28
2ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DISEÑO DE UN SISTEMA DE EVALUACION DE RENDIMIENTO Y SINTONIZACION PARA EL SISTEMA OPERATIVO UNIX SV"

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

GENARO ALONSO CHAVEZ FALCON

SOFIA DOMINGUEZ OLVERA

PEDRO GONZALEZ CELAYA

HECTOR TREJO ROMERO

ADOLFO VAZQUEZ VARGAS

Director: M.I. Juan Carlos Roa Beiza

MEXICO, D.F. AGOSTO 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

INTRODUCCION

Introducción.

El campo de los sistemas operativos ha sufrido muchas fallas costosas y frustrantes, pero, de la misma manera, ha logrado varios éxitos significativos. El sistema operativo UNIX, desarrollado en los laboratorios Bell, es uno de los éxitos más notables en el medio de la computación.

Aún cuando originalmente se diseñó con el objeto de ser una herramienta ideal para el desarrollo de programas, el contar con un lenguaje de comandos muy simple pero altamente poderoso, así como con un sistema de archivos independiente de los dispositivos, han hecho que el sistema sea ya muy popular en ambientes industriales, comerciales, universitarios y hasta pueden ser encontradas versiones para ser instaladas en computadoras personales.

Dado el repentino crecimiento y expansión que los sistemas UNIX han tenido, el software que se ha desarrollado para él no es tan sofisticado, todavía, como lo es para un computadora personal. Esto hace que en ocasiones los comandos asociados para dar información sobre monitoreo, estadísticas y el rendimiento en si del sistema, sean complicados y nada amigable para los usuarios, aún a nivel de administración, razón por la que creemos justificada la elaboración de programas que utilizando los recursos del mismo sistema, puedan hacer fácil la labor de administración y mantenimiento del rendimiento del sistema, y que además, se pueda

contar con una interfase visual amigable y de menúes altamente interactiva.

Deseamos enfatizar un poco en la evolución del sistema operativo, junto con sus limitantes, con objeto de tener las herramientas necesarias para justificar la necesidad de un sistema de monitoreo avanzado, y el porque los ajustes en el rendimiento del sistema deben ser hechos de manera más dinámica, utilizando todas esas herramientas para monitorear, por lo que a continuación incluimos una breve historia:

Historia

Entre 1965 y 1969, los laboratorios Bell junto con General Electric y el Instituto Tecnológico de Massachusetts, iniciaron un ambicioso proyecto para desarrollar un sistema operativo llamado MULTICS. Dicho sistema fue originalmente diseñado para operar en el computador GE-635, lo que hacía de MULTICS un sistema grande y complejo.

En 1969, debido a las muchas demoras, los laboratorios Bell se retiraron del proyecto, sin embargo algunos de los integrantes del equipo de investigación, que habían estado trabajando en el proyecto quedaron con la idea de hacer un sistema eficiente.

Ken Thompson, uno de los integrantes del grupo, había escrito un juego llamado "Viaje Espacial" (Space Travel) en la máquina GE-

635 con MULTICS, así que junto con Dennis Ritchie reescribieron el juego para ser utilizado por un PDP-7 de DEC (DIGITAL EQUIPMENT Co.), a la cual tenían acceso. Debido al primitivo medio ambiente de la PDP-7, para evitar el utilizar mucho tiempo en la reescritura, decidieron crear una estructura nueva para el control de archivos, así como algunas utilerías para la PDP-7, con lo que surgió una versión primitiva de un sistema operativo, al cual en 1970 fue dado el nombre de UNIX.

El significado de UNIX nunca ha sido revelado oficialmente, sin embargo puede ser una oposición al sistema MULTICS.

Thompson, Ritchie y algunas personas asociadas con ellos obtuvieron una computadora PDP-11 en un intercambio en el que prometieron agregar al sistema las capacidades de un procesador de textos, por lo que en 1973, el sistema UNIX fue escrito en un lenguaje de alto nivel llamado "C", facilitando su mantenimiento y su portabilidad a otras máquinas y surgiendo al mercado como la versión 4.

Después de la versión 7 UNIX llegó a ser un sistema comercial, y de éste se separaron algunas ramas que aportaron algunas mejoras en general.

Berkeley

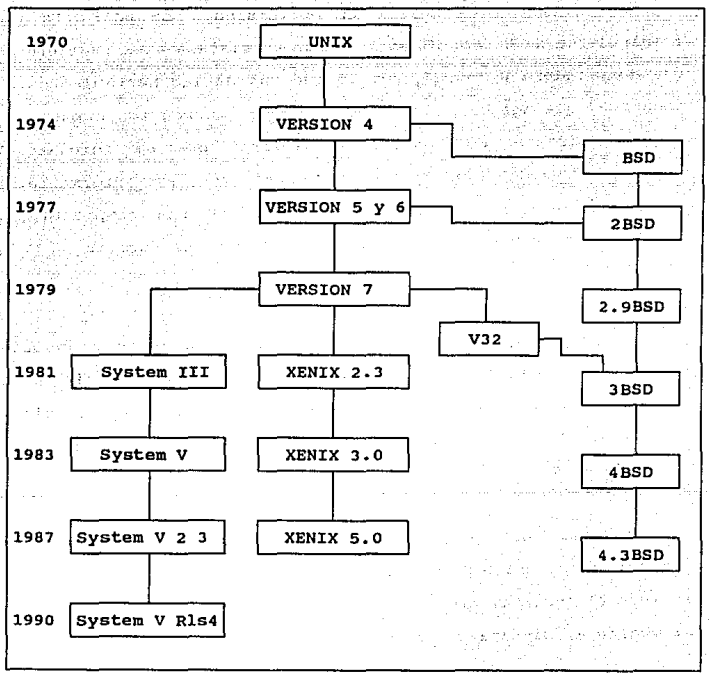
El profesor Fabry, en 1974 adquiere sin costo la versión 4 de

UNIX, con lo que la universidad de California, campus Berkeley inicia el uso del sistema operativo adicionando sus propias herramientas para hacerlo más eficiente.

Bill Joy y Chuck Haley, graduados de la universidad, crearon un editor de texto en línea llamado ex, que posteriormente modificaron para poder utilizar terminales de emulación, obteniendo el editor de pantalla completa vi. Joy también escribió el C-Shell que es un medio ambiente con sintaxis similar a la del lenguaje C, y a todo esto lo llamarón "Software de Distribución de Berkeley" (BSD).

La agencia de proyectos e investigaciones avanzados para la defensa (DARPA), proveyó los fondos que hicieron posible la siguiente versión, 4BSD, que incluyo una modificación al sistema de archivos hasta ese momento manejado, con un método llamado "**fast file system**", donde cada sistema de archivos es subdividido en grupos de cilindros, con el objeto de mantener los bloques del disco en la misma región física, aumentando la velocidad de acceso y previniendo la fragmentación.

Esta versión agrego el soporte de conexión en red con Ethernet, y la utilería para compartir disco duro con otras máquinas llamada "Network File System" (NFS)



Evolución del sistema UNIX

XENIX

Este sistema esta basado en la versión 7 de UNIX. La idea original fue de Microsoft en 1980 con un sistema para funcionar en una computadora personal, por lo que una de las herramientas más poderosas que agregó a las ya existentes, fueron los comandos de compatibilidad con MS-DOS, que hicieron que se iniciara la comunicación entre los dos sistemas operativos. Posteriormente, la unión con Santa Cruz Operations trató de lograr una compatibilidad en el código ejecutable escrito en XENIX con el escrito en UNIX.

AT&T

Es interesante hacer notar que a pesar de haber sido quién diseño a UNIX, AT&T no liberó ninguna versión comercial sino hasta 1982, cuando XENIX y 4.1BSD ya habían sido liberados, y le dio el nombre de System III, el cual ya contaba con el C shell y las capacidades de edición.

En 1983 surge el UNIX system V con el editor "vi" y una librería para manejo de pantalla de terminales llamada "curses". Se incluye el programa "init" para inicializar procesos para permitir el acceso al sistema operativo.

Es importante hacer notar que debido a que UNIX ha tenido un impulso importante de parte de varias empresa y universidades, la administración del sistema y sobre todo el enfoque que el sistema vi

ha tenido, que ha sido sin duda un enfoque no apto para usuarios finales, ha hecho que en utilerías de monitoreo y configuración sean complicadas y con una interfase difícil de comprender.

Esta es la razón que no ha llevado a crear el sistema de monitoreo y sintonización que pretende superar estas limitaciones y poner al servicio del administrador del sistema, a quién esta enfocado el desarrollo, una herramienta amigable y fácil de usar.

A continuación damos una breve descripción de la manera en que, capítulo con capítulo deseamos cubrir nuestro objetivo:
El capítulo I ofrece un panorama general de lo que son los sistemas operativos y sus elementos principales, esto con el fin de dar las bases para comprender el funcionamiento del sistema operativo UNIX lo cual se detalla en el capítulo II. En el capítulo III revisamos las herramientas para el monitoreo del sistema que se complementa con el capítulo IV donde se mencionan los elementos para la sintonización. En el capítulo V se presentan las especificaciones del análisis, diseño y la implementación del sistema y finalmente se presentan las conclusiones logradas como resultado de este trabajo.

I. SISTEMAS OPERATIVOS

I. SISTEMAS OPERATIVOS

I.1 ¿Qué es un Sistema Operativo?

Cuando se quiere comprender un concepto o una definición lo más importante es analizarla en su contexto, es decir en el lugar donde dicho concepto tiene su aplicación práctica.

Por ello, si vamos a tratar de comprender un Sistema Operativo interesará conocer antes el concepto de Sistema Informático del cual el Sistema Operativo es una parte muy importante.

I.1.1 Sistema Informático

Según la definición de la ISO **(International Standard Organization)** una computadora es un dispositivo para procesar datos, capaz de efectuar cálculos, y operaciones aritméticas y lógicas, sin intervención humana durante su funcionamiento.

Para llevar a cabo su cometido una computadora debe poseer una serie de recursos. Estos recursos pueden variar según el destino que le demos a la computadora. Así, una computadora que se destine para un proceso de investigación tendrá que contar con grandes elementos de cálculo que le permitan realizar operaciones complejas en un mínimo de tiempo, mientras que una computadora destinada al tratamiento de gran cantidad de datos, dispondrá de una memoria amplia donde almacenará los datos para su tratamiento.

Sin embargo, estas variaciones en la concepción de la computadora no son más que modificaciones de tipo cuantitativo pues cualitativamente todas las computadoras deben disponer de dos elementos básicos: Un Sistema Físico y un Sistema Lógico.

I.1.2 Sistema Físico.

El Sistema Físico es el conjunto de aparatos necesarios para el tratamiento eficaz de la información.

El Sistema Físico se conoce con el nombre de **Hardware** y se compone a su vez de varios elementos:

- La Unidad Central de Proceso (CPU)
- Memoria
- Dispositivo de entrada y salida (E/S)

El CPU es el elemento principal del sistema físico y tiene como misión realizar todas las operaciones del sistema, coordinar y controlar todas las partes que componen el Harwdare.

En el CPU se encuentran:

- La unidad de control que se encarga de analizar las operaciones y controlar a los demás elementos de la computadora, para que dichas operaciones sean ejecutadas eficazmente;
- la unidad lógica-aritmética que realiza las operaciones

aritméticas y lógicas;

- la memoria central, donde se almacenan los programas que hay que ejecutar para realizar un trabajo encomendado, y los datos necesarios para su ejecución.

La memoria auxiliar es el conjunto de dispositivos para almacenar de forma permanente datos y programas.

Los dispositivos de entrada y salida tienen como misión permitir el paso de información entre el CPU y el exterior y viceversa.

I.1.3 Sistema Lógico

El Sistema Físico por sí solo es un cuerpo sin vida, algo pasivo que necesita elementos para que lo pongan en marcha.

El Sistema Lógico es el conjunto de los recursos lógicos necesarios par el Sistema Físico se encuentra en condiciones de realizar todos los trabajos que se le vayan a encomendar.

Al Sistema Lógico se le conoce con el nombre de **Software**.

El **Software** se divide en dos tipos de programas: Los llamados programas del sistema, que manejan los recursos lógicos para que opere la propia computadora, y los llamados programas de aplicación

que son los que resuelven los problemas de los usuarios.

Algunos de los programas del sistema residen dentro de la memoria central y se conocen con el nombre de **Firmware**, tales programas suelen grabarse en el momento de la fabricación de la propia computadora. Otros programas sin embargo residen en memorias externas y son cargados en la memoria central cuando se pone en marcha la computadora o cuando se requiere alguna de sus ordenes.

Al conjunto de estos programas fundamentales del sistema se le conoce con el nombre de Sistema Operativo.

I.1.4 Sistema Operativo

El Sistema Operativo es el conjunto de programas del sistema que controla los recursos de la computadora y sirve de base para la escritura y ejecución de los programas de la aplicación.

Podemos decir que el Sistema Operativo es el conjunto de programas que actúan como intermediarios entre el **Hardware** de una computadora y los usuarios de la misma, permitiendo un uso eficaz y fácil del Sistema Informático.

Cuando un usuario necesita utilizar la computadora, requiere de una serie de recursos que han de ser administrados adecuadamente para obtener el máximo rendimiento, el Sistema Operativo actúa

entonces mediante una política adecuada óptima. También el Sistema Operativo controla los dispositivos de entrada/salida y supervisa la ejecución de los programas del usuario.

Aparece entonces una doble función del Sistema Operativo, por una parte como gestor de recursos y por otra como controlador de los programas y de su movimiento entre el CPU y los equipos periféricos.

En definitiva podemos definir de una manera formal lo que es un Sistema Operativo como: "El conjunto de programas de control y programas de proceso que relacionados entre si dan coherencia a las acciones que lleva a cabo una computadora optimizando el uso de la misma mediante un aprovechamiento eficiente de todos los recursos y una ordenada comunicación entre usuario y computadora".

I.1.5 Elementos de un Sistema Operativo.

Los elementos de un Sistema Operativo están formados por el conjunto de programas que utiliza la computadora para realizar un trabajo.

Estos programas se clasifican en tres tipos:

A) Programas de Utilidad

B) Programas de Servicio

C) Programas de Control

A) Programas de Utilidad.

Los programas de utilidad son los que van destinados a una función especifica, estos programas son diseñados par un **Hardware** especifico y elaborados totalmente dependientes de la instalación informática concreta.

Estos programas de utilidad realizan funciones de tipo repetitivo como pueden ser clasificaciones de los archivos, prueba de localización de errores, protección de archivos, etc.

Dentro de los programas de utilidad se distinguen dos clases:

- Los organizadores del sistema
- Los organizadores de los archivos

Los primeros se ejecutan bajo la supervisión del Sistema Operativo mismo y permiten manejar las librerias de programas asi como la protección de la información que maneja la computadora.

Los organizadores de los archivos se encargan de manipular los datos clasificando o reorganizando los registros de los diversos archivos.

B) Programas de Servicio.

El segundo tipo de programas son los programas de servicio,

estos están destinados a gestionar y permitir la ejecución de los programas de aplicación. En general los programas de aplicación están escritos en lenguaje comprensible para el usuario y poco comprensible para la computadora. Es por esto que es necesario que existan programas traductores que hagan comprensible los programas a la computadora y permitan su conservación, su corrección cuando existan errores, y su readaptación a los casos particulares.

Entre los programas de servicio podemos distribuir:

B.1) Programas Traductores

B.2) Programas de gestión de librerías

B.3) Programas editores

B.1) Programas Traductores.

Los programas traductores realizan el paso del lenguaje de alto nivel en que están escritos los programas de aplicación, al lenguaje de máquina que comprende la computadora.

Al programa original en lenguaje de alto nivel se suele denominar programa fuente y el que queda tras la traducción se le conoce con el nombre de programa objeto.

No todos los traductores actúan de la misma forma y según como realicen el proceso de traducción, se dividen en:

- Compiladores

- Interpretes
- Preprocesadores
- Ensambladores

Los compiladores traducen todo el programa fuente a código de máquina dejándolo dispuesto para su ejecución con poca o ninguna preparación posterior.

Un interprete realiza también la traducción del programa fuente pero lo hace paso a paso, ejecutando cada instrucción traducida.

Los compiladores hacen uso de mas memoria donde almacenar el programa objeto, pero actúa con mayor rapidez en las sucesivas ejecuciones del programa.

Los interpretes por el contrario requieren mucha menos memoria, pero actúan de forma mas lenta.

Los preprocesadores modifican ligeramente el programa fuente, traduciéndolo a otro lenguaje que aún comprende el usuario permitiéndole así mayor potencia antes de la traducción completa.

El ensamblador es un compilador sencillo que permite la traducción de una instrucción del lenguaje fuente a una instrucción en lenguaje de máquina.

B.2) Programa de Gestión de Librerías.

Son programas destinados a conservar y organizar el conjunto de programas de aplicación. Entre sus funciones caben destacar:

- Incorporación de programas
- Anulación o borrado de programas
- Organización de programas

Como los programas pueden estar en lenguaje de alto nivel, en módulos objeto, o preparados para la ejecución, se pueden clasificar las librerías en tres modalidades:

- Librería fuente
- Librería objeto
- Librería ejecutable

B.3) Programas Editores.

Son los programas que permiten la escritura, grabación y corrección de los programas de aplicación. Los editores combinan la diversidad de aplicaciones que poseen con la facilidad de su manejo, permitiendo todo tipo de comodidades para el programador.

Los editores son de propósito general, es decir no van destinados a un lenguaje concreto, sino que con un editor, cualquier programador puede tratar el lenguaje que desee.

C) Programas de Control

Por último los programas de control son los programas que facilitan y mejoran el rendimiento del uso de las computadoras. Son los encargados de llevar a cabo la administración de los recursos del sistema, facilitando el acceso ordenado y eficaz de los procesos a los recursos disponibles en el sistema en cada momento, impidiendo el interbloqueo de actividades.

Las rutinas de control permiten a muchos programas la utilización de los recursos de la computadora. Así, es posible ejecutar de forma concurrente dos o más programas cada uno de los cuales ocuparía por si solo toda la memoria. Todo es cuestión de administrar bien los recursos disponibles.

Los programas de control suelen estar formados por rutinas o módulos, cada uno de los cuales administra uno de los tras grupos fundamentales:

- **Hardware**
- Archivos
- Trabajos

El administrador del **Hardware** se encarga de controlar u administra el CPU, la memoria y las rutinas de entrada y salida.

La gestión de archivos maneja los datos externos moviéndolos

entre la memoria y los dispositivos periféricos.

El administrador de trabajos se encarga de asignar recursos a las diversas tareas y permite la comunicación del sistema con el exterior.

SISTEMA OPERATIVO

Programas de utilidad

- Programas organizadores del sistema
- Programas organizadores de archivos

Programas de servicio

- Traductores
  - Compiladores
  - Intérpretes
  - Preprocesadores
  - Ensambladores
- Programas de gestión de librerías
  - Incorporación de programas
  - Anulación o borrado de programas
  - Organización de programas
- Editores

Programas de control

- Administradores de hardware
- Administradores de archivos
- Administradores de trabajos

Figura I.1.1 Elementos de un Sistema Operativo

I.1.6. Evolución de los Sistemas Operativos

Las primeras computadoras estaban compuestas exclusivamente por el sistema físico que se reducía a una computadora de grandes dimensiones al que había que dotar de todas las instrucciones para que realizará un trabajo.

No existía un sistema operativo propiamente dicho, los usuarios ponían su nombre y el tiempo que necesitaba utilizar la máquina en una lista de espera. Durante el tiempo de utilización, cada usuario disponía de todos los recursos y empleaba la ejecución paso a paso para hacer correr sus programas. Toda la memoria era utilizada por el usuario.

Este modo de operación llamado "Puerta Abierta" era poco económico, ya que la inactividad del CPU, esperando decisiones del programador podía alcanzar hasta el 90% del tiempo. Al terminar su trabajo el procesador quedaba nuevamente inactivo hasta la llegada del siguiente usuario.

I.1.6.1 Pasos de Trabajo

Para realizar un programa o un conjunto de programas el usuario tenía que seguir una secuencia manual de trabajos (JOB STEP) que son básicamente los siguientes:

- Escribir el programa en un lenguaje de alto nivel
- Perforar las tarjetas correspondientes e introducirlas en el dispositivo lector
- Montar el carrete de cinta donde se encontraba el traductor de lenguaje de alto nivel a lenguaje ensamblador
- Ordenar la lectura y compilación del programa

Toda esta serie de pasos consumía una gran cantidad de tiempo al que se le denominó **SETUP TIME** (tiempo de preparación).

I.1.6.2 Controladores de Dispositivos

Con el tiempo aparecieron dispositivos de Entrada/Salida mucho más perfeccionados, como: Impresoras, Lectoras de Tarjetas, Cintas Magnéticas, etcétera. Para manejar con aprovechamiento dichos dispositivos, el programador tuvo que añadir a sus programas los programas necesarios para controlar cada uno de los dispositivos que iba a utilizar.

Se adoptó el criterio de grabar en la memoria los programas

específicos encargados de manejar cada dispositivo y tenerlos siempre a disposición de todos los usuarios.

A cada uno de estos programas específicos se le llamó "Controlador de Dispositivo". El esquema de la memoria quedo como se indica en la figura I.1.2.

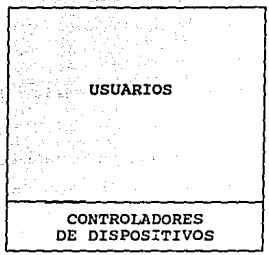


Fig. I.1.2 Esquema de memoria

I.1.6.3 Proceso por Lotes

Para reducir el tiempo de preparación se recurrió a la agrupación de los trabajos que poseían características similares o que necesitaban utilizar los mismos recursos lógicos. Asi surge el concepto de "Trabajos por Lotes" o "Procesc BATCH".

El operador reunía las tarjetas correspondientes a trabajos similares, transfiriendo luego la información de dichas tarjetas a una cinta magnética. Una vez que era procesado todo el lote, se transmitían los resultados a otra cinta para luego ser impresos.

Pero aún con este sistema se demoraban mucho las ejecuciones de los programas, ya que el operador debía cambiar carretes, ordenar las ejecuciones, recoger resultados, etcétera. Parecía conveniente que las secuencias de trabajo se hicieran de forma automática.

I.1.6.4 Secuencia automática de Trabajos.

Se realizaron programas que permitiesen la secuencia automática de trabajos. Estos programas se incorporaron a la memoria y tomaron la mayoría de las decisiones que antes estaban encargadas a los operadores. Pero era necesario indicar donde comenzaba cada trabajo, para ello se colocaron tarjetas con características especiales que marcaban el principio y el fin de cada programa.

A éstas tarjetas se les llamó de control y al programa que las interpretaba se le denominó "Lenguaje de Control de Trabajos" (JCL: Job Control Languaje). En la memoria deberían de estar los programas controladores de dispositivos, de gestión de errores interpretes, etcétera. La memoria tenía el aspecto mostrado en la

siguiente figura I.1.3.

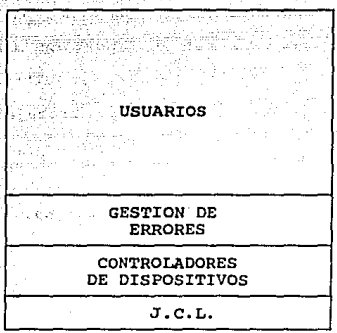


Fig. I.1.3 Esquema de memoria avanzado

Surgía entonces la pregunta... ¿Comó instalar todos estos programas en memoria?. La solución se llama IPL **(Initial Program Load)** o programas de carga inicial. Este programa contenido en la memoria ROM **(Read Only Memory)** se ejecuta automáticamente al conectar la computadora y su misión consiste en localizar el dispositivo periférico donde se encuentran los programas de control y grabarlos en la memoria RAM **(Random Access Memory)** para ser ejecutados en el momento oportuno.

I.1.6.5. Estados de Ejecución

El conjunto de programas del que hemos hablado antes constituye por si mismo un Sistema Operativo básico, pero aún así se requieren de funciones que perfeccionen dicho Sistema Operativo.

Cuando los programas necesitan utilizar dispositivos de entrada/salida (E/S) deben pedírselo al monitor mediante instrucciones denominadas "llamadas al Sistema". Una vez que concluye el servicio de E/S requerido, el control retorna al procesador.

De lo anterior se deduce que mientras se esta atendiendo una E/S el procesador permanece inactivo; por ello, es conveniente que las operaciones de entrada y salida se realicen lo mas rápidamente posible. Como los dispositivos de E/S son mas lentos que el procesador, se requiere un mecanismo para evitar las largas esperas de la unidad central.

Este mecanismo consiste en almacenar temporalmente la información que envía la unidad central y posteriormente dársela poco a poco a los dispositivos periféricos. Así la unidad central se ve pronto libre para realizar otros trabajos. Este mecanismo se conoce como **BUFFER** (área de almacenamiento temporal) y el sistema se conoce con el nombre de entradas-salidas buffereadas o buffering.

I.1.6.6 Interrupciones

Con el sistema de buffer se consigue un importante ahorro de tiempo; sin embargo para que dicho sistema sea eficaz, se requiere conocer lo antes posible cuando un dispositivo de E/S ha terminado una operación.

Este problema se resuelve con el servicio de interrupciones. Tan pronto como un dispositivo de E/S termina una operación, interrumpe al CPU y ésta cede el control a un programa residente en memoria conocido con el nombre de Servicio de Interrupciones que se encarga de iniciar la siguiente operación de E/S.

I.1.6.7 Spooling.

Cuando se utilizaban tarjetas perforadas y cintas magnéticas, el problema consistía en que la lectora de tarjetas no podía escribir en un extremo de la cinta mientras el CPU leía el otro. El uso del disco magnético elimina estas dificultades, ya que se puede pasar moviendo la cabeza lectora de una parte del disco donde se han grabado tarjetas a otra donde el CPU puede leer otras.

Este sistema de operaciones simultáneas se conoce con el nombre de **SPOOLING** o SPOOL (**Simultaneus Peripheral Operation On-Line**), y esencialmente lo que hace es utilizar el disco como si se tratase de un gran **BUFFER**.

Nada impide a esta forma de ejecución procesar un trabajo mientras se esta imprimiendo otro.

Con la combinación de los beneficios que proporciona el sistema **BUFFERING** y **SPOOLING**, el CPU puede planificar la forma y el orden de ejecutar los trabajos con vistas a obtener un mayor rendimiento.

I.1.7 Tipos de Sistemas Operativos

La posibilidad el CPU de poder planificar los trabajos que están en el disco proporciona una ventaja adicional, la cual consiste en que el CPU ya no permanece inactivo sino que cuando un trabajo ha de detenerse para efectuar una E/S o por cualquier suceso, tomará otro trabajo y lo ejecutará.

I.1.7.1 Multiprogramación.

La multiprogramación es un modo de trabajo en el que el Sistema Operativo permite ejecutar varios trabajos simultáneamente. En los Sistemas Operativos multiprogramados surge el concepto de planificar el procesador dando lugar a nuevas estructuras para los Sistemas Operativos.

I.1.7.2 Tiempo Compartido

La idea de trabajos por lote no esta en contradicción con la multiprogramación pero con la aparición de los discos se hizo posible el acceso inmediato a todos los trabajos y los procesos BATCH no se agrupasen por procesos similares sino por otras características.

El siguiente paso en el desarrollo de Sistemas Operativos fue la introducción de la Multiprogramación Interactiva. Aparecían al mismo tiempo las terminales conversacionales (teclado-pantalla), en los que el usuario ya no tenía que suministrar todos los datos al principio de la ejecución del proceso, sino que podía ir dándolos a media que el proceso los iba necesitando, de igual forma que iba recibiendo respuesta inmediata de sus datos.

En este modo de operación la organización no se realiza por trabajos, sino por sesiones. Una sesión es todo el conjunto de trabajos que se ejecuten desde que un usuario se conecte a la computadora hasta que se despide de la misma. Ver figura I.1.4

- Ser muy conversacionales
- Atender a varios usuarios simultáneamente
- Ofrecer tiempos de respuesta relativamente cortos (segs.)
- Utilizar técnicas de **BUFFERING** y **SPOOLING**

Fig. I.1.4 Sistema Operativo de tiempo compartido

I.1.7.3 Tiempo Real

El tiempo real es otra modalidad de los sistemas operativos multiprogramados, en que se necesita un tiempo de respuesta pequeño ante cualquier petición. Podemos decir que un sistema trabaja en tiempo real si el tiempo de respuesta permite controlar y regular el medio sobre el que opera. Ver figura I.1.5

Las características principales del tiempo real son:

- Fuertes restricciones en el tiempo de respuesta (ms)
- Manejo eficiente de interrupciones
- Manejo sencillo de prioridades
- El sistema debe permanecer prácticamente inactivo para atender lo antes posible cualquier evento de entrada.

I.1.7.4 Proceso Distribuido

El proceso distribuido consiste en la conexión de computadoras entre si a través de una gran variedad de dispositivos, existiendo varias modalidades entre las que podemos citar la conexión de varias computadoras compartiendo un mismo almacenamiento principal o aquellos que se conectan a una misma red nacional o internacional para el intercambio de información. Ver figura I.1.6

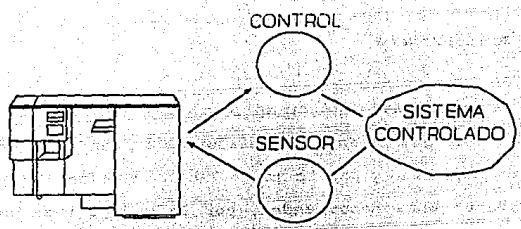


Fig. I.1.5 Sistema Operativo de Tiempo Real

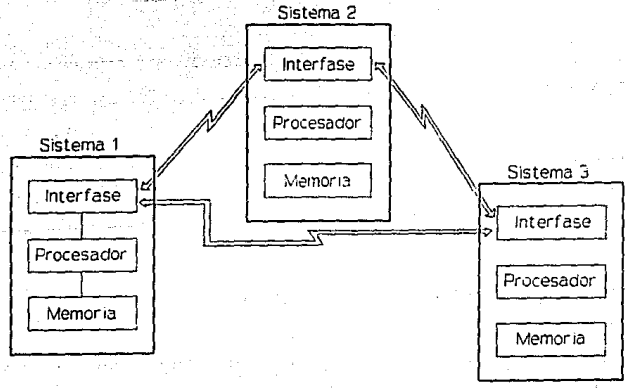


Fig. I.1.6 Sistema Operativo de Proceso distribuido

I.1.7.5 Multiproceso

En la actualidad existen aplicaciones que manejan tal cantidad de información que un solo procesador no es capaz de procesarla en el tiempo requerido y por lo tanto las computadoras convencionales no satisfacen esta necesidades, por lo que se hizo necesaria la descomposición de algoritmos en subalgoritmos mas sencillos, de manera que cada uno pueda tratar un subconjunto de los datos con cierta independencia de los otros. Al final se conjuntan esto datos obteniendo el resultado final de todo el proceso.

Para ello se están desarrollando actualmente diversas máquinas que siendo una sola computadora, contienen varios procesadores que pueden trabajar conjuntamente. Con ello queda satisfecha la necesidad de procesos de varios algoritmos simultáneamente (cada uno en un procesador) y con un ahorro sustancial en el tiempo de ejecución. Ver figura I.1.7.

I.1.8 Estructura de los Sistemas Operativos

A lo largo de la historia se han seguido diversos procedimientos para la implementación de los Sistemas Operativos, veamos algunos de ellos:

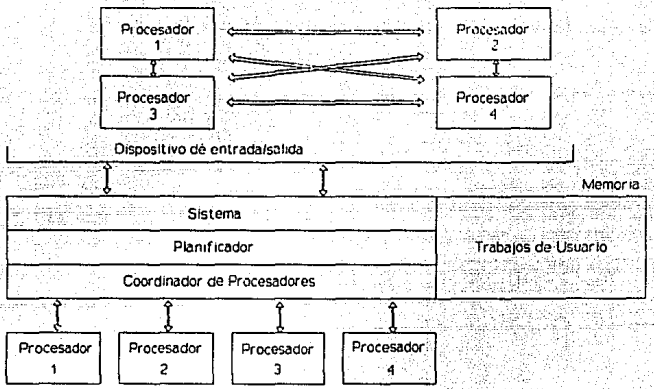


Fig. I.1.7 Multiprocesos

I.1.8.1 Sistemas Monolíticos

Consiste en escribir el sistema operativo como un conjunto de procedimientos cada uno de los cuales tiene una interfase que le permite llamar a cualquier otro procedimiento cuando lo necesite. Para construir el Sistema Operativo se compilan todos los procedimientos y se enlazan entre sí en un archivo único.

A pesar de ello posee una pequeña estructura que mediante una instrucción permite tener la máquina funcionando en dos modos: El

modo usuario y el modo supervisor. Ver figura I.1.8.

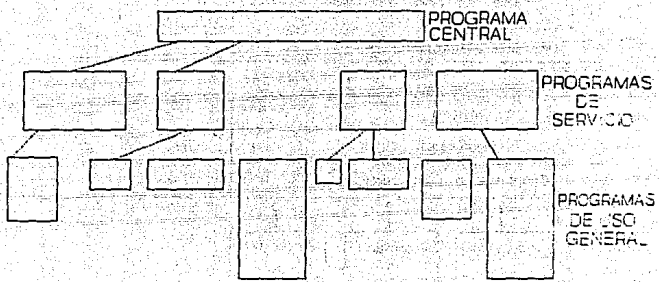


Fig. I.1.8 Estructura de un Sistema Monolitico

I.1.8.2 Sistemas Jerárquicos

Consisten en organizar el Sistema Operativo como una serie de estratos cada uno de los cuales interactúa con su entorno a través de interfases bien definidas.

En cada problema complejo se divide en módulos simples, cada uno de los cuales realiza una función única. Los módulos son pequeños, fáciles de comprender, diseñar y cambiar en caso necesario.

En cada nivel de la jerarquía se ignoran los detalles de actuación de los niveles inferiores, produciendo un principio de ocultar información que favorece la compartición multiusuario con independencia de los datos de cada uno de los usuarios del sistema. En general los niveles de todos los sistemas jerárquicos suelen tener la misma estructura:

- El primer nivel suele corresponder al núcleo que contiene un planificador para seleccionar el trabajo que se ejecutará a continuación y gestiona las interrupciones.
- Los niveles segundo, tercero y cuarto se encargan de E/S, gestión de memoria y sistema de almacenamiento externo.
- Los siguientes niveles constituyen la interfaz entre el Sistema Operativo y los usuarios. Ver figura I.1.9.

I.1.8.3 Máquinas Virtuales

Una máquina virtual consiste en un mecanismo que crea el Sistema Operativo de forma que una sóla máquina real aparenta funcionar como varias máquinas independientes cada una de las cuales es utilizada por uno o varios usuarios.

Para ello se comparten los recursos de la computadora de forma que cada usuario crea estar utilizando su propio procesador o su propia memoria. El centro del sistema se conoce como monitor de la máquina virtual y se implementa en el **Hardware** realizando multiprogramación con la creación de un estrato superior formado

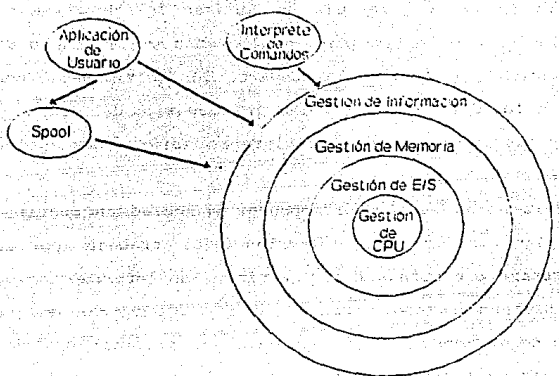


Fig. I.1.9 Estructura de Sistemas Jerárquicos

por varias máquinas virtuales que son copia exacta de la máquina real. Así cada máquina virtual puede disponer de su propio Sistema Operativo, y estos pueden ser diferentes para cada usuario. Ver figura I.1.10.

I.1.8.4 Servidores

En las máquinas virtuales el núcleo resulta demasiado complejo y por ello algunos Sistemas Operativos modernos desplazan parte de las rutinas de núcleo a estratos superiores; para ello implementan funciones del Sistema Operativo en procesos del usuario.

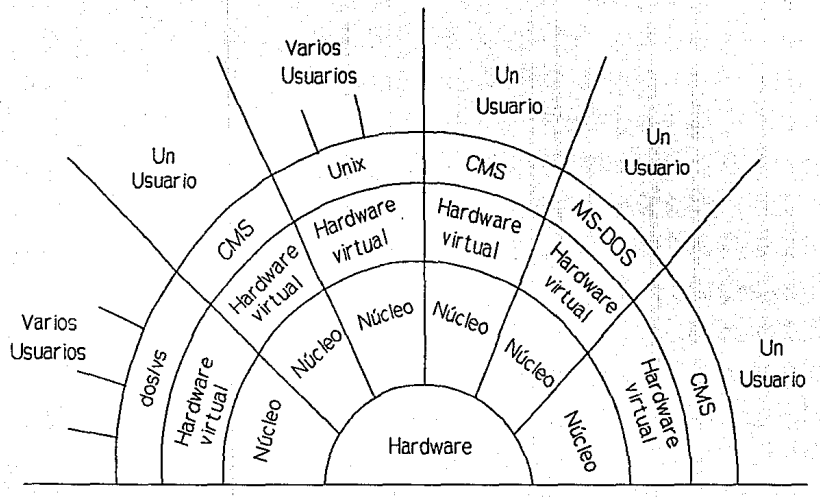


Fig. I.1.10 Máquina Virtual

Para solicitar un servicio el proceso del usuario envía una solicitud a un proceso llamado "Servidor" que realiza el trabajo y da respuesta al usuario. El núcleo sirve entonces de enlace entre usuario y servidores. Este modelo del Sistema Operativo se utiliza en los Sistemas Distribuidos. Ver figura I.1.11.

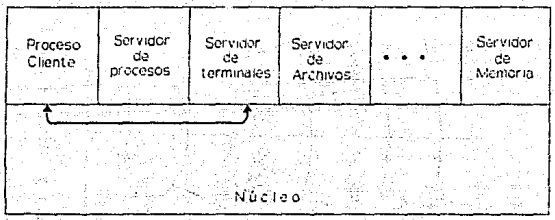


Fig. I.1.11 Servidores

I.2 Administración del procesador

Si se puede décir que una computadora tiene cerebro, ese cerebro sería la unidad central de proceso (Central Processing Unit, CPU). Es en la unidad central de proceso o CPU en donde la computadora lleva a cabo sus funciones aritméticas y lógicas y en donde se controla la operación del sistema de computación entero.

El CPU se divide en dos partes: la unidad de control y la unidad aritmética lógica. La función de la unidad de control consiste en obtener una sóla instrucción de la memoria principal y decodificarla. Una vez que la unidad de control sabe lo que hay que hacer, se da el control a la unidad aritmética lógica, la cual ejecuta la operación. El primer paso de este ciclo se llama tiempo de instrucción o tiempo "I". El segundo paso, que implica a la unidad aritmética lógica, se llama tiempo de ejecución o tiempo "E". La terminación de ambas partes representa un ciclo de máquina. Así es como trabaja la computadora: la unidad de control determina qué es lo que ha de hacer y da el control a la unidad aritmética lógica, misma que ejecuta la tarea y vuelve el control a la unidad de control, la cual decodifica la siguiente instrucción y entrega el control a la unidad aritmética lógica, y se repite ese patrón de ciclos muchas veces.

La administración del CPU es la base de los sistemas operativos multiprogramados. Alternando el uso del CPU entre los procesos, el sistema operativo puede hacer que el computador sea más productivo.

El concepto más importante en los sistemas operativos modernos es el de multiprogramación; es decir varios programas que permanezcan en memoria al mismo tiempo, pueden compartir el CPU. Este esquema mejora la eficiencia global del sistema informático realizando más trabajo en menos tiempo. Por ejemplo, un trabajo se

ejecuta hasta que deba esperar, normalmente para acabar alguna solicitud de E/S. En un sistema informático sencillo, el CPU permanecería entonces ocioso. Todo este tiempo de espera se desperdicia; no se realiza ningún trabajo útil. Con la multiprogramación se trata de emplear este tiempo productivamente, se mantienen en memoria varios trabajos al mismo tiempo; cuando uno de ellos tiene que esperar, el sistema operativo retira el CPU de ese trabajo y lo dedica a otro, y así sucesivamente. Cada vez que un trabajo tiene que esperar, hay otro que puede ocupar el CPU.

Los beneficios de la multiprogramación son el incremento de la utilización de el CPU y una mayor productividad. La productividad es la cantidad de trabajo realizado en un intervalo de tiempo dado.

La administración del CPU es una función fundamental del sistema operativo, puesto que casi todos los recursos del computador se planifican antes de utilizarlos. Por supuesto, el CPU es uno de los recursos primarios del computador. Por tanto, su administración es básica en el funcionamiento del sistema operativo.

El CPU ejecuta un gran número de trabajos o tareas. Aunque su principal cometido es la ejecución de los programas y trabajos de usuario, el CPU también es necesario para otras actividades del sistema. Cada acción del sistema informático es iniciada por el CPU. El CPU tiene que responder a excepciones de error, a llamadas

de programa e interrupciones de E/S. Las interrupciones pueden producirse por la interrupción de caracteres individuales por el teclado de una terminal en tiempo compartido, o ser el resultado del programa de canal transfiriendo grandes bloques entre la memoria y los dispositivos de almacenamiento periférico.

Un problema persistente con los sistemas operativos consiste en cómo llamar a todas las actividades del CPU. Un sistema en **batch** (lote) ejecuta trabajos, mientras que un sistema en tiempo compartido tiene programas de usuario. Aún en un sistema monousuario, un usuario puede ser capaz de ejecutar varios programas al mismo tiempo, uno interactivo y varios programas batch. Incluso si un usuario puede ejecutar solamente un programa simultáneamente, el sistema operativo necesita soportar sus propias actividades internas, como el **spooling** (poleo).

En muchos sentidos, todas estas actividades son similares, de modo que a todas ellas las llamamos procesos. Un proceso es un programa en ejecución. Típicamente, un trabajo en batch, un programa de usuario en tiempo compartido, una tarea del sistema como el spooling son procesos.

I.2.1 Ciclo de ráfagas cpu-e/s

El éxito de la administración del CPU descansa en las siguientes propiedades observadas de los procesos: la ejecución de

un proceso es un ciclo de ejecución en el CPU y espera para E/S. Los procesos alternan entre estos dos estados. La ejecución de un proceso empieza con una ráfaga de CPU, la siguiente una ráfaga de E/S, seguida a su vez por otra ráfaga de CPU, y luego por otra ráfaga de E/S, etc. Finalmente la última ráfaga de CPU terminará con una llamada al sistema para terminar la ejecución y no con otra ráfaga de E/S (fig I.2.1).

Se ha medido la duración de esas ráfagas de CPU. Aunque varían mucho de proceso a proceso y de computador a computador, tienden a tener una curva de frecuencias muy similar al de la figura I.2.2. Generalmente, la curva es exponencial. Hay un gran número de ráfagas de CPU muy cortas y un pequeño número de ráfagas muy largas. Un programa orientado a la E/S típicamente tendría muchas ráfagas de CPU muy cortas, mientras que un programa orientado al CPU las tendría muy largas. Esta distribución puede resultar muy importante en la selección del algoritmo apropiado para la administración del CPU.

```
LOAD
STORE
ADD                    >Ráfaga de CPU
STORE
READ

espera para E/S        >Ráfaga de E/S

STORE
INCREMENT INDEX        >Ráfaga de CPU
WRITE

espera para E/S        >Ráfaga de E/S

LOAD
STORE
ADD                    >Ráfaga de CPU
STORE
READ

espera para E/S        >Ráfaga de E/S
```

(Figura I.2.1)

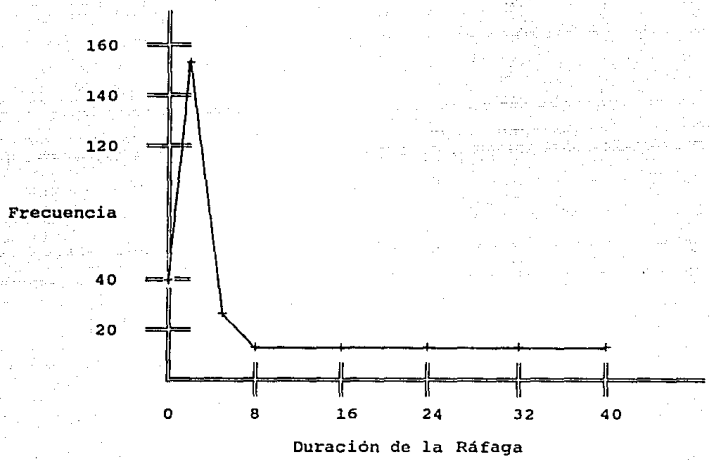


Fig. I.2.2 Histograma de Duración de Ráfagas de CPU

I.2.2 Estados del proceso

Un proceso es un programa en ejecución. A medida que se ejecuta el programa, el proceso cambia de estado. El estado de un proceso se define por su actividad actual. La ejecución de un proceso es una secuencia alternante de ráfagas de CPU y de E/S, empezando y acabando con una ráfaga de CPU. Así, todo proceso puede encontrarse en uno de los siguientes estados: **nuevo, activo, esperando o detenido.**

De hecho éstos estados aún pueden refinarse más. Puesto que el CPU puede ser compartido por varios procesos, un proceso activo puede estar esperando al CPU o ejecutándose. Un proceso que esta esperando al CPU está preparado, el que tiene el CPU asignado se encuentra en ejecución.

I.2.3 Bloque de control de proceso

Cada proceso esta representado en el sistema operativo por su propio bloque de control de proceso (también llamado bloque de control de trabajo o bloque de control de tarea). Un bloque de control de proceso (PCB) es una estructura o bloque de datos que contiene muchas de las informaciones asociadas a un proceso en especifico y que incluyen:

- El estado del proceso, puede ser nuevo, preparado, en ejecución, esperando o detenido.

- Los registros del CPU, varían en número y tipo, según sea la arquitectura del computador. Incluyen acumuladores, índices y registros de propósito general, más cualquier información sobre códigos de condición. Paralelamente al contador del programa, hay que salvar esta información de estado cuando se produce una interrupción, para permitir que el proceso continúe correctamente más tarde.
- Cualquier información de gestión de memoria, incluyendo registros base y límite o tablas de página.
- Cualquier información para contabilidad, incluyendo el tiempo real y de CPU empleados, límites de tiempo, identificadores de contabilización, de trabajo o proceso, etc.
- Información de estado E/S, incluyendo las solicitudes de E/S pendientes, los dispositivos de E/S asignados, archivos abiertos, etc.
- Información sobre la administración del CPU, que incluye prioridad del proceso, apuntadores a las colas de administración, etc.

El PCB sirve simplemente como depósito de cualquier información que pueda variar de proceso a proceso; tiene que almacenarse en la memoria del monitor. Esta memoria puede gestionarse de varias maneras. El enfoque más sencillo es, con mucho, la predeclaración del número máximo de procesos y la preasignación estadística del espacio suficiente para todos los

PCB. Pero más generalmente el número de PCB cambiará con el tiempo. Por tanto, puede resultar más conveniente una política de gestión de memoria dinámica.

I.2.4 Colas de administración

Los procesos que están preparados y esperando para ejecutarse se mantienen en una lista llamada cola de preparados. Esta es generalmente una lista enlazada. Una cabecera de cola contendrá los apuntadores al primero y al último PCB de la lista. Cada PCB tiene un campo apuntador que apunta al siguiente proceso en la cola de preparados.

Debemos darnos cuenta de que esta cola no es necesariamente una lista **first-in-first-out** (FIFO) (primero en entrar, primero en salir), puede implementarse como una cola con prioridades, una pila o simplemente como una lista enlazada no ordenada; no obstante, todos los procesos de la cola están alineados en espera de la oportunidad de ejecutarse en el CPU.

Una representación frecuente para la discusión de la administración del CPU es el diagrama de colas, como el de la figura I.2.3. Cada rectángulo representa una cola. Hay dos tipos de colas: la cola de preparados y un conjunto de colas de dispositivos. Los rectángulos E/S representan los recursos de las colas y las flechas indican el flujo de los procesos del sistema.

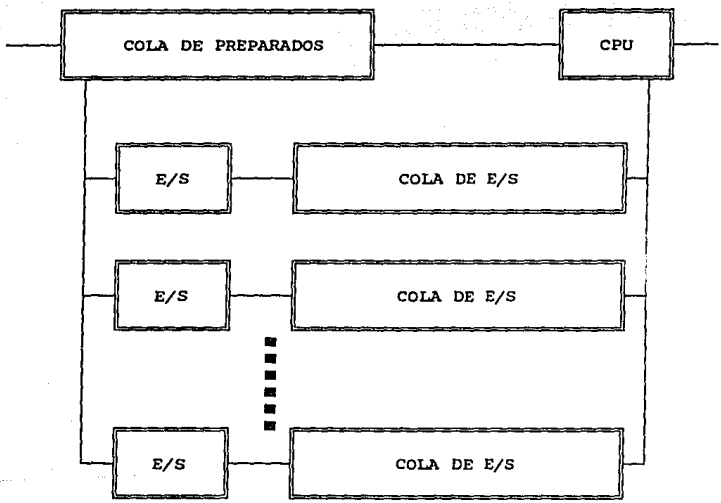


Fig. I.2.3 Representación de un diagrama de colas para la planificación del CPU.

Un proceso entra en el sistema desde el exterior y se sitúa en la cola de preparados. Espera en ella hasta que es seleccionado para el CPU. Después de ejecutarse en el CPU espera una operación de E/S pasando a una cola de E/S. Finalmente recibe servicio del dispositivo de E/S, y vuelve a la cola de preparados. El proceso continúa este ciclo CPU-E/S hasta que acaba; entonces sale del sistema.

Puesto que nuestro máximo interés es la administración del CPU, prescindiremos de los detalles del subsistema de dispositivos de E/S múltiples, substituyéndolo por una cola de espera de E/S y un servidor de E/S (fig I.2.4).

I.2.5 Planificadores

Un sistemas operativo tiene varios planificadores. Hay dos planificadores del CPU principales el de largo plazo y el de corto plazo.

El planificador a largo plazo (o planificador de trabajos) determina cuáles son los trabajos que se admiten en el sistema para su procesamiento. En un sistemas batch, con frecuencia hay más

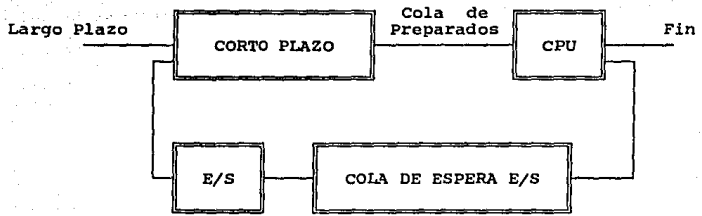


Fig. I.2.4 Diagrama de colas simplificado

trabajos de los que pueden ejecutarse inmediatamente. Estos trabajos se transfieren a un dispositivo de almacenamiento periférico, donde se mantienen para su posterior ejecución. El planificador a largo plazo selecciona los trabajos de este conjunto y los carga en memoria para su ejecución. El planificador a corto plazo (o planificador del CPU) selecciona entre los trabajos en memoria cuáles están prestos para la ejecución y asigna el CPU a uno de ellos.

La distinción básica entre estos dos planificadores es la frecuencia de su ejecución. El planificador a corto plazo tiene que seleccionar un nuevo proceso para el CPU con mucha frecuencia. Un proceso posiblemente se ejecuta durante sólo unos pocos milisegundos antes de tener que esperar por haber realizado una llamada de E/S. Con frecuencia el planificador a corto plazo se ejecuta al menos una vez cada diez milisegundos. Debido al corto período de tiempo entre ejecuciones, el planificador a corto plazo tiene que ser muy rápido. Si tarda un milisegundo en decidir ejecutar un proceso durante diez milisegundos, entonces 1/(10 + 1) = 90% del CPU se está utilizando (improductivamente) para planificar el trabajo.

Por otra parte, el planificador a largo plazo se ejecuta con mucha menor frecuencia. Pueden pasar minutos entre la llegada de nuevos trabajos al sistema. El planificador a largo plazo controla el grado de multiprogramación (número de procesos en memoria). Si

el grado de multiprogramación es estable, entonces la tasa media de entrada de trabajos al sistema tiene que ser igual a la tasa media de salida. De esta manera es posible que el planificador a largo plazo tenga que ser invocado solamente cuando un trabajo abandone el sistema. Debido al mayor intervalo entre ejecuciones, el planificador a largo plazo puede permitirse tardar más tiempo para decidir qué trabajo debe seleccionarse para su ejecución.

El hecho de que el planificador a largo plazo realice una selección cuidadosa puede ser de la mayor importancia. En general, la mayoría de los trabajos pueden clasificarse como orientados a la E/S u orientados al CPU. Es importante que el planificador a largo plazo seleccione un buen mix (mezcla) de trabajos entre los orientados a la E/S y al CPU. Si todos los trabajos son intensivos en E/S, la cola de preparados estará casi siempre vacía y el planificador de corto plazo tendrá poco trabajo. Si todos los trabajos son intensivos en CPU, la cola de espera para E/S estará casi siempre vacía y de nuevo el sistema estará desequilibrado. Para su mejor aprovechamiento, el sistema debe tener una combinación de trabajos orientados al CPU y a la E/S.

En algunos sistemas, el planificador a largo plazo puede ser mínimo o no existir. Por ejemplo, los sistemas en tiempo compartido frecuentemente carecen de planificador a largo plazo y se limitan a colocar en memoria a disposición del planificador a corto plazo, cualquier nuevo proceso. La estabilidad de este sistema depende,

bien de una limitación física (como un número limitado de terminales disponibles) o bien de la propia moderación de los usuarios humanos.

Algunos sistemas, especialmente aquellos que tienen memoria virtual o tiempo compartido, introducen un nivel intermedio adicional de planificación. Este planificador a medio plazo (fig. I.2.5) parte de la idea de que, en ocasiones puede resultar ventajoso eliminar procesos de memoria (y de la disputa activa por el CPU) y reducir asi el grado de multiprogramación. Más tarde el proceso puede reintroducirse en la memoria y continuar en el punto donde abandonó. A este esquema se denomina con frecuencia como **swapping** (intercambio). El planificador a medio plazo cambia el proceso haciéndolo salir de la memoria y lo vuelve a colocar más tarde. El cambio puede ser necesario para mejorar el mix de trabajos o bien por que un cambio en los requerimientos de memoria haya generado una sobreasignación de la memoria disponible, y sea preciso liberar memoria.

Otros componentes implicado en la función de administrar al CPU es el **dispatcher** (despachador). El **dispactcher** es el módulo que entrega verdaderamente el control del CPU al proceso seleccionado por el planificador a corto plazo. Esta función implica la carga de los registros del proceso, cambiando a modo usuario y saltando a la posición adecuada en el programa de usuario para reiniciarlo.

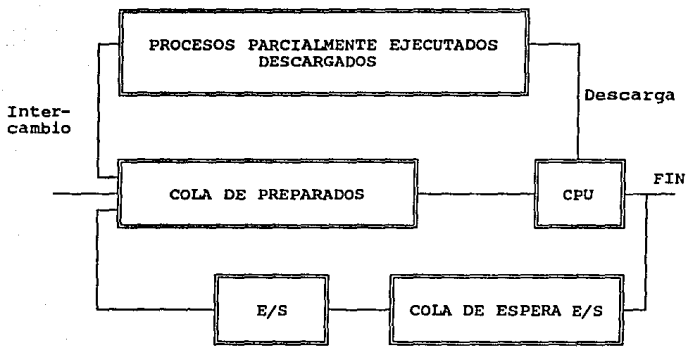


Fig. I.2.5 Adición de la planificación a medio plazo al diagrama de colas

I.2.6 Algoritmos de planificación

La administración del CPU consiste en decidir cuál de los procesos en la cola de preparados hay que asignar al CPU. Existen muchos algoritmos de CPU diferentes que tienen propiedades distintas que pueden favorecer a una clase de procesos sobre los demas. Al seleccionar el algoritmo a utilizar en una situación determinada, hay que tener en cuenta las propiedades de los mismos.

Se han sugerido muchos criterios para la comparación de algoritmos para la administración del CPU. Las caracteristicas que se utilicen en la comparación pueden introducir diferencias sustanciales en la determinación del mejor algoritmo. Entre los criterios que se emplean destacan:

- Utilización del CPU. Cuando el CPU es muy caro, deseamos mantenerlo tan ocupado como sea posible. El aprovechamiento del CPU puede ser desde el 0% al 100%. En un sistema real, esta gama deberia de ir desde el 40% (para un sistema con carga ligera) al 90% (para sistemas fuertemente cargados).
- Productividad. Si se mantiene ocupado el CPU entonces se está realizando trabajo. Una medida de este trabajo es la productividad. Para trabajos muy largos, esta productividad puede ser un trabajo por hora; para transacciones cortas, la productividad podria ser de 10 trabajos por segundo.

- Tiempo de retorno. Desde el punto de vista de un trabajo determinado, el criterio más importante es el tiempo que se tarda en ejecutar este trabajo. El intervalo de tiempo desde la entrega a la terminación es el tiempo de retorno, el cual esta dado por la suma de los periodos de tiempo utilizados en la espera para entrar en la memoria, la espera en la cola de preparados, la ejecución en CPU y la realización de la E/S.
- Tiempo de espera. El algoritmo de administración del CPU no afecta realmente a la cantidad de tiempo que un trabajo utiliza para ejecutarse o para realizar E/S, este afecta únicamente a la cantidad de tiempo que un trabajo invierte en esperar en la cola de preparados. Así, en lugar de mirar el tiempo de retorno, podríamos simplemente considerar el tiempo de espera de cada trabajo.
- Tiempo de respuesta. En un sistema interactivo, el tiempo de retorno quizá no sea el mejor criterio. Frecuentemente, un proceso puede producir alguna salida con bastante rapidez y puede continuar calculando nuevos resultados al tiempo que los primeros se envían al usuario. Así, otra medida es el tiempo desde el envío de una solicitud hasta que se produce la primera respuesta. Esta medida llamada tiempo de respuesta es la cantidad de tiempo que se tarda en iniciar la respuesta, pero no el tiempo que se lleva emitir esta respuesta. El tiempo de

retorno generalmente está limitado por la velocidad del dispositivo de salida.

Es deseable maximizar la utilización del CPU y la productividad o minimizar el tiempo de retorno, el tiempo de espera y el tiempo de respuesta. En la mayoría de los casos lo que se optimiza es la media. No obstante a veces puede ser deseable optimizar los valores mínimos y máximos, más que la media. Por ejemplo, para garantizar que todos los usuarios obtienen un buen servicio, desearemos minimizar el tiempo de respuesta máximo.

También se ha sugerido que en los sistemas interactivos, es más importante minimizar la varianza del tiempo de respuesta que minimizar el tiempo medio de la misma. Un sistema con un tiempo de respuesta razonable y predecible puede considerarse mejor que un sistema más rápido en promedio pero altamente variable. Se ha trabajado muy poco en los algoritmos de administración del CPU para minimizar la varianza.

I.2.6.1 First-come-first-served

El algoritmo de administración del CPU más sencillo es el First-Come-First-Served (FCFS). Esto es, el primer proceso en solicitar el CPU es el primero en recibir la asignación del mismo. La implementación del FCFS se realiza fácilmente mediante una cola FIFO. Cuando un proceso entra en la cola de preparados, su bloque

de control de proceso se enlaza al final de la cola. Cuando el CPU queda libre, se asigna el proceso situado al principio de la cola. Entonces el proceso en ejecución se elimina de la cola. El código para la administración de FCFS es sencillo de escribir y de comprender; sin embargo, los resultados son, con frecuencia, bastante pobres ya que el tiempo medio de retorno no es mínimo y puede variar sustancialmente, por otro lado si consideramos un FCFS en una situación dinámica se producirá un efecto "convoy" por el que todos los demás procesos esperan que un gran proceso libere el CPU traduciéndose en un aprovechamiento menor del CPU por los dispositivos que el que sería posible si los trabajos más cortos se arrancaran primero.

I.2.6.2 Shortest-job-first

Otro enfoque en la administración del CPU es el algoritmo **Shortest-Job-First** (SJF) el cual, asocia a cada trabajo la longitud de su siguiente ráfaga de CPU. Cuando el CPU queda libre, se asigna al trabajo que tenga la siguiente ráfaga de CPU más corta. Si dos trabajos tienen la misma ráfaga de CPU, se utiliza el FCFS.

El SJF es probadamente óptimo, en el sentido de que da el mínimo tiempo medio de espera para un conjunto dado de trabajos. La experiencia muestra que dando preferencia a un trabajo corto sobre uno largo, se reduce el tiempo de espera del trabajo corto más de

lo que se incrementa el tiempo de espera del largo (fig I.2.6). Por consiguiente, el tiempo de espera medio decrece.

La dificultad del SJF consiste en conocer la longitud de la siguiente petición de CPU. Para la planificación de trabajos (a largo plazo) en un sistema batch, podemos utilizar el límite de tiempo por trabajo. Motivando así a los usuarios para estimar el límite de tiempo de sus trabajos con precisión, puesto que un valor inferior puede significar un retorno más rápido. (Un valor demasiado bajo ocasionará un error por "tiempo límite excedido" y requerirá el relanzamiento). El SJF se utiliza frecuentemente en la planificación de trabajos.

Aunque el SJF es óptimo, no puede implementarse al nivel de planificación del CPU (corto plazo). No hay manera de saber la longitud de la siguiente ráfaga de CPU. Un enfoque consiste en tratar de aproximar la planificación del SJF.

Puede que no sepamos la longitud de la siguiente ráfaga de CPU, pero podemos predecir su valor como una media exponencial de las longitudes medidas de las ráfagas previas. Sea t_n la longitud de la enesima ráfaga de CPU y T_{n+1} nuestro valor calculado de la ráfaga de CPU siguiente. Entonces para (a), siendo 0 <= a <= 1, se define:

$$T_{n+1} = at_n + (1-a)_{Tn}$$

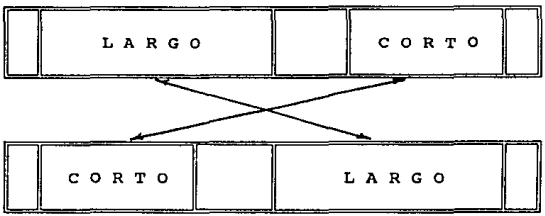


Fig. I.2.6 El SJF (Shortest Job First)

Esta fórmula define una media exponencial. El valor de tn contiene nuestra información más reciente; T_n almacena la historia pasada . El parámetro (a) controla el peso relativo de la historia reciente y de la anterior en nuestra predicción. Si a=0, entonces $T_{n+1} = T_n$, y la historia reciente no tiene efecto (las condiciones actuales se suponen que son transitorias); si a=1, entonces $T_{n+1} = T_n$, y entonces lo único que importa son las ráfagas de CPU más recientes (se supone que la historia es irrelevante). Más corrientemente, a=1/2, de modo que la historia reciente y la más antigua pesan igual. La figura I.2.7 muestra una media exponencial con a=1/2. La T_0 inicial puede definirse como una constante o como una media del sistema en conjunto.

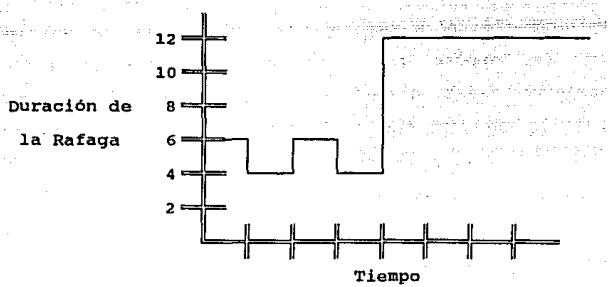
Para comprender el comportamiento de la media exponencial, podemos ampliar la fórmula para T_{n+1} sustituyendo por T_n, y llegamos a:

$$T_{n+1} = at_n + (1-a)at_{n-1} + \ldots + (1-a)^j at_{n+1} + \ldots$$

Puesto que tanto (a) como (1-a) son menores que 1, cada termino sucesivo tiene menor peso que su predecesor.

I.2.6.3 Prioridad

El SJF es un caso especial del algoritmo general de planificación por prioridad. Cada trabajo tiene asociada una prioridad, y el CPU se asigna al trabajo que tiene la



Ráfaga de CPU (ti)	6	4	6	4	13	13	13 ...
Pronóstico (T)	10	8	6	6	5	9	11	12 ...

Fig. I.2.7 Predicción de la ráfaga de CPU siguiente, mediante la media exponencial

prioridad más alta. Los trabajos con la misma prioridad se planifican por FCFS.

Un algoritmo SJF es sencillamente un algoritmo por prioridad en el que la prioridad (p) es la inversa de la siguiente ráfaga de CPU predicha (τ), $p=1/\tau$. Cuanto más larga sea la ráfaga de CPU, menor la prioridad y viceversa.

Obsérvese que discutimos la administración en términos de alta prioridad y baja prioridad. Las prioridades consisten generalmente en una gama determinada de números, tales como 0 a 7, ó 0 a 4095. Sin embargo, no existe un acuerdo general acerca de si 0 es la prioridad más alta o la más baja. Algunos sistemas utilizan los números inferiores para representar las prioridades bajas; otros los inferiores a las altas.

Las prioridades se pueden definir interna o externamente; las definidas internamente utilizan alguna cantidad o cantidades mensurables para calcular la prioridad de un proceso. Por ejemplo, en el cálculo de prioridades se han utilizado los límites de tiempo, el requerimiento de memoria, el número de archivos abiertos, y la razón entre la media de las ráfagas de CPU. Las prioridades externas se establecen mediante criterios ajenos al sistema operativo, tales como lo que se paga por el empleo del computador, el departamento que patrocina el trabajo, y otros factores externos, con frecuencia políticos.

Un problema importante de los algoritmos de planificación por prioridades es el bloqueo indefinido o inanición. Un proceso que está listo para ejecutarse pero que no dispone del CPU puede considerarse bloqueado, esperando al CPU. El algoritmo puede dejar algunos procesos de baja prioridad esperando indefinidamente. Una solución a este problema es el **aging** (envejecimiento), técnica que consiste en incrementar gradualmente la prioridad de los trabajos que esperan en el sistema durante mucho tiempo; es decir al aumentar la prioridad por periodos de tiempo dados el proceso envejece hasta que la prioridad alcanza el valor necesario para que el proceso sea ejecutado.

I.2.6.4 Algoritmos de apropiación

El FCFS, el SJF y los algoritmos por prioridad, son algoritmos de administración **non-preemtive** (no apropiativos). Una vez asignado el CPU a un proceso, este puede mantenerlo hasta que desee liberarlo, bien por haber terminado o bien por solicitar una E/S. El FCFS es intrínsecamente no apropiativo, pero los otros dos pueden modificarse para convertirse en algoritmos apropiativos.

El SJF puede ser tanto apropiativo como no apropiativo. La cuestión aparece cuando llega un nuevo trabajo a la cola de preparados mientras se está ejecutando uno anterior. El nuevo trabajo puede tener la siguiente ráfaga de CPU más corta que la del trabajo en ejecución. Un algoritmo SJF apropiativo revocará la

asignación del CPU al trabajo en ejecución, mientras que un SJF no apropiativo permitirá que el trabajo en ejecución acabe su ráfaga de CPU.

La administración por prioridades también puede ser apropiativa o no. Cuando llega un trabajo a la cola de preparados, su prioridad se compara con la prioridad del proceso en ejecución; un algoritmo apropiativo revocará la asignación de CPU si la prioridad del proceso recién llegado es mayor que la prioridad del proceso en ejecución. Un algoritmo de planificación por prioridades no apropiativo simplemente colocaría el nuevo proceso en la cabeza de la cola de preparados.

Los algoritmos de planificación del CPU no apropiativos (especialmente el FCFS) son particularmente molestos en los sistemas de tiempo compartido en donde, es muy importante que cada usuario consiga una participación del CPU a intervalos regulares. Sería desastroso permitir a un proceso que retuviera el CPU durante un periodo de tiempo prolongado.

I.2.6.5 Asignación round-robin

El algoritmo de planificación round-robin (RR) es apropiativo y esta especialmente diseñado para sistemas de tiempo compartido. Se define una pequeña unidad de tiempo común llamada quántum de tiempo o elemento de tiempo. El quántum de tiempo generalmente

tiene un valor entre 10 y 100 milisegundos. La cola de preparados se trata como una cola circular. El planificador del CPU recorre la cola, asignando el CPU a cada proceso durante un intervalo de tiempo de hasta un quántum.

Para implementar la planificación RR, la cola se mantiene como una cola de procesos FIFO. Los procesos nuevos se añaden al final de la cola. El planificador del CPU selecciona el primer trabajo de la cola, fija un temporizador para interrumpir tras un quántum de tiempo y despacha el proceso. En este tipo de planificación pueden ocurrir dos cosas. El proceso puede tener una ráfaga de CPU inferior a un quántum de tiempo, siendo el proceso mismo el que libera el CPU. Entonces pasamos al siguiente trabajo de la cola.

En el otro caso, si la ráfaga de CPU del proceso en ejecución es mayor que el quántum de tiempo, el temporizador sobrepasará su límite y ocasionará una interrupción al sistema operativo. Los registros del proceso interrumpido se salvan en su bloque de control del proceso, y el proceso se pone al final de la cola. Entonces el planificador del CPU selecciona el trabajo siguiente de la cola y le asigna el quántum de tiempo siguiente.

En este algoritmo de planificación no se asigna el CPU a ningún proceso durante más de un quántum de tiempo consecutivo, en cuanto el proceso excede el quántum asignado es devuelto a la cola de preparados.

Si hay n procesos en la cola y el quántum de tiempo es (q), entonces cada proceso obtiene (1/n) del tiempo de CPU en fragmentos de al menos (q) unidades de tiempo cada vez. Cada proceso tiene que esperar no más de (n-1) x (q) unidades de tiempo hasta su quántum de tiempo siguiente.

Los resultados de la planificación round-robin dependen fuertemente del quántum de tiempo. En un extremo si el quántum de tiempo es muy largo (infinito), la planificación round-robin es igual que el FCFS. Si el quántum de tiempo es muy corto (digamos un microsegundo), la planificación round-robin recibe el nombre de procesador compartido y aparece (en teoría) como si cada uno de los (n) procesos tuviera su propio procesador ejecutando a una velocidad equivalente a (1/n) de la velocidad del procesador real, resultando (n) procesadores lentos en lugar de uno rápido.

Al final de cada quántum de tiempo, nos encontramos con una interrupción procedente del temporizador. El procesamiento de la interrupción para conmutar el CPU a otro proceso requiere de salvar todos los registros del proceso antiguo y restaurar los del proceso nuevo. Este trabajo extra puro recibe el nombre de cambio de contexto, el cual tarda de 10 a 100 microsegundos.

Por lo tanto, el quántum de tiempo debe de ser largo respecto al tiempo de cambio de contexto. Si este tiempo fuera

aproximadamente del 10% del quántum de tiempo, entonces alrededor del 10% del tiempo de CPU se desperdiciaría en cambiar de contexto.

El tiempo de retorno también depende del quántum de tiempo (fig I.2.8). Puede mejorar si la mayoría de los trabajos acaban su ráfaga de CPU siguiente en un único quántum de tiempo. Por ejemplo, dados tres trabajos de 10 unidades de tiempo cada uno y un quántum de 1 unidad de tiempo, el retorno medio es de 29. No obstante, si el quántum de tiempo es 10, el retorno medio cae a 20. Si se añade el tiempo de cambio de contexto, las cosas empeoran cuanto menor sea el quántum de tiempo, dado que se requerirán más cambios de contexto.

Por otra parte, si el quántum de tiempo es demasiado largo, la asignación round-robin degenera en un FCFS. Hay una regla empírica que dice que el 80% de las ráfagas de CPU deben ser menores que el quántum de tiempo.

I.2.6.6 Colas multinivel

Se ha creado otro algoritmo de planificación para situaciones en las que los trabajos son fácilmente clasificables en diferentes grupos. Por ejemplo, una división frecuente es la que se realiza entre los trabajos **foreground** (interactivos) y los trabajos **background** (batch). Estos dos tipos de trabajos tienen unos requerimientos muy diferentes en lo que respecta al tiempo de respuesta, y por lo tanto pueden tener distintos algoritmos de

planificación. Además los trabajos foreground pueden tener prioridad sobre los background.

Este tipo de algoritmo divide la cola de preparados en colas separadas. Los trabajos permanecen asignados a una cola, en base generalmente a una propiedad de los mismos, como el tipo de trabajo o la necesidad de memoria. Cada cola tiene su propio algoritmo de planificación.

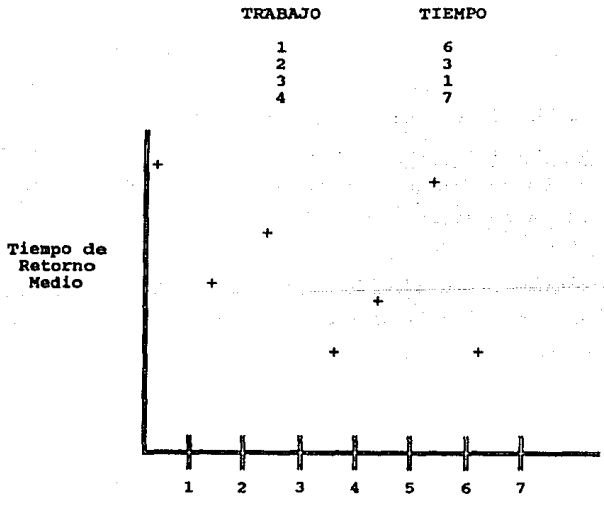


Fig. I.2.8 El tiempo de retorno medio varía con el Quántum de tiempo

Además debe de existir una planificación entre colas. Normalmente consiste en una planificación apropiativa con prioridad fija.

Otra posibilidad es el fraccionamiento de tiempo entre colas. Cada cola obtiene una parte del tiempo de CPU, que a su vez puede planificarse entre los distintos procesos de cada cola.

I.2.6.7 Colas multinivel con retroalimentación

Normalmente, en un algoritmo de planificación de colas múltiples, los trabajos permanecen asignados a una cola desde que entran al sistema. Los trabajos no cambian de colas. Si existen colas preparadas para los trabajos foreground y background por ejemplo, los trabajos no pasarían desde una cola a otra puesto que no cambian su naturaleza.

Sin embargo, este tipo de algoritmo con retroalimentación permite que un trabajo pase de una cola a otra. La idea consiste en separar los trabajos que tengan diferentes características en lo relativo a ráfagas de CPU. Si un trabajo utiliza mucho tiempo de CPU, pasará a una cola de menor prioridad. Este esquema deja a los trabajos interactivos e intensivos en E/S en las colas de mayor prioridad. Análogamente, un trabajo que permanezca demasiado tiempo en una cola de baja prioridad, puede trasladarsele a una cola de prioridad mayor.

En general, un planificador de colas múltiples con realimentación se define por los siguientes parámetros:

- El número de colas
- El algoritmo de planificación de cada cola.
- Un método para determinar cuándo se pasa un trabajo a una cola de mayor prioridad.
- Un método para determinar cuándo se pasa un trabajo a una cola de menor prioridad.
- Un método para determinar en qué cola entrará un trabajo cuando éste precise un servicio.

La definición de un planificador de colas múltiples con retroalimentación hace de él el algoritmo de planificación de CPU más general. Puede configurarse de modo que cumpla los requerimientos de un sistema concreto en fase de diseño. Aunque es el esquema más general, resulta también el más complejo.

I.3 Administración de memoria

La memoria tienen un carácter central en el funcionamiento de un sistema de computación. Como se muestra en la figura I.3.1, tanto el CPU como el sistema de E/S interactúan con la memoria. La memoria es una amplia tabla de palabras o bytes, cada una de las cuales tiene su propia dirección. La interactividad se obtiene mediante una secuencia de lecturas y escrituras en direcciones específicas de memoria. El CPU extrae de y almacena en la memoria

los datos que ha de modificar, y el sistema de E/S, también hace uso de la memoria, cuando el CPU no puede procesar la información de inmediato o bien cuando no está disponible algún dispositivo de salida.

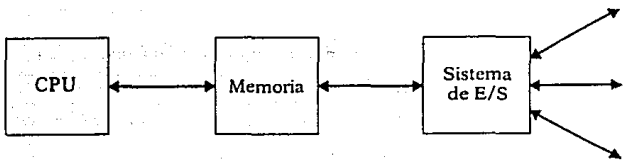


Fig. I.3.1 Carácter central de la memoria

I.3.1 Almacenamiento real

La organización y administración de la memoria principal, o memoria primaria o memoria real de un sistema de computación ha sido uno de los factores más importantes en el diseño de los sistemas operativos. Los términos memoria y almacenamiento han sido usados indistintamente, pero de los dos términos, almacenamiento parece ser el más utilizado actualmente.

Los programas y datos deben estar en el almacenamiento principal para poderlos ejecutar o hacer referencia directa a

ellos. El almacenamiento secundario, más conocido como disco, tambor o cinta, proporciona una capacidad muy grande y de bajo costo, para los numerosos programas y datos que deben estar disponibles para su procesamiento.

En este capítulo se verán algunos esquemas comunes de organización y administración de almacenamiento de un computador.

I.3.1.1 Organización de almacenamiento

Históricamente, el almacenamiento principal se ha considerado como un recurso costoso. Los primeros sistemas estaban interesados en la organización y administración del almacenamiento principal para su uso óptimo.

Por organización de almacenamiento se entiende la manera de considerar a éste. Por ejemplo:

- Si se coloca un solo usuario en la memoria principal o se colocan varios usuarios al mismo tiempo
- Cuánto espacio se concede a cada programa cuando se encuentran varios en el almacenamiento principal, si se da el mismo espacio o si se divide, en varias porciones, llamadas particiones, de diferentes tamaños
- Si estas particiones se hacen de una forma rígida con las particiones definidas para un largo período de tiempo, o

- Si se proporciona una partición más dinámica, que permita a los sistemas de computación adaptarse rápidamente a los cambios impuestos por las necesidades de los trabajos de los usuarios,
- Si se necesita que estos trabajos sean diseñados para funcionar en una partición especifica, o
- Si se permite que los trabajos se ejecuten en cualquier parte donde quepan o
- Si se requiere que cada trabajo sea colocado en un bloque contiguo de localización de almacenamiento o
- Si se permite dividir los trabajos entre varios bloques separados y colocarlos en cualesquiera ranuras disponibles del almacenamiento principal.

I.3.1.2 Administración del almacenamiento

Con independencia del esquema de organización del almacenamiento que se adopte para un sistema determinado, hay que decidir las estrategias que se van a usar para obtener un rendimiento óptimo.

Las estrategias de administración, del almacenamiento determinan el rendimiento de una organización del almacenamiento con varias estrategias:

- Cuándo se considera un nuevo programa para colocarlo en memoria principal,

- Cuándo el sistema los pide de forma específica o se intenta anticipar a las peticiones del sistema
- Dónde se colocará, dentro del mismo almacenamiento principal, el programa a ejecutarse a continuación
- Si se colocan los programas lo más apretados posible dentro de los espacios disponibles de memoria, o si se colocan los programas lo más rápido posible para ahorrar tiempo de ejecución
- Qué programa se desplazará, en la memoria principal cuando un nuevo programa necesita ser colocado en ésta, el más antiguo o el que se utiliza con menor frecuencia, o bien aquel que se ha utilizado hace más tiempo.

I.3.1.3 Jerarquías de almacenamiento

En las décadas de 1950 y 1960, el almacenamiento principal, por lo general de núcleo magnético era muy costoso.

Los programas y los datos tienen que estar en la memoria principal para poder ser ejecutados o para hacer referencia a ellos. Los programas o datos que no sean necesarios de inmediato pueden mantenerse en el almacenamiento secundario hasta el momento de traerlos al almacenamiento principal para su ejecución o referencia. Los medios de almacenamiento secundario como discos o cintas suelen ser más baratos que el almacenamiento principal,

tienen una capacidad mucho mayor. El almacenamiento principal es de acceso más rápido que el almacenamiento secundario.

Durante los años sesentas quedó claro que la jerarquía de almacenamiento aumentaría en un nivel más, con mejoras radicales de ejecución y utilización. Este nivel adicional, el **cache**, es un almacenamiento de alta velocidad, mucho más rápido que el principal. El almacenamiento cache es extremádamente costoso comparado con el principal, por lo que sólo se utilizan caches muy pequeños.

La figura I.3.2 muestra la relación entre los almacenamientos cache, primario y secundario. El almacenamiento cache impone al sistema en un nivel más de traspaso. Los programas del almacenamiento principal son traspasados al cache antes de su ejecución, donde pueden ser ejecutados mucho más rápido.

I.3.1.4 Estrategias de administración de almacenamiento

Los costosos recursos son administrados de forma intensiva para conseguir un mejor uso. Las estrategias de almacenamiento están dirigidas a la obtención del mejor uso posible del recurso del almacenamiento principal. Las estrategias de administración de este recurso se dividen en las siguientes categorías:

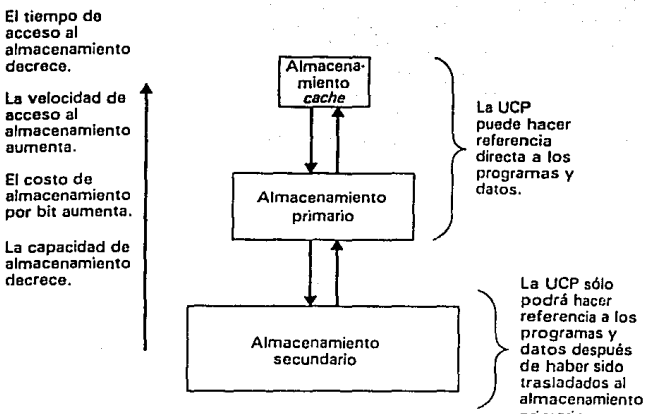


Fig. I.3.2 Organización jerárquica del almacenamiento

- **Estrategias de búsqueda.** Están relacionadas con el hecho de cuándo obtener el siguiente fragmento del programa, o de datos para su inserción en el almacenamiento principal. Durante muchos años, la estrategia convencional era la búsqueda por demanda, donde el siguiente fragmento de programa, o de datos, era traído al almacenamiento principal cuando algún programa en ejecución hacia referencia a él. Actualmente, muchos investigadores creen que la búsqueda anticipada provocará un mejor rendimiento del sistema.

- **Estrategias de colocación.** Están relacionadas con la determinación del lugar dentro del almacenamiento principal, donde será colocado un programa nuevo. Estas pueden ser de primer ajuste, mejor ajuste y peor ajuste.

- **Estrategias de reposición.** Están relacionadas con la determinación de qué fragmento de programa, o de datos, desplazar, para dejar sitio a los programas nuevos.

I.3.1.5 Asignación contigua de almacenamiento contra no contigua

Los primeros sistemas computacionales requerían una asignación contigua de almacenamiento. Cada programa tenía que ocupar un bloque sencillo y contiguo de localizaciones de almacenamiento. Hasta que se intentó la multiprogramación de partición variable no se hizo evidente que la asignación no contigua de almacenamiento podía ser útil. En la asignación no contigua de almacenamiento, un programa es dividido en varios bloques o segmentos que puedan ser colocados a lo largo del almacenamiento principal en partes que no tienen que ser necesariamente adyacentes. Es más difícil para un sistema operativo controlar la asignación no contigua de almacenamiento, la ventaja está en que el almacenamiento principal tiene muchos agujeros pequeños disponibles, en vez de un solo agujero grande, entonces el sistema operativo puede en muchos casos cargar y ejecutar un programa que de otro modo tendría que esperar.

I.3.1.6 Asignación contigua de almacenamiento de un solo usuario

Los primeros sistemas de computación sólo permitían usar la máquina a una persona a la vez. Todos sus recursos estaban a disposición del usuario, independientemente de si el trabajo los usaba o no.

En un principio, cada usuario escribía todo el código necesario para implementar un aplicación determinada, incluyendo las instrucciones de nivel de máquina de entrada/salida muy detalladas. Muy rápido éste código necesario para la implementación de funciones básicas, quedó consolidado en un sistema de control de entrada/salida (IOCS). Los usuarios que deseaban realizar una operación de entrada/salida ya no tenían que codificar las instrucciones de forma directa, sino que llamaban a las rutinas del IOCS para que hiciera el trabajo en sí. Esto simplificó y aceleró en gran medida el proceso de codificación. La implementación de los sistemas de control de entrada/salida, puede haber sido el comienzo del concepto actual de los sistemas operativos. La fig. I.3.3 ilustra la organización típica del almacenamiento de un sistema de asignación contigua de un solo usuario. El tamaño de los programa está limitado por la cantidad de almacenamiento principal, pero es posible ejecutar programas más grandes que este último por medio de recubrimiento. Si una sección particular del programa no es necesaria durante todo el transcurso de su ejecución, entonces puede traerse otra sección desde el almacenamiento secundario, para

ocupar el almacenamiento utilizado por la sección que ya no se necesita.

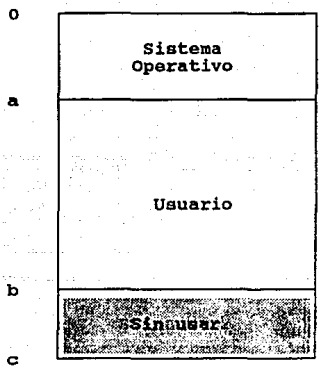


Fig. I.3.3 Asignación contigua de almacenamiento de un solo usuario

I.3.1.7 Multiprogramación en partición fija

Aún con los sistemas operativos de procesamiento por lotes, los sistemas de un solo usuario siguen desperdiciando una considerable cantidad de recursos computacionales.

Con esto se vio que se podían incrementar de forma considerable la utilización de el CPU (Unidad Central de Procesamiento) por medio de la administración intensiva. Esta vez, implementaron los sistemas multiprogramación, en los cuales varios usuarios compiten al mismo tiempo por los recursos del sistema. El trabajo que está esperando una operación de entrada/salida cederá a el CPU otro trabajo que se encuentre listo para realizar cálculos si, de hecho, hay otro trabajo esperando. Con esto, la operación de entrada/salida y la utilización de el CPU tienen lugar al mismo tiempo. De esta forma se incrementa en gran manera la utilización de el CPU y la capacidad de ejecución del sistema.

Para aprovechar al máximo las ventajas de la multiprogramación es necesario que varios trabajos residan a la vez en el almacenamiento principal del computador. Entonces, cuando un trabajo pida una operación de entrada/salida, el CPU puede ser cambiada de inmediato a otro trabajo y realizar los cálculos sin dilación cuando éste nuevo trabajo abandone la CPU, otro puede estar listo para usarla.

I.3.1.7.1 Multiprogramación de partición fija con traducción y carga absolutas

En los primeros sistemas de multiprogramación, el almacenamiento principal estaba dividido en una serie de particiones de tamaño fijo. Cada partición podía atender un solo

trabajo. La CPU se cambiaba rápidamente entre usuarios para crear la ilusión de simultaneidad.

Los trabajos se traducían con ensambladores y compiladores absolutos para ser ejecutados sólo dentro de una partición específica. (fig. I.3.4)

Si un trabajo estaba listo, para su ejecución y se ocupaba su partición, entonces ese trabajo debía esperar, aún cuando estuvieran disponibles otras particiones.

Esto daba como resultado un desperdicio del recurso de almacenamiento, pero el sistema operativo era de implementación relativamente sencilla.

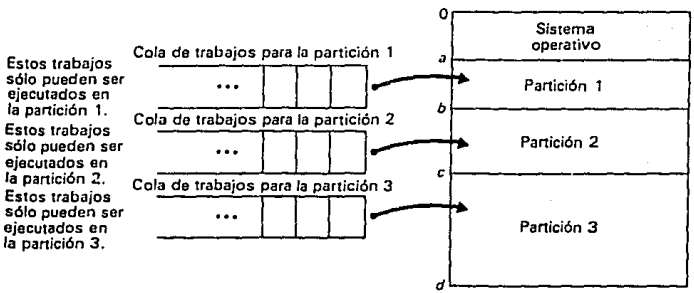


Fig. I.3.4 Multiprogramación de partición fija con traducción y carga absolutas

I.3.1.7.2 Multiprogramación de partición fija con traducción y carga relocalizable

Los compiladores, ensambladores y cargadores de relocalización se usan para producir programas relocalizables que pueden ser ejecutados en cualquier partición disponible de tamaño suficiente para aceptarlos (fig. I.3.5). Este esquema elimina cierta cantidad del desperdicio de almacenamiento inherente a la multiprogramación con carga y traducción absolutas. Los traductores y cargadores de relocalización son más complejos que sus equivalentes absolutos.

En los sistemas de multiprogramación de asignación contigua, la protección suele implementarse con varios 'registros de límite'. Los extremos superior e inferior de una partición del usuario pueden ser delineados con dos registros, o pueden indicarse el límite superior (o el inferior) y la longitud de la región.

La fragmentación de almacenamiento ocurre en todos los sistemas de computación con independencia de su organización de almacenamiento. En los sistemas de multiprogramación de partición fija, la fragmentación se produce cuando los trabajos del usuario no llenan completamente sus particiones designadas, o cuando una partición permanece sin usar porque es demasiado pequeña para alojar un trabajo que se encuentra en espera.

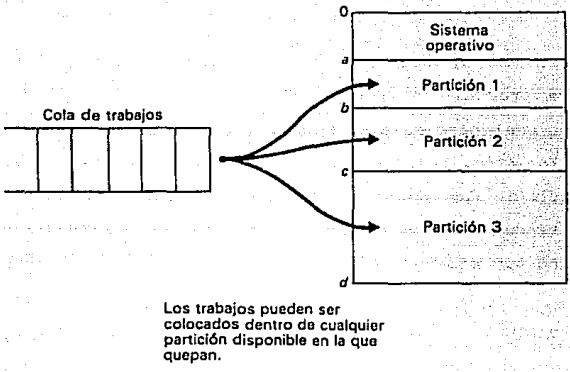


Fig. I.3.5 Multiprogramación de partición fija con traducción y carga relocalizables

I.3.1.8 Multiprogramación de partición variable

Al observarse los problemas de la multiprogramación de partición fija, decidieron que una mejora notable sería permitir que los trabajos ocuparan tanto espacio (menor que la totalidad del área de almacenamiento real) como necesitaran. Este esquema se denomina multiprogramación de partición variable. En esta multiprogramación no se hacen suposiciones sobre el tamaño de los trabajos (aunque no deben ser mayores que el espacio disponible en

el almacenamiento principal del computador). A medida que llegan los trabajos, si los mecanismos de planificación deciden que deben proceder, se les proporcionará todo el almacenamiento necesario. Aquí no hay desperdicio aparente, la partición de un trabajo es exactamente su tamaño, pero cuando terminan los trabajos dejan agujeros en el almacenamiento principal.

Estos pueden ser usados por otros trabajos, pero aún con esto, los agujeros restantes van haciéndose más pequeños, por tanto en la multiprogramación de partición variable hay desperdicio.

El proceso de fusionar agujeros adyacentes para formar uno sencillo más grande se llama **combinación**. Mediante este proceso se reclaman los bloques de almacenamiento contiguos más grandes posibles.

Aunque los agujeros separados distribuidos por todo el almacenamiento principal, constituyen una cantidad significativa de almacenamiento.

La técnica de **compresión** de almacenamiento implica pasar todas las áreas ocupadas del almacenamiento, a uno de los extremos del almacenamiento principal. La compresión también tiene sus desventajas:

- Consume recursos del sistema que podrían utilizarse de forma más productiva.

- El sistema debe detener todo mientras está realizando la compresión. Esto puede dar como resultado unos tiempos de respuesta erráticos para los usuarios interactivos y puede ser devastador en los sistemas de tiempo real.

I.3.1.9 Estrategias de colocación de almacenamiento

Estas se utilizan para determinar, en el almacenamiento principal, el lugar donde se colocarán los programas y datos que van llegando.

- **Best-fit** (Mejor ajuste). Un trabajo nuevo al llegar al almacenamiento principal es colocado en el agujero en el cual quepa de forma más ajustada y deje el menor espacio sin usar (fig. I.3.6.a).

-**First-fit** (Primer ajuste). Al llegar un trabajo nuevo al almacenamiento principal es colocado en el primer agujero disponible con tamaño suficiente para atenderlo (fig.I.3.6.b)

- **Worst-fit** (Peor ajuste). En principio, ésta parece ser una decisión caprichosa, pero después de un examen minucioso, el peor ajuste tiene atractivo, consiste en colocar un programa en el almacenamiento principal, en el agujero que quepa de la peor manera, es decir, en el agujero más grande posible (fig.I.3.6.c)

a) ESTRATEGIA DE PRIMER AJUSTE

Coloca el trabajo en el primer, agujero de la lista de almacenamiento libre en el que quepa.

Lista de almacenamiento libre manteniendo en orden las direcciones de almacenamiento o a veces, en orden aleatorio.

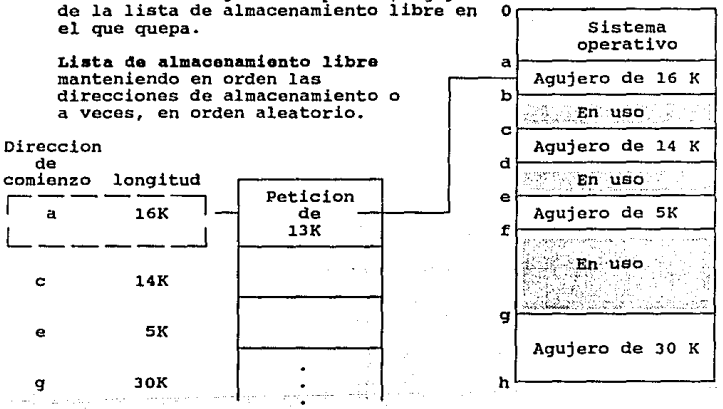


Fig. I.3.6.a Estrategia de colocación de almacenamiento

b) ESTRATEGIA DE MEJOR AJUSTE

Coloca el trabajo en el mas pequeño posible en el que quepa.

Lista de almacenamiento libre mantenida en orden ascendente por tamaño de agujero.

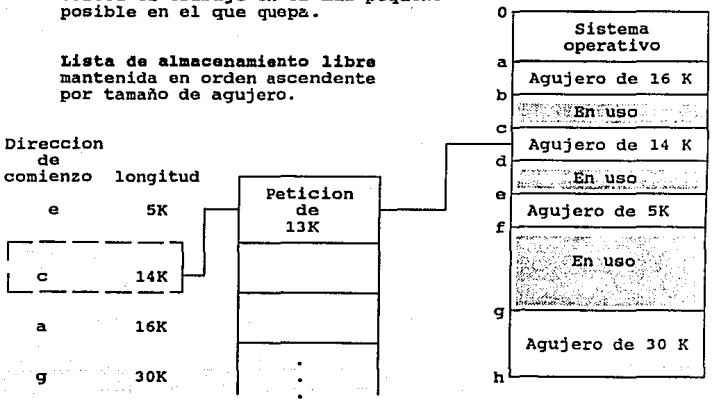


Fig.I.3.6.b Estrategias de colocación de almacenamiento

c) ESTRATEGIA DE PEOR AJUSTE

Coloca el trabajo en el más grande posible en el que quepa.

Lista de almacenamiento libre mantenida en orden descendente por tamaño de agujero.

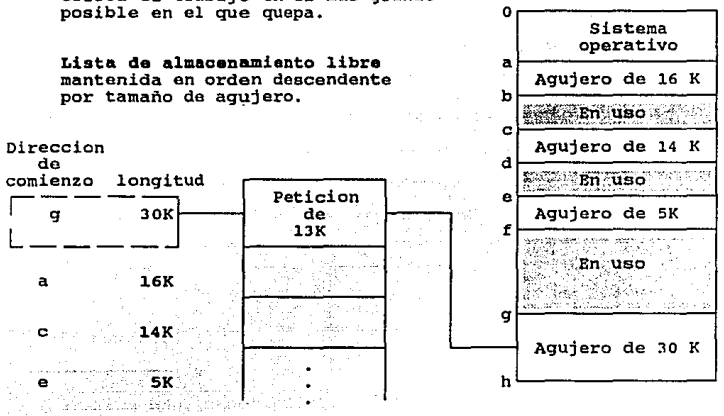
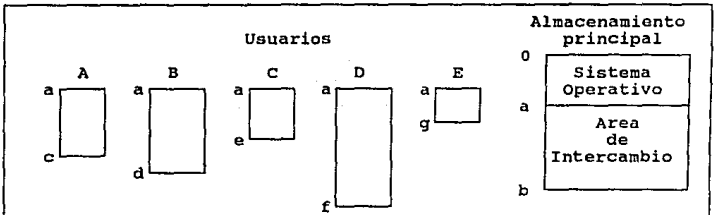


Fig. I.3.6.c Estrategias de colocación de almacenamiento

I.3.1.10 Multiprogramación con intercambio de almacenamiento

En cada uno de los esquemas de multiprogramación, los programas de usuario permanecen en la memoria principal hasta su terminación. En algunos sistemas de intercambio, un trabajo ocupa el almacenamiento principal de inmediato. Ese trabajo se ejecuta hasta que ya no pueda continuar y entonces cede el almacenamiento y el CPU al siguiente trabajo. Con esto, la totalidad del almacenamiento se dedica a un trabajo durante un breve periodo, ese

trabajo es entonces retirado (intercambiado) y entra el nuevo. Un trabajo puede intercambiarse varias veces, antes de llegar a su terminación (fig. I.3.7). Muchos de los primeros sistemas de tiempo compartido estaban implementados con ésta técnica de intercambio, que permitía garantizar tiempos de respuesta razonables para un número relativamente reducido de usuarios.



1. Solo hay usuario a la vez en el almacenamiento principal
2. Este usuario procede hasta que:
   a) se emite una peticion de E/S,
   b) el tiempo se agota,
   c) termina de forma voluntaria
3. El area de intercambio "imagen del nucleo" para ese usuario se copia entonces en el almacenamiento secundario
4. La imagen del nucleo para el siguiente usuario es ideal en el area de intercambio y el usuario procede hasta que es intercambiado y entra el nuevo usuario, etc.
5. Este esquema era comun en los primeros sistemas de tiempo compartido.

Fig. I.3.7 Multiprogramación en un sistema de intercambio

A principios de la década de los sesentas, los sistemas de intercambio dieron paso a los actuales sistemas de paginación de uso común. Actualmente se han desarrollado sistemas de intercambio

más sofisticados que permiten tener al mismo tiempo un número mayor de imágenes de usuario en el almacenamiento principal.

En estos sistemas, una imagen de usuario es retirada sólo cuando se necesita su almacenamiento para una nueva imagen.

I.3.2 Almacenamiento virtual

El almacenamiento virtual suele asociarse con la capacidad de direccionar un espacio de almacenamiento mucho mayor que el disponible en el almacenamiento primario de determinado sistema de computación.

Los métodos más comunes de implementación del almacenamiento virtual son la paginación y la segmentación. Algunos sistemas de almacenamiento utilizan una u otra técnica, y otros usan ambas.

La fig. I.3.8 muestra la evolución de las organizaciones de almacenamiento desde los sistemas dedicados a un solo usuario hasta los sistemas de almacenamiento virtual, combinando las técnicas de paginación y segmentación. A continuación se examinarán diversas organizaciones de almacenamiento virtual y más adelante las técnicas empleadas para su administración.

La clave del concepto de almacenamiento virtual está en la disociación de las direcciones a las que hace referencia un

programa de las direcciones disponibles en el almacenamiento primario.

Las direcciones referidas por un proceso de ejecución se denominan direcciones virtuales. Las direcciones disponibles dentro del almacenamiento primario se denominan direcciones reales. El número de las direcciones virtuales a que puede hacer referencia un proceso, se llama espacio de direcciones virtuales, v, de este proceso. Aún cuando los procesos sólo hacen referencia a direcciones virtuales, estas deben ejecutarse en el almacenamiento real, por eso, las direcciones virtuales deben ser transformadas a direcciones reales, mientras el proceso se está ejecutando.

Los mecanismos de traducción dinámica de direcciones (DAT), convierten las direcciones virtuales en reales al ejecutarse el proceso, pero las direcciones virtuales, no necesariamente deben ser contiguas dentro del almacenamiento real, esto se llama contigüidad artificial.

La organización del almacenamiento de niveles múltiples se refiere a utilizar un esquema de almacenamiento de dos niveles. El primero es el almacenamiento real, y el segundo se conoce como almacenamiento auxiliar, secundario o adicional.

Como el almacenamiento real es compartido por varios procesos, y como cada uno de ellos puede tener un espacio de direcciones

virtuales mucho más grande que el almacenamiento real, entonces sólo se mantiene al mismo tiempo una pequeña parte de los programas y datos de cada proceso en el almacenamiento real. La fig. I.3.9 muestra un sistema de almacenamiento de dos niveles en el cuál se han colocado los elementos de los almacenamientos virtuales de varios usuarios en el almacenamiento real. Los mecanismos de traducción dinámica de direcciones deben mantener mapas que ilustren qué direcciones del almacenamiento virtual se encuentran en el almacenamiento real y dónde se encuentran. La información se agrupa en bloques y el sistema está informado del lugar del almacenamiento real donde han sido colocados los bloques de almacenamiento virtual.

<table>
<tr><td></td><td colspan="2">Real</td><td colspan="3">Virtual</td></tr>
<tr><td rowspan="2">Sistemas dedicados a un solo usuario</td><td colspan="2" rowspan="2">Sistemas de Multiprogramación en almacenamiento real</td><td colspan="3">Multiprogramación en almacenamiento virtual</td></tr>
<tr><td>Paginación pura</td><td>Segmentación pura</td><td>Combinación paginación/ segmentación</td></tr>
<tr><td></td><td>Multiprogración en partición fija</td><td>Multiprogramación en partición variable</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td></td><td>Absoluta</td><td>Relocalizable</td></tr>
</table>

Fig. I.3.8 Evolución de las organizaciones de almacenamiento

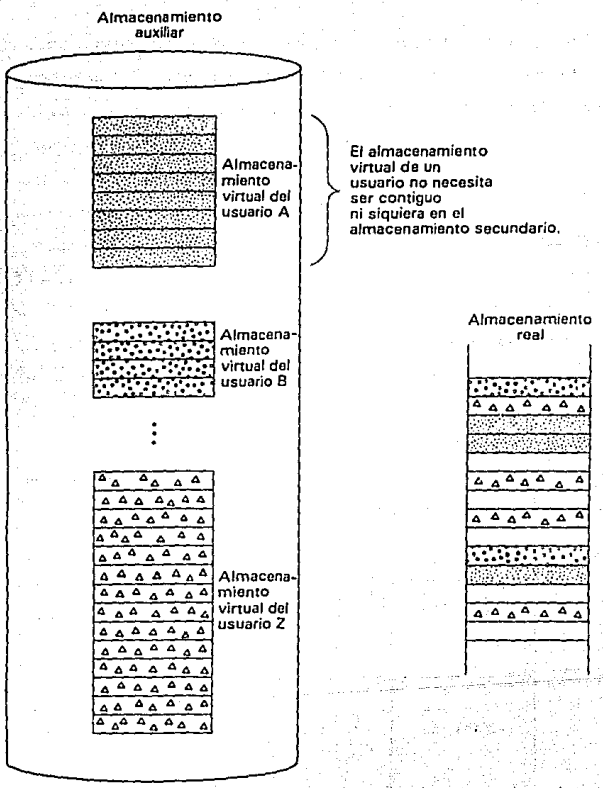


Fig. I.3.9. Almacenamiento de dos niveles

Cuando los bloques son del mismo tamaño, reciben el nombre de páginas y la organización del almacenamiento virtual asociada, se denomina paginación.

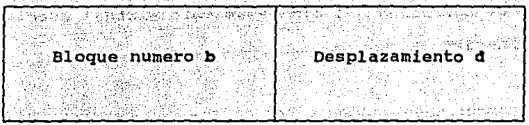
Cuando los bloques pueden tener diferentes tamaños, se llaman segmentos, y la organización del almacenamiento virtual asociada se conoce como segmentación.

Algunos sistemas combinan las dos técnicas, implementando los segmentos como entidades de tamaño variable, compuestas de páginas de tamaño fijo, esto se conoce como paginación/segmentación.

Las direcciones en un sistema de transformación de bloques son bidimensionales. Para referirse a un elemento determinado, un programa especifica el bloque en el cual reside el elemento y el desplazamiento de éste a partir del inicio del bloque.

Una dirección virtual, v, se indica con un par ordenado (b, d), siendo b el número del bloque donde reside, y d el desplazamiento a partir del inicio del bloque. (fig. I.3.10)

Es importante observar que la transformación de bloques se realiza de forma dinámica mientras se ejecuta un proceso.



Dirección virtual

v = (b,d)

Fig. I.3.10 Formato de la dirección virtual dentro de un sistema de transformación de bloques

I.3.2.1 Paginación

Una dirección virtual en un sistema de paginación es un par ordenado (p, d) donde p es el número de la página en el almacenamiento virtual en que reside el elemento al que se está haciendo referencia y d es el desplazamiento en la página p. Las páginas se transfieren del almacenamiento secundario al primario en bloques llamados estructuras de páginas, que tienen el mismo tamaño que las páginas.

Las técnicas para la realización de la planificación de páginas son las siguientes:

- **Transformación directa.** Aquí la tabla de mapa de páginas contiene una entrada para cada una de las páginas de almacenamiento virtual de éste proceso, si éste tiene n páginas dentro del almacenamiento

virtual, entonces la transformación directa de la tabla de mapa de páginas del proceso contiene entradas sucesivas para la pág. 0, pág. 1, hasta la página n-1.

- **Transformación asociativo.** Una forma de acelerar la traducción dinámica de páginas consiste en colocar la tabla completa del mapa de páginas en un almacenamiento asociativo que tiene un tiempo de ciclo de un orden de magnitud más rápido que el almacenamiento primario. Cada una de las células de almacenamiento asociativo se registra de manera simultánea para buscar una correspondencia con p.

- **Transformación asociativa/directa.** Los sistemas actuales que utilizan un mapa asociativo parcial han obtenido rendimientos del noventa por ciento o mayores del que se puede conseguir con un mapa asociativo de página completo, de hecho éste no necesita ser tan grande para conseguir un buen rendimiento. La utilización de un mecanismo de transformación combinada asociativa/directa es una decisión de ingeniería basada en la economía de las tecnologías de hardware existentes.

I.3.2.1.1 Compartimiento de recursos en un sistema de paginación

En los sistemas de compartimiento multiprogramados, especialmente en los de tiempo compartido, es común que muchos usuarios se encuentren ejecutando los mismos programas. En los sistemas actuales los programas se encuentran divididos en áreas separadas de procesos y de datos. Los procedimientos no

modificables se llaman procedimientos puros o reentrantes, éstos y los datos no modificables pueden ser compartidos, mientras los datos como los procedimientos modificables no se pueden compartir. (fig. I.3.11).

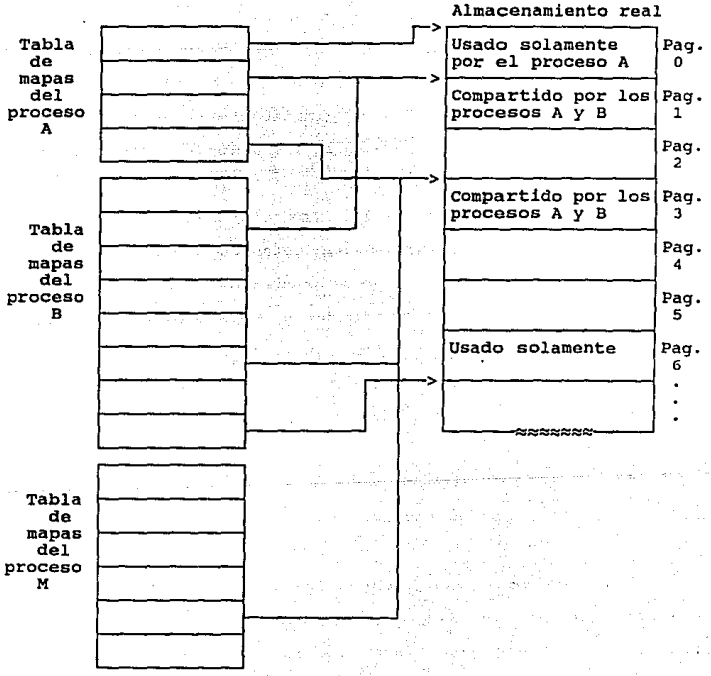


Fig. I.3.11 Compartimiento en un sistema de paginación

Esto indica la necesidad de identificar cada página como compartible ó no. Al hacer que las entradas de la tabla de páginas de varios procesos apunten al mismo marco, ese marco de página es compartido por esos procesos. El compartimiento reduce la cantidad de almacenamiento primario necesario para la ejecución eficaz de un grupo de procesos y puede hacer posible que un sistema determinado mantenga una cantidad mayor de usuarios.

I.3.2.2 Segmentación

La multiprogramación de partición variable, aún se encuentra restringida para ejecutar programas en un solo bloque de posiciones contiguas de almacenamiento real. En los sistemas de segmentación, ésta restricción se elimina y un programa puede ocupar varios bloques separados de almacenamiento real. Los bloques mismos no necesitan ser del mismo tamaño y deben, además, estar compuestos de posiciones contiguas de almacenamiento, pero los bloques separados no necesitan ser adyacentes.

Esto introduce una serie de nuevos e interesantes problemas. Por ejemplo, para proteger a cada usuario ya no es suficiente un par de registros de límites. Un esquema para la implementación de la protección del almacenamiento en sistemas de segmentación es el uso de claves de protección del almacenamiento, como muestra la figura I.3.12.

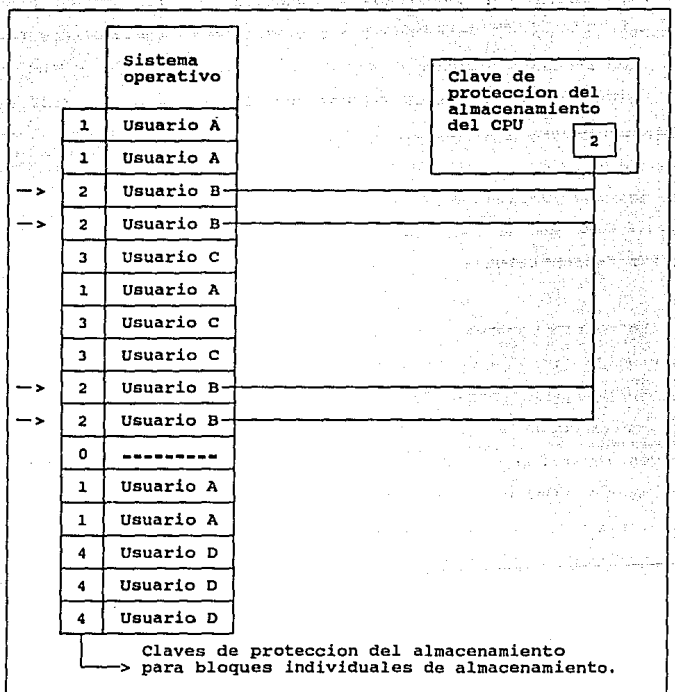


Fig. I.3.12 Uso de claves de protección del almacenamiento

Una dirección virtual de un sistema de segmentación es un par ordenado v = (s, d), donde s es el número del segmento del almacenamiento virtual en el cual residen los elementos referidos, y d es el desplazamiento en el segmento s en el cual se encuentra el elemento referido.

Los segmentos se transfieren del almacenamiento secundario al primario como unidades completas. Las estrategias de colocación para la segmentación son idénticas a las utilizadas en la multiprogramación de partición variable, siendo las más comunes el primer ajuste y el mejor ajuste.

No es recomendable otorgar a cada proceso un acceso ilimitado a todos los segmentos del sistema. El control de acceso se consigue al darle a cada proceso ciertos derechos de acceso a todos los segmentos y al negarle por completo el acceso a muchos otros. Los tipos más comunes de control de acceso en los sistemas actuales son:

- Acceso de lectura
- Acceso de escritura
- Acceso de ejecución
- Acceso de adición

Con éstos se pueden crear 16 diferentes modos de control de acceso.

Al igual que en los sistemas de paginación , existen varias estrategias para la implementación de la traducción de direcciones de segmentación. Esto puede hacerse mediante

= Transformación directa
= Transformación asociativa o por la combinación de
= Transformación directa/asociativa

Puede hacerse con almacenamiento caché de tamaño suficiente, para alojar a la tabla completa de mapa de segmentos, o puede hacerse con almacenamientos caché parciales grandes suficientes para contener sólo las entradas de los segmentos de referencia más reciente.

I.3.3.2.1 Compartimiento en un sistema de segmentación

Una de las ventajas de la segmentación sobre la paginación, es que se trata más de un concepto lógico que físico, además compartir segmentos es más sencillo, comparado con el compartimiento en un sistema de paginación. En su forma más general, los segmentos no están restringidos a un tamaño fijo, sino que se les permite (dentro de límites razonables) tener el tamaño que necesiten. Un segmento correspondiente a un arreglo es tan grande como el arreglo mismo. En un sistema de segmentación, una vez que el segmento ha sido declarado como compartido, entonces las estructuras de datos pueden aumentar o disminuir a voluntad sin cambiar el hecho lógico de que residen en segmentos compartidos. (fig. I.3.13)

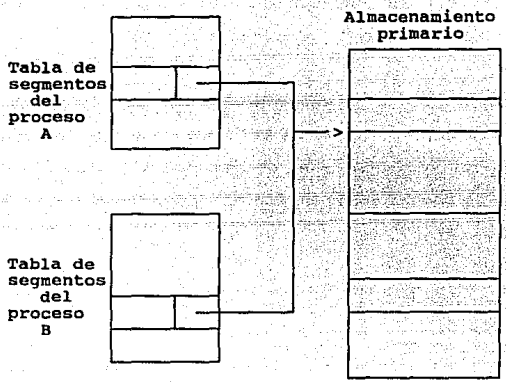


Fig. I.3.13 Compartimiento en un sistema de segmentación

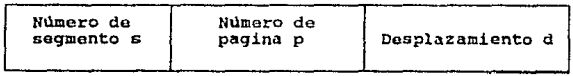
I.3.2.3 Paginación/Segmentación

Ambas paginación y segmentación, ofrecen ventajas significativas como organizaciones de almacenamiento virtual.

Actualmente muchos sistemas de computación combinan la paginación y la segmentación, ofreciendo ventajas de las dos técnicas de organización del almacenamiento virtual. En general, el

tamaño de los segmentos es múltiplo del de las páginas, no es necesario que todas las páginas de un segmento se encuentren al mismo tiempo en el almacenamiento primario, y las páginas de almacenamiento virtual, que son contiguas en este almacenamiento, no necesitan ser contiguas en el almacenamiento real.

El direccionamiento es tridimensional con una dirección de almacenamiento virtual, v, siendo un trío ordenado v = (s, p, d), donde s el número del segmento, p es el número de la página y d el desplazamiento en la página donde se encuentra asignado el elemento deseado. (fig. I.3.14)



Dirección virtual
v = (s,p,d)

Fig. I.3.14 Formato de la dirección virtual en un sistema de paginación/segmentación

I.3.2.3.1 Copartimiento en un sistema de paginación/segmentación

Para un sistema de paginación/segmentación, las ventajas de compartir segmentos son algo importante. El compartimiento se implementa disponiendo entradas en tablas de mapa de segmentos para

diferentes procesos que apunten a la misma tabla de mapa de páginas. (fig. I.3.15)

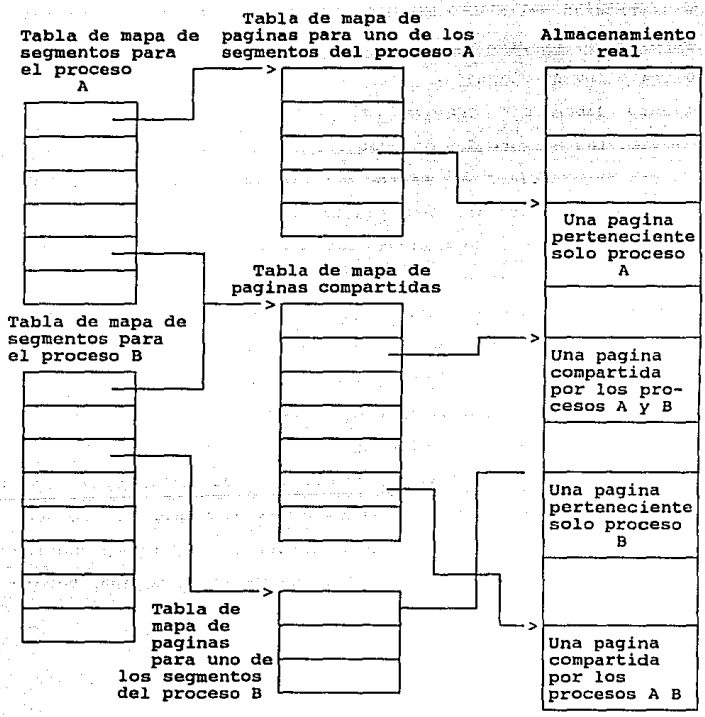


Fig. I.3.15 Dos procesos compartiendo un sistema de paginación/segmentación.

I.3.2.4 Estrategias de administración de almacenamiento virtual

Como en el almacenamiento real, las estrategias de administración del almacenamiento que son de búsqueda, colocación y reposición, se presentan en el almacenamiento virtual.

- **Estrategias de búsqueda.** Tratan de los casos en que una página o segmento deben ser traídos del almacenamiento secundario al primario. Las estrategias de búsqueda por demanda esperan a que se haga una referencia a una página o segmento por un proceso, antes de traerlos a almacenamiento primario. Los esquemas de búsqueda anticipada intentan determinar por adelantado a qué páginas o segmentos hará referencia un proceso.
- **Estrategias de colocación.** Tratan del lugar del almacenamiento primario donde se colocará una nueva página o segmento. Los sistemas de paginación toman las decisiones de colocación de una forma trivial, pues una nueva página puede ser colocada dentro de cualquier marco de página disponible.
- **Estrategias de reposición.** Tratan de la decisión de cuál página o segmento desplazar para hacer sitio a una nueva página o segmento cuando el almacenamiento primario está completamente comprometido.

Dentro de éstas estrategias (de reposición de página) tenemos las siguientes:

- Principio de optimización
- Reposición de páginas al azar
- Primero en entrar-primero en salir (FIFO)
- Menos recientemente usada (LRU)

- Menos frecuentemente usada (LFU)
- No usada recientemente (NUR)
- Conjuntos de trabajo (WS)

El principio de optimización indica que para obtener un rendimiento óptimo, la página que se va a reponer es una que no se va a utilizar en el futuro durante el periodo de tiempo más largo.

Si buscamos una estrategia de reposición de página con poca sobrecarga y que no sea discriminatoria con determinados usuarios, una técnica simple es elegir al azar la página que será reemplazada. Todas las páginas del almacenamiento principal deben tener la misma probabilidad de ser reemplazadas.

En la reposición de página por el sistema de primero en entrar-primero en salir, se pone un sello de tiempo en cada página al entrar en el almacenamiento primario. Cuando una página necesita ser reemplazada, escogemos aquella que ha estado más tiempo almacenada.

La estrategia de reposición de página menos-recientemente-usada (LRU) selecciona para ser reemplazada la página que no ha sido usada durante el mayor período de tiempo. Esta estrategia requiere que cada página reciba un sello de tiempo cada vez que se referencia.

Una aproximación a la LRU es la estrategia menos-frecuentemente-usada (LFU). En esta lo importante es la intensidad de uso que ha tenido cada página. La página que será reemplazada es aquella que ha sido usada con menos frecuencia o que ha sido referida con menos intensidad.

Un esquema usual de aproximación a LRU con poca sobrecarga es el de las páginas no usadas recientemente (NUR). Las páginas que no han tenido uso reciente, tienen poca probabilidad de ser usadas en un futuro próximo y pueden ser reemplazadas por otras nuevas.

Un conjunto de trabajo es una colección de páginas a las cuales un proceso hace activamente referencia. En la estrategia de conjunto de trabajo trata de mantener el conjunto de trabajo de los programas activos en el almacenamiento primario, si una página no está dentro de este conjunto de páginas favorecidas de un proceso, es reemplazada.

I.3.2.5 Localidad

Un aspecto central en casi todas las estrategias de administración del almacenamiento es el concepto de localidad: los procesos tienden a hacer referencia al almacenamiento en patrones no uniformes y muy localizados. Este concepto muestra los procesos en ejecución, es decir, los procesos tienden a favorecer a un subconjunto de sus páginas durante un intervalo de ejecución. La

localidad se manifiesta en el tiempo y en el espacio. La localidad temporal es la localidad en el tiempo. La localidad en el espacio significa que elementos cercanos tienden a ser similares. De hecho, ésta es bastante razonable dentro de los sistemas de computación cuando se considera la forma en que se escriben los programas y organizan los datos. Sobre todo:

- **Localidad temporal.** Significa que si un proceso hace referencia a una página, probablemente pronto hará referencia a esa página nuevamente.
- **Localidad en el espacio.** Significa que si un proceso hace referencia a una página, probablemente hará referencia a las páginas cercanas de su espacio de direcciones virtuales.

I.3.2.6 El modelo de conjuntos de trabajo

Denning desarrolló el concepto de conjuntos de trabajo para explicar el comportamiento de un programa en función de la localidad. Las estrategias de administración del almacenamiento por medio de conjuntos de trabajo intentan mantener el conjunto de trabajo de páginas del proceso,(es decir, las páginas más recientemente referenciadas) dentro del almacenamiento primario para que el proceso pueda ser ejecutado con rapidez. Podrán iniciarse nuevos procesos sólo si hay lugar dentro del almacenamiento primario para sus conjuntos de trabajo. Los procesos que intenten su ejecución sin contar con suficiente espacio para sus conjuntos de trabajo, suelen experimentar hiperpaginación, un

fenómeno en el cual se reemplazan continuamente páginas que vuelven a ser llamadas de inmediato al almacenamiento primario.

I.3.2.7 Paginación por demanda

Lo más común es que las páginas sean cargadas por demanda. Ninguna página debe ser traída del almacenamiento secundario al primario hasta que sea referenciada de forma explícita por el proceso en ejecución. Algunas ventajas de ésta estrategia se mencionan a continuación:

- El intento de precargar páginas con anticipación a su uso puede dar como resultado cargar páginas equivocadas.
- La paginación por demanda garantiza que las únicas páginas traídas al almacenamiento principal sean las que en realidad necesita el proceso.
- La sobrecarga implicada en la decisión de qué páginas traer al almacenamiento principal en mínima.

La paginación por demanda no está exenta de problemas, por ejemplo el tiempo que se emplea en buscar las páginas es mayor que el tiempo que tarda la ejecución del proceso. Un proceso debe acumular sus páginas una por una. Cada vez que se referencia una nueva página, el proceso debe esperar a que se transfiera al almacenamiento primario.

I.3.2.8 Paginación anticipada

En la paginación anticipada, el sistema operativo intenta predecir las páginas que un proceso va a necesitar y a continuación precarga éstas páginas cuando hay espacio disponible. Si se toman las decisiones correctas, el tiempo total de ejecución de un programa puede ser reducido considerablemente. Mientras el proceso ejecuta sus páginas actuales, el sistema carga páginas nuevas que estarán disponibles cuando el proceso las pida. Esta estrategia tiene las siguientes ventajas:

- Si se toman las decisiones correctas en la mayoría de los casos, el tiempo de ejecución de un proceso puede ser reducido de forma considerable.
- Si en muchas ocasiones pueden tomarse decisiones precisas, con una sobrecarga relativamente baja, entonces la ejecución de un proceso dado puede acelerarse considerablemente sin efectos adversos sobre los demás proceso activos.
- Con el hardware más económico, las consecuencias de una mala decisión serán menos graves.

I.3.2.9 Liberación de página

En una administración del almacenamiento por conjunto de trabajo, los programas nos dicen qué páginas desean usar por medio de referencias explícitas. Los programas que ya no necesitan páginas específicas deben sacarlas de su conjunto de trabajo.

Cuando se hace evidente que ya no se necesitará una página, un usuario puede emitir una liberación voluntaria de página, para liberar el marco de página, aunque la incorporación de los datos de liberación de página dentro de los programas puede retrasar el desarrollo de las aplicaciones.

I.3.2.10 Tamaño de la página

En los sistemas de paginación, el almacenamiento real suele estar dividido en estructuras de página de tamaño fijo. Aún cuando no hay una necesidad apremiante para que en todos los sistemas de computación se tenga el mismo tamaño de página, o un sólo tamaño de página, existen algunas consideraciones que determinan el tamaño óptimo de una página para un sistema dado:

- Cuanto más pequeño sea el tamaño de una página, más páginas y estructuras de página habrá y mayores habrán de ser las tablas de página. El desperdicio de almacenamiento debido al tamaño excesivo de las tablas se llama **fragmentación de tablas**; las transferencias de E/S son más eficientes con páginas grandes.
- La localidad tiende a ser pequeña; por lo tanto, un tamaño más pequeño de página ayudaría a un programa a establecer un conjunto de trabajo más cerrado.
- Cuanto más pequeña sea la página, menor será la fragmentación interna.

I.4 Administración de unidades de almacenamiento

En los sistemas de computación multiprogramados, la ineficiencia es a menudo resultado de un uso inapropiado de los dispositivos de almacenamiento rotacionales como discos y tambores. La planificación dinámica de estos recursos permite que el ordenador opere más eficientemente.

I.4.1 Planificación de discos

La mayor parte del procesamiento en las computadoras gira alrededor del sistema de discos, pues facilitan el almacenamiento de información en línea, tanto de programas como de datos. La mayoría de los programas, como los compiladores, ensambladores, rutinas de clasificación, editores, formateadores, etc. se encuentran almacenados en un disco hasta que se cargan en memoria, y luego utilizan el disco como fuente y como destino en su procesamiento. De aquí que la adecuada administración del almacenamiento en disco tiene mucha importancia para un sistema de computación.

I.4.1.1 Características físicas

Físicamente los **discos magnéticos** son relativamente sencillos (fig. I.4.1). Cada disco tiene una forma plana, circular como un disco de fonógrafo, sus dos superficies se hallan recubiertas con

un material magnético, similar al de la cinta magnética. La información se registra en las superficies. Estos discos están conectados por un eje común que gira a alta velocidad (por ejemplo 3600 revoluciones por minuto).

La superficie del disco se halla lógicamente dividida en **pistas.** En una superficie de disco puede haber cientos de pistas. Un disco de cabezas fijas tiene una cabeza para cada pista. Esta

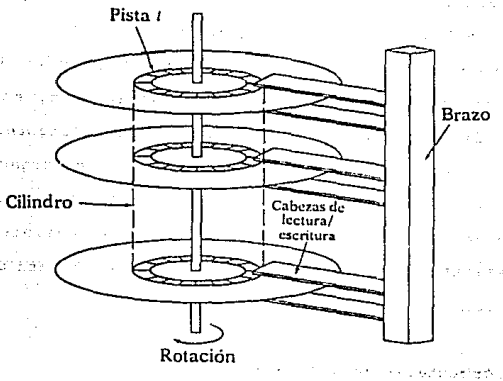


Fig. I.4.1 Esquema del mecanismo de las cabezas de un disco

disposición permite que el ordenador cambie de una pista a otra rápidamente pero precisa de un gran número de cabezas. Actualmente por lo general, se utiliza una cabeza, (para cada superficie del disco) que se mueve hacia adentro y hacia afuera para acceder a diferentes pistas, este **disco** se conoce como **de cabezas móviles** y precisa del hardware para mover la cabeza.

Un **disco de cabezas fijas** es lógicamente igual que un tambor, este dispositivo tiene forma de un cilindro en lugar de un platillo, y el registro se realiza sobre su lado, en lugar de sobre sus extremos, éstos suelen tener unas velocidades de transferencia mayores que los discos, pero tienen menos capacidad de almacenamiento. Los tambores se utilizaron generalmente como memoria auxiliar en los primeros sistemas de paginación, por eso la memoria auxiliar se llamaba tambor de paginación. Ahora muchos sistemas utilizan un disco de alta velocidad o un dispositivo de memoria de núcleos, o semiconductores de alta velocidad que actúa como un tambor, pero que tiene una velocidad de transferencia y acceso aún mayor. Algunos sistemas han empezado a usar los **discos CD-ROM's** (Compact Disk Read Only Memory) que pueden almacenar hasta 600 Megabytes (Mb) de información. Estos discos, por ahora pueden ser de dos tipos: aquellos grabados en fábrica y otros donde se puede escribir en ellos una sola vez. Algunas compañías ya han desarrollado los discos que pueden borrarse y escribirse, aunque todavía no son comerciales. Estos discos probablemente llegarán a complementar a los discos magnéticos, con la ventaja de ser

completamente removibles. Como permiten el acceso aleatorio, cualquier información grabada en ellos puede recuperarse en pocos segundos (pero no tan rápidamente como en el caso de los discos magnéticos). Esto los ha hecho especialmente útiles para difundir información de referencia. La forma en que se opera el CD-ROM es igual a la del disco compacto de audio.

Regresando a los discos magnéticos, para facilitar una capacidad de almacenamiento adicional, se ha mejorado la densidad de grabación, permitiendo almacenar más bits por superficie. La densidad queda reflejada por el número de pistas por pulgada y por lo tanto, por el número total de pistas sobre una superficie.

Finalmente, el disco puede ser removible, con lo que es posible montar discos diferentes según sea necesario. Los paquetes de discos removibles pueden consistir en uno o varios discos (llamados platos). Estos discos grandes son platos rígidos de aluminio cubiertos con material magnético. Las cabezas de lectura/escritura se mantienen tan próximas como sea posible a la superficie del disco.

Los **floppy disks** (ó diskettes) se basan en una idea diferente. Los discos se hallan cubiertos por una superficie resistente, con lo que la cabeza de lectura/escritura puede descansar directamente sobre la superficie del disco, sin destruir los datos. Aunque el recubrimiento (y la cabeza de lectura/escritura) se irá desgastando

con el uso y precisará ser reemplazado con el tiempo. Los floppies son por lo regular mucho más pequeños que los discos duros, van desde 100 bytes a 1.44 Mb bytes (por el momento), y son de muchos tipos (simple o doble cara, simple, doble o alta densidad) y tamaños (5 1/4", 3 1/2", 2 1/2"). Los discos duros van desde 5 Mb hasta más de 300 Mb por unidad (hasta la fecha).

El hardware de un sistema de discos puede dividirse en dos partes:

- El **disk drive (unidad de disco)** es la parte mecánica, e incluye el motor, las cabezas de lectura/escritura y la lógica asociada.
- El **controlador de disco**, que determina la interacción lógica con la computadora.

Cada una de las diversas cabezas de lectura/escritura mientras se encuentran fijas en una posición, trazan una pista circular de datos, sobre una superficie del disco. Todas las cabezas de lectura/escritura están montadas sobre una sola barra, o brazo móvil, éste puede moverse hacia adentro o hacia afuera. Cuando se encuentra en una posición particular el brazo, la serie de pistas trazadas en las superficies, por todas las cabezas de lectura/escritura, forman un cilindro vertical. Dentro de una pista, la información se escribe en bloques, que pueden ser de tamaño fijo, especificado en el hardware, éstos reciben el nombre

de sectores. Cada sector puede escribirse o leerse independientemente.

La velocidad del disco se compone de tres partes:

- **Tiempo de búsqueda (T.B,** o posicionamiento). Tiempo que tarda la cabeza en posicionarse en la pista apropiada.
- **Tiempo de latencia (T.L),** en el que la cabeza espera por el sector específico. Y
- **Tiempo de transferencia (T.T),** es el tiempo que tarda la verdadera transferencia de datos entre el disco y la memoria principal.

El tiempo total de servicio de una solicitud de acceso a disco es la suma los tres tiempos anteriores:

Tiempo total de servicio = T.B. + T.L. + T.T.

El sistema operativo puede mejorar el tiempo medio de servicio del disco planificando las solicitudes de acceso al mismo, esto es ordenar la cola de peticiones de alguna forma. Este proceso se conoce como Planificación de disco. Los tipos de planificación más comunes son:

- Optimización de búsqueda
- Optimización rotacional

I.4.1.2 Optimización de búsqueda

La siguiente tabla resume algunas de las estrategias más populares de optimización de búsqueda.

- FCFS (primero en llegar, primero en ser servido): no hay reordenamiento de la cola.
- SSFT (menor tiempo de búsqueda primero): el brazo del disco se sitúa en la siguiente petición que minimice el movimiento del brazo.
- SCAN: el brazo del disco se mueve de un lado a otro sobre la superficie del disco, sirviendo a todas las peticiones que encuentra a su paso. Cambia de dirección sólo cuando ya no hay más peticiones que servir en la dirección actual.
- C-SCAN (búsqueda circular): el brazo del disco se mueve en una sola dirección por toda la superficie del disco hacia la pista interna. Cuando ya no hay más peticiones que servir al frente del brazo, éste salta a servir la petición más cercana a la pista externa y procede de nuevo hacia adentro.
- SCAN de n-pasos: el brazo del disco se mueve de un lado al otro como en SCAN, pero todas las peticiones que lleguen durante un movimiento en una dirección, son colocadas en lotes y reordenadas para un servicio óptimo durante el movimiento de regreso.

Tabla 1.4.2 Políticas básicas de planificación de disco

I.4.1.2.1 Planificación First Come First Served

La forma más sencilla de planificación de disco es, el FCFS (First Come First Served: Primero en llegar, primero en ser servido). Este algoritmo es fácil de programar e intrínsecamente justo, pero cuando la tasa de peticiones se sobrecarga, FCFS puede dar como resultado tiempos de espera muy largos.

I.4.1.2.2 Planificación Shortest Seek Time First

En la planificación SSTF (Shortest Seek Time First: Menor tiempo de búsqueda primero) se sirve a todas las solicitudes cuyas pistas se encuentran cercanas a la posición actual de la cabeza, antes de mover lejos la cabeza para servir otra solicitud, aunque la siguiente en ser servida, no sea la primera de la cola de espera. SSTF tiende a discriminar de manera clara ciertas peticiones. Los patrones de búsqueda SSTF tienden a estar muy localizados, dando como resultado que las pistas más internas y más externas reciban un pobre servicio, en comparación de las pista centrales. Sin embargo, ésta planificación da como resultado mejores tasas de capacidad de ejecución que FCFS, y la media de tiempos de respuesta tiende a ser más baja, para cargas moderadas. STF es útil en sistemas de procesamiento por lotes en los cuales capacidad de ejecución es lo más importante, pero la alta varianza de los tiempos de respuesta (es decir, su falta de

predicibilidad) lo hacen inaceptable para los sistemas interactivos.

I.4.1.2.3 Planificación SCAN

La cabeza de lectura/escritura empieza en un extremo del disco y se mueve hacia el otro, sirviendo las solicitudes a medida que llega a cada pista, hasta que se encuentra en el otro extremo del disco. Entonces, se invierte la dirección del movimiento y continúa el servicio. La cabeza barre continuamente la superficie de un extremo a otro. SCAN ha sido la base de la mayoría de las estrategias de planificación que se implementan en la actualidad, esta planificación se compara de manera muy parecida a SSTF desde el punto de vista de la mejora en la capacidad de ejecución y de la media de los tiempos de respuesta, pero elimina mucho de la descriminación inherente a los esquemas SSTF y ofrece una varianza menor.

I.4.1.2.4 Planificación SCAN de n-pasos

Una modificación interesante de la estrategia SCAN básica se denomina SCAN de n-pasos. En esta estrategia el brazo del disco se mueve de un lado a otro, como en SCAN, pero sólo da servicio a aquellas peticiones que se encuentran en espera cuando comienza un recorrido particular. Las peticiones que llegan durante el recorrido son agrupadas y ordenadas para un servicio óptimo durante

el recorrido de regreso. SCAN de n-pasos ofrece un buen rendimiento de la capacidad de ejecución y de la media de los tiempos de respuesta, su característica más significativa es una menor varianza de los tiempos de respuesta que en las planificaciones SSTF y SCAN convencionales.

I.4.1.2.5 Planificación C-SCAN

Otra modificación a la estrategia SCAN básica se denomina C-SCAN (Circular SCAN: Búsqueda circular), la cual elimina la discriminación contra los cilindros exterior e interior.

En la estrategia C-SCAN, el brazo se mueve del cilindro exterior al interior, sirviendo las peticiones sobre una base de búsqueda más corta. Cuando el brazo ha completado su recorrido hacia adentro, salta a la petición más cercana al cilindro exterior y a continuación reanuda su recorrido hacia adentro procesando peticiones.

C-SCAN puede implementarse de manera que las peticiones que llegan durante un recorrido sean servidas en el siguiente. De esta forma C-SCAN elimina completamente la discriminación contra peticiones de los cilindros exterior e interior. Tiene una varianza de los tiempos de respuesta muy pequeña.

I.4.1.3 Optimización Rotacional

En condiciones de carga pesada, las probabilidades de que ocurran referencias al mismo cilindro aumentan y resulta útil considerar la optimización rotacional y la optimización de búsqueda. La optimización rotacional se ha utilizado durante muchos años en dispositivos de cabeza fija como tambores.

Una forma paralela a la estrategia de optimización de búsqueda, SSTF es la estrategia SLTF (Shortest Latency Time First: Menor tiempo de latencia primero) de optimización rotacional. Una vez que el brazo del disco llega a un cilindro en particular, pueden existir muchas peticiones pendientes en las distintas pistas de ese cilindro. La estrategia SLTF examina todas estas peticiones y sirve primero a aquella que tiene el retraso rotacional más corto. Esta estrategia ha demostrado estar muy cerca del óptimo teórico y es relativamente fácil de implantar.

Todas las estrategias mencionadas persiguen las metas de planificación siguientes:

- Maximizar la capacidad de ejecución
- Minimizar los tiempos de respuesta
- Minimizar la varianza de los tiempos de respuesta

La planificación de disco no siempre es útil, debido a los problemas de rendimiento que pueden causar otras partes del

hardware o software. Los embotellamientos pueden estar en los discos, los controladores o los canales. O pueden ser causados por distribuciones no uniformes de peticiones como las normales en el procesamiento de archivos secuenciales.

Los métodos de acceso como el ISAM (Index Sequential Access Method) pueden ser convenientes para las aplicaciones, pero pueden requerir un gran número de largas búsquedas, por funcionar durante el tiempo de ejecución.

I.4.2 Sistemas de archivos

Para la mayoría de los usuarios del sistema de archivos es el aspecto más visible de un sistema operativo. Los archivos almacenan datos y programas.

El sistema operativo implementa el concepto abstracto de: archivo, administrando dispositivos de almacenamiento masivo, tales como cintas y discos.

I.4.2.1 Concepto de archivo

La administración de archivos es uno de los servicios más evidentes de un sistema operativo. Las computadoras pueden almacenar información de distintas formas físicas; la cinta magnética, el disco y el tambor son las más comunes.

Cada uno de éstos dispositivos tiene sus propias características y organización física.

El sistema operativo hace abstracciones de las propiedades físicas de sus dispositivos de almacenamiento y define una unidad lógica de almacenamiento: el archivo.

El sistema operativo acopla, o plasma los archivos a los dispositivos físicos (fig. I.4.3).

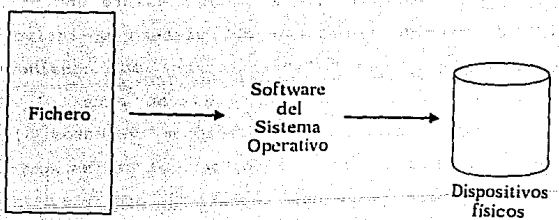


Fig. I.4.3 Acoplamiento de archivos lógicos sobre un dispositivo físico.

Un archivo es una colección de datos relacionados entre sí y definidos por su creador. Normalmente los archivos representan programas y datos. En general, un archivo es una secuencia de bits, bytes, líneas o registros cuyo significado ha sido definido por su creador y su usuario.

Un archivo tiene una cierta estructura que se define según el uso que se vaya a hacer de él, y el sistema operativo debe conocer esta estructura para que pueda operar sobre él, aunque cuando se definen muchos tipos de estructuras, el sistema operativo crece, pues debe contener el código para administrar correctamente estas estructuras de archivos, el otro extremo consiste en no imponer ningún tipo de archivo por parte del sistema operativo, este enfoque ha sido adoptado en UNIX entre otros. UNIX considera cada archivo como una secuencia de bytes de ocho bits; el sistema operativo no realiza ninguna interpretación de éstos bits. El esquema permite la máxima flexibilidad, pero el mínimo apoyo. Cada programa de aplicación tiene que incluir su propio código para conocer la estructura de un archivo de entrada.

I.4.2.2 Jerarquía de datos

Todos los datos manipulados por las computadoras están compuestos por **bits**, los dígitos binarios 0 y 1.

El siguiente nivel superior en la jerarquía de datos está compuesto por **bytes** o **caracteres** que son patrones de bits de longitud fija. Muchos sistemas actuales usan bytes de 8 bits.

La asignación de patrones de bit a los caracteres que representan se llama **conjunto de caracteres**. Los dos conjuntos de caracteres más populares hoy día son **ASCII** (American Standard Code for Information Interchange) y **EBCDIC** (Extended Binary Coded Decimal Interchange Code). Ambas representaciones son de uso tan común que muchos sistemas contienen mecanismos para utilizar los dos códigos y convertirlos libremente entre sí.

Un **campo** es un grupo de caracteres relacionados. Pueden ser numéricos, alfabéticos o alfanuméricos.

Un **registro** es un grupo de campos relacionados. Una clave de registro es un campo de control que identifica de manera exclusiva al registro.

Un **archivo** es un grupo de registros relacionados.

El nivel más alto en la jerarquía de datos suele denominarse **base de datos**, que consiste en un grupo de archivos relacionados.

I.4.2.3 Operaciones con archivos

Para definir adecuadamente un archivo se deben considerar las operaciones que pueden realizarse sobre él, ya que puede ser manipulado como una unidad por operaciones como:

- **open.** Preparar un archivo para ser referenciado.
- **close.** Evitar más referencias a un archivo hasta abrirlo de nuevo.
- **create.** Construir un nuevo archivo.
- **destroy.** Destruir un archivo.
- **copy.** Crear otra versión del archivo con un nombre nuevo.
- **rename.** Cambiar el nombre de un archivo.
- **list.** Imprimir o desplegar el contenido de un archivo.

Los elementos de datos individuales dentro del archivo pueden ser manipulados por operaciones como:

- **read.** Introducir un dato de un archivo en un proceso.
- **write.** Introducir un dato de un proceso en un archivo.
- **update.** Modificar un dato existente.
- **insert.** Añadir un nuevo dato.
- **delete.** Eliminar un dato de un archivo.

I.4.2.4 Características de los archivos

Los archivos pueden caracterizarse por:

- **Volatilidad.** Se refiere a la frecuencia con que se realizan adiciones y eliminaciones en un archivo. Cuando esta frecuencia es baja, el archivo es estático, y cuando es alta, el archivo es volátil.

- **Actividad.** Se refiere al porcentaje de registros de un archivo que son procesados en una ejecución dada.

- **Tamaño.** Se refiere a la cantidad de información almacenada en el archivo.

La información concreta que, para cada archivo se mantiene en el directorio varía de un sistema operativo a otro, ésta es utilizada por un descriptor de archivos o bloque de control de archivos, para administrar un archivo. Un descriptor de archivos puede incluir:

- Nombre simbólico del archivo.
- Localización del archivo en el almacenamiento secundario.
- Organización del archivo (secuencial, secuencial indexada, etc.).
- Tipo de dispositivo.
- Datos de control de acceso.
- Tipo de archivo.
- Diposición (permanente o temporal).
- Fecha y tiempo de creación.
- Fecha de destrucción.
- Fecha de la última modificación.
- Suma de las actividades de acceso.

I.4.2.5 Organización de archivos

La organización de archivos se refiere a la manera en que se disponen los registros de un archivo dentro de un almacenamiento secundario. Los esquemas más comunes de organización de archivos utilizados actualmente son los siguientes:

- **Secuencial.** Los registros se colocan en orden físico. El siguiente registro es el que sigue físicamente al registro previo. Esta organización se utiliza para archivos almacenados en cinta magnética, cita perforada, y salidas impresas. Los archivos de disco también pueden organizarse en secuencia.

- **Secuencial indexado.** Los registros se disponen en una secuencia lógica de acuerdo con una clave contenida en cada registro. El sistema mantiene un índice que contiene la dirección física de ciertos registros principales. Los registros secuenciales indexados pueden ser accedidos de manera secuencial por orden de clave o pueden ser de acceso directo, por medio de una búsqueda a lo largo del índice creado por el sistema.

- **Directo.** Los registros son de acceso directo (al azar) según sus direcciones físicas en un dispositivo de almacenamiento de acceso directo (DASD). El usuario de aplicaciones coloca los registros en el DASD, en un orden cualquiera apropiado para una aplicación específica.

- **De partición.** Este es, en esencia un archivo de subarchivos secuenciales; cada subarchivo secuencial se denomina, miembro. La dirección de comienzo de cada miembro está alamacenada en el

directorio del archivo. Los archivos de partición suelen utilizarse para alamcenar bibliotecas de programas o macrobibliotecas.

Existen muchas variaciones de uso común de las organizaciones de archivos antes mencionadas.

I.4.2.6 Sistemas de archivos

El sistema de archivos es un componente importante de un sistema operativo, es la parte del sistema de administración del almacenamiento responsable, principalmente de la administración de archivos del almacenamiento secundario. Es responsable de proporcionar los medios para un almacenamiento privado de información y de compartir la información entre los usuarios de una manera controlada.

Algunas de las funciones que suelen atribuirse al sistema de archivos son las siguientes:

1. Los usuarios deben poder crear, modificar, y borrar los archivos.
2. Los usuarios deben poder compartir sus archivos de una manera controlada para poder basarse en el trabajo de los demás.
3. El mecanismo encargado de compartir los archivos debe proporcionar varios tipos de acceso controlado, de lectura, de escritura, de ejecución o combinación de éstos.

4. Los usuarios deben poder estructurar sus archivos de manera más apropiada a cada aplicación.
5. También deben poder ordenar la transferencia de información entre archivos.
6. Deben proporcionarse posibilidades de respaldo y recuperación de información.
7. Los usuarios deben poder hacer referencia a sus propios archivos por medio de nombres simbólicos, en lugar de tener que usar nombres de dispositivos físicos.
8. En ambientes donde la información debe ser mantenida segura y probada, el sistema de archivos también debe proporcionar posibilidades de cifrado y descifrado (poner la información en clave).
9. Debe proporcionar un acoplamiento favorable al usuario. Debe proporcionar a los usuarios una visión lógica de sus datos y de las funciones que van a ser ejecutadas, en lugar de una visión física. Los sistemas de archivos suelen contener:

- **Métodos de acceso.** Relacionados con la manera de acceder a los datos almacenados en archivos.

- **Administración de archivos.** Se refiere a la provisión de mecanismos para que los archivos sean almacenados, referenciados, compartidos y asegurados.

- **Métodos de asignación.** Tratan de la asignación de espacio a los archivos en los dispositivos de almacenamiento secundario.

- **Integridad del archivo.** Se ocupa de garantizar la integridad de la información del archivo.

I.4.2.6.1 Métodos de acceso

Los grandes sistemas operativos suelen proporcionar muchos métodos de acceso. Estos son agrupados en dos categorías:

- Métodos de acceso por colas (secuenciales)
- Métodos de acceso básico

Métodos de acceso secuenciales

Se utilizan cuando se puede anticipar la secuencia en la que van a ser procesados los registros, como en los accesos secuencial y secuencial indexado. Estos métodos realizan las operaciones de buffer y de planificación de entrada/salida de manera anticipada. Están basados en un modelo de archivo en cinta (fig. I.4.4)

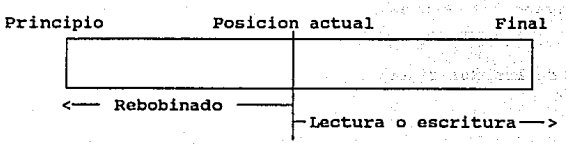


Fig. I.4.4 Archivo de acceso por colas

Métodos de acceso básico

Estos métodos suelen usarse cuando no es posible anticipar la secuencia en la que van a ser procesados los registros, sobre todo en el acceso directo. En los métodos básicos, el archivo se ve como

una secuencia numerada de bloques de registros, éstos bloques son generalmente de longitud fija que permite leer o escribir en bloques arbitrarios físicos. Están basados en un modelo de archivo en disco (fig. I.4.5)

Fig. I.4.5 Archivo de acceso básico

No todos los sistemas operativos soportan los dos tipos de acceso a los archivos, pues la mayoría de los sistemas permiten sólo uno de los dos tipos.

Sobre el método de acceso directo pueden construirse otros métodos de acceso. Estos métodos adicionales implican generalmente la construcción de un índice de archivos, el cual es como un índice de un libro, que contiene apuntadores a los distintos bloques. Para hallar una entrada a un archivo, se explora primero el índice y luego se utiliza el apuntador para acceder directamente al archivo y encontrar la entrada deseada. Con archivos grandes, el propio

archivo de índices puede llegar a ser demasiado grande como para mantenerse en memoria. Una solución consiste en crear un índice para el archivo de índices. El archivo del índice primario contendría los apuntadores a los archivos de índice secundario que a su vez apuntarían a los datos reales. Por ejemplo el **ISAM** (Indexed Sequential Access Method) utiliza un pequeño índice maestro que apunta a los bloques del disco de un índice secundario. A su vez, éstos apuntan a los bloques de archivos reales. De esta forma cualquier registro puede ser localizado por su clave en dos lecturas de acceso directo como máximo.

I.4.2.6.2 Métodos de asignación

Hay dos grandes divisiones de los métodos de asignación del espacio de disco que se emplean por lo general:

- Asignación Contigua
- Asignación no Contigua

Cada método tiene sus ventajas y sus inconvenientes. Debido a ésto, algunos sistemas soportan métodos de los dos tipos, pero por lo general los sistemas utilizan un método determinado para todos los archivos.

Durante la operación de un sistema informático los archivos se crean y borran con frecuencia. Debido a que el espacio es limitado, es necesario reutilizar el espacio que ocupaban los archivos

borrados para almacenar archivos nuevos. Para seguir la pista del espacio libre en un disco, el sistema operativo mantiene una lista de espacios libres, que contiene todos los bloques del disco que están libres. Para crear un archivo, se explora la lista de espacio libre para buscar el espacio necesario o bien, cuando se borra un archivo se añade a la lista el espacio desocupado. Otro enfoque consiste en enlazar todos los bloques libres, manteniendo un apuntador al primero de ellos. Este bloque contiene un apuntador al siguiente bloque libre puesto que para recorrer la lista se debe leer cada bloque, lo que requiere tiempo de entrada/salida importante. Una variante de este enfoque almacenaría las direcciones de n bloques libres; en el primer bloque libre. Otro enfoque más, consiste en aprovechar el hecho de que generalmente se asignan o liberan varios bloques continuos simultáneamente, en particular cuando se utiliza la asignación contigua.

Asignación contigua

En la asignación contigua, los archivos son asignados a áreas contiguas de almacenamiento secundario. Un usuario especifica por adelantado el tamaño del área necesario para contener un archivo que se va a crear, esta asignación se define por la dirección del primer bloque y su longitud. Una ventaja de la asignación contigua es que los registros sucesivos lógicos son, por lo general físicamente adyacentes entre sí. Esto acelera el acceso, en comparación con los sistemas en los que los registros sucesivos

lógicos están dispersos por todo el disco. Otra ventaja es que ésta asignación soporta tanto el acceso directo como el secuencial.

Para asignar y liberar los archivos, es común que el espacio de disco se fragmente cada vez más. Una técnica para resolver este problema consiste en realizar una condensación o recolección de basura periódica. Los archivos pueden ser reorganizados para que ocupen áreas adyacentes del disco y las áreas libres pueden ser condensadas en un sólo bloque o en un grupo de grandes bloques. Esta recolección de basura suele hacerse fuera de las horas de trabajo. Cuando se pueda parar el sistema. Algunos sistemas lo realizan de forma dinámica mientras están en operación.

En los sistemas paginados, la cantidad mínima de información transferida entre los almacenamientos primario y secundario es una página, por lo que es lógico asignarle al almacenamiento secundario bloques del tamaño de una página, o de un múltiplo del tamaño de la página; además, es deseable almacenar las páginas lógicamente contiguas del almacenamiento virtual de un usuario como páginas físicamente contiguas en el almacenamiento secundario.

Los esquemas de asignación contigua presentan los mismos problemas de fragmentación inherentes a los sistemas de multiprogramación de partición variable.

Asignación dinámica de almacenamiento

El espacio en disco puede verse como una gran tabla de bloques de disco. En un momento determinado, algunos bloques se asignan a archivos otros quedan libres. Entonces el espacio en disco puede verse como una colección de segmentos libres y segmentos utilizados.

El problema de asignación dinámica de almacenamiento consiste en cómo satisfacer una solicitud de tamaño n con una lista de espacios libres.

Las estrategias más comunes utilizadas para seleccionar un espacio libre del conjunto, son:

- **First-fit (Primer ajuste).** Asigna el primer espacio que sea suficientemente grande.
- **Best-fir (Mejor ajuste).** Asigna el espacio más pequeño de todos los que sean suficientemente grandes.
- **Worst-fit (Peor ajuste).** Asigna el espacio más grande.

Estos algoritmos provocan la fragmentación externa a medida que los archivos se asignan y borran, el espacio libre en el disco se fragmenta en pequeñas piezas. La fragmentación externa se produce cuando hay suficiente espacio en disco como para satisfacer una solicitud. El problema más importante de la asignación contígua consiste en determinar cuánto espacio es preciso para un archivo. En algunos casos, determinar ésto puede

ser sencillo, pero en general, el tamaño de un archivo de salida puede resultar bastante difícil de estimar.

Asignación no contígua

Debido a que los archivos muestran tendencia a crecer o encogerse con el tiempo, y como los usuarios rara vez conocen por adelantado todo el tamaño que va a tener sus archivos, los sistemas de asignación contígua de almacenamiento están siendo reemplazados por sistemas de asignación no contígua de almacenamiento, más dinámicos. Se utilizan varios esquemas para facilitar la asignación no contígua de almacenamiento.

Asignación encadenada orientada hacia un sector

Cuando un archivo necesita crecer, el proceso pide más sectores a la lista de espacios libres. Los archivos que se reducen devuelven sectores a la lista de espacio libre. No hay necesidad de condensación. La asignación no contígua resuelve algunos de los problemas inherentes a los esquemas de asignación contígua, pero también tiene sus desventajas. Debido a que los registros de un archivo pueden estar dispersos por el disco, la recuperación de registros lógicamente contíguos puede involucrar largas búsquedas, que causan una sobrecarga en el tiempo de ejecución.

Asignación por bloques

Un esquema usado para administrar el almacenamiento secundario con mayor eficiencia y reducir la sobrecarga en el tiempo de

ejecución es la asignación por bloques. Esta es una mezcla de los dos métodos de asignación contigua y no contigua. En este esquema, en lugar de asignarse sectores individuales, se asignan bloques de sectores contiguos (algunas veces llamados extensiones). El sistema trata de asignar nuevos bloques a un archivo, eligiendo bloques libres lo más próximos posible a los bloques del archivo existentes.

Existen varias formas comunes para implementar los sistemas de asignación por bloques. Estas incluyen:

- **Encadenamiento de bloques**
- **Encadenamiento de bloques indexados y**
- **Transformación de archivos orientados hacia bloques.**

- En el **encadenamiento de bloques**, las entradas en el directorio de usuarios apuntan al primer bloque de cada archivo. Cada uno de los bloques de longitud fija que forman un archivo, contiene dos partes: un bloque de datos y un apuntador al bloque siguiente.

Algunos sistemas usan listas de encadenamiento doble, para facilitar la búsqueda; los bloques se enlazan hacia adelante y hacia atrás de manera que la búsqueda puede proceder en cualquier dirección. (fig. I.4.6)

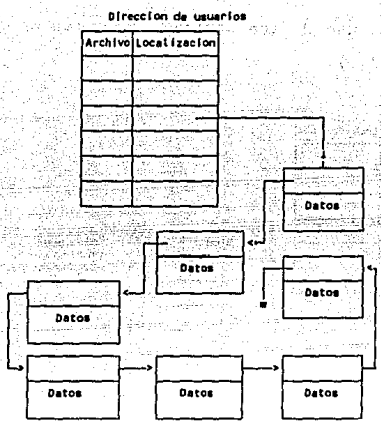


fig. I.4.6 Encadenamiento de bloques

- Con el **encadenamiento de bloques indexados** los apuntadores son colocados en varios bloques de índices separados. Cada entrada contiene un identificador de registros y un apuntador a ese registro. Si es necesario utilizar más de un bloque de índices para describir un archivo, entonces se encadenan una serie de bloques de índices. La gran ventaja que presenta el encadenamiento de bloques de índices sobre el encadenamiento de bloques de índices simples es que la búsqueda puede realizarse en los propios bloques de índices.

La desventaja de este esquema es que las inserciones pueden requerir la reconstrucción completa de los bloques de índices. El

sistema operativo UNIX es uno de los sistemas que usan un esquema análogo a éste. (fig. I.4.7)

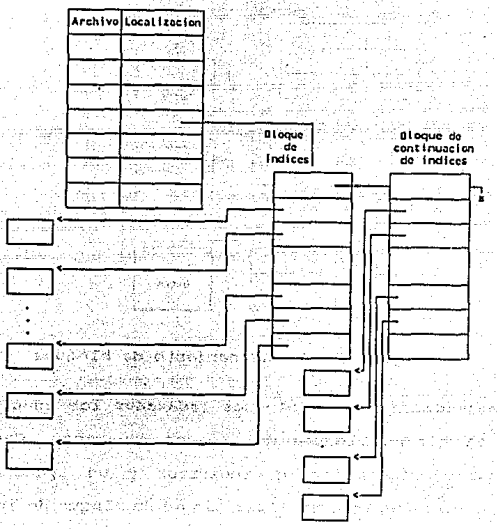


Fig. I.4.7 Encadenamiento de bloques indexados

- En la **transformación de archivos orientada hacia bloques**, en lugar de apuntadores el sistema utiliza número de bloques. En general, éstos se convierten con facilidad a direcciones de bloques, gracias a la geometría del disco. Se conserva un mapa del archivo conteniendo una entrada para cada bloque del disco. Las

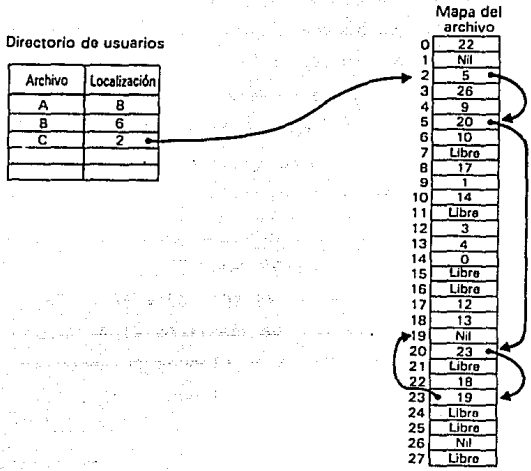
entradas en el directorio del usuario apuntan a la primera entrada del mapa del archivo para cada archivo. De esa forma, todos los bloques de un archivo pueden ser localizados siguiendo las entradas al mapa del archivo. La entrada al mapa del archivo correspondiente a la última entrada de un archivo determinado se ajusta a algún valor centinela, como NILL, para indicar que se ha alcanzado el último bloque de un archivo.

La verdadera ventaja de éste esquema es que, debido a la cercanía física del disco cuando se va a asignar un nuevo bloque, es relativamente fácil localizar un bloque de almacenamiento libre con una buena proximidad a los otros bloques del archivo. En este esquema las inserciones y retiros son inmediatos. (fig. I.4.8)

Hay diferencias importantes entre los diversos métodos de asignación por bloques, respecto a la eficiencia en el almacenamiento. También difieren en el tiempo requerido para acceder a un bloque de disco. Este es un factor particularmente importante en el caso de sistemas de discos pequeños y lentos.

I.4.3 Sistemas de directorio

Los archivos están representados por entradas en un directorio de dispositivo o tabla de contenido del volumen. El directorio de dispositivo guarda información tal como el nombre, situación, tamaño y tipo de todos los archivos de este dispositivo.



Bloques físicos en el almacenamiento secundario

Bloque 0 B(4)	Bloque 1 B(10)	Bloque 2 C(1)	Bloque 3 A(4)	Bloque 4 B(8)	Bloque 5 C(2)	Bloque 6 B(1)
Bloque 7 Libre	Bloque 8 A(1)	Bloque 9 B(9)	Bloque 10 B(2)	Bloque 11 Libre	Bloque 12 A(3)	Bloque 13 B(7)
Bloque 14 B(3)	Bloque 15 Libre	Bloque 16 Libre	Bloque 17 A(2)	Bloque 18 B(6)	Bloque 19 C(5)	Bloque 20 C(3)
Bloque 21 Libre	Bloque 22 B(5)	Bloque 23 C(4)	Bloque 24 Libre	Bloque 25 Libre	Bloque 26 A(5)	Bloque 27 Libre

Fig. I.4.8 Transformación de archivos orientados hacia bloques

Un sistema de archivos para un ambiente de tiempo compartido grande, puede estar organizado en una estructura de directorios que nos dota de un mecanismo para organizar muchos archivos en un sistema de archivos.

En el sistema de archivos de estructura jerárquica, el nombre del sistema para un archivo suele estar formado como el nombre de la trayectoria desde el directorio raíz hasta el archivo. De hecho, muchos sistemas tienen dos estructuras de directorio distintas:

- **Directorio de dispositivo**
- **Directorio de archivos**

El **directorio de dispositivo** se almacena en cada dispositivo físico y describe todos los archivos de ese dispositivo, sus entradas se concentran principalmente en la descripción de las propiedades físicas de cada archivo: dónde está, su tamaño, cómo está asignado, etc.

Los **directorios de archivos** son una organización lógica de los archivos de todos los dispositivos, y sus entradas se concentran en las propiedades lógicas de cada archivo: nombre, tipo de archivo, propietario, información para contabilización, código de protección de acceso, etc. Se han propuesto y se utilizan muchas estructuras de directorio de archivo diferentes.

El directorio es esencialmente una tabla de símbolos, cuando se considera una estructura en particular, debemos tener presentes las operaciones que se han de realizar en el directorio.

- Buscar un archivo
- Crear un archivo
- Borrar un archivo
- Listar el directorio
- Hacer respaldos (backup)

I.4.3.1 Directorio de un nivel

La estructura de directorios más sencilla es el directorio de un sólo nivel. Todos los archivos están contenidos en un mismo directorio, por lo que resulta muy sencillo de administrar y comprender. Pero la principal desventaja de esta estructura, es la confusión de nombres de archivos entre usuarios diferentes. La solución estándar consiste en crear un directorio por separado para cada usuario. (fig. I.4.9)

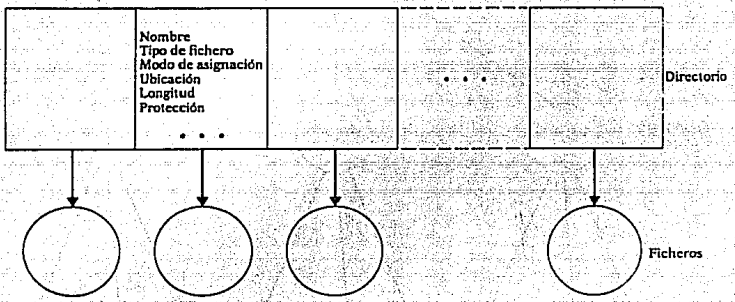


Fig. Directorio de un nivel

I.4.3.2 Directorio de dos niveles

En esta estructura se usa una raiz para indicar en qué parte del disco comienza el directorio raiz. Este directorio apunta a varios directorios de usuarios. Cada directorio de usuario contiene una entrada para cada archivo del usuario.

(fig. I.4.10)

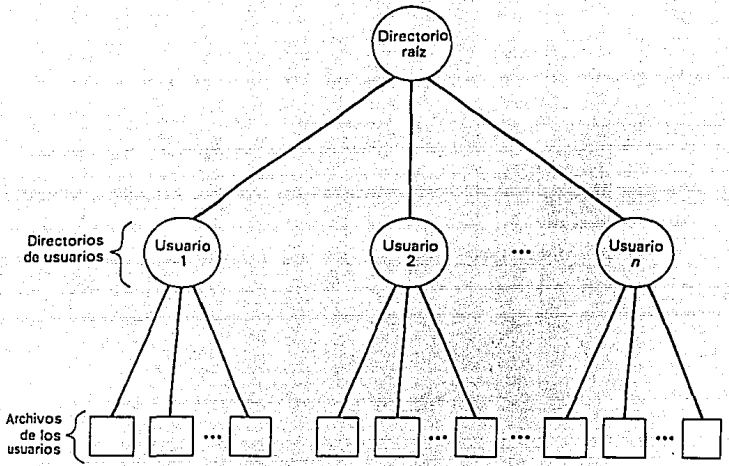


Fig. I.4.10 Directorio de dos niveles

I.4.3.3 Directorio con estructura arborescente

Una vez que se ha concebido un directorio de dos niveles como un árbol de dos niveles, la generalización natural consiste en extender la estructura del directorio hasta un árbol cualquiera (fig. I.4.11). Esto permite que los usuarios puedan crear sus propios subdirectorios y organicen consecuentemente sus archivos, por ejemplo el sistema de archivos de UNIX, es arborescente. El

árbol tiene un directorio raiz. Todo archivo tiene un nombre de trayectoria único.

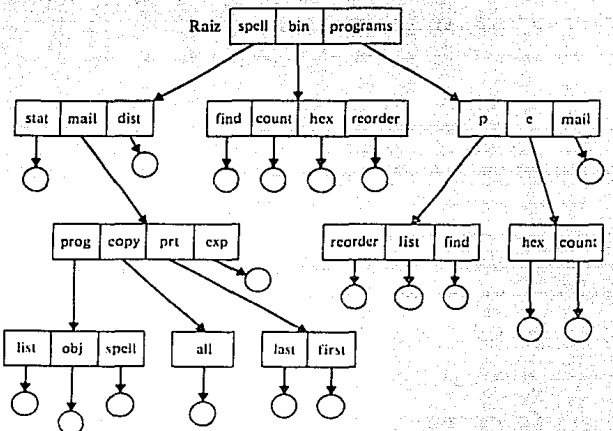


Fig. I.4.11 Directorio con estructura arborescente

I.4.3.4 Directorio en forma de gráfica acíclica

Los archivos asociados a un proyecto de dos o más usuarios, pueden almacenarse en un sólo subdirectorio separándolos de otros proyectos y archivos de los usuarios involucrados. El subdirectorio común tendría que ser compartido, el cuál existiría en dos o más lugares del sistema de archivos al mismo tiempo, que en realidad no

serían dos copias del mismo archivo, sino uno sólo. Una estructura de árbol prohíbe compartir los archivos o directorios. Una gráfica aciclica permite que los directorios tengan subdirectorios y archivos compartidos. (fig. I.4.12)

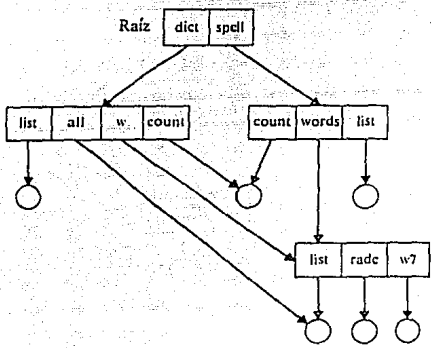


Fig. I.4.12 Directorio en forma de gráfica aciclica

I.4.3.5 Directorio con forma de gráfica general

Es fácil ver que la simple adición de nuevos archivos y subdirectorios en un directorio estructurado en árbol preserva su

naturaleza arborescente, sin embargo, cuando añadimos enlaces a un directorio en árbol, se destruye la estructura arborescente convirtiéndose en una estructura en gráfica simple.
(fig. I.4.13)

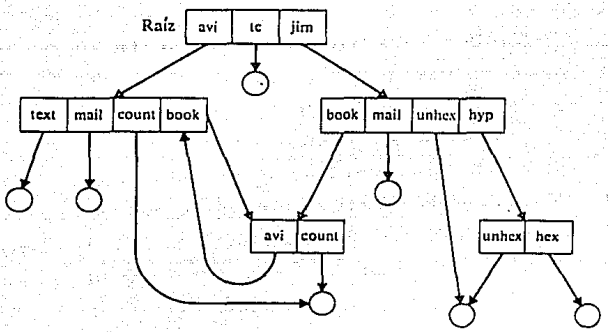


Fig. I.4.13 Directorio en forma de gráfica general

I.5 Administración de Entrada / Salida

Una de las funciones principales de un sistema operativo consiste en controlar todos los dispositivos de entrada/salida de

la computadora. Este debe emitir comandos a los dispositivos, capturar interrupciones y manejar errores. También debe proporcionar una interfase entre los dispositivos y el resto del sistema que sea simple y fácil de utilizar. Hasta donde sea posible, la interfase debe ser la misma para todos los dispositivos. El código de E/S representa una fracción significativa del Sistema Operativo en su totalidad.

I.5.1 Principios del Hardware de Entrada / Salida.

Los dispositivos de E/S pueden dividirse en dos categorías: dispositivos de bloque y dispositivos de caracteres. Un dispositivo de bloque es aquel que almacena información en bloques de tamaño fijo, cada uno con su dirección. Los tamaños comunes de bloque van de 128 bytes a 1024 bytes. La propiedad esencial de un dispositivo de bloque es que es posible leer o escribir cada bloque en forma independiente de los demás (Ej. discos).

El otro dispositivo de E/S es el dispositivo de caracteres. Un dispositivo de caracteres entrega o acepta un flujo de caracteres, sin importar de cuál estructura de bloque se trate. Este no es direccionable y no tiene ninguna operación de localización (Ej. terminales). Las unidades de E/S por lo general constan de un controlador del dispositivo o controlador, y del dispositivo mismo.

Muchos controladores, en especial los de los dispositivos de bloque, tienen DMA (Direct Memory Access, acceso directo a la memoria). En una situación de lectura de disco donde no se utiliza DMA, ocurre lo siguiente: Primero el controlador lee el bloque de la unidad en serie, bit a bit, hasta que todo el bloque esté en el buffer interno del controlador; después, realiza el cálculo de la suma de comprobación para verificar que no hayan ocurrido errores en la lectura; paso seguido el controlador genera una interrupción al CPU; cuando el S.O. inicia su ejecución, éste puede leer el bloque de discos del buffer del controlador, un byte o una palabra a la vez, ejecutando un ciclo, donde en cada iteración se lee un byte o una palabra de un registro del dispositivo controlador y se almacena en la memoria. Cuando se utiliza DMA, ésta da al controlador dos elementos de información, además de la dirección en el disco del bloque: la dirección de la memoria a donde se dirige el bloque y el número de bytes a transferir.

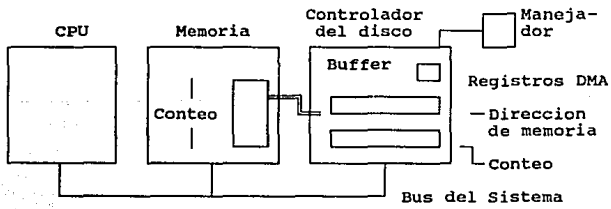


Fig. 1.5.1 Acceso directo a memoria

Después de que el controlador ha leído todo el bloque del dispositivo en su buffer y verificado la suma de comprobación, éste copia el primer byte o palabra en la memoria central en la dirección especificada por la dirección de memoria de DMA. Después incrementa la dirección DMA y determina el conteo DMA por el número de bytes que acaba de transferirse. Este proceso se repite hasta que el conteo DMA se vuelve 0, y en ese instante el controlador da origen a una interrupción. Cuando el S.O. inicia su proceso, no tiene que copiar el bloque de memoria , éste ya se encuentra ahí. La razón de necesitar un buffer interno es que cuando se ha iniciado la transferencia del disco, los bits siguen llegando del disco a intensidad constante, ya sea que el controlador esté listo para ellos o no. Si el controlador intentara escribir datos directamente en la memoria, tendría que examinar el bus del sistema para analizar cada palabra transferida. Si el bus estuviera ocupado debido a que algún otro dispositivo lo estuviese utilizando, el controlador tendría que esperar. Si la siguiente palabra del disco llegara antes de que la anterior hubiera sido almacenada, el controlador tendría que almacenarla en algún sitio. Si el bus estuviera muy ocupado, el controlador podría terminar el almacenamiento de unas cuantas palabras y también tendría que realizar una intensa labor de administración. Cuando el bloque se separa internamente, el bus no se necesita hasta que el DMA da comienzo, de manera que el diseño del controlador es mucho más simple porque la transferencia DMA a la memoria no se lleva mucho tiempo.

I.5.2 Principios del Software de Entrada / Salida.

Desde el punto de vista del software de E/S, la idea básica consiste en organizarlo como una serie de estratos, donde a los inferiores concierne el ocultamiento de las peculiaridades del hardware de los superiores, y a los superiores la presentación de una interfase activa, limpia y regular a los usuarios.

Un concepto importante en el diseño del software de E/S es la independencia del dispositivo, esto es, debe ser transparente para el programa el tipo de dispositivo físico al cual accede.Otro aspecto del software de E/S es el manejo de errores. En términos generales, los errores deben manejarse lo más apegados al hardware que sea posible; si el controlador descubre un error en la lectura debe intentar corregirlo, si es posible; si no puede hacerlo, entonces el manejador del dispositivo debe hacerse cargo de él. Sólo si los estratos inferiores no pueden controlar el problema, se debe informar de él a los superiores.

Otro aspecto importante es la diferencia entre las transferencias síncronas (de bloqueo) contra las asíncronas (activadas por una interrupción).

La mayor parte de la E/S física es asíncrona (el CPU da inicio a la transferencia y sale para hacer algo más hasta que llegue la interrupción). Los programas de usuario son mucho más sencillos de

escribir si se bloquean las operaciones de E/S; es tarea del S.O. hacer que la operaciones que en realidad son activadas por una interrupción parezcan bloqueadas a los programas de usuarios.

El sistema operativo debe hacerse cargo de los dispositivos compartidos y dedicados en forma tal que se eviten problemas; estos objetivos se pueden lograr en forma inteligible y eficiente mediante la estructuración del software de E/S en cuatro estratos.

I.5.2.1 Manejadores de Interrupciones.

Las interrupciones deben ocultarse de manera que la menor parte posible del sistema sepa de ellas, esto consiste en hacer que un proceso, comúnmente un manejador de dispositivo, se bloquee siempre que se haya emitido un comando de E/S y se espere una interrupción. Cuando ésta se presenta, el procedimiento de interrupciones desbloquea el manejador, permitiendo así que el proceso continúe su ejecución.

I.5.2.2 Manejadores de Dispositivos.

El trabajo de un manejador de dispositivo consiste en aceptar solicitudes abstractas del software independiente del dispositivo que está sobre él y observar que se cumpla la requisición. Si el manejador está ocioso en el momento en que llega una requisición, inicia su ejecución de inmediato. Sin embargo, si ya está ocupado

con una solicitud, normalmente meterá la nueva solicitud en una lista de espera de requisiciones pendientes con la cual trabajará lo más pronto posible. El primer paso en la ejecución real de la solicitud de E/S es decidir qué operaciones del controlador se requieren y en que orden. Una vez hecho esto, el controlador inicia su emisión escribiendo en los registros del dispositivo del controlador.

Algunos controladores pueden hacerse cargo sólo de un comando a la vez; otros pueden aceptar una lista enlazada de comandos, que después ejecutan sin ayuda del sistema operativo.

Una vez emitido el comando o comandos, se pueden presentar dos situaciones: que el manejador del dispositivo deba esperar hasta que el controlador realice algún trabajo por él, de modo que se autobloquee hasta que la interrupción llegue para desbloquearlo, o que la operación se realice sin retraso, en cuyo caso el manejador no necesite autobloquearse. De cualquier forma, después de que la operación se ha completado debe verificar si hay errores. Si todo está bien, el manejador puede hacer que se pasen datos al software independiente del dispositivo. Por último, produce información sobre la condición para transmitirla a su solicitante. Si alguna otra requisición se forma en lista de espera, una de ellas puede ser seleccionada e iniciada ahora. Si no se coloca nada en la lista de espera, el manejador se bloquea esperando la siguiente solicitud.

I.5.2.3 Software de E/S independiente del dispositivo.

Una fracción considerable del software de E/S del dispositivo es independiente de él. La frontera exacta entre los manejadores y el software independiente del dispositivo depende del sistema, ya que algunas funciones que podrían realizarse en forma independiente del dispositivo quizá tengan que efectuarse en realidad en los manejadores. Las funciones que se realizan comúnmente en el software independiente del dispositivo son:

Sincronización uniforme de los manejadores de dispositivos
Nominación de dispositivos
Protección de los dispositivos
Proporción de un tamaño de bloque independiente del dispositivo
Separación
Asignación del almacenamiento en dispositivos de bloque
Asignación y devolución de dispositivos dedicados
Informe de errores

Sincronización uniforme de los manejadores de dispositivos.

Consiste en ejecutar las funcione de E/S que son comunes a todos los dispositivos y proporcionar una interfase uniforme a nivel de usuario.

Nominación de dispositivos.

Es un aspecto fundamental de un sistema operativo la forma en que objetos como archivos y dispositivos de E/S son nombrados. El software independiente de dispositivo se hace cargo de describir los nombres simbólicos de dispositivos en el manejador adecuado.

Protección de los dispositivos.

Esta protección se relaciona íntimamente con la asignación de nombres. El manejo de estas protecciones varía entre sistemas operativos. Es la forma en que el sistema operativo controla el acceso a los dispositivos por parte de los usuarios o programas de usuario.

Proporción de un tamaño de bloque independiente del dispositivo.

Diferentes discos pueden tener diferentes tamaños de sector. Es tarea del software independiente del dispositivo ocultar este hecho y proporcionar un tamaño de bloque uniforme a los estratos superiores, por ejemplo, considerando varios sectores como un solo bloque lógico. En esta forma, los estratos superiores sólo trabajan con dispositivos abstractos que utilizan el mismo tamaño de bloque lógico, independiente del tamaño del sector físico. Análogamente, algunos dispositivos de caracteres entregan sus datos en un byte a la vez, mientras que otros entregan los suyos en unidades mayores. Estas diferencias también deben de ocultarse.

Separación.

La separación también es un aspecto que conviene considerar, tanto para dispositivos de bloque como para los de caracteres. Para los de bloque, el hardware suele insistir en la lectura y escritura de bloques enteros al mismo tiempo, pero los procesos de los usuarios tienen la libertad de leer y escribir en unidades arbitrarias. Si un proceso de usuario escribe medio bloque, el sistema operativo normalmente mantiene los datos en rotación en el interior hasta que se escriben los datos restantes, momento en el cual el bloque puede salir del disco. Para dispositivos de caracteres, los usuarios pueden escribir datos en el sistema más aprisa de lo que los pueden sacar y necesitan separación. La entrada del teclado también puede arribar antes de que se necesite, lo cual también requiere separación.

Asignación del almacenamiento en dispositivos de bloque.

Cuando se crea un archivo y se llena con datos, se tienen que asignar al archivo nuevos bloques de disco. Para realizar esta asignación, el sistema operativo necesita un mapa de bits de bloques libres por cada disco, pero el algoritmo para localizar un bloque libre es independiente del dispositivo y puede ejecutarse arriba del nivel del manejador.

Asignación y devolución de dispositivos dedicados.

Algunos dispositivos como las unidades de cinta magnética, pueden ser utilizados por un solo proceso en un momento dado; corresponde al sistema operativo examinar requisiciones de uso de dispositivo y aceptarlas o rechazarlas, según si el dispositivo solicitado está disponible o no.

Informe de errores.

El manejo de errores es efectuado por los manejadores. La mayoría de los errores son altamente dependientes del dispositivo, de manera que sólo el manejador sabe como manipularlos. Un error común es provocado por un bloque de disco que se ha dañado y no se puede leer. Después de que el manejador ha intentado leer el bloque cierto número de veces, se retira e informa al software independiente del dispositivo. Si el error ocurriera mientras se lee el archivo de un usuario, quizá baste informar el error al solicitante. No obstante, si ocurriera mientras se lee una estructura de datos importante del sistema, como el bloque que contiene el mapa de bits que muestra qué bloques están libres, el sistema operativo quizá tenga que imprimir un mensaje de error y terminar.

I.5.2.4 Software de E/S en el espacio del usuario.

Aunque la mayor parte del software de E/S está dentro del sistema operativo, una pequeña parte de él consta de bibliotecas

enlazadas con programas de usuarios y hasta programas completos que se ejecutan fuera del kernel. Las llamadas al sistema, entre ellas las llamadas al sistema de E/S, normalmente son realizadas por procedimientos de biblioteca. Mientras que estos procedimientos hacen poco más que colocar sus parámetros en el sitio adecuado para la llamada al sistema, hay otros procedimientos de E/S que efectivamente realizan el trabajo real. La figura I.5.2 resume el sistema de E/S, mostrando todo los estratos y funciones principales de cada uno de ellos.

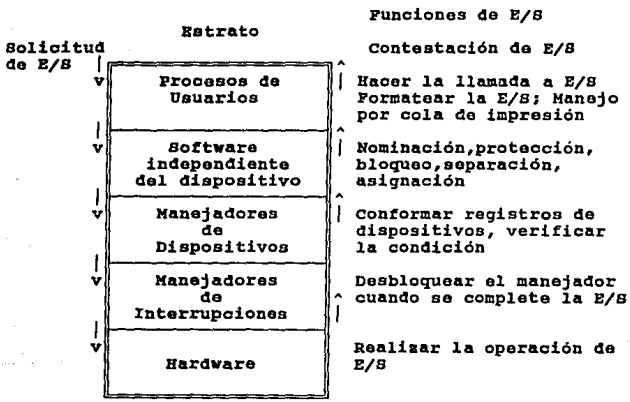


Fig. I.5.2 Estratos del sistema de E/S y las funciones principales de cada estrato.

I.5.3 Interbloqueo

Se dice que un proceso dentro de un sistema de multiprogramación está en un estado de interbloqueo si está esperando por un evento determinado que no va a ocurrir. Cuando los recursos son compartidos entre una población de usuarios, donde cada uno mantiene un control exclusivo sobre recursos determinados asignados a ese usuario, pueden producirse interbloqueos en los cuales los procesos de algunos usuarios nunca podrán llegar a su término. Se denominan recursos para éste fin a cualesquiera elementos que sólo pueden ser empleados por un sólo proceso en cualquier momento.

Coffman, Elphick y Shoshani (1971) establecieron las cuatro siguientes condiciones necesarias que deben darse para que se produzca un interbloqueo :

- **Condición de exclusión mutua.-** Los procesos reclaman control exclusivo de los recursos que piden.
- **Condición de espera por.-** Los procesos mantienen los recursos que ya les han sido asignados mientras esperan por recursos adicionales.
- **Condición de no apropiatividad.-** Los recursos no pueden ser extraídos de los procesos que los tienen hasta su completa utilización.
- **Condición de espera circular.-** Existe una cadena circular de procesos en la cual cada uno de ellos

mantiene uno o más recursos que son requeridos por el siguiente proceso de la cadena.

En principio, hay cuatro aspectos importantes en la investigación del interbloqueo, a saber :

- **Prevención del interbloqueo.**
- **Evitación del interbloqueo.**
- **Detección del interbloqueo.**
- **Recuperación del interbloqueo.**

En la **prevención del interbloqueo**, el interés se centra en condicionar un sistema para que elimine toda posibilidad de que éstos se produzcan. La prevención es una solución limpia en lo que respecta al interbloqueo propiamente dicho, pero los métodos de prevención pueden dar como resultado una pobre utilización de los recursos. Sin embargo, los métodos de prevención se utilizan con gran frecuencia.

En la **evitación del interbloqueo**, la meta es imponer condiciones menos estrictas que en la prevención, para intentar lograr una mejor utilización de los recursos. La evitación no precondiciona al sistema para que evite todas las posibilidades de que se produzca un interbloqueo, en su lugar, los métodos de evitación permiten la aparición del interbloqueo, pero siempre que se produce una posibilidad de interbloqueo, éste se esquiva.

Los métodos de **detección del interbloqueo** se utilizan en sistemas que permiten que estos ocurran, ya sea de forma voluntaria o involuntaria. La meta de la detección del interbloqueo es determinar si ha ocurrido, y detectar precisamente aquellos procesos y recursos implicados en él. Una vez determinado, éste puede ser eliminado del sistema.

Los métodos de **recuperación del interbloqueo** se utilizan para despejar interbloqueos de un sistema de manera que pueda seguir operando libre de ellos, y los procesos estancados lleguen a su terminación, liberando así sus recursos. En el mejor de los casos, la recuperación es un problema sucio y la mayoría de los sistemas se liberan del interbloqueo extrayendo en su totalidad a uno o varios de los procesos bloqueados. Entonces, los procesos extraídos se reinician en forma normal, desde el principio, con la pérdida de la mayor parte (si no de la totalidad) del trabajo previamente realizado por éstos.

Los interbloqueos pueden evitarse negando una o más de las condiciones necesarias para que éste exista. Sólo la condición de exclusión mutua no puede ser negada, ya que debe permitirse el uso mutuamente exclusivo de varios tipos de recursos del sistema. La condición de espera por, puede negarse al requerir que los usuarios pidan por adelantado todos los recursos que van a necesitar; el sistema o los concede como un grupo o niega la petición, hasta que queda disponible el grupo completo de recursos. La condición de no

apropiativo puede ser negada al requerir que un usuario que tiene algunos recursos y le es negada la petición de recursos adicionales, libere los que tiene, y pida de nuevo todos los recursos requeridos.

La condición de espera circular puede ser negada al requerir que todos los usuarios pidan los recursos en un orden predeterminado definido por la instalación.

I.6 Administración de equipo periférico.

Los equipos periféricos forman una parte vital en la configuración de cualquier equipo de computo, al grado de que una de las funciones principales de un sistema operativo, es la de controlar el flujo hacia estos dispositivos, verificar el desempeño que estos estan teniendo y realizar el control de errores de los mismos.

Los equipos periféricos representan la interfase entre el usuario y la computadora. Es, basicamente, la manera de indicar al equipo cuáles son los requerimientos de proceso, y la manera que éste tiene de mostrar los resultados a dichos requerimientos. Permiten, a su vez, el almacenamiento de grandes cantidades de información en forma permanente y confiable, ayudando aún a la memoria misma en sus funciones de almacenamiento temporal.

I.6.1 Historia.

Los sistemas operativos, así como los dispositivos periféricos, han sufrido una serie de cambios evolutivos llamados generaciones. Cada generación ha ido acompañada de reducciones sustanciales en los costos, el tamaño, la eficiencia y un incremento notable en velocidad y capacidad, tanto de sus componentes principales así como los componentes complementarios. Estos últimos, generación con generación han ido evolucionando e inovando las formas en que las instrucciones pueden ser enviadas al procesador y como el procesador puede mostrar la respuesta a dichas instrucciones.

A continuación revisaremos ligeramente, los cambios que fueron surgiendo en algunas de las generaciones de la computación, y las características de éstas.

I.6.1.1 Generación Cero (década de 1940).

Los primeros sistemas computacionales no poseían sistema operativo. Los usuarios tenían acceso a través del lenguaje de la máquina, ya que todas las instrucciones eran codificadas a mano. Los dispositivos periféricos eran rudimentario,y en la mayoría de los casos se limitaban a unidades con tarjeta de interruptores configurables para la entrada, y una impresora para la salida.

I.6.1.2 Primera Generación (década de 1950).

Este fué el comienzo de los sistemas de procesamientos por lotes, donde los trabajos se reunían por grupos para obtener más fluidez en la transición entre éstos. Una vez que el trabajo estaba en ejecución, dicho trabajo tenía control total del equipo. Al terminar el trabajo (ya sea normal o anormalmente), el control era devuelto al sistema operativo, el cual leía e iniciaba el trabajo siguiente.

Esta generación comenzó a hacer uso de lo que propiamente llamaron dispositivos periféricos, que básicamente se reducían a lectores de tarjetas perforadas, impresoras y unidades de almacenamiento magnético.

Los dispositivos eran siempre asignados por el usuario, haciendo una referencia específica a este y dejando el manejo de la asignación del mismo a la tarea de acuerdo con la demanda de procesos y la prioridad de estos.

I.6.1.3 Segunda Generación (primera mitad de la década de 1960).

Durante esta generación se desarrollaron los sistemas compartidos con multiprogramación, apareciendo los sistemas operativos interactivos, permitiendo que ahora varios procesos pudieran estar almacenados al mismo tiempo en la memoria, haciendo

al procesador capaz de cambiar rápidamente la atención entre un proceso y otro.

Junto con esto, otro de los adelantos importantes fué que surgieron máquinas con más de un procesador, lo que llevó a un desarrollo de los principios de multiproceso y a sistemas operativos que utilizáran completamente dichos recursos.

En esta generación de máquinas, también el manejo de los periféricos fué radicalmente diferente. Ahora el usuario solamente especificaba la tarea que utilizaría el dispositivo, y el sistema operativo era el encargado de localizarlo, asignarlo y liberarlo al término de la misma. Con el surgimiento de la interactividad con el equipo, surge también el uso de terminales y teletipos, reduciendo, significativamente, el tiempo de corrección a los procesos y el re-envio de estos para ser vueltos a procesar. Surgen también, en esta generación, los sistemas de tiempo real, donde el uso de los periféricos se hace crítico, pues se requiere de estos una respuesta inmediata a lecturas de, tal vez, hasta miles de puntos al mismo tiempo.

I.6.1.4 Tercera Generación (mitad década de 1960 a mitad década de 1970).

Estos sistemas se caracterizaron por ser de modos múltiples, es decir, eran capaces de soportar simultáneamente procesos por

lotes (batch), procesos interactivos, procesamiento en tiempo real y procesos en paralelo.

Esto trajo como consecuencia lógica, la creación de Sistemas Operativos complejos que requerían familiarizarse con un complicado lenguaje de control de trabajos, para poder especificar las tareas y los recursos requeridos. Surgen en paralelo controles más eficientes de colas para los dispositivos periféricos, así como algoritmos de administración de las misma que reducían enorme el tiempo de espera de una tarea.

I.6.1.5 Cuarta Generación (mitad década de 1970 hasta nuestros días).

Cobran un auge muy grande las microcomputadoras y las minicomputadoras, con un poder de computo mayor y un tamaño y costo mucho menor.

Surgen también las redes de computadoras, tanto locales como remotas y los programas que permiten compartir los dispositivos periféricos de unas y otras. Surgen programas que distribuyen información y la procesan independientes o concentrada.

Se hace cada vez más frecuente el uso de Menues para facilitar el control de la administración y el movimiento a través de las diferentes opciones ofrecidas.

Crece el número de dispositivos periféricos que pueden ser conectados en un equipo, así como las opciones para compartir los mismos con otras computadoras. La figura I.6.1 muestra los dispositivos periféricos más comunes.

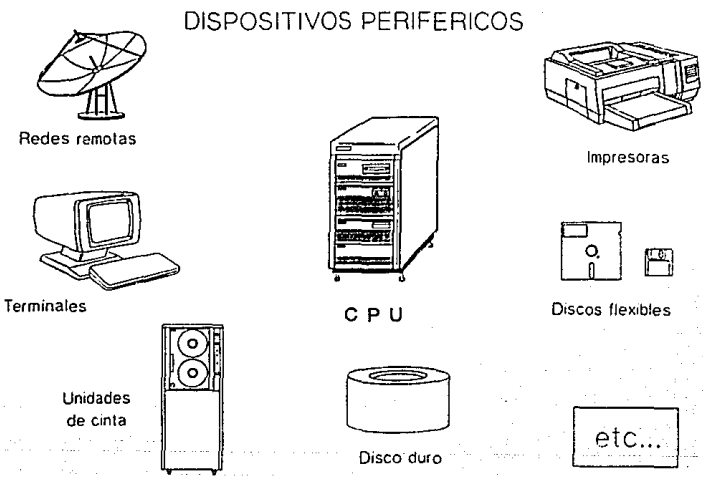


Fig. I.6.1 Diferentes dispositivos periféricos

I.6.3 Definición.

Dentro de un equipo de cómputo, los dispositivos periféricos pueden dividirse en dos categorías:

I.6.3.1 Dispositivos de bloques.

Se define como un dispositivo que utiliza un bloque de tamaño fijo para almacenar o procesar información, cuya principal propiedad consiste en la posibilidad que ofrecen de leer o escribir en forma independiente a los demás bloques, es decir, en forma aleatoria.

I.6.3.2 Dispositivos de caracteres.

Se define como un dispositivo que entrega o acepta un flujo de caracteres, sin importar de cuál estructura de bloque se trate. Entre sus características principales, se encuentra el que no es direccionables, y que no tiene ninguna operación de localización aleatoria, es decir opera secuencialmente en todos los casos.

Es necesario mencionar que esta clasificación tiene dificultad de ubicar algunos de los dispositivos, pues su funcionamiento no es similar a ninguno de los casos mencionados anteriormente, como el reloj de la máquina, por ejemplo, que no direcciona bloques, ni acepta flujos de caracteres de ningún tipo.

La figura I.6.2 muestra la clasificación de algunos dispositivos.

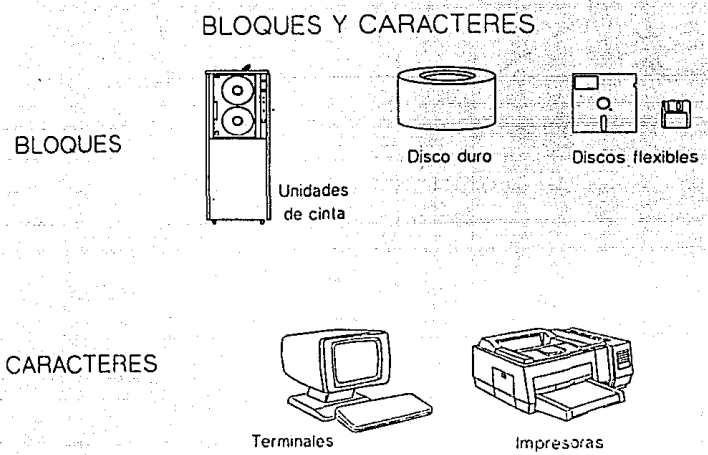


Fig. I.6.2 Dispositivos de bloqueo y caracteres

I.6.3.3 Manejadores de dispositivos.

Por otra parte, sin importar cual es el tipo en el que pueda ser identificado un dispositivo, existe dentro del sistema operativo un conjunto de programas que manejan su operación, los cuales son llamados **drivers (manejadores)**. Existe un manejador por cada clase diferente de dispositivo y aún puede haber un manejador para un solo dispositivo, como es el caso del ratón.

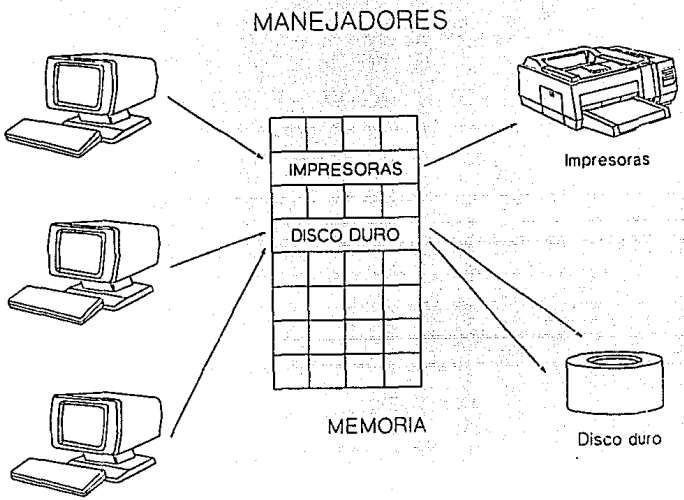


Fig. I.6.3 Funcionamiento de un manejador

La función principal del manejador de un dispositivo consiste en aceptar solicitudes de servicio del software, a fín de emitir los comandos para su operación y verificar que se complete la requisición normalmente. Por ejemplo, en una solicitud común de lectura del bloque n. En el caso en el que el manejador esté ocioso en el momento en que llega la requisición, la atención será inmediata se iniciará una busqueda en la tabla del disco para saber el número de cilindro y sector donde se buscará el dato, se verificará si el motor de la unidad de disco esta disponible, se

determinará la posición actual de la cabeza para poder hacer el ajuste e iniciar la lectura, es decir, se ejecutan cierto número de comandos con un órden ya establecido para satisfacer el requerimiento. Sin embargo, en el caso en el que el dispositivo esté ocupado con alguna solicitud, normalmente meterá la nueva solicitud en una lista de espera de requisiciones pendientes (cola), con la cual trabajará tan pronto le sea posible.

Algunos controladores pueden hacerse cargo sólo de un comando a la vez, mientras que otros (llamados inteligentes) pueden aceptar una lista enlazada de requerimientos, que después ejecutan por sí mismos sin ayuda del procesador principal del equipo.

I.6.3.4 Estado del dispositivo.

Después de que el comando se ha terminado el manejador del dispositivo puede realizar cualquiera de éstas dos situaciones:

- Esperar hasta que el controlador del dispositivo le envíe otra tarea, lo que lo hace entrar en un estado de bloqueo que se termina al recibir una interrupción.
- La operación finaliza sin retraso alguno, de manera que el manejador nunca entra en un estado de bloqueo.

Un ejemplo de ésto es el deslizamiento de la pantalla (**scroll**) en algunas terminales, el cual requiere que sean escritos algunos **bytes** en los registros del controlador, pero no necesita de ningún

movimiento mecánico de manera que toda la operación se puede completar en unos cuantos microsegundos.

I.6.4 Procesos Residentes.

Otra de las formas en que el manejador atiende las tareas, es por medio de un proceso especial residente en la memoria y un directorio especial creado en el disco duro, llamado directorio de manejo de la cola de impresión. Para imprimir un archivo, existe un proceso que genera todo el archivo que se imprimirá y lo coloca en el directorio de manejo de la cola. En ese momento, el proceso que está residente en memoria y que es el único que tiene autorización para usar el archivo especial de la impresora, se encarga de imprimir los archivos que encuentre en el directorio.

El propósito de restringir el acceso al directorio especial por parte de los usuarios, es eliminar el problema de que alguien lo mantenga abierto por un periodo de tiempo largo que impide el trabajo del programa residente. Este tipo de manejo de dispositivos es muy usado en impresoras, correo electrónico y transferencias de archivos dentro de una red.

I.6.5 Política de Asignación de Recursos.

Cuando el manejador hace uso de uno o varios dispositivos, cualquiera de éstos se puede utilizar para satisfacer la

solicitudes enviadas, lo que hace necesario una política de asignación del recurso, dado que, un recurso sólo puede ser empleado por un sólo proceso en cualquier momento. Dicha política establece que es necesario respetar lo siguiente:

- Solicitar el recurso y bloquearlo.
- Utilizar el recurso y satisfacer la tarea.
- Liberar el recurso, quitando el bloqueo.

Si algún recurso no está disponible cuando es requerido, el proceso que lo solicita necesitará esperar. Dependiendo del sistema operativo el proceso tratará de ser asignado a cualquiera de los recursos que sea liberado primero y permanecerá intentándolo hasta obtener una respuesta, mientras que en otros sistemas la requisición fallará enviando un código de error y dejando que el usuario decida la acción siguiente.

I.6.6 Manejo del Spool.

Muy estrechamente relacionado con la política de asignación de un recurso, se encuentra un metodo para conseguir una óptima operación de periférico de baja velocidad llamado operación simultánea de periféricos en línea (**spool: simultaneus peripheral operation on line**), con lo que se consigue que un dispositivo de alta velocidad, como es un disco duro, se sincronice por medio del programa residente en memoria, con un dispositivo de baja velocidad compartiendo información.

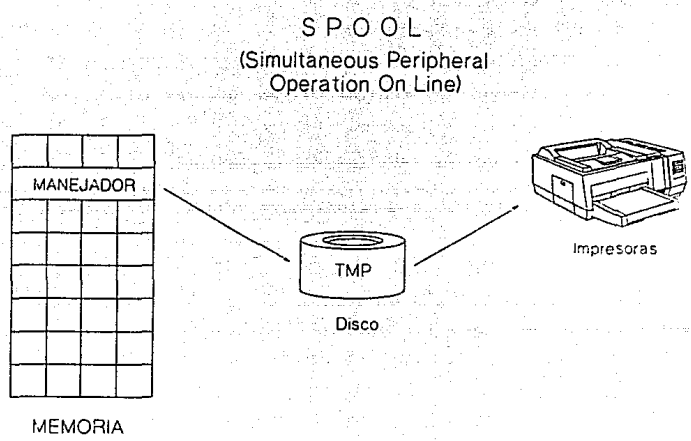


Fig. I.6.4 Spool de impresión

La manera en que trabaja es que en lugar de escribir las líneas directamente en la impresora, por ejemplo, las líneas son escritas sobre el disco con lo que el proceso solicitante puede terminar mas rapidamente, y los procesos que dependen de el pueden iniciarse antes, pues cuando la impresora se encuentre disponible leerá la información dejada sobre el disco para ese propósito. Es decir , si un programa está enviando líneas a una impresora, dicho programa necesitará esperar la señal de que la línea fue impresa antes de enviar la siguiente línea, con lo que la ejecución del

programa será lenta. Para asegurar la ejecución del programa dichas líneas son enviadas a un dispositivo más rápido, como una unidad de disco, donde son almacenadas temporalmente hasta que puedan ser impresas. El nombre spool es bastante apropiado para éste procedimiento ya que se parece mucho a un hilo enrollado en un carrete que puede ser desenrollado cuando sea necesario.

I.6.7 Problemas de interbloqueo.

Los sistemas de spool están propensos al interbloqueo, lo que significa que se estorban unos a otros. Por ejemplo, en algunos sistemas los datos completos de salida de un programa deben de estar disponibles antes de comenzar el proceso de impresión. De ésta manera, varios trabajos parcialmente completos que generan líneas de impresión a un archivo de spool pueden estorbarse si el espacio disponible se llena antes de completarse alguno de éstos trabajos. La recuperación de éste interbloqueo puede implicar reinicializar el sistema con la pérdida de todo el trabajo realizado hasta el momento. Los sistemas actuales son mucho más sofisticados, pues permiten que la impresión comience antes de terminar un trabajo, de manera que un archivo de spool lleno, o casi lleno, pueda ser vaciado o despejado en cierta medida aunque un trabajo se encuentre todavía en ejecución. En muchos sistemas, la asignación del espacio de spool se ha hecho más dinámica de forma que si el espacio existente ha empezado a llenarse, pueda disponerse de más espacio.

I.6.8 Gráficas de Asignación de Recursos.

Para facilitar la detección de interbloqueos, se utiliza una anotación popular en la cual una gráfica dirigida indica las asignaciones y peticiones de recursos. Los procesos son representados por cuadros y las clases de recursos son representadas por círculos grandes. Los círculos pequeños, dibujados dentro de los grandes representan el número de recursos idénticos dentro de cada clase. Por ejemplo, un círculo grande llamado D1 que contiene tres círculos pequeños indica que hay tres dispositivos equivalentes del tipo D1, disponibles para su asignación al sistema. La figura I.6.5 indica las relaciones que pueden ser encontradas en una gráfica de asignación y petición de recursos.

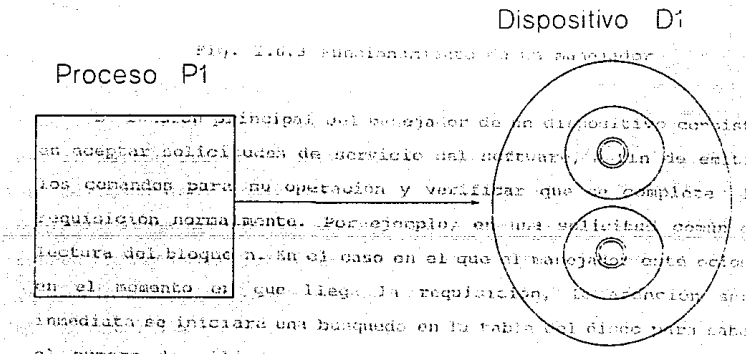


Fig I.6.5 Requerimientos de un dispositivo

En la figura I.6.5, el proceso P1 está pidiendo un dispositivo del tipo D1. La flecha de P1 toca solamente el extremo del círculo grande, indicando que esta petición está siendo considerada.

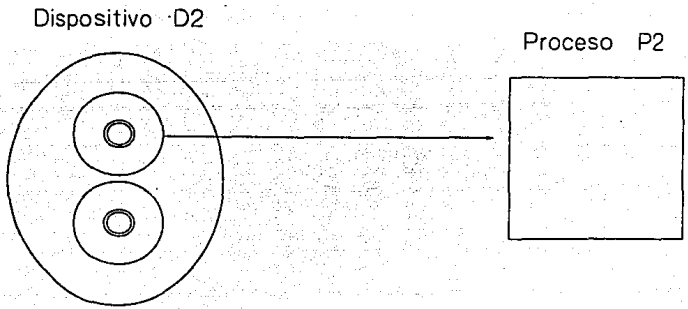


Fig. I.6.6 Dispositivo asignado

De la figura el proceso P2 le ha sido asignado un dispositivo del tipo D2, del cual existen dos equivalentes. La flecha sale del círculo pequeño indicando que el recurso ha sido asignado.

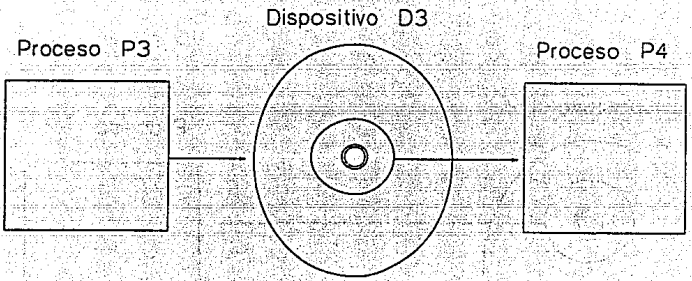


Fig. I.6.7 Asignación y solicitud de un dispositivo

La figura I.6.7 indica una situación próxima a un interbloqueo, ya que el dispositivo R3, que ha sido asignado al proceso P4, está siendo pedido por el proceso P3.

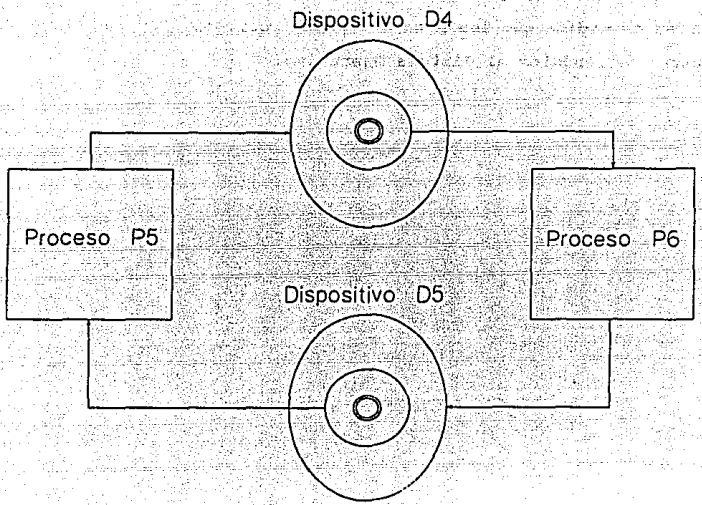


Fig. I.6.8 Deadlock (Interbloqueo)

La figura I.6.8 indica un pequeño sistema deadlock (interbloqueo) en el cual el dispositivo R4 asignado al proceso P6, está siendo pedido por el proceso P5, al cual se le ha asignado el dispositivo R5, que a su vez está siendo pedido por el proceso P5.

Esta es una espera circular típica de un sistema interbloqueado. Las gráficas de asignación y petición de recursos cambian a medida que los procesos piden recursos, los adquieren o acaban cediéndolos al Sistema Operativo.

II. EL SISTEMA OPERATIVO UNIX SYSTEM V

II.1 Introducción a UNIX y al núcleo del sistema (Kernel)

II.1.1 Núcleo y superestructura del sistema operativo UNIX.

Al analizar los sistemas operativos resulta conveniente separar aquellas partes que proporcionan el medio del usuario de las relacionadas en forma más directa con el hecho de compartir recursos y la interfase con el Hardware de la máquina, como el manejo de interrupciones, almacenamiento, control del procesador, etcétera.

Esta segunda área se describirá como núcleo y al resto se le denominará superestructura.

En esta definición la superestructura abarca aquellas partes del sistema operativo que proporcionan la base de servicios al usuario (sistema de archivo, lenguaje de comandos, etc.), pero no incluye Software del sistema.

La distinción entre núcleo y superestructura resulta muy útil; permite describir un sistema operativo de aplicación general dividiéndolo en dos partes con una interfase evidente; siempre y cuando se comprenda que hace el núcleo, es posible seguir una explicación de la superestructura sin necesidad de saber claramente las categorías de los sistemas operativos. Ver figura. II.1.1.

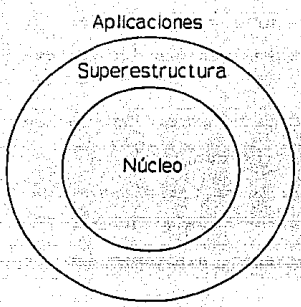


Fig. II.1.1 Núcleo y superestructura

Las principales posibilidades ofrecidas por la superestructura son el manejo de datos y control de trabajos. Una parte importante del manejo de datos es el control de los dispositivos de entrada/salida. A menudo resulta conveniente particionar la superestructura en cierto número de subsistemas:

- Uno o más supervisores de fila de trabajos por lote
- Un manejador de sistemas de archivo
- Un supervisor de múltiple acceso
- Un supervisor de procesamiento de transacciones
- Otros.

Debe considerarse que estos subsistemas corren en forma concurrente y que cada uno de ellos estará constituido por componentes que corren a su vez en forma concurrente.

II.1.2 Estructura de UNIX

El sistema operativo UNIX es un sistema de tiempo compartido, y por lo tanto, multiusuario, en el que existe portabilidad para la implantación de distintas computadoras. Está formado por una serie de elementos que pueden representarse en forma de capas concéntricas donde, en primer lugar, alrededor del hardware de la máquina se encuentra el núcleo (kernel), que interactúa, directamente con el hardware, aislando a éste de los usuarios, además de adaptar el resto del sistema operativo a la máquina debido a la portabilidad que existe en el mismo. Ver figura II.1.2.

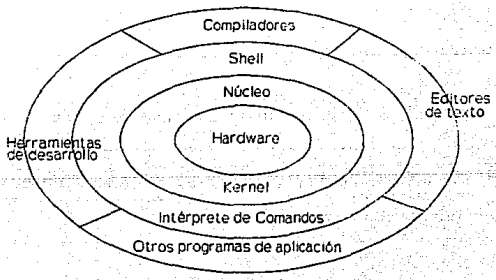


Fig. II.1.2 Capas concéntricas de UNIX

En una segunda capa se encuentran los comandos, que no son otra cosa que la interfase entre los programas de aplicación y el núcleo del sistema operativo. La última de las capas contiene los programas de aplicación.

El sistema operativo UNIX se compone de bloques funcionales que se encuentran representados en la figura II.1.3.

Para entender mejor la estructura de UNIX se deben describir los subsistemas que lo conforman, separando y analizándolos funcional y arquitectónicamente. Es muy importante tomar en cuenta la diferencia entre estas dos perspectivas; por una parte el aspecto funcional describe las capacidades de cada subsistema, por otro lado, el punto de vista arquitectónico determina la forma en que los módulos del subsistema están trazados y las relaciones con otros módulos.

En el modelo del sistema UNIX existen dos entidades básicas que conforman los conceptos centrales de los subsistemas: Archivos y Procesos. La figura II.1.3 sirve como ayuda para "ver lógicamente" éstos conceptos.

Esta figura muestra los tres niveles del sistema: nivel usuario, nivel Kernel y nivel hardware; los llamados al sistema y la interfase de librerías representan la frontera entre los programas del usuario y el Kernel.

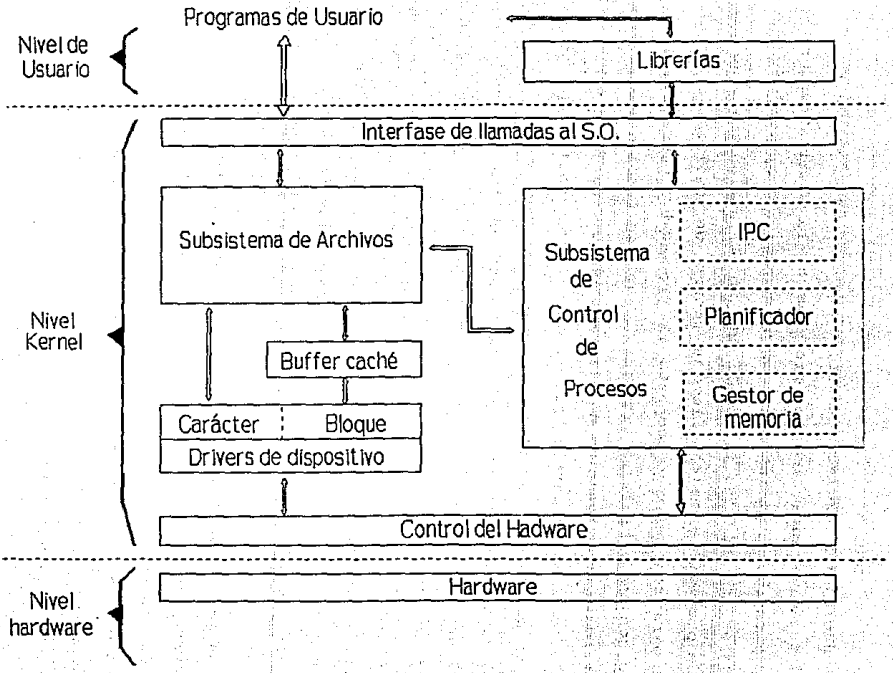


Fig. II.1.3 Bloques funcionales de UNIX

II.1.3 Funciones del núcleo

El núcleo del sistema (KERNEL) tiene un doble papel; simula actividad concurrente para los procesos de la superestructura y también proporciona la interfase con el hardware, recibiendo todas las señales provenientes del mundo externo. El núcleo mantiene una lista de procesos actuales, lleva un control de aquellos que están libres para continuar y comparte el procesador entre ellos de acuerdo con una estrategia escogida. El núcleo también administra la estrategia de asignación de almacenamiento y es responsable de controlar cómo se comparten los recursos. Esto significa que todas las operaciones que implican procesos son controladas por el núcleo, este representa solamente una pequeña porción del código de todo el sistema operativo, pero se encuentra entre los códigos de más amplio uso. Por esta razón el núcleo suele permanecer en el almacenamiento primario, mientras otras porciones del sistema operativo son transportadas de un lado a otro, entre el almacenamiento primario y secundario, según las necesidades. Ver figura II.1.4.

Una de las funciones más importantes del núcleo es el proceso de interrupciones. En los grandes sistemas de múltiples usuarios hay una corriente constante de interrupciones dirigidas al procesador. La rápida respuesta a estas interrupciones es esencial para mantener bien utilizados los recursos del sistema, y para proveer tiempos de respuesta aceptables a los usuarios interactivos

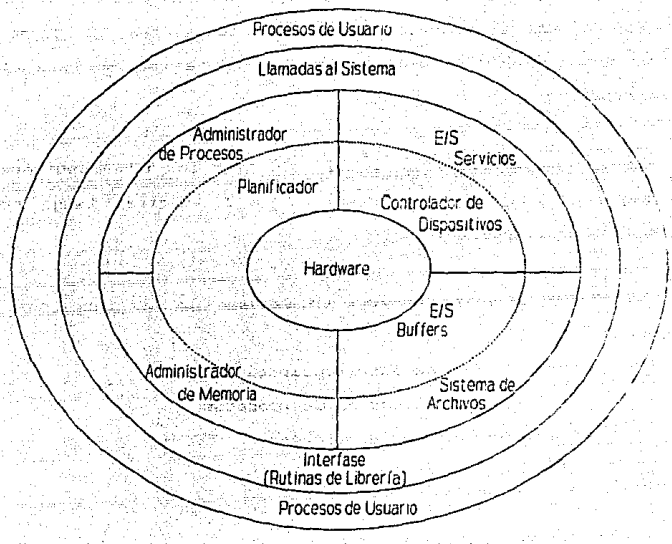


Fig. II.1.4 Funciones del Kernel

El núcleo inhabilita las interrupciones mientras responde a una interrupción; las interrupciones son habilitadas de nuevo después de completar el proceso de una interrupción. Con una corriente constante de interrupciones es posible que el núcleo tenga interrupciones inhabilitadas durante un periodo de tiempo grande; esto se mejora si el núcleo realiza la "mínima" cantidad de

proceso posible en cada interrupcion, y después pasa el resto del proceso de cada interrupción a un proceso del sistema apropiado que puede seguir operando mientras se habilita el núcleo para otras interrupciones.

Esto da como resultado poder habilitar las interrupciones en un mayor porcentaje de tiempo, y que el sistema tenga mejor capacidad de respuesta.

A continuación se muestra un resumen de las funciones del núcleo:

- Manipulación de interrupciones
- Creación y destrucción de procesos
- Cambio de estados de procesos
- Ejecución de procesos
- Suspensión y reanudación de procesos
- Sincronización de procesos
- Comunicación entre procesos
- Manipulación de bloques de control de proceso
- Soporte de las actividades de entrada/salida
- Soporte de la asignación y designación de almacenamiento
- Soporte del sistema de archivos
- Soporte del mecanismo de llamada/regreso de procedimientos
- Soporte de ciertas funciones contables del sistema.

II.1.4 Modos de operación

Como hemos visto UNIX ha sido implementado bajo un modelo arquitectónico que se basa en descripciones funcionales, si observamos la figura II.1.3 y eliminamos el nivel más inferior, es decir, el nivel hardware tendremos dos niveles propios del sistema operativo. Estos niveles representan los dos espacios más importantes que posee UNIX.

II.1.4.1 Espacio de usuario.

Es el espacio en memoria donde los procesos de los usuarios son ejecutados, consiste de la memoria situada al final del espacio asignado al Kernel e incluye el resto de la memoria disponible. Este espacio de memoria está protegido de tal forma que el sistema verifica que un usuario no interfiera con las tareas de otro usuario, evitando así daños en los procesos. Un proceso operando en este espacio se le conoce como Proceso de Usuario, es decir está operando en Modo Usuario.

II.1.4.2 Espacio de Kernel.

Es el espacio en memoria donde todos los servicios del Kernel son ejecutados vía los procesos del Kernel. Cualquier proceso que requiera uso del código del Kernel opera en este espacio y es conocido como Proceso de Kernel. El espacio de Kernel es una área

privilegiada a la cual los usuarios no tienen acceso, tampoco tienen acceso a las instrucciones directas para manejo de dispositivos. Sin embargo, los procesos del Kernel si tienen acceso. Similarmente el proceso de Kernel puede modificar el mapa de memoria, una operación frecuentemente requerida para la ejecución de procesos. Un proceso de usuario activa a un proceso Kernel cuando se efectúa un llamado al sistema, en ese momento se ejecuta el código del Kernel, es decir se está operando en Modo Kernel.

II.1.4.3 Flujo de datos entre el espacio de usuario y espacio de Kernel.

Como el espacio de usuario y de Kernel son diferentes direcciones de memoria, el esquema de movimiento de los datos resulta muy importante. El dato puede ser representado por un bit, un byte, un entero, un arreglo o una cadena de caracteres.

Cuando existe una llamada al sistema, el argumento de la llamada corresponde a una Rutina del Kernel (cuya identificación se relaciona con un número en la tabla de llamadas al sistema), este argumento es pasado del espacio de usuario al espacio de Kernel. La figura II.1.5 muestra este evento.

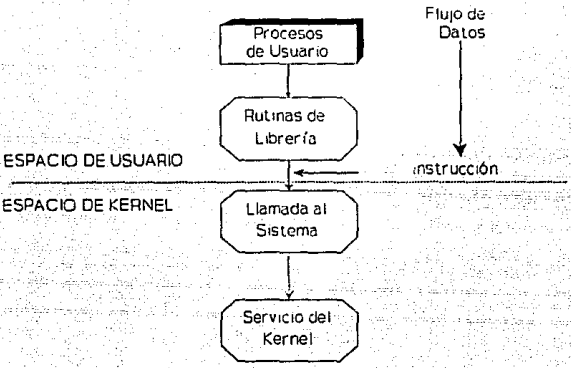


Fig. II.1.5 Relación entre espacio de usuario y espacio Kernel

II.1.4.4 Opciones de acceso al Kernel.

Las opciones de acceso al Kernel de UNIX, descrito en la Figura II.1.6, permite tener a nuestra disposición las funciones y servicios que ofrece el Kernel. Existen 3 tipos diferentes de estos accesos:

1. Llamadas al sistema realizada por un programa de aplicación del usuario o por una utilería del sistema.
2. Un requerimiento de servicio de hardware desde los manejadores de dispositivos.

3. Condiciones de error.

Estos 3 accesos son mecanismos de interrupción a la actividad del Kernel.

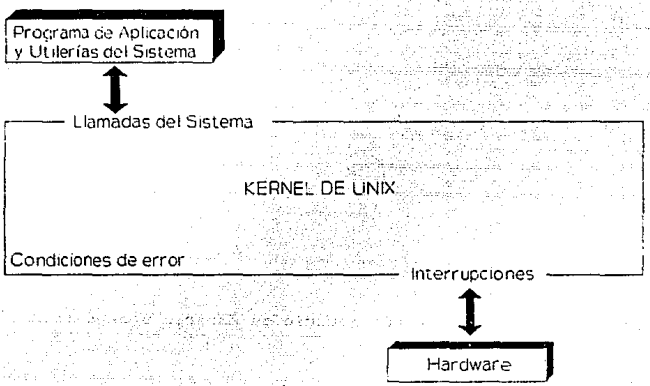


Fig. II.1.6 Opciones de acceso al Kernel

II.1.4.5 Estados de Proceso y Transiciones

El tiempo de vida de un proceso puede ser dividido en estados que básicamente son los siguientes:

1. Proceso ejecutándose en modo usuario
2. Proceso ejecutándose en modo Kernel

3. Proceso no ejecutándose, en espera de ser atendido
4. Proceso "durmiendo", esperando a que se complete una operación de entrada/salida.

Una transición es el evento que produce un cambio de estado en los procesos. La figura II.1.7 muestra un diagrama de estados, y transiciones.

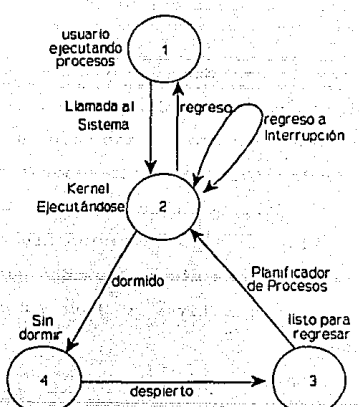


Fig. II.1.7 Estados de procesos y transiciones

II.1.5 Objetivos de diseño

Son dos los aspectos del origen de los sistemas operativos UNIX los que más han contribuido a su diseño exclusivo:

- La primera versión de un sistema UNIX se construyó en sólo unos cuantos años de trabajo
- Los que construyeron el sistema UNIX fueron los principales usuarios del sistema

La coherencia de los sistemas UNIX es atribuible a estos factores. El primer objetivo de diseño fue conservar la sencillez del sistema operativo y que apoyara tan solo una cantidad mínima de funciones. El segundo objetivo fue la generalidad, un solo método debería servir a diversos propósitos, esta generalidad se manifiesta en los sistemas UNIX en varias áreas por ejemplo:

- El sistema usa las mismas llamadas para leer (o escribir archivos, dispositivos y buffers de mensajes entre procesos.
- Se aplican los mismos mecanismos de nomenclatura, formación de seudónimos y protección de acceso a los archivos de datos, directorios y dispositivos.
- Se usa el mismo mecanismo para atrapar interrupciones de software y trampas del procesador.

El tercer objetivo es el de crear un ambiente en el cual las grandes tareas puedan ser cumplidas combinando pequeños programas

existentes, en vez de desarrollar nuevos programas.

II.1.6 Generalidades del Sistema Operativo.

Varias características han contribuido a la extensa difusión de UNIX; entre ellas cabe destacar:

- Flexibilidad para adaptarse a aplicaciones diversas
- Potencia y capacidad
- Portabilidad que permite usarse en casi todas las computadoras a partir de un cierto tamaño
- Compartición de archivos con el sistema operativo MS-DOS

Aunque los requerimientos de 4 megabytes de memoria RAM y unos 40 Mb de espacio en disco duro que necesita UNIX para mejorar sus más de 400 comandos y 2500 archivos le hace poco accesible a pequeños usuarios, cada vez está siendo más usado por profesionales que se sirven de la computadora como ayuda o preparación de sus trabajos y desde luego es bastante frecuente encontrarlo en empresas de cualquier tipo.

En el desarrollo de UNIX se partió de que el apoyo al programador ha de ser total y por ello además del sistema operativo propio de UNIX posee un software de apoyo potente, de gran calidad y fácil manejo. En el software se pueden encontrar dos vertientes; la destinada a la oficina que posee procesadores de textos, correo eléctrico, redes de comunicación y bases de datos, y la que se

destina a la programación, que comprende la posibilidad de ejecutar todos los lenguajes principales, la detección de errores, o paquetes de usos diversos.

El sistema operativo está escrito en C en lugar de ensamblador como la mayoría de los sistemas; esto le permite una fácil lectura y una completa adaptación a las diferentes computadoras.

Como otros sistemas, consta de dos partes ; el núcleo y los programas del sistema.

El CPU está organizado de forma que favorece los procesos interactivos, de tal suerte que para la planificación de procesos se usan algoritmos basados en prioridades..

Los archivos se organizan en directorios estructurados en árbol (sistema jerárquico de procesos y archivos) y se manejan mediante bloques de datos que tienen dos tamaños en el disco.

El tamaño grande guarda todos los bloques del archivo menos el último que es menor para disminuir la fragmentación interna. Existen dos tendencias en la planificación de la memoria dependiendo de las versiones de UNIX. En unas versiones se utiliza el intercambio (swapping) entre los diversos procesos. Los procesos que se elijan como víctimas para ser descargados de la memoria. Cuando es necesario espacio para nuevos procesos se eligen por

estar ociosos, llevar mucho tiempo en memoria o ser grandes, y se cargan sin llevar mucho tiempo fuera o si son pequeños.

En las versiones de Berkeley se utiliza un sistema de memoria virtual con paginación bajo solicitud.

UNIX no está ligado a una marca comercial fabricante de computadoras.

Posee una interfase con periféricos constante y uniforme.

II.1.7 Historia y versiones.

El sistema operativo UNIX ha evolucionado desde hace más de 20 años de forma continua hasta convertirse en uno de los más importantes en la actualidad.

La primera versión de UNIX surgió en 1964 cuando Ken Thompson, miembro del equipo de los laboratorios Bell, intentó utilizar una computadora llamada PDP-7 para ejecutar un programa que habitualmente era ejecutable en una máquina grande bajo el sistema operativo Multics. La adaptación de dicho programa a la PDP-7 necesitó un nuevo sistema operativo que Thompson bautizó como UNIX.

Pronto se le unió en el trabajo Dennis Ritchie y el UNIX se comenzó a utilizar en los laboratorios Bell en 1971, así se fue

adaptando a otras máquinas PDP entre ellas la PDP-11/70 que se utilizaba en las universidades norteamericanas con lo que se impulsó extraordinariamente su desarrollo.

Al principio, el sistema operativo UNIX se escribió en ensamblador pero pronto fue reescrito en lenguaje C. La primera versión disponible para el usuario fue la llamada versión 6 en 1976. En 1978 se distribuyó la versión 7 con diversas mejoras sobre la versión anterior.

Paralelamente apareció el PWP/UNIX que desarrolló otro equipo diferente al de Thompson y Ritchie, la universidad de Berkeley presenta la versión 4.1 BSD Y 4.2 BSD para computadoras VAX.

Tras la versión 7 la compañía Bell sigue el desarrollo de UNIX con dos versiones denominadas Sistema III y Sistema V de las cuales hace diferentes revisiones.

Microsoft crea en 1980 una nueva versión denominada XENIX para microprocesadores de 16 bits. En Francia el Instituto Nacional de Investigaciones desarrolla el sistema SQL compatible con UNIX.

En la mayoría de los casos, cada fabricante tiene su propia versión de UNIX, entre ellas citaremos:

Digital Equipment Corporation	ULTRIX
IBM	AIX
Data General	DG/UX
National Semiconductor	GENIX
Gould Concept 32/6750	UTX
Hewlett Packard	HP-UX

Tabla II.1.1 Sistemas operativos en diferentes equipos

II.1.8 Sistemas UNIX estándar.

Los sistemas UNIX estándar son sistemas operativos multiprogramables de tiempo compartido, diseñados principalmente para las computadoras PDP-11/34. 40, 45 y 70 de DEC. Proporcionan el ambiente de un centro de computación a pequeña escala que soporta hasta 40 usuarios. Los sistemas UNIX estándar proporcionan un sistema de archivos jerárquico con protección total de volúmenes desmontables, independencia de dispositivos y características que facilitan la sencillez de la programación.

Los sistemas UNIX permiten a los usuarios el encaminar la salida de un programa directamente a la entrada de otro. Esto facilita la solución de los problemas de programación a gran

escala, al componer pequeños programas disponibles, en vez de desarrollar programas completamente nuevos.

Los sistemas UNIX estándar se distribuyen con una serie de programas empaquetados que incluyen un editor de textos, un interpretador de lenguaje de mandatos programables, varios compiladores, un ensamblador, un editor encadenador, depuradores, formateadores de documentos (con posibilidades matemáticas), programas de procesamiento de textos, un dispositivo de clasificación, programas administrativos, y de mantenimiento, bibliotecas normales del sistema y rutinas del usuario y un paquete de juegos.

Mucha de la documentación está disponible en línea para los usuarios interactivos. Una configuración típica basada en UNIX contiene:

- Una unidad central con el sistema
- Una consola
- Terminales
- Líneas de comunicación
- Impresoras.

II.1.9 Sesión UNIX.

El trabajo de un usuario en el sistema operativo UNIX se organiza por sesiones. Una sesión UNIX comprende todo el trabajo

realizado por el usuario en la computadora, desde que se identifica hasta que se despide.

Vamos a exponer brevemente cuales deben ser las acciones que ha de llevar a cabo el usuario para dar comienzo a una sesión, operar a lo largo de ella y finalizarla correctamente. La figura II.1.8 muestra un ejemplo de inicio y terminación de una sesión de usuario, en el que además el sistema pide la correspondiente palabra o clave de usuario para proteger el acceso de otros usuarios. Una vez reconocida la clave por el sistema, dará comienzo la sesión.

Aparece en la terminal el carácter que indica espera de mandato o comando, que será interpretado por el intérprete de comandos o Shell y que se denomina "prompt", representándose en general por los caracteres $ o %. Cuando se termina el trabajo que se pretendía hacer, basta con despedirse de la sesión por medio del carácter de control Ctrl-d (^D).

Una vez iniciada una sesión en UNIX y estando presente el prompt $, el intérprete de comandos Shell está preparado para recibir un comando.

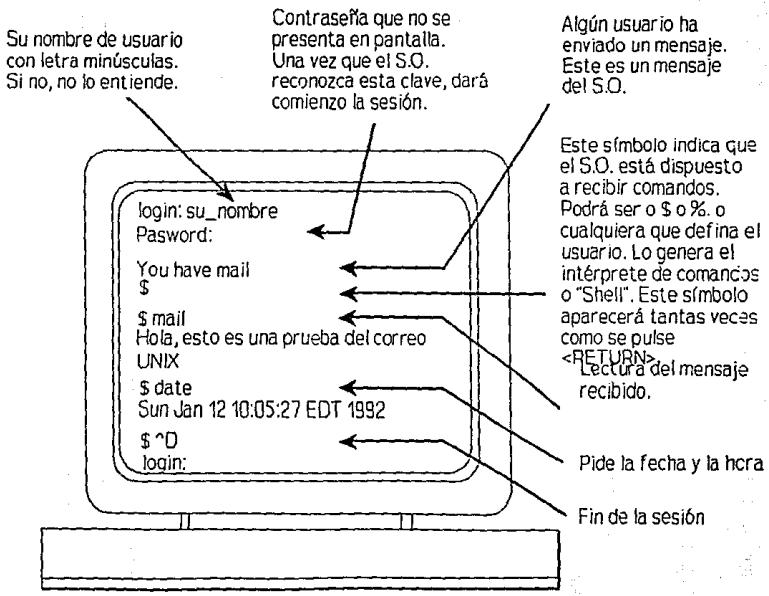


Fig. II.1.8 Inicio y fin de una sesión UNIX

II.1.10 El Shell

Se conoce con el nombre de Shell al programa que interpreta las peticiones del usuario para ejecutar comandos, utilerías o programas; es decir es el intérprete de comandos del sistema operativo y es la interfase entre el usuario y el sistema.

El Shell es un programa del Sistema Operativo, pero no forma parte del núcleo del mismo. Se ejecuta cada vez que un usuario se identifica ante el sistema y comienza una sesión.

Es también un lenguaje de programación que soporta todas las estructuras propias de los lenguajes modernos. Además permite la utilización de las instrucciones del Sistema Operativo de control de procesos, interrupciones y utilerías para diseñar programas de comandos por el usuario. Existen varios tipos de Shell con diferentes características:

Bourne Shell	Es el intérprete de comandos básico
C-Shell	Es el intérprete de comandos creados por Berkeley para el sistema operativo BSD y para XENIX, un poco más completo que el anterior. Su programación es prácticamente lenguaje C.
Korn Shell	Se basa en los dos anteriores, siendo compatible con el Bourne en un 95%. Añade posibilidades de programación avanzada, facilidades aritméticas y mayor rapidez de ejecución.

Tabla II.1.2 Interpretes de comandos

II.1.11 Herramientas de desarrollo de Software.

El Sistema Operativo UNIX proporciona una serie de herramientas o programas de utilidad para el desarrollo de Software. Estas herramientas son:

VI: Es un editor de pantalla que tiene como misión principal la creación de archivos fuente.

CC: Es el compilador del lenguaje C.

LINT: Es un analizador sintáctico de lenguaje C más estricto que el compilador.

AR: Es un programa para mantener librerías o bibliotecas de programas objeto o cualquier otro tipo de archivos.

MAKE: Es un programa de utilidad de UNIX que acepta especificaciones de las dependencias existentes entre módulos de un programa y establecer mecanismos para mantener las versiones, trasladando al resto de los módulos los cambios que se realicen en uno de ellos.

SCCS: (Sistema de Control de Código Fuente): Son un conjunto de utilidades para el desarrollo de aplicaciones con muchos módulos que se desarrollan en distintos equipos.

AUK: Es el intérprete de un lenguaje de programación cuyo fin es tabular, dar formato y preprocesar archivos para crear un analizador léxico que basa su análisis en ciertas reglas de lenguaje definidas por el usuario.

YACC: Es un analizador semántico cuya misión es transformar un

archivo con instrucciones de un lenguaje especificado, que cumpla con las reglas semánticas dadas, en un módulo compatible en C.

II.1.12 Administración del Sistema

En computadoras que funcionan bajo el Sistema Operativo UNIX, existe un usuario que se distingue de los demás por ser el encargado de realizar la administración del sistema. Las funciones propias del administrador son:

- Actualización y mantenimiento de usuarios
- Realización periódica de copias de seguridad
- Suministro de soporte técnico al resto de los usuarios
- Otros.

En UNIX existe un directorio de uso exclusivo del administrador del sistema donde se encuentra una serie de comandos para la realización de dichas funciones, que no pueden ser utilizadas por el resto de los usuarios.

El administrador del sistema, denominado más comúnmente "superusuario", tiene asociada una cuenta que se identifica en la terminal por un símbolo diferente al resto de los usuarios, normalmente # en lugar de $.

En general UNIX puede tener además una serie de usuarios que gocen de cierta libertad para la gestión del sistema; estos son denominados usuarios especiales.

El superusuario es quien organiza y designa el arranque del sistema editando un archivo de comandos de uso exclusivo, en el que se ordenan las siguientes acciones:

- Comprobar si el sistema de archivos es correcto, para que si se encuentran archivos erróneos proceda a su restauración
- Limpiar el registro de usuarios activos
- Montar el sistema de archivos
- Limpiar los directorios temporales
- Arrancar los procesos iniciales
- Presentar la fecha en la consola maestra

De igual forma el superusuario establece el proceso de apagado del sistema cuando termina una jornada. Este proceso suele comprender las siguientes acciones:

- Enviar mensajes de aviso a las terminales activas
- Terminar o interrumpir los procesos activos excepto el de la consola maestra
- Asegurar que toda la actividad sobre el sistema de archivos ha terminado
- Desmontar el sistema de archivos

Otras actividades de la administración del sistema son:

- Cambiar las características de las terminales, símbolos de teclado, velocidad de transmisión, etcétera.
- Ejecutar periódicamente determinados procesos, existe un comando (CRON) que analiza en un archivo especial (crontab) determinados procesos que tienen que ejecutarse automáticamente cada minuto, hora, día, etcétera.

II.2 Manejo del procesador

Un problema de diseño importante en los sistemas operativos es la representación de los procesos. Una diferencia relevante entre Unix y otros muchos sistemas es la facilidad de crear y manipular múltiples procesos. Estos procesos están representados en Unix por varios bloques de control. No hay bloques de control del sistema accesibles en el espacio de memoria virtual de un proceso de usuario; los bloques de control asociados a un proceso están almacenados en el núcleo. La información de estos bloques de control es utilizada por el núcleo para la planificación del CPU y el control de los procesos.

II.2.1 Bloques de control de proceso

La estructura de datos más básica asociada a los procesos es la estructura de proceso (**process structure**). Una estructura de proceso contiene todo lo que es necesario saber sobre un proceso

cuando se descarga en un intercambio, como su identificador de proceso, información para la planificación (como la prioridad del proceso) y apuntadores a otros bloques de control. Existe una tabla de estructuras de proceso cuya longitud se define en el momento de montar el sistema. Las estructuras de proceso de los procesos dispuestos para ejecución se mantienen enlazadas por el planificador en una lista doblemente enlazada (la cola de preparados), y hay apuntadores desde cada estructura de proceso al padre del proceso, a su hijo vivo más joven y a otros parientes de interés, tales como una lista de procesos que comparten el mismo código de programa (texto).

El espacio virtual de direcciones de un proceso de usuario esta dividido en los segmentos de texto (código del programa), datos, y pilas. Los segmentos de datos y de pilas se ubican siempre en el mismo espacio de direcciones, pero pueden crecer por separado, y usualmente en sentidos opuestos: muy frecuentemente la pila crece hacia abajo y los datos ascienden hacia ella. El segmento de texto se halla en ocasiones en un espacio de direcciones diferentes del de los datos y la pila y es normalmente de sólo lectura.

Todo proceso con texto compartible tiene un apuntador desde su estructura de proceso a una estructura de texto. La estructura de texto registra el número de procesos que están utilizando el segmento de texto, un puntero a una lista de sus estructuras de

proceso, y dónde puede encontrarse en el disco la tabla de páginas del segmento de texto cuando se intercambie. La estructura de texto misma reside siempre en la memoria principal: al montar el sistema se define una tabla de tales estructuras. Los segmentos de texto, datos y pilas de los procesos pueden ser intercambiados al disco. Cuando se recuperan los segmentos, se paginan.

Las tablas de páginas registran información sobre el **mapping** (mapeo) de la memoria virtual del proceso a la memoria física. La estructura del proceso contiene apuntadores a la tabla de páginas, para utilizarla cuando el proceso resida en la memoria principal, la dirección del proceso en el dispositivo de intercambio, cuando se intercambie. No existe una tabla de páginas especial para un segmento de texto compartido; cada proceso que lo comparte tiene entradas para sus páginas en la tabla de páginas del proceso.

La información sobre el proceso que se precise solamente cuando el proceso sea residente (es decir, no descargado) se mantiene en una **u structure** (estructura de usuario) en lugar de la estructura de proceso. La **u structure** se mapea en el espacio virtual de datos del núcleo. Ahí se almacena una copia del bloque de control de procesos para salvar los registros generales del proceso, el apuntador a la pila, el contador de programa y los registros base de la tabla de páginas cuando el proceso no se esté ejecutando. Hay espacios para guardar los parámetros de las llamadas al sistema y los valores a devolver. Todos los

identificadores de usuario y de grupos asociados al proceso (no sólo el identificador de usuario efectivo situado en la estructura de proceso) se almacenan. Las señales, los temporizadores y las cuotas tienen sus estructuras de datos. De más obvia relevancia para el usuario ordinario, el directorio actual y la tabla de ficheros abiertos se guardan en la estructura de usuario.

Cada proceso tiene una fase en modo sistema y una en modo usuario. La mayoría del trabajo ordinario se realiza por el proceso en modo usuario, pero cuando se hace una llamada al sistema, es el proceso en modo sistema el que ejecuta la llamada al sistema. Las fases de sistema y de usuario de un proceso nunca se ejecutan simultáneamente. El proceso en modo sistema tiene una pila distinta que el proceso en modo usuario. El **kernel stack** (la pila del núcleo) del proceso, sigue inmediatamente a la estructura de usuario: la pila del núcleo y la estructura de usuario conjuntamente componen el segmento de datos del sistema para el proceso. La figura II.2.1 ilustra cómo se utiliza la estructura de proceso para encontrar las diversas partes de un proceso.

La llamada del sistema **fork** (bifurcada) asigna al proceso hijo una nueva estructura de proceso (con un nuevo identificador de proceso) y copia la estructura de usuario. Ordinariamente no hay necesidad de una nueva estructura de texto, puesto que los procesos comparten su texto; los contadores y las listas afectados simplemente se actualizan.

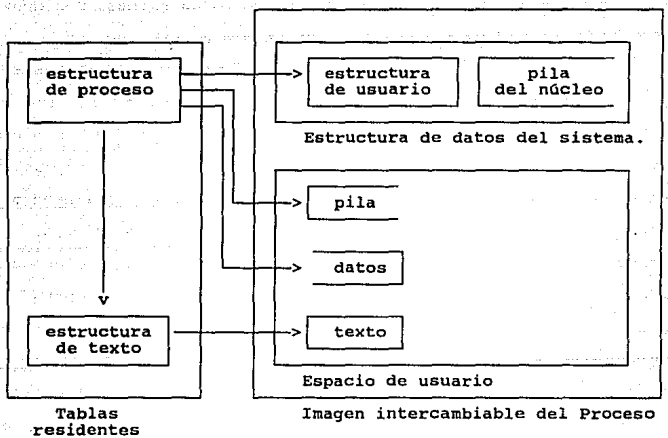


Fig. II.2.1 Estructura de un proceso

Se construye una nueva tabla de páginas, y se asigna nueva memoria principal para los segmentos de datos y de pila del proceso hijo. El copiado de la estructura de usuario preserva los

descriptores de los ficheros abiertos, los identificadores del usuario y del grupo, signals, (señales) y muchas de las prioridades análogas de un proceso.

La llamada al sistema **vfork** no copia los datos y pilas para el nuevo proceso, sino que el nuevo proceso comparte simplemente la tabla de páginas del antiguo. Se crean, de todas formas, una nueva estructura de usuario y una nueva estructura de proceso. Un empleo corriente de esta llamada al sistema asociada a los números mayores significan menor prioridad. Los procesos los realiza el shell para ejecutar un comando y esperar que acabe. El proceso padre utiliza vfork para producir al proceso hijo. Puesto que el proceso hijo desea utilizar inmediatamente una **execve** (llamada del sistema) para cambiar completamente su espacio virtual de direccionamiento, no es precisa una copia completa del proceso padre. Estructuras de datos como las necesarias para la manipulación de tubos, entubados pipes pueden guardarse en los registros entre vfork y execve. Los ficheros pueden cerrarse en un proceso sin que afecte al otro proceso, puesto que las estructuras de datos del núcleo implicadas dependen de la estructura de usuario, que no se comparte. El padre utiliza **wait** (espera) para suspender su ejecución hasta que termine el hijo, de modo que el padre no pueda alterar la memoria que el hijo necesite.

Cuando el proceso padre es grande, vfork puede producir ahorros sustanciales en tiempo de CPU. Sin embargo, es una llamada

al sistema algo peligrosa puesto que cualquier cambio en memoria se produce en ambos procesos hasta que tenga lugar la execve. Una alternativa consiste en compartir todas las páginas duplicando la tabla de páginas, pero marcando las entradas en ambas tablas de página como **copy-on-write** (copia en escritura). Los bits de protección por hardware se ponen en ON para atrapar cualquier tentativa de escritura en estas páginas compartidas. Si ésta se produce, se asigna una nueva celda y la página compartida se copia en la nueva celda. Las tablas de páginas se modifican para indicar que esta página ya no se comparte (y en consecuencia ya no necesita protección contra escritura) y la ejecución puede reanudarse. Errores por hardware en el VAX/750 impidieron a la versión 4.2BSD incluir una operación fork con copia dependiente de escritura.

Una llamada al sistema execve no crea una nueva estructura de proceso o estructura de usuario, sino que el texto y datos del proceso son reemplazados. Los ficheros abiertos se preservan (aunque existe un medio para especificar que ciertos descriptores de ficheros tienen que cerrarse ante una execve).

La mayoría de las propiedades de la gestión de excepciones por **sofware** (signals) se preservan, pero las disposiciones para llamar a una rutina especial del usuario ante una señal se cancelan, por razones obvias. El identificador del proceso y la mayoría del resto de las prioridades del proceso permanecen intactas.

II.2.2 Planificación del CPU

En Unix la administración del CPU está diseñada para favorecer a los procesos interactivos. Los procesos reciben pequeñas divisiones del tiempo del CPU por un algoritmo de prioridades, que se reduce a una asignación **round-robin** para trabajos intensivos en CPU.

Cada proceso tiene una **scheduling priority** (prioridad de planificación) asociada a él; los números más altos suponen menor prioridad. Los procesos que realizan E/S a disco u otras tareas importantes tienen prioridades negativas y no pueden ser eliminados por las excepciones de softwere. Los procesos de usuario ordinarios tienen prioridades positivas y por lo tanto es menos probable que se ejecuten antes que cualquier proceso del sistema, aunque los procesos pueden tener precedencia unos sobre otros.

Cuanto más tiempo de CPU acumule un proceso, más baja (más positiva) se hace su prioridad y a la inversa, de modo que existe una realimentación negativa en la administración del CPU y es difícil que un solo proceso acapare todo el tiempo de CPU. Para impedir la inanición se emplea el envejecimiento de los procesos.

Los sistemas Unix más antiguos utilizaban un quántum de un segundo para la planificación del round robin. La versión 4.BSD de la Universidad de Berkeley replanifica los procesos cada décimo de

segundo y recalcula las prioridades cada segundo. La planificación por round robin se realiza por medio de un compás de espera (timeout) que indica al controlador de interrupciones de reloj que llame a una subrutina del núcleo tras un intervalo especificado; la subrutina a llamar en este caso ocasiona la replanificación y se somete de nuevo a un compás de espera antes de llamarse a si misma de nuevo. El recálculo de la prioridad también está temporizado por una subrutina que se somete a si misma a un compás de espera.

En el núcleo no hay apropiación de un proceso por oro. Un proceso puede ceder el CPU por que esté esperando E/S o por que su división haya expirado. Cuando un proceso elige ceder el CPU, se va a "dormir" esperando un suceso. La primitiva del núcleo utilizada para esto se llama **sleep** (dormido) (no confundir con la rutina de biblioteca del nivel del usuario del mismo nombre). Tiene un argumento que, por convenio, es la dirección de una estructura de datos del núcleo asociado a un suceso que el proceso desea que ocurra antes de ser despertado. Cuando se produce el suceso, el proceso del sistema que lo sabe llama a **wakeup** (despertar) con la dirección correspondiente al suceso y todos los procesos que habían hecho una sleep sobre la misma dirección se sitúan en la cola de preparados.

Por ejemplo, un proceso que espera que acabe la E/S de disco, se debe dormir sobre la dirección de la cabecera del **buffer** (almacenamiento temporal) correspondiente a los datos, que se

estén transfiriendo. Cuando la rutina de interrupción del controlador de disco observa que la transferencia se ha completado, llama a wakeup sobre la cabecera del buffer. La interrupción utiliza la pila del núcleo de un proceso cualquiera que se esté ejecutando en ese momento, y se realiza wakeup desde ese proceso en modo sistema.

El proceso que en este momento se ejecute puede ser elegido por el planificador. Sleep tiene un segundo argumento, que es la prioridad de planificación a utilizar para este propósito. Si la prioridad es negativa, impide también que el proceso sea despertado prematuramente por algún suceso excepcional, como un signal (señal)

Cuando se genera un signal, (Señal) se pone en cola hasta que se ejecute la fase en modo sistema del proceso afectado. Esto suele ocurrir pronto, puesto que el signal (Señal) normalmente despierta al proceso si éste estaba esperando a alguna otra condición.

No existe memoria asociada a sucesos y el proceso que llama a la rutina que realiza el **sleep** (dormir) ante un suceso, tiene que estar preparado para afrontar un retorno prematuro, incluyendo la posibilidad de que la razón para la espera haya desaparecido.

Existen **race conditions** (condiciones de carrera) que afectan al mecanismo de sucesos. Si un proceso decide (por ejemplo, debido

a la verificación de un indicador flag (bandera) en memoria) dormir en espera de un suceso y éste se produce antes de que el proceso pueda ejecutar la primitiva que implementa la operación de dormir hasta el suceso, puede ocurrir que el proceso duerma indefinidamente. Esto se impide elevando la prioridad hardware del procesador durante la sección crítica de modo que no puedan producirse interrupciones, y así pueda ejecutarse sólo el proceso que desea el suceso hasta que se haya dormido. De esta manera, se utiliza la prioridad del procesador para proteger las regiones críticas en el núcleo. Esto constituye el mayor obstáculo para transferir Unix a máquinas con múltiples procesadores.

Hay muchos procesos, como editores de texto, que son intensivos en E/S y que serán planificados principalmente sobre la base de la espera de E/S. La experiencia sugiere que el administrador de Unix funciona mejor con trabajos de E/S, como puede observarse cuando hay ejecutando varios trabajos intensivos en CPU, como formateadores de texto o interpretes de lenguajes.

Lo que hemos llamado administración del CPU se corresponde estrechamente con la planificación a corto plazo, aunque la realimentación negativa del esquema de prioridades supone una cierta planificación a largo plazo, puesto que determina la mezcla de trabajos a largo plazo. La planificación a medio plazo se realiza mediante el mecanismo de **swapping** (intercambio) que a continuación se describe.

II.2.3. Swapping

Los sistemas UNIX anteriores al 3.BSD utilizaban el intercambio exclusivamente para gestionar la competencia por memoria entre procesos: si había demasiada competencia, se descargaban procesos hasta que hubiera suficiente memoria disponible. Además, algunos pocos procesos grandes pueden forzar a muchos procesos más pequeños a abandonar la memoria, y un proceso mayor que la memoria principal, fuera del núcleo no puede ejecutarse en lo absoluto.

El segmento de datos del sistema (la estructura u y la pila del núcleo) y el segmento de datos de usuario (texto [si no es compatible], datos y pila) se mantienen en memoria principal contigua por razones de eficiencia en los intercambios, de modo que la fragmentación externa de la memoria puede ser un problema importante.

La asignación de la memoria principal y del espacio para intercambio se realiza por **first-fit** (primer ajuste). Cuando el tamaño de la imagen de memoria crece (debido bien a la expansión de la pila o a la expansión de los datos), se asigna un nuevo fragmento de memoria suficientemente grande para la imagen en conjunto. La imagen de memoria se copia, se libera la antigua memoria y se actualizan las tablas apropiadas. (En algún sistema se realiza una tentativa para encontrar memoria contigua al final

del fragmento, para evitar copias). Si no hay un solo fragmento de memoria principal suficientemente grande, el proceso se descarga para ser recargado con el nuevo tamaño.

No hace falta descargar el segmento del texto compartible, porque es de sólo lectura, y no es preciso cargar un segmento de texto compartible para un proceso cuando ya hay otra instancia en memoria. Esta es una de las principales razones para seguir la pista de los segmentos compartibles: menos tráfico de intercambio. La otra razón es la reducida cantidad de memoria principal requerida para varios procesos que utilizan el mismo segmento de texto.

Las decisiones sobre que procesos se descargan o reincorporan son tomadas por el scheduler process (proceso planificador) que es el proceso 0 (también conocido como el proceso intercambiador o swapper). El planificador despierta al menos una vez cada cuatro segundos para verificar que procesos hay que descargar o reincorporar. Un proceso tiene más probabilidades de ser descargado si esta ocioso, ha estado en memoria durante mucho tiempo, o si es grande; si no se encuentran fácilmente candidatos, se eligen otros procesos por su edad. Un proceso tiene más probabilidades de ser reincorporado si ha estado descargado durante mucho tiempo, o si es pequeño. Hay verificaciones para impedir el thrashing (descargado), básicamente no permitiendo que el proceso sea descargado si no ha estado en memoria durante un cierto tiempo.

Si no es preciso descargar trabajos, se explora la tabla de procesos para buscar uno que merezca ser reincorporado (se determina por su tamaño y por el tiempo que hace que fué descargado). Si no existe suficiente memoria disponible, se descargan procesos hasta que la haya.

En el 4.2BSD, el espacio de intercambio se asigna en fragmentos que son múltiplos de una potencia de dos y desde un tamaño mínimo (por ejemplo, 32 páginas) hasta un máximo que viene determinado por el tamaño de la partición del espacio del disco dedicada al intercambio. Por esta razón, si hay varias particiones lógicas del disco que pueden ser utilizadas para intercambio, tienen que ser del mismo tamaño. Las distintas particiones lógicas del disco, también deben encontrarse bajo brazos diferentes para minimizar las búsquedas en disco.

Muchos sistemas Unix usan aún el esquema de intercambio descrito arriba. Todos los sistemas USG, incluyendo el System V, lo hacen. Todos los sistemas Unix de Berkeley, por otra parte, incluyendo el 4.2BSD, dependen primeramente de la paginación para la gestión de la competencia por la memoria y solo secundariamente dependen del intercambio. Para determinar que procesos se descargan o reincorporan, se utiliza un esquema muy similar, en líneas maestras, al tradicional, pero los detalles difieren y la influencia del intercambio es menos.

II.3 Manejo de memoria

El sistema operativo UNIX está soportado sobre una amplia variedad de arquitecturas de hardware con un rango, de capacidad de memoria virtual. La memoria física está dividida en segmentos de tamaño igual, con este tipo de mapeo, todas las páginas de memoria pueden direccionarse con un número de página.

En cualquier momento dado, puede haber un número de procesos de UNIX en el sistema, aunque sólo un número limitado de procesos puede estar recidente en la memoria física disponible en cualquier momento y el resto se guarda en el disco.

Esta situación da origen a los conceptos de la memoria virtual. Un proceso es almacenado en disco como una imagen completa o parcial del proceso en la memoria. Las **page tables** (tablas de páginas) son proyectadas de la memoria virtual a la memoria física y facilita una carga rápida de un proceso dentro de la memoria física.

Existen tres esquemas de **swapping** (intercambio) empleados por la mayoría de las implementaciones de UNIX.

- El esquema de swapping que se usó en la versión 7 de UNIX de AT&T, a través del sistema V.
- Berkeley adirió la paginación por demanda en UNIX 4.2 bsd (Berkeley Software Distribution)

- El sistema V de UNIX de AT&T ha sido mejorado al proporcionar una combinación de **swapping** y paginación por demanda.

II.3.1 Organización de la memoria virtual y física

Antes de entrar en los esquemas de swapping y paginación por demanda, para el manejo de la memoria, se debe revisar cómo es que el kernel de UNIX organiza y asigna la memoria física.

Los conceptos básicos de la organización y dirección de la memoria física y las estructuras de datos asociadas son muy independientes del esquema de manejo de memoria.

Las estructuras de datos para manejar la memoria física están diseñadas para proporcionar cuatro capacidades significativas:

1. Direccionamiento de grandes espacios de memoria física (32 MB o más).
2. Asignación rápida y desasignación de la memoria física.
3. Utilización de grandes espacios de memoria virtual (en el rango de los Gigabytes, Gb).
4. Protección del kernel, así como de la memoria de los procesos individuales.

Un sistema UNIX por lo regular soporta un gran número de usuarios. La memoria requerida por éstos usuarios es generalmente mucho mayor que la memoria física en el sistema. Un proceso que no está incluído en una lista para atenderlo, es almacenado

temporalmente en disco en un área llamada **swap device** (dispositivo de intercambio), el cual es una partición configurable en el disco manejada de manera similar a la memoria.

Los programas que se corren, son compilados y ligados en base a ciertas suposiciones acerca de los rangos del direccionamiento de memoria. Los procesos que se intercambian, se meten o se sacan en la memoria sobre un fragmento base que se repite de la memoria física y hace imposible garantizar que exactamente los mismos bloques de memoria física estarán disponibles para un proceso específico, cuando se cargue de nuevo en la memoria principal.

Sin ésta garantía, el kernel tendría que recalcular todas las direcciones en los programas cada vez que se cargara de nuevo el proceso en la memoria principal, a menos que un esquema alternativo se emplee. El uso del **process virtual address space** (espacio de la dirección virtual del proceso), soluciona éste problema al asignar bloques contiguos de memoria virtual a cada proceso. Estos bloques son proyectados a una memoria física, esto es, los procesos son **swapping in** (intercambiados) hacia la memoria principal y cuando no se han de correr se **swapping out** (intercambian) hacia el dispositivo de intercambio.

Las tareas más importantes del kernel, son las siguientes:

1. El rápido **tracking** (rastreo) de la memoria física desocupada.
2. Asignación y desasignación de la memoria física.

3. **Mapping** (proyectar) el proceso del espacio de memoria virtual a la memoria física.
4. Asignación y desasignación de memoria de intercambio (en el **swapping device**, dispositivo de intercambio).
5. Proyección del proceso que está en el espacio de dirección virtual a la memoria de intercambio.

El kernel lleva a cabo las dos primeras metas anteriores, al mantener arreglos de bits para determinar rápidamente si una página de memoria está en uso, a través de una lista de memoria libre que rastrea la memoria física desocupada.

Las tablas de páginas son usadas para proyectar las páginas de memoria virtual a páginas de memoria física. Una tabla de páginas contiene una lista secuencial de las estructuras de datos llamados "accesos de la tabla de páginas". Una tabla de páginas determina el espacio de memoria virtual para el texto, datos y los segmentos de **stack** (pila), llamados "región de un proceso".

Un proceso de UNIX está en modo usuario, mientras está ejecutando código de usuario o código de una librería de **runtime** que pasa por una llamada del sistema. (Con la instrucción **trap**, se cambia de modo usuario a modo kernel).

El código para las rutinas de servicio del kernel se llama código del kernel. Los procesos en modo usuario no pueden activar

ningún código directamente en el kernel, ni pueden accesar ninguna información en el kernel. Sin embargo, el kernel puede accesar la información de los procesos de modo usuario. Esta diferencia entre el usuario y el kernel, se mantiene al agrupar las tablas de páginas para el kernel en una tabla de página del kernel y al agrupar las tablas de páginas para el usuario en una tabla de páginas, de modo usuario.

II.3.1.1 Estructuras de datos

Un acceso de la tabla de páginas, consiste en los siguientes elementos:

1. Número de **frame page** (página de la estructura)
2. Edad
3. Bit modificado de página
4. Protección de la página
   a. Modo del kernel de sólo lectura
   b. Modo del kernel de sólo escritura
   c. Modo del usuario de escritura
   d. Modo del kernel de escritura, modo del usuario de lectura
   e. Modos del kernel y del usuario de sólo lectura.
5. **Page valid bit** (bit valido de página).

El número de **frame page** (página de la estructura) pertenece a la página de la memoria física, el resto de los accesos de la tabla de páginas describen los atributos de la asignación común de ésa

página de memoria.

Un **region table entry** (acceso de tablas de regiones) consta de los siguientes elementos:

1. Apuntador al inodo del archivo cargado en esta región
2. Tipo de región (ejemplo: texto, datos y pila)
3. Tamaño de la región
4. Ubicación de la región a la memoria física (apuntadores a las tablas de páginas)
5. Estado de la región (ejemplo: **locked, in demand**)
6. Contador de referencia

II.3.1.2 Proyección de la memoria física a la memoria virtual

La figura II.3.1 describe cómo se proyecta la estructura de la dirección virtual a la memoria física, vía tablas de región y tablas de páginas. El espacio de dirección virtual de un proceso se divide en texto, datos, stack y regiones de memoria compartida. Para cada proceso, los accesos de tablas de regiones son conectados a la tabla común de regiones correspondiente, (regiones de texto, datos, stack, o de memoria compartida). Los accesos de tabla de regiones contienen una lista secuencial de direcciones virtuales, que se proyectan dentro de los accesos de tablas de páginas. Los accesos de las tablas de páginas contienen números de páginas físicas (que corresponden a las páginas de la memoria física).

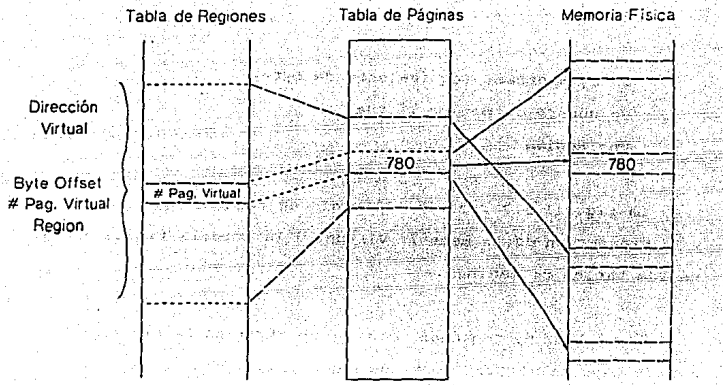


Fig. II.3.1 Ejemplo de la translación de una dirección virtual a una física

Este mecanismo traduce el espacio de dirección virtual contiguo de un proceso, a páginas de memoria física, direccionadas aleatoriamente. Una dirección en el espacio de dirección virtual, consiste de tres componentes:

- Un **pointer** (apuntador) al acceso de tablas de regiones
- Un número de página virtual
- Un **byte offset** (equivalente de byte) dentro de la página.

Estos tres componentes proporcionan la información suficiente para la translación de la dirección virtual a la física. Una **virtual address structure** (estructura de dirección virtual) descrita a continuación define completamente una dirección de la memoria:

- Un **byte offset** (equivalente de byte)
- Un número de página virtual
- Una Región

La fig. II.3.2 describe la tabla de regiones para un proceso y la proyección de la memoria virtual a la memoria física a través de las tablas de páginas.

Las tablas de regiones, como se muestran en la figura anterior asumen memoria virtual contigua para un proceso.

Las regiones de texto, datos y pila se proyectan a los accesos respectivos en la tabla de páginas. Existe un acceso de tabla para cada bloque de la memoria especificado en la tabla de regiones. La tabla de páginas proporciona una interrelación entre la dirección virtual y la dirección física. El número de la página de memoria física ocupa el campo del número de la estructura de la página, así como la página de la memoria virtual se proyecta dentro de la memoria física. Además, el espacio del kernel es común a todos los procesos y se puede usar un direccionamiento más simple y directo; para las regiones del kernel la mayoría de las implantaciones usan el mismo mecanismo para los procesos de los usuarios y del kernel. La proyección de los procesos que se llevan a disco se archiva al rastrear las direcciones de bloque de disco correspondiente a sus direcciones virtuales. Dependiendo de la implementación, cualquiera de los números de las estructuras de páginas se reemplazan por

direcciones de bloque de disco para las páginas de un proceso que se intercambian o bien se utiliza un campo descriptor de bloque de disco adicional.

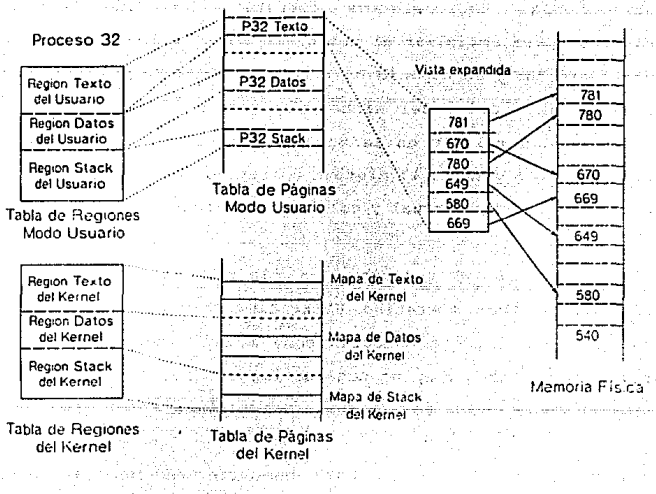


Fig. II.3.2 Ejemplo de proyección de la memoria

II.3.2 Organización de la memoria del sistema UNIX

La proliferación de versiones del sistema UNIX sobre una variedad de sistemas de computadoras han simulado una amplia variedad en la organización física de la memoria para el sistema operativo UNIX. Este sistema operativo ha sido diseñado liberalmente para optimizar el uso de la memoria disponible si un sistema tiene un gran espacio de memoria física (o primaria), la mayor parte del código del sistema operativo y las estructuras centrales pueden residir en la memoria física. Si el espacio de memoria no es tan grande, el sistema operativo reside parcialmente en disco y en memoria virtual y sólo se carga cuando sea necesario.

La figura II.3.3 describe una distribución típica de la memoria del sistema operativo UNIX. La memoria usada por los procesos es generalmente conocida como **memoria del usuario** y la memoria para los tres componentes más importantes (o más grandes) del sistema es conocida como **memoria del sistema.**

La memoria del sistema UNIX consiste en los siguientes componentes:

1. **Low core components** (Componentes bajos del núcleo)
   - Vectores de interrupción
   - Manejadores de dispositivos
   - Rutinas de proyección de memoria física.
2. Código del kernel

- Manejo de memoria
- Manejo de procesos
- Manejo de disco
- Manejo de E/S
- Comunicación de interprocesos

3. Datos del sistema
   - **Process structures** (tablas de procesos)
   - **User structure** (estructura del usuario)
   - **Kernel stacks** (pilas del kernel)
   - **Buffer pool**
   - **In-core inodes** (inodos del núcleo)
4. **In-core processes** (procesos del núcleo)

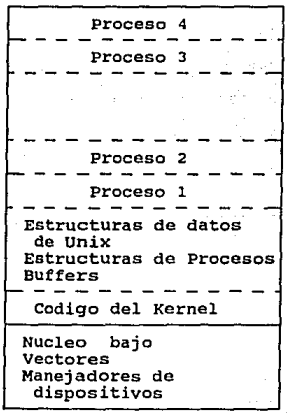


Fig. II.3.3 Mapa de memoria del Sistema UNIX

II.3.2.1 Arquitectura de la asignación de memoria

La **memory free list structure** (estructura de la lista de memoria libre) graba las **page frames** (estructuras o marcos de páginas) de la memoria disponible. Esta estructura se usa en conjunción con otro arreglo de información, un arreglo secuencial de bits de memoria libre llamado "**memvad**". Cada bit de este arreglo llamado **page frame bit** (bit de la estructura de página) se proyecta hacia una estructura de página de la memoria física. Este bit se prende (con 1) si la estructura esta en uso, o se pone en cero, si la estructura está libre.

La estructura de la lista de memoria libre incluye los siguientes componentes:

1. El contador de páginas libres de memoria en el arreglo **pnum**
2. Apuntador a la primera estructura de página de memoria libre, en el mismo arreglo **pnum**
3. Apuntador a la última estructura de página de la memoria en el arreglo **pnum**
4. Apuntador a la última asignación de memoria
5. El total de estructuras de páginas de memoria disponible
6. El arreglo **pnum** (con hasta cien números de estructuras de páginas almacenadas)

En la inicialización del sistema, la estructura de la lista de memoria libre, registra el primer número de estructura de página,

el último número de estructura de página y el total de estructuras de página disponibles.

II.3.3 Memory Swapping (Intercambio de memoria)

Los sistemas UNIX de AT&T, es sus primeras versiones del sistema V, usaban sólo un dispositivo de intercambio, que se define como un dispositivo de bloques contiguos en el disco reservado para los procesos de intercambio. Las versiones posteriores del sistema V soportaban dispositivos de intercambio múltiple y de paginación por demanda. La estructura del dispositivo de intercambio difiere de la estructura normal en los siguientes puntos:

1. El tamaño del dispositivo de intercambio es un parámetro configurable
2. El espacio en disco se asigna en la inicialización del sistema de archivos
3. Las transferencias de disco ocurren en grandes bloques

Estos aspectos requieren un mecanismo diferente del usado para accesar bloques de información para archivos de disco. Además, los dispositivos de intercambio se manejan a través de mapas que rastrean bloques de las estructuras de páginas sobre el dispositivo de intercambio.

II.3.3.1 Arquitectura de asignación de mapas de memoria

Los mapas de memoria se usan para dispositivos de intercambio y para ciertas funciones especiales de E/S. Como hemos visto, los dispositivos de intercambio son tratados esencialmente como la memoria. Un mapa de memoria llamado "mapa de intercambio para un dispositivo de mapa" consiste de estructuras de mapas que tienen dos parámetros:

- Una dirección de mapa
- Un tamaño de mapa

Cada estructura de mapa define el inicio de un bloque continuo de unidades de intercambio libres de memoria. Una unidad de intercambio es de 512 bytes para el sistema V de UNIX de AT&T para dispositivos de intercambio tipo 1. Puede variar en otras implementaciones, pues las unidades de intercambio de 1024 bytes son usadas en muchas implementaciones.

El kernel inicia con una estructura de mapa que apunta al principio de la memoria (esto es, la página inicial en el dispositivo de intercambio) y especifica el tamaño como el tamaño total del dispositivo. El kernel entonces usa secciones de bloques continuos desde esta memoria para una rápida descarga de los procesos. La fragmentación es reducida al combinar los fragmentos de memoria cada vez que alguna memoria es liberada.

La rutina **malloc** asigna la memoria de intercambio; es importante notar que esta rutina asigna memoria en el dispositivo de intercambio, no en memoria física, ya que ésta sólo es asignada cuando el proceso es **re-scheduled** (puesto en la lista de espera otra vez). Cuando la memoria de intercambio es asignada, la tabla de mapa de memoria es revisada para conocer el tamaño requerido. La primera estructura que proporciona el tamaño lo suficientemente grande es usada y el número requerido de unidades de intercambio es asignado al inicio de este bloque. El parámetro **size** (tamaño) en la estructura de mapa se reduce por la cantidad de las unidades de intercambio asignadas y la dirección de inicio se enciende para la nueva dirección de inicio del bloque libre de unidades de intercambio. Si no se encuentra memoria se regresa un "0", en el parámetro **size**.

Cuando la memoria de intercambio es liberada, los bloques de unidades de intercambio que se liberan son incluídas en la tabla de mapa de memoria. Sin embargo, la rutina "**m free**" primero clasifica la tabla de mapa de intercambio por la dirección inicial de la unidad de intercambio para cada acceso a la estructura de mapas y revisa si la estructura de mapa para cada bloque liberado puede ser combinado dentro de la estructura previa o la siguiente (estructura). Esta puede combinar con la estructura previa si la dirección de inicio de la primera dirección de la unidad de intercambio, en el bloque liberado, es mayor que la última dirección de la unidad de intercambio, en la estructura de mapas.

El parámetro **size** en la estructura de mapa previa, se incrementa para combinarlas. Esta combinación con el próximo bloque puede darse si la dirección de la última unidad de intercambio en el bloque liberado, es menor que la dirección inicial del mismo bloque y el parámetro **size** se incrementa por el número de unidades liberadas de intercambio. La figura II.3.4 ilustra la proyección entre las estructuras de mapas y los bloques de dispositivos de intercambio.

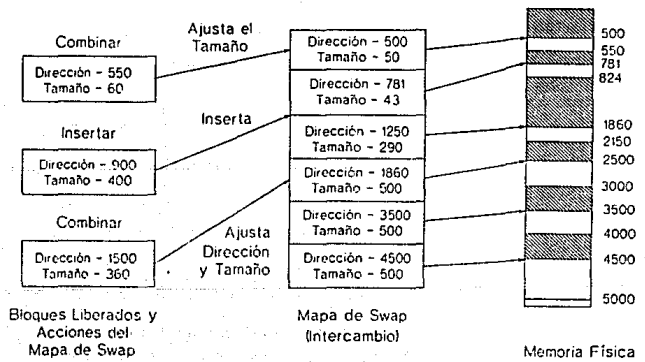


Fig. II.3.4 Ejemplo de un dispositivo de intercambio de memoria

II.3.4 Sistemas de paginación por demanda

La versión de sistema operativo de UNIX de Berkeley 4.2 bsd proporcionó el primer rediseño más grande del kernel de UNIX al soportar la paginación por demanda. Primero veamos algunos conceptos generales usados en los sistemas de paginación por demanda.

II.3.4.1 Conceptos de la arquitectura de paginación por demanda

Como el término lo implica, la arquitectura de paginación por demanda requiere que se carguen las páginas de memoria de disco a la memoria física por demanda. En lugar de intercambiar el proceso completo, en realidad sólo se intercambian las partes que se requieren de éste. Una ventaja de este acercamiento, es que un gran número de procesos pueden ser realmente intercambiados hacia la memoria en cualquier momento. Por otra parte, la operación de la paginación por demanda causa que un proceso sea bloqueado si la página requerida no está en memoria. Los beneficios de conservar un gran número de procesos en memoria, parecen ser más importantes que las desventajas del bloqueo de recursos ciertamente más frecuente, el volumen del intercambio de disco se reduce significativamente en los sistemas de paginación por demanda. Las novedades más significativas en la paginación por demanda son:

- **page faults** (fallas de página)
- **cache memory** (memoria cache)

El mecanismo de "falla de página" es usado para determinar cuándo el código cruza las fronteras de una página, y la siguiente página no está en memoria física. Esto es cuando ocurre lo anterior se genera una falla de página, que a su vez ocaciona las siguientes actividades:

1. Ajustar las tablas de paginación del núcleo para aceptar una nueva página.
2. Organizar para que una página sea leída del disco
3. "Dormir" hasta que la página sea cargada
4. Iniciar la ejecución en el inicio de la nueva página cuando sea **re-scheduled**.

La memoria cache es empleada por algunas implantaciones para anticipar las páginas requeridas y darlas a la memoria, antes de que sean requeridas por el CPU para eliminar, en lo posible las fallas de páginas.

El concepto de regiones y tablas de páginas es muy similar al descrito para los sistemas de intercambio. Sin embargo, los componentes de los accesos a la tabla de páginas son diferentes. Dos campos adicionales se requieren para la paginación por demanda, como el campo **age** (edad) que se usa para traer la duración para la cual la página ha sido retenida en memoria y es usada en el algoritmo para reclamar memoria, y la bandera de referencia que indica si la página fue accesada por un proceso. Cada acceso de la tabla de páginas está asociado con un "descriptor de bloque de

disco" que describe el bloque del almacenamiento en disco para esa página virtual. Los accesos de tablas de página y los descriptores de bloque de disco son combinados dentro de las tablas similares a las tablas de páginas usadas por los sistemas de intercambio. Los sistemas de paginación por demanda usan una estructura de datos del marco (o estructura) de la página. Una tabla de datos de la estructura de una página consiste de una entrada a la estructura de datos del marco de la página para cada página en uso y contiene información acerca del estado de la página, el contador de referencia y el almacenamiento en disco correspondiente. La tabla de datos del marco de la página mantiene una lista de estructuras de datos de los marcos de las páginas, llamada **free list** (lista libre) para estructuras con un contador de referencia de "0". En caso de que la misma página se necesite de nuevo, la lista libre de estructuras de datos de los marcos de páginas, se revisa para ver si el marco de la página está disponible todavía. Si es así, se reasigna y no hay necesidad de copiarla de nuevo desde el disco. La operación de las tablas de datos de marcos de páginas, es análogo al mecanismo de **buffer pool**. La figura II.3.5 describe el manejo de éstas tablas en la arquitectura de paginación por demanda. Otra estructura de datos requerida específicamente por los sistemas de paginación por demanda, es la **swap-use table** (tabla de uso de intercambio) que rastrea el número de accesos de tablas de página que apunta a las unidades de almacenamiento del dispositivo de intercambio.

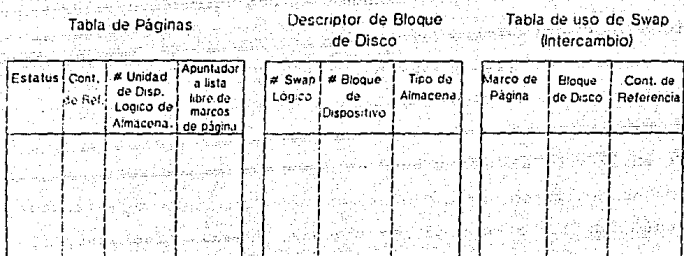


Fig. II.3.5 Utilización de las Tablas de información de las estructuras de páginas.

II.3.4.2 Estructuras de datos para la paginación por demanda

Mientras algunas estructuras similares existen para otras implementaciones, las descripciones presentadas aquí, se aplican a las versiones del sistema V de UNIX de AT&T. La estructura de accesos de tablas de páginas consisten en los siguientes

componentes:

1. **Page frame number** (número de estructura o marco de página)
2. **Age** (edad)
3. **Copy on modify** (se hace otra copia, cuando se modifico con un proceso)
4. **Modify** (Modifica, cuando se ha modificado y se necesita escribir lo anterior)
5. **Reference** (contador de referencia)
6. Validación
7. Protección

Los descriptores de bloques de disco se usan como una extensión de los accesos de tablas de páginas y cada descriptor está asociado con cada uno de los accesos de la tabla de estructuras de páginas. Los componentes de los descriptores de bloques de disco son los siguientes:

1. Dispositivo de intercambio (nombre lógico del dispositivo)
2. Número de la unidad de almacenamiento del dispositivo de intercambio
3. Tipo de la unidad de almacenamiento

Las estructuras de datos de los marcos de página contienen información acerca de la utilización del marco de página de memoria. Esta información incluye lo siguiente:

1. Estado de la página
   a) sobre el dispositivo de intercambio

b) en el archivo ejecutable
c) DMA en progreso
d) disponible para su uso

2. Contador de referencia (el número de procesos que realmente lo utilizan)
3. Dispositivo lógico y número de la unidad de almacenamiento
4. Apuntadores a la lista libre de marcos de páginas.

Las estructuras de uso de intercambio, una para cada página o unidad de almacenamiento sobre el dispositivo de intercambio, contienen información del uso de cada página y consiste en lo siguiente:

1. Contador de referencia (accesos a la tabla de páginas apuntando al dispositivo de intercambio)
2. Número de unidad de página/almacenamiento.

La fig. II.3.6 describe las tablas usadas para la paginación por demanda.

I.3.4.3 Manejo de los marcos de páginas

Como vimos anteriormente, el mecanismo de las fallas de marcos de páginas, se utiliza en la mayoría de las implementaciones de paginación por demanda para introducir nuevas páginas. Una falla de página puede ocurrir si una página que se está modificando y se prende el **"bit copy on modify"**, esta situación generalmente se

llama una "falla de protección", ya que el bit de protección para la memoria indica "copiar" en el estado "modificar" si se asigna el bit no separado para este estado.

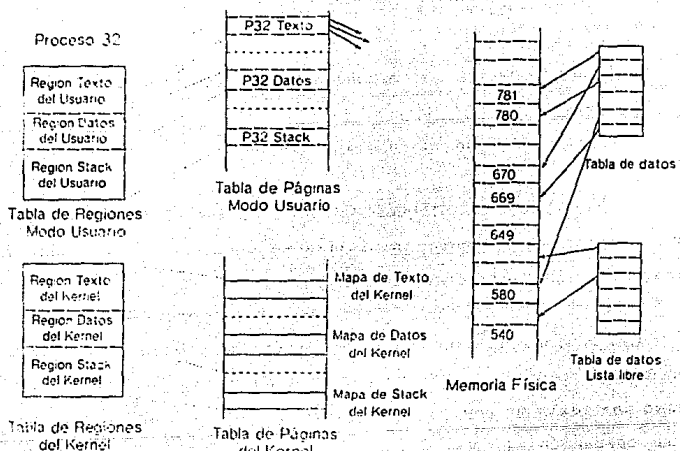


Fig. II.3.6 Utilización de las Tablas para sistemas de paginación por demanda

Otro tipo de falla de página puede ocurrir si la página requerida no está en la memoria, o está fuera del espacio de la dirección virtual para ese proceso. El **valid bit** (bit de

validación) se usa para indicar el tipo de falla de página. Una página que está en el espacio de la dirección virtual, válido para el proceso, se puede encontrar en una de las siguientes tres localidades:

1. En la lista libre de páginas (**free pool**) en la memoria
2. En un dispositivo de intercambio (en el que ha sido cargada una vez, pero ya se ha **swapped out**, intercambiado hacia la memoria)
3. Aún en el archivo ejecutable (que no se ha cargado del todo)

El descriptor de bloques de disco define el tipo de página, por ejemplo, si requiere inicializarse con cero, o si puede ser cargada sin inicializarse.

Las rutinas de manejo de las fallas de página, son llamadas con la dirección de la página que originó el proceso, que a su vez causó una falla de página. Esto indica el uso de tablas diferentes y la operación del **valid bit** (bit de validación) y el **copy on modify bit** (copiar en el estado de modificar).

II.4 Manejo del sistema de archivos

El sistema de archivos es uno de los subsistemas más importantes del sistema operativo. Es responsable de almacenar la información en los **disk drives** (discos) y recuperar y actualizar esta información directamente por el usuario o por un programa.

II.4.1 Definiciones del sistema de archivos

Una unidad física de disco puede consistir de varias secciones o particiones. Cada sección del disco puede contener un sistema lógico de archivos de un bloque de **boot** (inicialización), un superbloque, una lista de inodos y bloques de datos. Cada sección del disco tiene un nombre de archivo dispositivo. El manejador de discos está diseñado para controlar particiones múltiples.
El sistema UNIX tiene los siguientes tipos de archivos:

1. **Archivos ordinarios**
2. **Archivos de directorio**
3. **Archivos especiales (de dispositivos)**
4. **Archivos FIFO para los pipes (tubos)**

Archivos ordinarios. Son archivos que contienen información almacenada por el usuario, un programa de aplicación, o un programa de utilerías del sistema.

Los **archivos de directorio** manejan la función de catalogar el sistema de archivos. Los directorios asocian nombres con grupos de archivos que reflejan generalmente la estructura de la aplicación del usuario. Los archivos de directorios son archivos ordinarios con privilegios especiales de protección de escritura, tales que solo el sistema puede escribir en ellos, mientras el acceso de lectura está disponible para los usuarios bajo el control normal de acceso.

Los archivos especiales son usados para accesar dispositivos periféricos. Cada dispositivo de entrada está asociado con un archivo especial.

Archivos FIFO. Estos archivos usados por los pipes (tubos) son también especiales, debido a su misma naturaleza FIFO (primero en entrar, primero en salir).

II.4.2 Estructura de los archivos

El sistema operativo UNIX tiene una estructura de directorio jerárquica de multiniveles construida como una raiz invertida, con un nodo raiz único (fig. II.4.1). Este nodo es llamado **root (raiz)** y se hace referencia a éste, con un "/".

El directorio "/usr" es el directorio primario para los programas y datos del usuario . Los usuarios pueden crear niveles adicionales de directorios, bajo éste, para catalogar sus grupos o archivos.

Todos los directorios son creados con dos accesos de archivos. Un acceso llamado ".." que se refiere al directorio de nivel superior, comunmente llamado **directorio padre**; el otro acceso llamado "." se refiere a sí mismo.

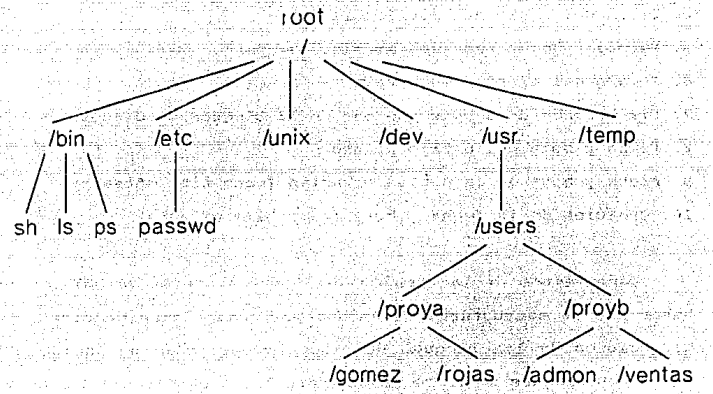


Fig. II.4.1 Estructura del directorio de UNIX

II.4.3 Inodos e identificación de archivos

Cada archivo tiene una estructura de inodo, que es identificada por un número i. El inodo contiene la información que se requiere para accesar el archivo. Esta información incluye lo siguiente:

1. Tipo de archivo y modo (ordinario, directorio, especial, etc.)
2. Número de **links** (ligas) al archivo
3. El **ID** (identificador) del propietario
4. El **ID** del grupo del propietario
5. Permisos de accesos del archivo
6. Tamaño del archivo en bytes
7. Direcciones de 3 bytes de más de 13 bloques de discos
8. Fecha y hora del último acceso
9. Fecha y hora de la última creación (como **time offsets**)
10. Creación de la fecha y hora (como **time offsets**)

Dependiendo de la versión del sistema operativo UNIX, puede haber 8 o 16 estructuras de inodos por cada bloque de disco. Las direcciones de los bloques de disco proporcionan un índice para todos los bloques de disco usados para la información del archivo.

El inodo del disco, descrito anteriormente, contiene información acerca de un archivo y puede cambiar, si esta información cambia, aún si el archivo mismo no es modificado. Cualquier cambio en el archivo también requiere cambiar el inodo debido a los parámetros del tamaño del archivo y el momento del acceso.

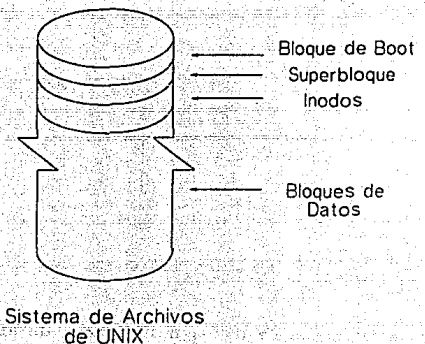


Fig. II.4.2 Organización del Sistema de Archivos de UNIX

La figura II.4.3 proporciona un vistazo de las estructuras de datos utilizadas para el manejo del sistema de archivos. Veamos ahora cómo el kernel maneja éstas estructuras.

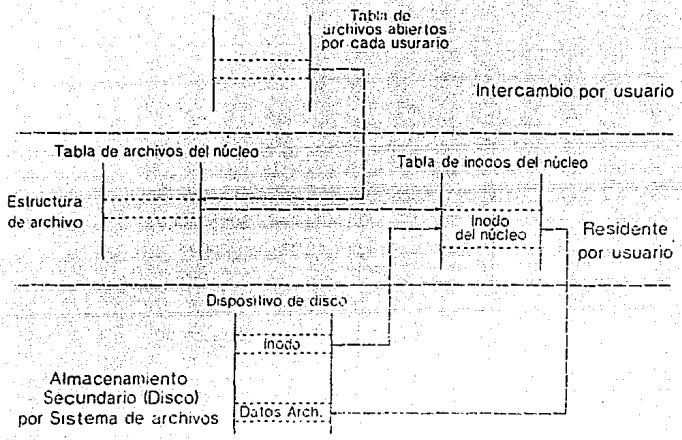


Fig. II.4.3 Estructuras de datos del Sistema de archivos de UNIX

Cada archivo en uso tiene otras dos estructuras asociadas a él, ésta son:

- **File structure** (estructura del archivo)
- **In-core inode** (inodo del núcleo)

Para cada archivo abierto o creado se establece, una estructura de archivo y un acceso en la tabla de archivos. Un archivo puede tener más de una estructura de archivo asociada a él. El inodo del núcleo, además de la información contenida en el inodo de disco, contiene lo siguiente:

1. **Banderas del estado del acceso de los inodos.**
   a) El inodo esta en **locked** (cerrado) esto es, en modificación en proceso.
   b) Un proceso está esperando por un inodo para estar en **looked**
   c) La información del archivo en el inodo del núcleo está modificada (puede ser diferente del inodo del disco)
   d) El contenido del archivo está cambiado, y el inodo modificado, ambos pueden ser diferentes del inodo del disco.
   e) El archivo es un dispositivo **"moutable"**
2. Contador de referencia adquirido de los accesos de tablas de archivos
3. ID del dispositivo donde reside el inodo del disco.
4. Las direcciones de disco para el apuntador del archivo
5. La última lectura lógica de bloque (posición del archivo)

El **lock** (candado) del inodo es usado para prevenir que dos procesos intenten modificar un archivo o su inodo, al mismo tiempo. La bandera **process waiting** (espera de proceso) es equivalente a una **call back** (llamada) asi que, cuando el inodo esta abierto, es modificada la bandera **process waiting**.

El acceso al inodo de núcleo se crea al leer el inodo del disco dentro de la memoria y almacenándolo dentro de la tabla de inodos del núcleo. El número máximo de accesos dentro de ésta tabla es un parámetro configurable del sistema y define el número máximo

de archivos únicos que pueden abrirse al mismo tiempo. La variable del contador de referencia es usada para mantener el rastreo de las referencias de archivos. El inodo queda activo hasta que el contador de referencia tienda a cero, el inodo del núcleo se regresa a la lista de inodos libres. El inodo de disco se actualiza siempre que la información del archivo cambia o el mismo archivo es modificado. Si solo la información del archivo cambia, las banderas de modificación del inodo del núcleo, son usadas para determinar si el inodo del disco debe ser actualizado.

II.4.4 Inode free list (Lista de inodos libres)

Dependiendo de la versión del sistema operativo UNIX, los bloques son de 512 bytes o 1 Kbytes (Sistema V de AT&T), o de 4 Kbytes (Berkeley 4 bsd). El número de inodos por bloque varía de una versión a otra. El número total de inodos es un parámetro configurable, determinado cuando el sistema de archivos es creado. Un inodo es del orden de los 64 bytes de longitud para el Sistema V de UNIX. El total de bloques requerido para mantener a todos los inodos se calcula y los bloques de discos son asignados como bloques contiguos. Los inodos pueden entonces ser accesados como registros de **fixed length** (longitud variable), en un espacio de datos secuencial. La posición relativa del inodo en este espacio de datos es su número de inodo. La lista de inodos libres, así como el tamaño de los inodos, se mantiene en el superbloque, el cual también tiene el número de índice del siguiente inodo libre en la

lista de inodos libres. No existe una lista para inodos activos. El sistema asume que cualquier inodo que no esté en ésta lista, esta en uso. Los inodos activos son accesados a través de la estructura de directorios, que asocia un nombre de archivo con un número de inodo.

II.4.5 Organización física del sistema de archivos

La organización física del sistema de archivos ha cambiado con cada versión nueva liberada del Sistema UNIX. Estos cambios incluyen tamaños de bloques, algoritmos de direccionamiento de bloques de disco, y la organización de los inodos. Mientras la versión de UNIX de Berkeley 4.2 bsd, utiliza bloques de 4 bytes, el Sistema V de AT&T utiliza bloques de 1 Kbytes. Las versiones anteriores utilizaban bloques de 512 bytes.

II.4.5.1 Direccionamiento Lógico

El sistema de archivos UNIX se construye como un disco lógico. Una unidad física de disco puede consistir de varias secciones lógicas de disco, donde cada una es establecida como un sistema de archivos. Esto puede manejarse bajo una jerarquía común, como se muestra en la figura II.4.4, al ejecutar la llamada al sistema "mount". Un disco lógico puede verse como un arreglo dimensional de bloques, sin embargo, solo las partes "llave" del disco lógico asignadas al momento de la creación del sistema de archivos, son

establecidas, como bloques contiguos en el disco físico. El resto de los bloques del disco lógico son asignados como sea necesario. El disco físico no provee un sistema de archivos físicamente contiguo. El tamaño de los archivos del disco lógico es esencialmente usado para definir el límite superior del tamaño del disco lógico. El disco físico puede consumirse antes de que alcance éste límite superior y restringir el tamaño actual del disco lógico.

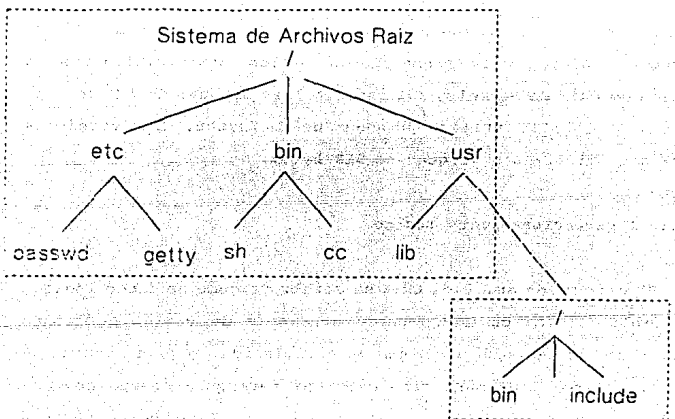


Fig. II.4.4 Organización del Sistema de Archivos "Mounted" de UNIX

Un disco lógico está dividido en las siguientes 4 secciones:

- Bloque 0 - Bloque de boot (inicialización)
- Bloque 1 - Superbloque
- Bloque 2-m - Inodos
- Bloque (m + 1) a n - File space (Espacio de archivos)

El bloque de inicialización generalmente contiene un programa que carga el kernel de UNIX desde el sistema de archivos, y trasfiere el control a la dirección de inicio del kernel.

El superbloque contiene toda la información que define el estado actual del sistema de archivos. Esta información se lista en la tabla II.4.5.

Como hemos visto anteriormente, el número total de los inodos en un sistema de archivos es un parámetro configurable. Los inodos son establecidos para iniciar en el bloque 2 y el número requerido de bloques contiguos es asignado hasta el bloque "m" como se describió anteriormente.

- Tamaño en bloques de la lista-i (lista de inodos)
- Tamaño en bloques del sistema de archivos
- Numero de bloques libres en el sistema de archivos
- Una lista de bloques libres disponibles en el sistema de archivos
- La lista de inodos libres
- Banderas de seguridad de los inodos
- Bandera para indicar que el superbloque ha sido modificado
- Bandera para indicar que el sistema de archivos ha sido "mounted" for (modificado para) solo-lectura.
- Fecha y hora de la ultima actualizacion del superbloque (time offset)
- Total de los bloques libres en el sistema de archivos
- Total de los inodos libres en el sistema de archivos
- Nombre del sistema de archivos
- Nombre del disco del sistema de archivos

Tabla II.4.5 Lista parcial del contenido en la estructura de superbloque

La figura II.4.6 muestra la evolución del direccionamiento de bloques de disco y como el sistema UNIX lo adoptó para archivos más grandes de disco y para sistemas de archivos mayores.

El concepto de "direcciones intrinsecas" es el mecanismo adoptado para tratar los archivos grandes.

```
UNIX de AT&T Nivel 6:
          Archivos pequeños: 8 bloques directos
          Archivos grandes:  7 bloques indirectos,
                             el 8o. es doble indirecto

UNIX de AT&T Nivel 7:
          Bloques de 512 bytes
          13 direcciones de bloques de disco:
                        10 bloques directos
                        1 bloque indirecto
                        1 bloque doble indirecto
                        1 bloque triple indirecto

System V de UNIX de AT&T:
                  Bloques de 1K bytes
                  El mismo esquema de direcciones que el
                        UNIX de AT&T Nivel 7

Versión 4.1 bsd del UNIX de Berkeley:
                  Bloques de 1K bytes
                  El mismo esquema de direcciones que el
                        UNIX de AT&T Nivel 7

Versión 4.2 bsd del UNIX de Berkeley:
                  Bloques de 4K bytes
                  Los inodos estan distribuidos en Cilindros
                  para búsquedas más rápidas.
```

fig. II.4.6 Diferencias entre las implementaciones (o versiones) de UNIX

II.4.6 Acceso de archivos y rutinas de control

Algunas rutinas son usadas para crear archivos, asignar inodos y espacio en disco, y para controlar el acceso a los archivos. Las rutinas principales se mencionan a continuación:

1. **NAMEI** Traduce los nombres de los archivos a inodos
2. **IGET** Asigna y accesa la tabla de inodos

3. **ALLOC** Asigna y libera bloques de disco e inodos
4. **FIO** Rutinas para manipular las estructuras de las tablas de archivos
5. **SYS2** Rutinas de interfase para llamadas al sistema

II.4.7 Performance issues (Problemas en el desempeño)

Algunas caracteristicas diferentes de la arquitectura del sistema UNIX afectan su funcionamiento operacional. Las características principales de la arquitectura del manejo del sistema de archivos que chocan con el desempeño incluyen las siguientes:

1. Archivos regulares y pipes (tubos)
2. **Swap memory** (memoria de intercambio)
3. Partición de discos
4. **Buffer pool size**

II.4.7.1 Archivos regulares y pipes

Los archivos son usados para almacenar información del usuario, así como para comunicar esta información entre los programas. La información puede ser comunicada, también a través de los archivos regulares o tubos.

Un archivo regular consiste de:

- 10 bloques directos y
- 3 bloques indirectos

Un bloque directo requiere menos accesos a disco que un bloque indirecto. La información, en pequeñas cantidades, que es requerida por los programas frecuentemente, puede ser almacenada en bloques directos, y conservada en archivos regulares.

La información transitoria, sin embargo, debe ser comunicada por vía **pipe** nombrando o sin nombre, (o a través de archivos **FIFO**).

II.4.7.2 Memoria de intercambio

La memoria de intercambio como ya hemos visto anteriormente, es usada para almacenar temporalmente texto, datos, y **stack** en el disco para un proceso que está dividido en bloques. La memoria de intercambio esta establecida de forma contigua. El tamaño de la memoria de intercambio es un parámetro configurable. En un ambiente de operación típico, la memoria de intercambio se establece en 2 o 3 veces el tamaño de la memoria física. Separa la memoria de intercambio en algunos discos, acelera el acceso en un sistema con múltiples **disk drives** y múltiples controladores de disco.

II.4.7.3 Particionamiento de disco

El tamaño de las particiones de disco es un parámetro configurable, así como el tamaño de las particiones de la raiz, del espacio de intercambio, y del usuario (**usr**).

II.4.7.4 Buffer pool size (tamaño del pool de buffers)

Un gran **buffer pool** aloja más bloques de disco que son retenidos en la memoria por periodos de tiempo más grandes. Esto reduce el potencial de un proceso, al bloquearlo por una E/S de bloque de disco. En un sistema con aplicaciones múltiples que usa un gran número de archivos de varios sistemas de archivos, reduce el tiempo de acceso a la información, al aumentar significativamente el tamaño del **buffer pool**.

II.5 Manejo de Entrada / Salida

En UNIX, el sistema de E/S se relaciona tan íntimamente con el sistema de archivos que se puede decir que están unidos en un mismo proceso. La unidad básica utilizada para organizar la información en el Sistema UNIX se denomina archivo. El sistema de archivos del Sistema UNIX proporciona un método lógico para organizar, almacenar, recuperar, manipular y gestionar la información. Los archivos están organizados en un sistema de archivo jerárquico, agrupados en directorios. Una característica de simplificación importante del Sistema UNIX es la forma general de tratamiento de los archivos. Por ejemplo, los dispositivos físicos se tratan como archivos; esto permite que las mismas órdenes operen sobre archivos ordinarios y sobre dispositivos físicos, es decir, la impresión de un archivo se trata de manera similar a la visualización en una terminal de video.

AT&T ofreció en la versión de UNIX Sistema V Versión 2 (1985) protección de archivos durante cortes de potencia, bloqueo de archivos y registros para uso exclusivo de un programa. En 1987, AT&T ofreció, bajo el UNIX Sistema V Versión 3.0, bajo un enfoque de redes simple y consistente, capacidades como STREAMS, utilizado para construir software de redes, y la interfase a nivel de transporte (TLI), utilizada para construir aplicaciones que utilizan redes. En la Versión 3.1 se incorporaron funcionalidades para la recuperación de archivos. Dentro de la versión 4.2BSD (Berkeley, 1983) se introdujeron características de red, incluyendo redes TCP/IP (Transmission Control Protocol / Internet Protocol), así como un nuevo sistema de archivos que acelera el acceso a los mismos. Sun Microsystems añadió características de red tales como el sistema de archivos de red (NFS, Network File System) a la versión 4.2BSD; éstas características han sido incorporadas a UNIX Sistema V Versión 4.

II.5.1 Manejo del Software de Entrada/Salida

La Versión 4 incluye ampliaciones que hacen más fácil a los programadores utilizar nuevos dispositivos. Esto se lleva a cabo utilizando el Device-Kernel Interface (DKI) entre el Kernel y el software del dispositivo. Esta Versión introduce una interfase común para acceso al sistema, que puede ser a través de una terminal, a través de una red de área local (LAN, Local Area Network), o por acceso remoto. Esto lo lleva a cabo el Service

Access Facility, que proporciona un mecanismo de acceso consistente y monitorea los puntos de acceso externos.

Una de las áreas claves de las ampliaciones de la Versión 4 son las redes. Se han incorporado muchas capacidades de red de los Sistemas BSD y SunOS. Las capacidades de red del Sistema BSD incluyen al paquete Internet TCP/IP, que a su vez incluye órdenes para transferencia de archivos, presentación remota y ejecución remota. Esta versión también incluye la interfase de red de enchufes (sockets) BSD, que se utiliza para construir aplicaciones basadas en red. Del SunOS, se han incorporado características de redes que incluyen al sistema de archivos de red (NFS) para compartir archivos remotos, el protocolo estándar de industria Remote Procedure Call (RPC), para que un procedimiento sobre una computadora llame a un procedimiento sobre una computadora remota, y el External Data Representation (XDR), que especifica un formato para los datos que permite su intercambio entre sistemas que incluso tengan diferente arquitectura de hardware, diferente sistema operativo y diferente lenguaje de programación. Dentro de las nuevas características de la Versión 4 se incluye la de Network Selection (selección de red), que permite a una aplicación seleccionar una red sobre la cual comunicarse; un mecanismo de traslación nombre-a-dirección utilizado para que las máquinas cliente determinen la dirección de los servidores que proporcionan servicios particulares; el Service Access Facility (SAF), que proporciona un único proceso que gestiona todos los accesos

externos a un sistema; y uucp (Unix to Unix Communication Protocol) independiente del medio, que permite la transferencia de archivos sobre cualquier clase de red.

La Versión 4 también proporciona muchas ampliaciones al sistema de archivos y a las operaciones sobre archivos; el sistema de archivos tiene una arquitectura que ofrece flexibilidad y modularidad con el resto del núcleo. Así mismo se ha implementado una nueva arquitectura **"file system switch"** (conmutación de sistema de archivos); éste es el Virtual File Switch (VFS) que procede del Sistema BSD. Se pueden soportar sistemas de archivos con formatos y características ampliamente variables. Entre los sistemas de archivos soportados están los sistemas de archivos tradicionales, el **Fast File System** (sistema de archivo rápido) de BSD, y los sistemas de archivos remotos. En la Versión 4 se pueden configurar rápidamente nuevos sistemas de archivos y tipos, tales como el sistema de archivos DOS (Disk Operating System). Esta es una capacidad única entre los sistemas operativos comerciales.

II.5.2 Manejo de Interrupciones.

Algunos procesos ejecutan programas que ejecutan comandos tecleados por un usuario. Otros procesos son parte del sistema y realizan tareas como solicitudes de servicios de archivo o el manejo de los detalles de la ejecución de un disco o de una unidad de cinta.

Cuando ocurre una interrupción del disco, el sistema toma la decisión de suspender la ejecución del proceso corriente y corre el proceso del disco, que se bloqueó cuando esperaba esa interrupción. Cuando se ha leído el bloque de disco o se ha tecleado el caracter, el proceso que lo espera se desbloquea y puede ser ejecutado nuevamente.

Todo el manejo de las interrupciones y los detalles del inicio y suspensión reales de los procesos se ocultan en el planificador de procesos. En asociación con cada clase de dispositivo de E/S (por ejemplo discos flexibles, discos duros, relojes, terminales) está una localidad en la proximidad de la parte inferior de la memoria llamada **vector de interrupción**.

Este contiene la dirección del procedimiento de servicio de las interrupciones. Supongamos que un proceso de usuario está en ejecución cuando ocurre una interrupción del disco.

El contador de programa, la palabra de condición del programa y posiblemente uno o más registros se meten en la pila por medio del hardware de interrupciones. Después, la computadora salta a la dirección especificada en el vector de interrupción del disco. El procedimiento de servicio a interrupciones inicia su operación guardando todos los registros en el registro de la tabla de procesos del proceso corriente. El número del proceso corriente y un apuntador a su registro se guardan en variables globales de

manera que puedan encontrarse rápidamente.

Más tarde, la información depositada por la interrupción se retira de la pila y el apuntador de esta pila se asigna a una pila temporal que es utilizada por el manejador del proceso. A continuación se construye un mensaje que se enviará al proceso del disco, el cual se bloqueará para esperarlo. El mensaje indica que ocurrió una interrupción, para distinguirlo de los mensajes que provienen de procesos del usuario que solicitan la lectura de bloques de disco y cosas semejantes.

El estado del proceso del disco se cambia ahora de bloqueado a listo, y se llama al planificador de procesos.

Los controladores de dispositivos tienen algunos registros que se utilizan para establecer comunicación con la unidad central de procesamiento. En los computadores IBM PC se utiliza un espacio de dirección especial para E/S, donde a cada controlador se le asigna una porción determinada de ella.

A continuación se muestran las direcciones de E/S y los vectores de interrupción asignados a algunos de los controladores de la IBM PC.

Controlador de E/S	Direcciones de E/S	Vector de Interrupción
Reloj	040 - 043	8
Teclado	060 - 063	9
RS232 secundaria	2F8 - 2FF	11
Disco duro	320 - 32F	13
Impresora	378 - 37F	15
Despliegue monocromático	380 - 3BF	-
Despliegue de color	3D0 - 3DF	-
Disco flexible	3F0 - 3F7	14
RS232 primaria	3F8 - 3FF	12

Fig.II.5.1 Ejemplos de controladores, direcciones de E/S y sus vectores de interrupción en la IBM PC.

La asignación de direcciones de E/S a dispositivos se hace por medio de la lógica de decodificación de bus asociada con el controlador. El sistema operativo efectúa la E/S escribiendo comandos en los registros del controlador. Muchos de los comandos tienen parámetros que también se cargan en los registros del controlador. Cuando se ha aceptado un comando, la unidad central de proceso (CPU) puede dejar solo al controlador y realizar otro trabajo. Cuando el comando se ha completado, el controlador produce una interrupción con el fin de permitir que el sistema operativo tome el control de la CPU y compruebe los resultados de la operación. La CPU determina los resultados y la condición del dispositivo leyendo uno o más bytes de información de los registros del controlador.

II.5.3 Manejadores de Dispositivos.

En el Sistema UNIX todos los procesos tiene dos partes: una parte del espacio de usuario y una parte del espacio del kernel. La parte del espacio de usuario llama a la parte del espacio del kernel. El sistema de archivos llama al manejador del dispositivo como procedimiento. Todo el sistema operativo es parte de cada proceso.

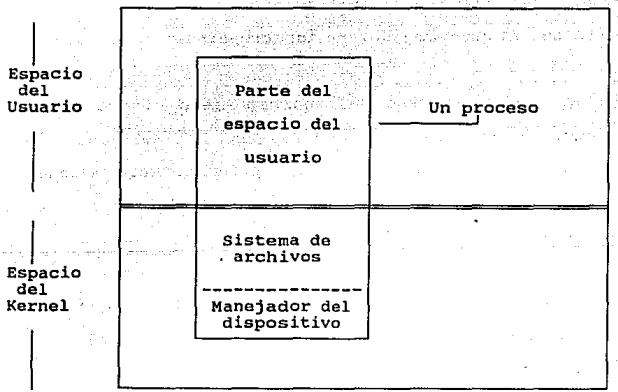


Fig. II.5.2 Comunicación entre el usuario y el sistema.

Cuando se hace una llamada al sistema, el sistema operativo cambia de la parte del espacio del usuario a la del espacio del kernel en forma un tanto mágica. Esta estructura es un residuo del diseño de MULTICS, en el cual el cambio es simplemente una llamada ordinaria al procedimiento, en vez de una trampa seguida de guardar el estado de la parte del usuario, como se hace en UNIX.

Los manejadores de dispositivos en UNIX son simplemente procedimientos del kernel que son llamados por la parte del espacio del kernel del proceso. Cuando un manejador necesita esperar una interrupción, éste llama a un procedimiento del kernel que lo bloquea hasta que algún manejador de interrupciones lo desbloquea.

Nótese que es el proceso del usuario mismo el que se está bloqueando aquí, puesto que las partes del kernel y del usuario son en realidad partes diferentes del mismo proceso. Este método es eficiente, puesto que las llamadas a los procedimientos son mucho más rápidas que la emisión de mensajes.

En la implementación del UNIX V Versión 4 de Santa Cruz Operation, los archivos que controlan a los dispositivos son ubicados en el directorio /dev.

A continuación se muestra una lista parcial de dichos manejadores de dispositivos.

Dispositivo UNIX	Nombre
/dev/console	system console
/dev/rdsk/*	raw devices
/dev/dsk/0s0	entire disk on drive 0
/dev/dsk/0s1	first disk partition on drive 0
/dev/dsk/0s2	second disk partition on drive 0
/dev/dsk/1s0	entire disk on drive 1
/dev/dsk/1s1	first disk partition on drive 1
/dev/dsk/1s2	second disk partition on drive 1
/dev/dsk/f05d9	360 floppy drive 0
/dev/dsk/f05q	720 floppy drive 0
/dev/dsk/f05h	1.2 Mb floppy drive 0
/dev/f03h	1.44 Mb floppy drive 0
/dev/lp	lineprinter
/dev/kmem	kernel virtual memory
/dev/mem	physical memory
/dev/null	null device
/dev/rmt0	QIC tape device
-	QIC-40 tape device
-	minicartridge tape device
/dev/root	root file structure
/dev/swap	swap area
/dev/ttynn	terminals

Lista II.5.3 Manejadores de dispositivos en SCO UNIX

II.5.4 Software de E/S independiente del dispositivo.

Un archivo es la estructura básica utilizada para almacenar información en UNIX. Es una secuencia de bytes que se almacenan en algún lugar de un dispositivo de memoria, tal como un disco. Un archivo no tiene porque ser almacenado en un único sector físico de un disco, sino que UNIX mantiene en una secuencia especifica la información. De esta forma, un archivo puede contener cualquier clase de información que se pueda representar como una secuencia de bytes.

El sistema UNIX contempla los archivos de una manera extremadamente simple y general dentro de un modelo único. Ve de la misma manera los directorios, los archivos ordinarios, los dispositivos, tales como impresoras y discos, y los teclados y terminales de video. El sistema de archivos oculta al usuario detalles del hardware subyacente.

Para UNIX existen cuatro tipos diferentes de archivos: archivos ordinarios, directorios, vinculos simbólicos y archivos especiales.

Un archivo especial representa un dispositivo físico. Puede una terminal, un dispositivo de comunicaciones, o una unidad de almacenamiento como un disco. Desde la perspectiva del usuario, el Sistema UNIX trata los archivos especiales como archivos

ordinarios; esto es, puede leer o escribir los dispositivos exactamente como lee y escribe los archivos ordinarios. Se pueden tomar los caracteres pulsados en el teclado y escribirlos de la misma forma en un archivo ordinario o una pantalla de video. El Sistema UNIX toma las órdenes de lectura y escritura y produce la activación del hardware conectado al dispositivo. Esta forma de tratar al hardware del sistema tiene una consecuencia importante para los usuarios, puesto que UNIX trata casi todo como si fuese un archivo, el usuario no necesita conocer las particularidades del hardware de la computadora. Estos archivos asociados a dispositivos residen normalmente bajo el subdirectorio /dev del árbol del sistema de archivos. Además de hacerse cargo de la interfase con los manejadores, separación, asignación de bloques, etc., el sistema de archivos también maneja la protección y el manejo de los inodos, directorios y sistemas de archivos montados.

El sistema de archivos contiene archivos y directorios que son representados por estructuras especiales denominadas "inodes" (**inodos**) y "data blocks" (**bloques de datos**). Estas estructuras hacen posible que el sistema operativo pueda crear y mantener los sistemas de archivos.

Un **bloque de datos** es una unidad de 1024 bytes de datos guardados en disco, que puede contener ya sea componentes de un directorio o datos de un archivo. Un componente de un directorio consiste de un número de inodo y de un nombre de archivo. Para

minimizar el tiempo de búsqueda de datos en un disco duro, los bloques de datos utilizados en forma más reciente son guardados en estructuras de memoria, denominados buffers, para mejorar el desempeño del sistema operativo. Cuando los datos suficientes son acumulados para escribir uno o más bloques de disco completos, el buffer es escrito a disco. Un bloque de datos especial, llamado el super bloque, contiene información general acerca del sistema de archivos, no conteniendo la ubicación de un pedazo en particular de un archivo. El super bloque contiene la información necesaria para "montar" un sistema de archivos y accesar su información. Contiene también el tamaño del sistema de archivos, el número de inodos libres, e información del espacio libre disponible. La información es leída de la versión en disco del super bloque cuando el sistema de archivos es montado, y se mantiene y modifica en memoria conforme haya actividad en el sistema. La información se vuelve a escribir a disco a intervalos de tiempo regulares por el comando update, el cual es ejecutado normalmente por los procedimientos en /etc/rc2 cuando el sistema es activado. El comando update llama al comando sync cada 30 segundos, el cuál forza que la versión en memoria del super bloque y los buffers sean escritos a disco. Si el sistema se detiene y la información almacenada en disco no fue actualizada en ese momento, el sistema de archivos puede llegar a corromperse.

Un **inodo** contiene información acerca de un archivo, que incluye su ubicación, su tamaño, el tipo de archivo, y el número de

componentes de directorios ligados a él. Un inodo no contiene el nombre de un archivo, estos nombres son guardados en los directorios. El inodo contiene las localidades de todos los datos que componen un archivo, para que el sistema operativo pueda recolectarlos cuando sea necesario.

II.5.5 Software de E/S en el espacio del usuario.

El Sistema UNIX trata la entrada y salida de una forma simple y consistente, utilizando entrada y salida estándar.

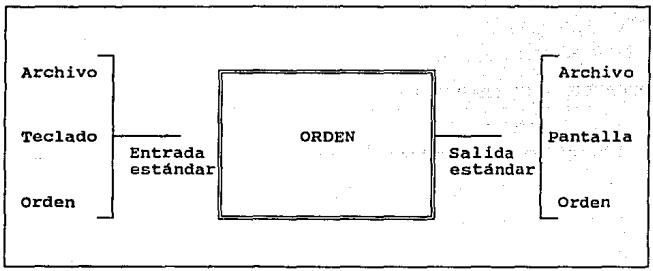


Fig. II.5.4 Modelo para entrada y salida estándar.

Una de las características más típicas del Sistema UNIX es el "pipe". Esta característica es un mecanismo general que permite utilizar la salida de una orden como entrada a otra. A continuación se listan los operadores de redirección del shell.

Símbolo	Ejemplo	Función
<	cmd < file	Toma la entrada para cmd de file
>	cmd > file	Envía la salida de cmd a file
>>	cmd >> file	Añade la salida de cmd a file
¦	cmd1 ¦ cmd2	Ejecuta cmd1 y envía la salida a cmd2

Fig. II.5.5 Operadores de redirección del shell

A cualquier archivo está asociado un nombre de archivo que lo identifica de manera única, y por este nombre se puede hacer referencia a él. Cabe agregar que a un archivo se le pueden asociar otros nombres, proceso conocido como vínculo (link).

En cuanto a la seguridad de acceso a los archivos, UNIX prevé tres clases de autorizaciones que corresponden con las tres clases del usuario: el propietario (owner) del archivo, el grupo (group) al que pertenece el propietario, y los otros (other) usuarios del sistema. Por cada una de estas clases se proporcionan tres tipos de permisos: lectura (r), escritura (w) y ejecución (w). Estos permisos son modificables mediante instrucciones del sistema operativo (chmod, chgrp, chown). Por otro lado, en cuanto a la E/S a redes, se proveen diferentes servicios que enriquecen el uso del sistema. Uno de ellos es uucp (UNIX to UNIX Copy). Este sistema se puede utilizar para transferir archivos entre computadoras o para ejecutar una orden en una máquina remota.

Las órdenes de usuario que junto a uucp proveen este servicio se muestran en la figura II.5.6. Una característica de uucp es que está basado en comunicaciones punto a punto entre equipos, lo cual lo hace lento y poco sofisticado, aunado a que no está disponible en muchos sistemas en la actualidad (sistemas no UNIX). UNIX Sistema V Versión 4 incluye capacidades de red que pueden ser utilizadas para proporcionar una variedad de servicios sobre una red de alta velocidad. Estas capacidades se proporcionan mediante el paquete Internet TCP/IP, basado en la Internet Protocol Suite.

Orden	Acción
cu	Llama a otro sistema y gestiona un diálogo, incluyendo transferencia de archivos ASCII
ct	Marca el número de una terminal remota y genera un proceso de login
uucp	Copia archivos de un sistema UNIX a otro
uuto	Copia archivos de un sistema a otro, permitiendo al sistema remoto controlar el acceso del archivo
uupick	Busca en un directorio predeterminado los archivo que le han sido enviados y pregunta que hacer con ellos
uux	Ejecuta una orden en un sistema remoto
uuname	Imprime los nombres de todos los sistemas conocidos a uucp
uustat	Visualiza el estado de los trabajos uucp actuales, cancela trabajos previos, proporciona información sobre el rendimiento del sistema

Fig. II.5.6 Ordenes asociadas a uucp

Utilizando este paquete, se pueden realizar transferencias de archivos ASCII o binarios, ejecución remota de órdenes, correo electrónico, y presentación remota. Una de las bondades en el uso de TCP/IP es que está disponible en plataformas de equipo heterogéneas, haciéndolo más versátil.

Algunos de los comandos de TCP/IP se muestran en la figura II.5.7.

Comando	Propósito
ftp	Transferencia de archivos entre máquinas corriendo TCP/IP; estas máquinas pueden o no ejecutar el mismo sistema operativo
rcmd	Ejecución remota de órdenes en otro equipo con UNIX
rcp	Copiado de archivos entre dos sistemas UNIX
rlogin	Presentación remota en otro sistema UNIX
ruptime	Despliegue del estado de equipos en la red local
rwho	Despliegue de la relación de usuarios conectados a equipos en la red local
telnet	Presentación remota en un equipo corriendo TCP/IP, sin que ejecuten el mismo sistema operativo

Fig. II.5.7 Comandos más usuales de TCP/IP

La Versión 4 contiene un amplio rango de facilidades que pueden ser utilizadas para desarrollar aplicaciones de comunicación en red, entre ellas:

STREAMS, es un mecanismo normalizado para construir aplicaciones en red; incluye llamadas al sistema, recursos del núcleo y utilerías, para servicios y recursos de comunicación, entre un proceso usuario y un controlador en el núcleo, que interactúa directamente con el hardware de comunicaciones. STREAMS retiene las familiaridades llamadas de sistema de E/S de UNIX Sistema V e introduce nuevas clases de E/S. Cuando se utiliza, los módulos de red pueden ser reutilizados en la implementación de diferentes pilas de protocolos; los módulos de red pueden ser reemplazados por nuevos módulos con la misma interfase de servicio; los módulos de red pueden ser transportados a nuevas máquinas y las aplicaciones de red pueden ser utilizadas transparentemente con respecto a los servicios de red sobre los que corren.

TLI (Transport Layer Interface), proporciona a los usuarios comunicación de red confiable de extremo a extremo, de modo que el usuario puede construir aplicaciones que son independientes de la red física; las aplicaciones no necesitan saber cual es el medio subyacente o cuáles son los protocolos de las capas inferiores. Existen dos modos de servicio soportados por TLI: Circuitos virtuales (servicio orientado a conexión), y el Servicio sin conexión, que soporta transferencia de datos en paquetes.

Enchufes (sockets), son una interfase de programación utilizada para construir aplicaciones de red que llegan a UNIX Sistema V Versión 4 desde el Sistema BSD; los enchufes fueron originalmente incorporados al sistema BSD para soportar

comunicación de red TCP/IP. Han sido utilizados para programar comunicaciones de circuito virtual y comunicaciones cliente-servidor.

Las aplicaciones que requieren acceso directo a la capa de transporte de una red, pueden escribirse utilizando TLI o enchufes. Ambos manejan las comunicaciones interproceso generalizando la E/S de archivos; de igual forma soportan los modos orientados a conexión y sin conexión.

Por otro lado, el servicio NFS (Network File System), que hace uso de TCP/IP, permite al usuario exportar sistemas de archivos en la red, para que usuarios en otros equipos de la red puedan usarlos como si fueran sistemas de archivos locales, y viceversa, evitando el tener que recurrir a una presentación remota (login) o a efectuar una copia de archivos remotos.

Por el hecho de estar basado en TCP/IP, NFS puede utilizarse entre equipos que, sin tener que ejecutar el mismo sistema operativo, soporten el uso de TCP/IP.

El servicio de traslación nombre-a-dirección (BIND, Berkeley Internet Name Domain) proporciona la manera de enrutar información entre equipos que, a diferencia de TCP/IP, no requiere de mantener en todos los equipos un archivo de definición de todos los equipos disponibles en la red.

II.5.6 Deadlock (Interbloqueo)

En cuanto al interbloqueo, UNIX simplemente ignora, en general, esta situación. El único sitio en que pueden ocurrir interbloqueos es con lo recursos compartidos implícitos, como las entradas de la tabla de procesos, entradas de la tabla de inodos, etc. Ninguno de los algoritmos de interbloqueo que se conocen puede trabajar con recursos como éstos que no se soliciten en forma explícita. En realidad lo anterior no es estrictamente verdadero.

Existen algunos sitios donde se debe tener considerable precaución para evitar problemas. El principal es la interacción entre el sistema de archivos y el manejador de la memoria. El manejador de la memoria envía mensajes al sistema de archivos para leer un archivo binario (programa ejecutable); si el sistema de archivos no está ocioso cuando el manejador de la memoria intenta enviarle un mensaje, el manejador de la memoria se bloqueará. Si el sistema de archivos debe después intentar enviar un mensaje al manejador de la memoria, también descubriría que la cita falla y se bloquearía, provocando un interbloqueo. Este problema se ha evitado mediante la construcción del sistema en forma tal que el sistema de archivos nunca envíe mensajes de solicitud al manejador de la memoria, solo contestaciones, con la única excepción de que al activarse, el sistema de archivos informa su tamaño al manejador de la memoria, que se garantiza que lo está esperando.

II.6 Manejo de equipo periférico

II.6.1 ¿ Que es un manejador de dispositivos ?

Para cada dispositivo periférico (como una terminal o una impresora), debe existir un manejador que provea la interfase de software entre el dispositivo y el sistema operativo. De ahí la importancia de definir primeramente lo que es un manejador de dispositivos:

- Un conjunto de rutinas que comunican con un dispositivo físico y proveen una interfase uniforme hacia el kernel del sistema operativo.
- Un manejador de flujo y control de datos entre un programa de usuario y un dispositivo periférico.
- Una sección del kernel, definida por el usuario, que permite a un programa o a un dispositivo físico aparecer como un archivo en el directorio /dev para el resto del software de la computadora.

Normalmente los manejadores de dispositivos crean código que será agregado en el kernel del sistema, pasando a ser parte integral del nuevo kernel. Además, cada vez que el sistema operativo es cargado en memoria para iniciar sus funciones, el manejador también ocupará un espacio en memoria permanentemente, hasta que el sistema sea dado de baja. Esto hace que un manejador sea muy eficiente y rápido en dar respuesta, pero que se corra el

riesgo de dañar algunas de las funciones básicas del sistema.

II.6.2 Agregando un Manejador

Dado que el código del kernel aumenta cada vez que un manejador es instalado, antes de poder instalar un nuevo manejador de dispositivos, es necesario mencionar las precauciones que deberán ser tomadas:

- Cuando se desee agregar algún manejador al kernel actual, es necesario asegurarse de que los recursos y las direcciones en memoria, no altere el uso que, de éstos, los manejadores existentes están haciendo, ni altere el medio ambiente y el funcionamiento de lo actualmente contiene el kernel.
- El manejador debe permitir la portabilidad y el fácil manejo de sus funciones.
- Siempre se deberá respaldar el software existente antes de instalar un nuevo manejador.
- Se deberá conocer los parámetros de sintonización para dar a los usuarios finales la capacidad de utilizar al máximo las características del manejador. Siempre es adecuado revisar las configuraciones con las que, por default, se instala un manejador y verificar que estas sean las más adecuadas para su funcionamiento, tomando en cuenta la carga de trabajo del equipo específico.
- Deberá documentarse su propósito y modo de operación.
- Se deberá probar para asegurarse que esta trabajando

correctamente y monitorear su desempeño durante las horas de más demanda.

El siguiente diagrama muestra la relación entre un manejador y el sistema operativo:

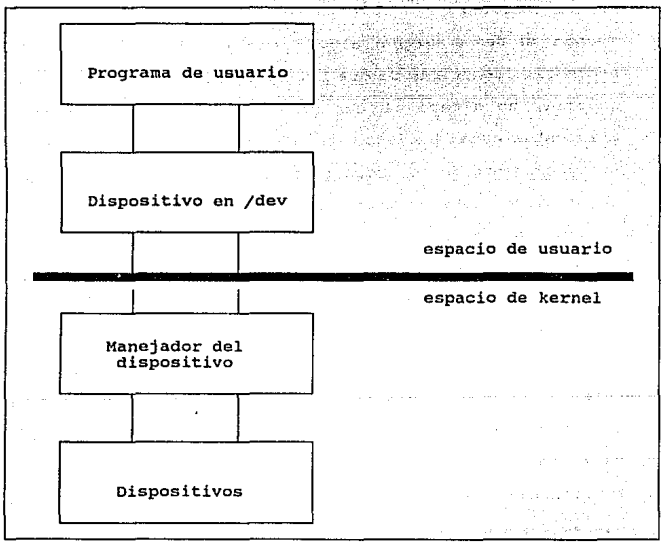


Fig. II.6.1 Relación entre un manejador y el sistema operativo

II.6.3 Rutinas de un manejador de dispositivos

Un manejador consiste en rutinas propias del kernel que son utilizadas por el, y en rutinas del manejador mismo. Las primeras están implementadas en el sistema operativo y son utilizadas como llamadas externas que utiliza para su funcionamiento interno. Las segundas son rutinas escritas para cubrir necesidades de control específicas, manejar la entrada y la salida y ofrecer servicios específicos para los dispositivos.

Las funciones más comunes de un manejador incluyen servicios para dispositivos de bloques, dispositivos de caracteres, para el video, así como para "streams".

Un manejador puede ser escrito ya sea para dispositivos de bloque o para dispositivos de caracteres dependiendo de la transferencia de información que se llevara a cabo entre el dispositivo y el sistema operativo, ya que de necesitarse un buffer, se deberá usar un dispositivo de bloques y en caso en el que la información no tenga una longitud fija, ya sea que utilice el buffer o no, se deberá utilizar un dispositivo de caracteres como es el caso de las terminales, impresoras y tarjetas de red.

II.6.4 Manejo de dispositivos de bloques y caracteres.

La operación de ambos dispositivos dentro del sistema UNIX, se

reduce principalmente a cinco llamadas al sistema que son: abrir (open), cerrar (close), leer (read), escribir (write) y buscar (seek).

Por ejemplo para abrir un archivo dentro del sistema se utilizaría la instrucción:

```
fd = open (nombre_del_archivo, modo)
```

donde el modo indica si el archivo se abrirá de lectura, de escritura, o con ambos permisos. El fd es el descriptor de archivo que identifica a este y se utiliza en referencias posteriores.

En el caso de la lectura y la escritura de un archivo, se utilizan las instrucciones:

```
nobytesread = read (fd, buffer, nobytesdesired)
```

además de:

```
nobyteswritten = write (fd, buffer, nobytesdesired)
```

Es necesario mencionar que en el caso de una lectura existen tres posibilidades:

- Si es la primera lectura, entonces la lectura es secuencial desde el principio del archivo.
- Si el archivo ha sido leído, entonces la lectura actual obtiene los datos que siguen de la lectura anterior.
- Si acaba de realizarse una búsqueda, entonces la lectura es

secuencial tomando en cuenta el desplazamiento especificado en la llamada.

Esto mismo es válido para la escritura, tomando en cuenta que toda lectura y escritura es secuencial, pero el efecto de acceso directo se logra usando la llamada de búsqueda para ajustar el desplazamiento dentro del archivo, de la manera siguiente:

seek (fd, desplazamiento, tipo_de_desplazamiento)

donde el tipo de desplazamiento especifica si es relativo o absoluto y si esta dado en unidades de bytes o en bloques de 512 bytes.

La operación de búsqueda funciona bien tanto en cinta magnética como en disco, pero esta claro que los programas de acceso directo son mucho mas lentos en cinta. Los archivos se cierran escribiendo: **close (fd)**

Estos son todos los comandos disponibles para el manejo de dispositivos secuenciales, y aunque pudiera parecer pocos comparados con los Sistemas Operativos grandes, utilizando estos, se han implementado sofisticados sistemas de administración de datos.

La misma manera de operar se utiliza para dispositivos físicos, los cuales se representan en el sistema como archivos

especiales dentro de una estructura denominada "Sistema de Archivos", esto significa que los usuarios no pueden escribir rutinas dependientes de dispositivos a menos que las dependencias se reflejen directamente en la operación. Por ejemplo, los programas no tienen que abrir sus archivos normales de entrada o de salida ya que estos están asignados a su terminal de usuario, la cual es abierta por el sistema.

II.6.5 Operaciones de flujo contra operaciones de registro

Todas las operaciones realizadas dentro de los sistemas UNIX, están orientadas a manejar el flujo en lugar de hacerlo manejando registros, como lo hacen la mayoría de los sistemas. Un flujo es una secuencia de bytes terminados por un delimitador (carácter de fin de flujo), lo que facilita la implementación de la independencia de dispositivos y la transparencia entre archivos, dispositivos y tuberías (pipes). También otorga flexibilidad a los usuarios para trabajar con una gran cantidad de datos.

A pesar de ser esta la manera de operación, si el usuario lo desea. puede implementar con bastante facilidad una operación por registros, pues para hacerlo solamente necesita especificar una longitud constante en todas las lecturas y escrituras. La operación de acceso directo con registros de longitud fija se implementa multiplicando la longitud del registro por el número de registros y llamando a la función de búsqueda para situar el

apuntador en el registro deseado. Los registros de longitud variable pueden ser utilizados precediendo cada registro con un campo de tamaño de registro de longitud fija. Por ejemplo, usando un campo de tamaño de dos bytes, los registros de longitud variable pueden leerse por pares de lecturas como sigue:

```
read (fd, tamaño, 2)
read (fd, buffer, tamaño)
```

II.6.6 Modo de Operación de Dispositivos Caracteres

En general, un dispositivo con un conjunto de registros fijos de direccionamiento aleatorio puede ser considerado un dispositivo de bloques; cualquier otro tipo de dispositivo será considerado de caracteres.

Los manejadores de los dispositivos de caracteres puede comunicarse directamente con los programas de los usuarios. El acceso al dispositivo comienza cuando un programa de usuario hace un requerimiento de transferencia de datos entre una sección de la memoria y un dispositivo específico. En ese momento, el sistema operativo transfiere el control al manejador del dispositivo apropiado y el programa del usuario se encarga de pasarle los parámetros del requerimiento, para que la tarea sea realizada. Sin embargo, el sistema operativo tiene una función mínima en el requerimiento, ya que la transferencia de datos es una transacción propia del proceso del usuario y del manejador del dispositivo.

II.6.7 Modo de Operación de Dispositivo de Bloques

Los manejadores de dispositivos de bloques requieren de más participación del sistema operativo para realizar sus tareas. Parte de la razón de esta participación adicional es a causa de que los dispositivos de bloque transfieren sus datos en bloques de tamaño fijo y son normalmente capaces de hacer accesos aleatorios.

En un dispositivo de bloque, la transferencia de datos entre el kernel y el dispositivo se hace utilizando la memoria principal, en múltiplos de un bloque de memoria de tamaño "BSIZE", que es el tamaño fijo de bloque del sistema, y un buffer para el almacenamiento temporal de la información.

La información contenida en el encabezado del buffer y la información de los bloques que la memoria esta utilizando, forman una lista ligada. La información que contiene el encabezado del buffer se encuentra definida en una estructura de datos denominada "buf". Uno de los campos de este encabezado apunta al bloque de datos y al siguiente y último elemento de una lista ligada, mientras que los otros campos proveen información sobre el estado de los procesos.

El mecanismo de buffer simplifica el uso excesivo (overhead) del dispositivo debido a que es manejado por el sistema, y no necesita una rutina de asignación y manejo de memoria. En suma, el

sistema transfiere el dato hacia y desde el dispositivo.

Frecuentemente, los dispositivos de bloque soportan interfases para caracteres para realizar funciones de entrada y salida en modo "raw" en las cuales el dato es transferido entre un programa de usuario y el dispositivo sin hacer uso de un buffer.

Este tipo de uso del dispositivo es común en los programas de respaldo que copian sectores de datos desde el disco.

Los dos factores que distinguen las operaciones de bloque de las operaciones de caracteres son:

- El tamaño del requerimiento de transferencia de datos desde el kernel hacia el dispositivo es siempre el múltiple del tamaño del bloque del sistema (llamado BSIZE), sin importar el tamaño del proceso de usuario original. Un requerimiento de usuario sencillo puede generar muchos requerimientos del sistema hacia un manejador. El valor del bloque del sistema (BSIZE) esta definido en el archivo **sys/fs/s5param.h** y varia dependiendo del sistema de archivo. El tamaño del bloque físico del dispositivo puede ser menor que el tamaño del bloque del sistema en cuyo caso el manejador del dispositivo iniciara múltiples transferencias físicas para cubrir un solo bloque lógico.
- Las transferencias nunca son hechas directamente al área de memoria que ocupa el proceso del usuario. Son siempre

almacenadas en buffers implementados en memoria cache. Los requerimientos de entrada y salida de un programa son satisfechos directamente por estos buffers. En el sistema V de UNIX el manejador se encarga de leer y escribir de los buffers tantas veces como sea necesario. El kernel maneja estos buffers para realizar servicios como el bloqueaje y desbloqueaje de datos y el acceso a la cache.

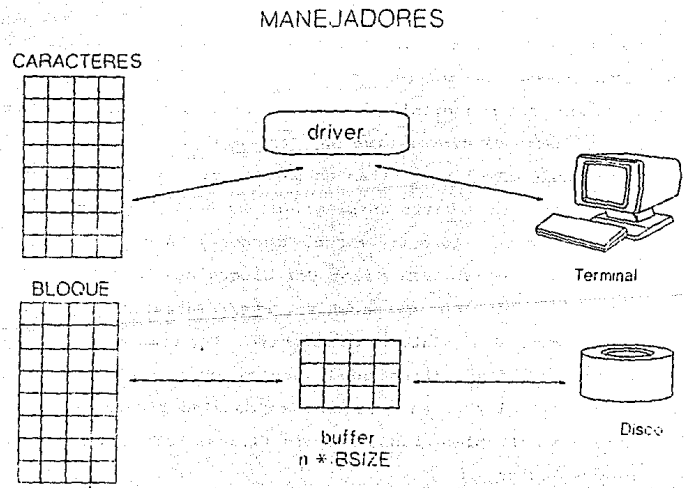


Fig. II.6.2 Operación de los dispositivos

II.6.8 Archivos especiales de dispositivos.

Para un usuario del sistema V de UNIX, un dispositivo puede ser tratado como si fuera un archivo. Por definición, un archivo consiste de una secuencia ordenada de bytes. Los archivos que contienen datos son llamados archivos regulares y los archivos que representan dispositivos son llamados archivos especiales de dispositivos. Cada archivo tiene al menos un nombre, y los nombres de los archivos especiales están, por convención puestos en un directorio llamado /dev.

Cada archivo especial tiene un número de dispositivo que identifica de manera única a ese dispositivo. Dicho número consiste de dos partes, el número mayor y el número menor.

El número mayor le indica al kernel cual es el manejador del dispositivo que se encarga de los requerimientos para ese archivo especial, y solo varía entre un dispositivo y otro diferente, un disco y una impresora por ejemplo. El número menor puede ser usado por el manejador para dar más información acerca de una unidad particular sobre la que tiene control. Por ejemplo, si se cuenta con una tarjeta con ocho puertos seriales, todos los puertos tienen el mismo número mayor de dispositivo, pero tienen ocho diferentes numeros menores. El número menor del dispositivo normalmente describe el número de la unidad. Sin embargo, un manejador de dispositivo puede dedicar algunos de los bits en el número menor

para indicar opciones especiales, como el uso de doble densidad en el caso de un disco flexible.

II.6.9 Examinando el numero del dispositivo.

Antes de que el proceso de usuario pueda hacer un requerimiento de entrada o salida, debe primero haber abierto un archivo especial de dispositivo. Un archivo especial se ve como un archivo ordinario grabado en un disco, excepto que fué creado por una utilería llamada **mknod**. El archivo aparece en el directorio que tiene campos de propietario y permisos de archivos, como los tiene un archivo normal, pero no contiene un tamaño de archivo. En lugar de tener el tamaño del archivo aparece el número mayor y el número menor asociado con él. El comando **ls -l** despliega una información como la siguiente:

```
brw------- 2 sysinfo   sysinfo 1, 15 Aug 21 05:34 /dev/hd01
crw-rw-rw- 1 bin       bin     5,  0 Aug  7 18:20 /dev/tty1a
crw-rw-rw- 1 bin       bin     5,  1 Aug 10 17:33 /dev/tty1b
```

Aquí el dispositivo **/dev/tty1a**, tiene un número mayor de 5 y un número menor de 0. El dispositivo **/dev/tty1b** tiene un número mayor de 5 igual al del dispositivo anterior, y un número menor de 1. El dispositivo **/dev/hd01** tiene un número mayor de 1 y un número menor de 15.

II.6.10 Accesando un dispositivo

Cuando un proceso de usuario abre uno de los archivos especiales, el kernel reconoce de que archivo se trata y utiliza el número mayor para indexar una tabla de puntos de entrada.

Si el archivo especial esta designado como un dispositivo de carácter, este usa la tabla cdevsw, pero si esta designado como un dispositivo de bloque, utilizará la tabla **bdevsw**. Estas dos tablas están definidas en el archivo **sys/conf.h**.

Cuando un proceso utiliza la función **open** o la función fopen sobre un archivo, el kernel hace una llamada al punto de entrada donde fué abierto el dispositivo utilizando la tabla **cdevsw** o la tabla **bdevsw** proporcionando el número mayor y menor del dispositivo como un argumento.

Estos archivos especiales tienen nombres significativos y todos deben residir en el directorio **/dev**.

II.6.11 Como los datos son pasados al Manejador

La porción tiempo-tarea del manejador del dispositivo tiene acceso a una área denominada "u-área" del usuario, desde donde es mapeada dentro de un espacio de la dirección del kernel, donde

están ubicados los servicios que el manejador proporciona. Las rutinas del kernel que atienden el requerimiento de proceso del usuario proporciona información que describe el requerimiento en la "u-área".

EL siguiente diagrama ilustra como los parámetros de "u-área" son pasados al manejador:

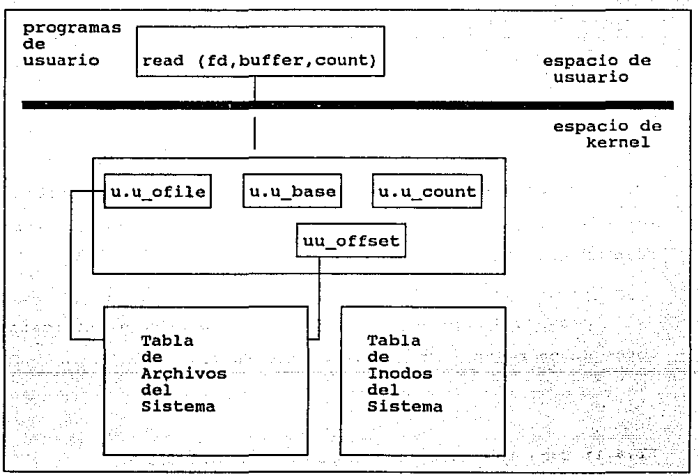


Fig. II.6.3 Area de usuario y Kernel

Los parámetros pasados en la "u-área" son:

Parámetro	Contenido
u.u_base	Especifica la dirección en los datos del usuario para leer o escribir en una transferencia.
u.u_count	Especifica al número de bytes para transferir.
u.u_offset	Especifica la dirección de inicio dentro del archivo para transferencia.
u.u_segflg	Indica la dirección de la transferencia. Los posibles valores son 0, que indica una transferencia entre el kernel y un espacio para datos de usuario; 1 indica una transferencia del espacio del kernel entre las direcciones del kernel, y 2 para indicar una transferencia entre el kernel y el área de instrucciones del usuario.

Adicionalmente para los parámetros pasados en la "u_área", las rutinas de entrada y salida del kernel pasan el número del dispositivo, que contiene el número mayor y el número menor, como un parámetro para el manejador. De esta manera, el manejador tiene toda la información que necesita para realizar la tarea.

Solamente los manejadores que no usen caracteres estándares necesitan una interfase de bloque de entrada y salida en el kernel, necesario para examinar la "u-área". Las rutinas del kernel proveen esas interfases estándares también, y convierten los valores pasados a la "u_área", en valores que el driver espera. En el caso de que un bloque estándar de interfase de entrada salida, estos parámetros son puestos en el encabezado del buffer, el cual describe la transferencia de datos.

Los manejadores que utilizan la interfase de carácter estándar y la lista ligada de memoria y buffer, están basados en el esquema llamado **clist-buffering** y son las rutinas que manipulan el clist las que efectúan la transferencia de los datos.

III. METODOS DE EVALUACION DE RENDIMIENTO

III.1 Conceptos básicos

Debido a que un sistema operativo es en primer lugar un administrador de recursos, es importante para los diseñadores de sistemas operativos poder determinar con que efectividad administra sus recursos un sistema operativo.

Hace veinte años, gran parte de los costos de un ambiente computacional estaban relacionados con el hardware. Con los avances en la tecnología de hardware, estos costos han decrecido de forma drástica. Pero los costos de trabajo han ido aumentando sin cesar, haciendo que los costos de trabajo absorban gran parte del costo de la computación. Como resultado, debemos formular de nuevo el aspecto del rendimiento del hardware base y comenzar a medir el rendimiento de manera más adaptada a la productividad humana.

Con el advenimiento del microprocesador en la década de 1970, se ha hecho posible proporcionar ciclos de CPU a un costo nominal. Ahora, la evaluación del rendimiento debe centrarse en otras áreas, como la utilización de dispositivos de entrada/salida, donde los costos siguen siendo altos.

Los microprocesadores han hecho posible el desarrollo de computadoras personales en la década de 1970. Una computadora personal se utiliza sobre todo para la conveniencia de un solo individuo. Así es que la utilización de la CPU o de los

dispositivos de entrada/salida resulta menos importante que el hecho de que la computadora esté lista y disponible en el momento que el usuario lo necesita.

Está claro que los aspectos de la construcción de redes y de procesamiento distribuido influyen en nuestros puntos de vista sobre la evaluación del rendimiento. Ahora que las computadoras personales se encuentran en gran cantidad de hogares y las terminales individuales de las estaciones de trabajo son algo corriente en las organizaciones, los usuarios suelen conectarse con redes en lugar de computadoras específicas. Estas redes pueden contener cientos, o tal vez miles, de sistemas de computación, además de utilizar comunicaciones complejas controladas por computadora.

Para la evaluación del rendimiento se han propuesto tres objetivos comunes:

- Evaluación de selección. Aquí , el evaluador del rendimiento debe decidir si la adquisición de un sistema de computación de un proveedor en particular es apropiada.

- Proyección del rendimiento. La meta del evaluador, en este caso, es estimar el rendimiento de un sistema inexistente. Puede ser un nuevo sistema de computación, o un nuevo componente de hardware o software.

- Control del rendimiento. El evaluador acumula datos del rendimiento de un sistema o componente existente con el fin de

asegurar que el sistema cumple con sus metas del rendimiento, para ayudar a estimar el impacto de los cambios planeados y proporcionar a la administración los datos necesarios para tomar decisiones estratégicas, como la conveniencia o no de modificar un sistema existente de prioridades de trabajos.

La evaluación y predicción del rendimiento son necesarias desde los primeros momentos de la concepción de un nuevo sistema, en la operación diaria del sistema después de su instalación, y en las consideraciones de su modificación eventual reemplazo por un sistema mejor. En las primeras fases del desarrollo de un nuevo sistema, el proveedor intenta predecir:

- la naturaleza de las aplicaciones que van a correr en el sistema y
- las cargas de trabajo anticipadas que éstas aplicaciones deberán manejar.

Al comenzar el desarrollo e implementación de un nuevo sistema, la evaluación y predicción del rendimiento se usan para determinar:

- la mejor organización del hardware
- las estrategias de administración de recursos que deberán implementarse en el sistema operativo y
- si el sistema evolutivo cumple o no son sus objetivos de rendimiento.

Una vez que el producto se lanza al mercado, el proveedor debe estar preparado para responder a diversas preguntas de usuarios potenciales, acerca de si el sistema puede manejar ciertas aplicaciones con ciertos niveles de rendimiento. Los usuarios suelen estar interesados en la elección de una configuración apropiada de un sistema que pueda servir a sus necesidades. Esta adecuación se denomina a veces proceso de configuración.

III.1.1 Mediciones del rendimiento

Por rendimiento queremos expresar la manera o la eficiencia con que un sistema de computación cumple sus metas. Así pues, el rendimiento es una cantidad relativa, más que absoluta, aunque suele hablarse de medidas absolutas de rendimiento como el número de trabajos por hora que un sistema de computación dado puede servir. Pero siempre que se hace mención al rendimiento, en general se utiliza como base de comparación.

Algunas medidas de rendimiento como la facilidad de uso, son difíciles de cuantificar. Otras, como los accesos al disco por minuto son fáciles de enumerar. El evaluador del rendimiento debe tener cuidado al considerar ambos tipos de medidas, aún cuando sólo es posible presentar estadísticas claras y precisas para las últimas medidas.

Algunas de las medidas del rendimiento más comunes son:

- Tiempo de regreso. En un sistema de procesamiento por lotes, éste se define como el tiempo desde la entrega del trabajo hasta su regreso al usuario.

- Tiempo de respuesta. Es el tiempo de regreso de un sistema interactivo, y a menudo se define como el tiempo transcurrido desde que el usuario presiona la tecla ENTER hasta que el sistema comienza a imprimir una respuesta.

- Tiempo de reacción del sistema. En un sistema interactivo, éste suele definirse como el tiempo transcurrido desde que el usuario presiona la tecla ENTER hasta que se da la primera sección de tiempo de servicio a la petición del usuario.

Se trata de cantidades probabilísticas, y en los estudios de simulación y modelado de los sistemas se consideran como variables aleatorias. Cuando hablamos de valor esperado de una variable aleatoria, nos referimos a su valor promedio o medio.

Pero las mediciones pueden a veces ser decepcionantes, debido a que cierto valor medio puede ser producido al promediar una serie de valores idénticos o casi idénticos, o puede ser el producto de promediar una gran variedad de valores, unos mucho mayores y otros mucho menores que la media calculada por lo cual otra medida del rendimiento utilizada a menudo es:

- Varianza de los tiempos de respuesta (o de cualquiera de las otras variables aleatorias que hemos tratado). La varianza de los tiempos de respuesta es una medida de dispersión. Una varianza

pequeña indica que los diferentes tiempos de respuesta experimentados por los usuarios están relativamente próximos a la media. Una varianza grande indica que algunos usuarios pueden estar experimentando tiempos de respuesta que difieren mucho de la media. Mientras algunos pueden estar recibiendo un servicio relativamente rápido, otros pueden estar experimentando retrasos prolongados. Así pues, la varianza de los tiempos de respuesta es una medida de su predecibilidad y, desde el punto de vista del factor humano, ésta puede ser una medida del rendimiento muy importante en sistemas interactivos.

- Capacidad de ejecución. Es la medida de la ejecución del trabajo por unidad de tiempo.

- Carga de trabajo. Es la medida de la cantidad de trabajo que ha sido introducida en el sistema, y que el sistema debe procesar normalmente para funcionar de manera aceptable.

- Capacidad. Es la medida de la capacidad de rendimiento máxima que un sistema puede tener, siempre y cuando el sistema esté listo para aceptar más trabajos, y haya alguno inmediatamente disponible.

- Utilización. Es la fracción de tiempo que un recurso está en uso. Otro punto de vista sobre la utilización del CPU también da lugar a aspectos interesantes. Podemos ver al CPU en cualquier instante como:

a) disponible

b) en estado de programa

c) en estado de supervisión.

III.1.2 Técnicas de evaluación del rendimiento

Las técnicas más importantes de evaluación del rendimiento se resumen en la tabla III.1.1.

TECNICA DE EVALUACION	PROPOSITOS DE LA EVALUACION					
	EVALUACION DE SELECCION (EXISTE)		PROYECCION DEL RENDIMIENTO (NO EXISTE)		CONTROL DEL RENDIMIENTO (EN OPERACION)	
	HW NVO	SW NVO	DISEÑO HW	DISEÑO SW	CAM HW	CAM SW
TIEMPOS	1	0	1	0	0	0
MEZCLAS	1	0	1	0	0	0
NUCLEO	2	1	2	1	0	0
MODELOS	2	1	2	1	2	0
REFERENCIA	3	3	0	2	2	2
P. SINTETICOS	3	3	2	2	2	2
SIMUALCION	3	3	3	3	3	3
MONITOR (HW Y SW)	2	2	2	2	3	3

0 : Técnica no aplicable

1 : Ha sido usada pero es inadecuada

2 : Proporciona alguna ayuda, pero es insuficiente; debe ser usado en conjunción con otras técnicas

3 : Satisfactoria

Tabla III.1.1 Técnicas de evaluación de rendimiento

III.1.2.1 Tiempos

Los tiempos proporcionan los medios para realizar comparaciones rápidas del hardware de la computadora. Los primeros sistemas de computación solían evaluarse, por sus tiempos de suma, o por sus tiempos de ciclo de memoria.

Ultimamente, se ha hecho común categorizar a una computadora particular por el número de MIPS (millones de instrucciones por segundo) que realiza, dentro de unos años, algunas máquinas trabajarán en la gama de BIPS (miles de millones de instrucciones por segundo). Los tiempos se usan sobre todo en comparaciones rápidas. Para evaluar los más significativos y complejos sistemas actuales de hardware y software, deben utilizarse otras técnicas.

III.1.2.2 Mezclas de instrucciones

La técnica de mezclas de instrucciones usa un promedio ponderado de varios tiempos de las instrucciones más apropiadas para una aplicación determinada. El evaluador del rendimiento estudia la mezcla de trabajos de una instalación determinada e intenta formar un promedio ponderado de los tiempos de aquellas instrucciones usadas con más frecuencia en la instalación.

Las mezclas proporcionan poca o ninguna información útil para evaluar el software. La validez de las mezclas de instrucciones se

está haciendo cada vez más difícil de establecer con los complicados diseños actuales de hardware. Con el uso de las memorias cache y puesta en canalización, la misma instrucción puede ejecutarse en diferentes cantidades de tiempo, dependiendo de su contexto en diferentes ejecuciones.

III.1.2.3 Programas del núcleo

Un programa de núcleo es un programa típico que puede ser ejecutado en una instalación. Utilizando los tiempos estimados que suministra el fabricante, se cronometra el programa núcleo para una máquina dada. Entonces pueden hacerse las comparaciones entre máquinas basándose en las diferencias de los tiempos de ejecución del programa núcleo en diferentes máquinas. Los programas núcleo pueden ser útiles para la evaluación de ciertos componentes del software de un sistema.

III.1.2.4 Modelos analíticos

Los modelos analíticos son representaciones matemáticas de sistemas de computación o de componentes de sistemas de computación. Se usan muchos tipos de modelos, pero los de la teoría de colas y los procesos de Markov parecen ser los más manejables y útiles. El evaluador del rendimiento debe entender muchas técnicas diferentes y debe usarlas concordando unas con otras. Los resultados de una evaluación que utiliza una técnica determinada

pueden, a veces, ser invalidados por estudios que utilicen otras técnicas. Pero ocurre a menudo que varias evaluaciones diferentes tienden a reforzarse entre sí, y esto ayuda a demostrar la validez de las conclusiones del evaluador.

III.1.2.5 Puntos de referencia

Un punto de referencia (o programa de comparación del rendimiento) es un programa real que el evaluador ejecuta en una máquina que se está evaluando.

Por lo común, un punto de referencia es un programa de producción (es decir, uno que se ejecuta con regularidad en una instalación) típico de muchos trabajos de la instalación. El evaluador está muy familiarizado con el rendimiento del punto de referencia del equipo existente y cuando se ejecuta en un nuevo equipo, el evaluador puede llegar a conclusiones significativas.

En ambientes complejos con multiprogramación, tiempo compartido, multiprocesamiento, bases de datos, comunicaciones de datos y sistemas de tiempo real, los puntos de referencia pueden ser particularmente valiosos por ejecutarse en la máquina bajo circunstancias reales. Los efectos del software pueden experimentarse directamente, en vez de ser estimados. Los puntos de referencia son útiles tanto en la evaluación del hardware como del software, y aún en ambientes de operación compleja.

III.1.2.6 Programas sintéticos

Los programas sintéticos combinan las técnicas de los núcleos y los puntos de referencia. Son programas reales que han sido diseñados para ejercitar características específicas de una máquina. Una gran ventaja que tienen sobre los puntos de referencia es que un programa de referencia que pueda ser aplicable para probar una característica particular de una nueva máquina puede no existir.

El evaluador, desafortunadamente, no siempre dispone de tiempo suficiente para codificar y depurar programas sintéticos; por eso, a veces buscan programas de referencia existentes que se adapten lo mejor posible a las características deseadas de un programa sintético.

III.1.2.7 Simulación

La simulación es una técnica con la cual el evaluador desarrolla un modelo computarizado del sistema que se está evaluando.

El modelo se ejecuta entonces en un sistema de computación que pueda realizar millones de operaciones detalladas con rapidez y precisión, reflejando así el comportamiento del sistema que se está evaluando durante un periodo de tiempo simulado. Con la simulación

es posible preparar un modelo de un sistema que aún no exista, y ejecutarlo para ver cómo se comportaría en ciertas circunstancias.

Los simuladores computarizados se han hecho populares en las industrias espacial y de transportes, debido sobre todo a las tremendas consecuencias que podría tener construir un sistema que pudiera fallar.

III.1.2.8 Control de rendimiento

El control de rendimiento es la recolección y análisis de información, relativa al rendimiento de sistemas existentes. Es útil en la determinación del rendimiento de un sistema en relación con la capacidad de ejecución, tiempos de respuesta, predecibilidad, etc. El control del rendimiento puede hacerse por medio de técnicas de hardware o de software. Los monitores de software son de construcción o adquisición económica, pero pueden distorsionar las lecturas del rendimiento, debido a que consumen recursos del sistema. Los monitores de hardware suelen ser más costosos, pero tienen la ventaja de que su influencia sobre la operación del sistema es mínima. Cabe hacer notar que ésta es la técnica que se utilizará para la evaluación del rendimiento, ya que UNIX proporciona las herramientas apropiadas (como **sar**, **sa1**, etc.) además de que esta técnica resulta satisfactoria en el control de rendimiento del sistema en operación, que es precisamente el objetivo de este trabajo.

III.2 Tablas del Sistema Operativo UNIX Sistem V.

La mayoría de las estructuras de datos del Kernel son tablas cuya longitud puede asignarse dinámicamente, la ventaja de esta característica es que el código del Kernel puede tomar una nueva definición de las tablas, reconfigurando el sistema. El manejo de estas estructuras de datos, junto con la simplicidad del código del Kernel dan por resultado que si se pretende ejecutar un proceso cuyo volumen de información excede a los límites de las tablas se reportará un error al requerimiento del usuario y se podrá, si así se desea, modificar el tamaño de la tabla.

El modificar las tablas del sistema es una acción delicada que debe realizarse por personas conocedoras de las implicaciones que lleva consigo una reconfiguración.

Las tablas del Sistema son usadas para almacenar en disco y memoria información importante referente a usuarios, archivos y procesos. Una gran parte de las actividades en el Kernel requiere de la información almacenada en las tablas. Algunas veces es necesario modificar las tablas para satisfacer necesidades específicas, esto se logra modificando el valor de los parámetros configurables del Kernel. Lo cual significa que al modificar el valor de los parámetros, en realidad estamos incrementando o decrementando el tamaño de las estructuras de datos (tablas) del sistema.

III.2.1 Tablas Principales

III.2.1.1 Tabla de Acceso a páginas.

El manejo de memoria en la arquitectura de UNIX está basado en la división de toda la memoria en segmentos de igual tamaño llamados páginas. Generalmente el tamaño típico de estas páginas está en el rango de 512 bytes a 16 kbytes, dependiendo de la implementación. La tabla de acceso a páginas especifica el uso de las tablas de memoria. Una tabla de páginas posee una estructura bien definida en la cual se determina el tamaño de la página, así como también los modos de protección al núcleo del sistema (Kernel de lectura, Kernel de escritura).

El parámetro asociado a esta tabla es MAXPMEM.

III.2.1.2 Tabla de Archivos.

Una de las tablas más importantes, es la tabla de Archivos, la cual es una estructura global del Kernel. Cuando un Proceso abre o crea un archivo el Kernel asigna una entrada a la tabla de archivos, de tal forma que la tabla tendrá tantos registros como archivos abiertos existan. El parámetro NFILE determina el número de archivos abiertos permitidos para la tabla.

III.2.1.3 Tabla de inodos.

Esta tabla tiene una relación directa con la tabla de archivos, por cada archivo abierto que exista, se generará un registro en la tabla de inodos. El número de entradas en esta tabla depende del número de archivos abiertos.

El parámetro asociado a esta tabla es NINODE. Ver figura III.2.1.

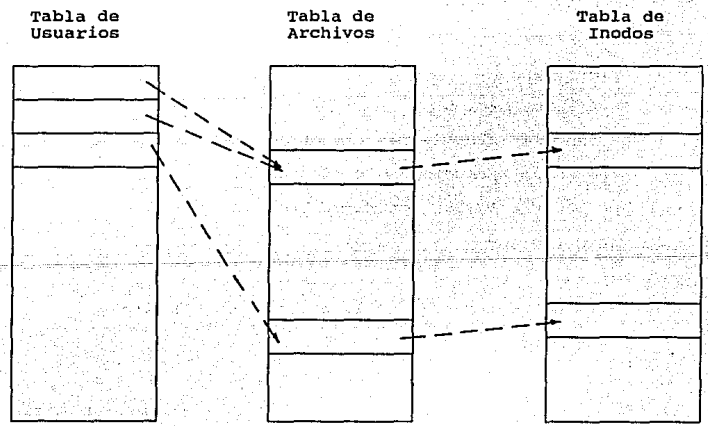


Fig. III.2.1 Relación entre tabla de usuarios, archivos y inodos

III.2.1.4 Tabla de procesos.

La tabla de procesos es también llamada estructura de procesos, está descrita en el archivo **PROC.h.** se tiene una estructura de procesos por cada proceso y es usada por el Kernel para determinar aspectos como la prioridad de actividades, estados de ejecución del proceso y los recursos requeridos para su ejecución.

Algunos de los componentes de esta tabla son los siguientes:

- Estado del Proceso
- Banderas del Proceso
- Prioridades
- Parámetros para determinar el tiempo que estará detenido en memoria antes de ser ejecutado
- Señales de Ejecuciones Pendientes
- Apuntador a la Estructura de Usuarios
- Parámetro que permite modificar la tabla: NPROC

III.2.1.5 Tabla de Región modo usuario y tabla de Región Modo Kernel

La memoria virtual usada en un proceso se divide en tres regiones: de texto, de datos y de pila. Estas tres regiones son mapeadas en la tabla de región modo usuario para los procesos de los usuarios y en la tabla de región del Kernel para el mismo.

El parámetro NREGION determina los límites de estas tablas. Ver figura III.2.2

III.2.1.6 Tabla de dispositivos.

La tabla de dispositivos es creada cuando el Kernel es compilado, esta tabla contiene un arreglo de estructuras que se encuentran en el archivo **/USR/INCLUDE/SYS/CONF.H.**

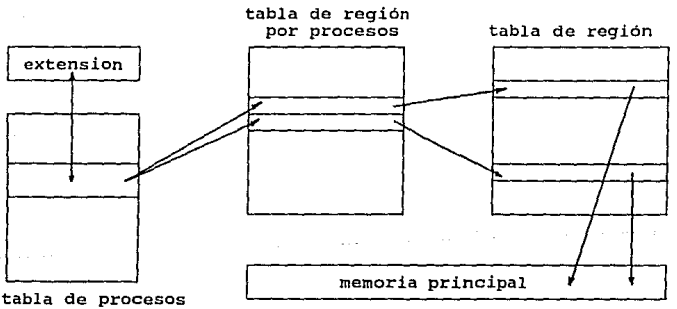


Fig. III.2.2 Relación entre tablas de Regiones y Procesos

La tabla está indexada por el número de dispositivos y almacena los nombres de las rutinas para apertura y cierre de dispositivos, así como un apuntador que permite la transferencia de datos, NCALL es el parámetro que permite modificar esta tabla.

III.2.2 Estructura de usuarios.

La estructura de usuarios es una base de datos usada por el Kernel para localizar cualquier información requerida por un proceso de un usuario mientras el proceso está siendo transferido a memoria. La estructura de usuarios está descrita en el archivo user.h veamos los componentes de esa estructura:

- Código de error de retorno en una llamada al sistema (ERRNO).
- Valores del retorno de una llamada al sistema
- Manejo de direcciones y bit de conteo para operaciones I/O.
- Descripciones del archivo de usuarios
- Apuntador a la estructura de procesos

III.2.3 Cola de mensajes

Los procesos pueden comunicarse usando la cola de mensajes, estos son dispuestos conforme a la secuencia en que son escritos y conforman el modo más importante de comunicación interprocesos. Los componentes de la cola de mensajes son:

- Permisos de operación
- Número total de mensajes
- Identificadores del último mensaje enviado y recibido
- Tiempo de operación en la cola del último mensaje enviado y recibido

III.2.4 Super-bloque

El super-bloque es siempre el bloque 1 del disco del sistema operativo UNIX. Contiene información que define el estado actual del sistema. Los principales componentes de esta información son:

- Tamaño en bloques de los inodos
- Número de bloques libres en el sistema
- Número total de inodos libres
- Bandera que indica si el super-bloque ha sido modificado
- Hora y fecha de la última modificación del superbloque.

III.3 Utilerías para obtención de estadísticas de operación

La administración eficiente del rendimiento de las computadoras es una de las responsabilidades principales del Administrador del Sistema. Las herramientas más importantes y más comúnmente utilizadas son la familia de comandos **sar** (system activity report, reporte de actividad del sistema). Estos reportes de actividad del sistema son obtenidos ejecutando comandos **sar** con las opciones adecuadas. De forma adicional se utilizan los comandos **ps** y **timex** para determinar mayor información sobre procesos.

Los comandos **sar** utilizan contadores del sistema operativo que son incrementados periódicamente por el sistema en ejecución. Estos contadores contienen información relacionada con la utilización de la Unidad Central de Proceso (CPU), la actividad del Sistema de

Entrada/Salida, la actividad de los dispositivos de terminales (tty), la actividad de los llamados al sistema, el acceso a archivos, la actividad en colas, la actividad de comunicación entre procesos, la actividad de pagineo, y la actividad de compartimiento remoto de archivos.

Monitorear el rendimiento del equipo no solo ayuda al Administrador a identificar causas de degradación del Sistema, posiblemente reportada por el usuario, sino que también puede ayudar a prevenir problemas severos en la operación del sistema.

III.3.1 Familia de Comandos sar

Los comandos sar recolectan información periódica del sistema operativo relacionados con la utilización de los recursos en el equipo. Un administrador puede ver esta información y determinar si es necesaria alguna acción para mejorar el rendimiento del sistema.

Por ejemplo, si el Administrador determina que un recurso en particular está exhausto, puede llegar a ajustar algún parámetro para corregir el problema. Si el acceso a disco no está repartido en los diferentes dispositivos, puede ser necesario cambiar el uso de los sistemas de archivos.

Cuando el sistema es cargado, los contadores que registran la información de la actividad del sistema son inicializados con valor

0. El archivo /var/spool/cron/crontabs/sys también contiene entradas por omisión que periódicamente recolectan información. El Administrador puede entonces usar el comando sar para verificar la información generada.

Comando	F u n c i ó n
sar	Reportar el uso de un recurso en el sistema.
sadc	Recolectar información del uso de recursos por un periodo de tiempo dado.
sa1	Crear archivos de datos conteniendo información de uso de recursos para ser consultados posteriormente por el comando sar.
sa2	Convierte información generada por sa1.

Fig. III.3.1 Comandos de la familia sar

III.3.1.1 El comando sar

Sintaxis:

sar [-ubdycwaqvmprDSAC] [-o file] t [n]

sar [-ubdycwaqvmprDSAC] [-s time] [-e time] [-i sec] [-f file]

El comando **sar**, en primera instancia, recolecta la información acumulada por los contadores del sistema operativo a "n" intervalos de "t" segundos, donde "t" debe ser mayor ó igual a 5. Si la opción -o es especificada, salva la recolección en el archivo "file" en un formato binario. El valor por omisión de "n" es 1. En segunda instancia, sin intervalo de tiempo especificado, **sar** extrae

información de un archivo **file** previamente generado, ya sea el especificado en la opción **-f** ó, por omisión, el archivo estándar diario de actividad del sistema **/usr/adm/sa/sadd** , para el día actual **dd**. Las fechas de inicio y término para el reporte pueden ser precisadas con los argumentos **-s** y **-e time** de la forma **hh[:mm[:ss]]**. La opción **-i** selecciona todos los registros en intervalos de tiempo **sec**. De otra manera, todos los intervalos encontrados en el archivo son reportados.

Para todos los casos, el subconjunto de datos a mostrar son seleccionados por las opciones de la figura III.3.2.

Opc	Información reportada
-u	**Utilización del CPU** (default). **%usr, %sys, %wio, %idle** - porción de tiempo corriendo en modo usuario, corriendo en modo sistema, ocioso con algunos procesos esperando E/S, y de otra manera ocioso. Cuando es usado con la opción **-D**, **%sys** es dividido en porciento de tiempo atendiendo requerimientos de equipos remotos (**%sys remote**) y porciento por todos los demás (**%sys local**).
-b	**Actividad de buffers. bread/s, bwrit/s** - transferencias por segundo de información entre buffers del sistema y disco u otros dispositivos de bloque; **lread/s, lwrit/s** - accesos a buffers del sistema; **%rcache, %wcache** - tasas de aciertos en memoria cache; **pread/s, pwrit/s** - transferencias via dispositivos físicos (raw). Usado con **-D**, buffers en memoria cache son reportados para recursos remotos localmente montados.
-d	**Actividad para cada dispositivo de bloque.** La información reportada es: **%busy, avqueue** - porción de tiempo que el dispositivo estuvo ocupado atendiendo un requerimiento de transferencia, número promedio de requerimientos presentes en el periodo; **r+w/s, blks/s** - número de transferencias desde/hacia un dispositivo, número de bytes transferidos en unidades de 512 bytes; **avwait, avserv** - tiempo promedio en ms. que un requerimiento de transferencia espera ocioso en cola, y tiempo promedio para ser atendido.
-y	**Actividad de dispositivos tty.** **rawch/s, canch/s, outch/s** - tasa de entrada de caracteres, tasa de entrada de caracteres procesados canónicamente, tasa de salida de caracteres; **rcvin/s, xmtin/s, mdmin/s** - tasas de recepción, transmisión, e interrupción de modems.
-c	**Llamados al sistema. scall/s** - llamados al sistema de todo tipo; **sread/s, swrit/s, fork/s, exec/s** - llamados al sistema específicos; **rchar/s, wchar/s** - caracteres transmitidos por llamadas al sistema de lecturas o escrituras. Cuando es usada con la opción **-D**, las llamadas al sistema son divididas en recibidas, emitidas, y estrictamente locales.
-w	**Swapeo en el sistema. swpin/s, swpot/s, bswin/s, bswot/s** - número de transferencias y número de unidades de 512 bytes transferidas para swapin y swapout (incluyendo carga inicial de algunos programas); **pswch/s** - conmutación de procesos.

Fig. III.3.2 Opciones del comando **sar**.

Opc	Información reportada
-a	**Uso de rutinas del sistema de acceso a archivos.** **iget/s, namei/s, dirblk/s.**
-q	**Tamaño promedio de cola ocupada y % de tiempo ocupada.** **runq-sz, %runocc** - cola (run) de procesos en memoria y ejecutables; **swpq-sz, %swpocc** - cola (swap) de procesos swapeados fuera de memoria pero listos para ejecución.
-v	**Estado de procesos, inodos, tablas de archivos.** **text-sz, proc-sz, inod-sz, file-sz, lock-sz** - entradas/tamaños para cada tabla evaluados una vez en puntos de muestra; **ov** - desbordamientos (overflows) que ocurren entre puntos muestra para cada tabla.
-m	**Actividad de mensajes y semáforos.** msg/s, sema/s - primitivas por segundo.
-p	**Actividad de pagineo.** **vflt/s** - page faults por traslación de dirección (página válida no en memoria); **pflt/s** - page faults de protección de errores (acceso ilegal a páginas); **pgfil/s** - vflt/s satisfechas del sistema de archivos; **rclm/s** - páginas válidas reclamadas para lista de páginas libres.
-r	**Páginas de memoria no usadas y bloques de disco.** **freemem** - páginas disponibles promedio para procesos de usuario; freeswap - bloques de disco disponibles para swapeo de procesos.
-D	**Actividad de Compartimiento de Archivos Remotos.** Cuando es usada en combinación con **-u, -b** ó **-c**, provoca que sar produzca la versión de compartimiento de archivos remotos del reporte correspondiente. **-Du** se asume cuando solo **-D** es especificada.
-S	**Estado de cola de requerimientos y de servidor.** **serv/lo-hi** - Número promedio de servidores de compartimiento de archivos remotos en el sistema; **request % busy** - % de tiempo que descriptores de recepción están en cola de requerimientos; **request avg lgth** - número promedio de descriptores de recepción esperando por servicio cuando la cola está ocupada; **server %avail** - % de tiempo que hay servidores ociosos; **server avg avail** - número promedio de servidores ociosos cuando los hay.
-A	**Toda la información.** Equivalente a **-udqbwcayvmprSDC.**

Fig. III.3.2 (continuación)

III.3.1.2 El comando sadc.

Sintaxis:

/usr/lib/sa/sadc [t n] [ofile]

Este comando, colector de información, resume información n veces, con un intervalo de t segundos entre muestras, y escribe en formato binario al archivo **ofile** ó a la salida estándar. El intervalo de muestra t debe ser mayor que 5 segundos; de otra manera la propia actividad de sadc puede afectar la muestra. Si t y n son omitidos, un registro especial es generado. Esta facilidad es utilizada al momento de iniciar el sistema, cuando se pasa a modo multiusuario, para marcar el momento en el cual lo contadores son inicializados a cero. Por ejemplo, el archivo **/etc/init.d/perf** escribe la marca de reinicio a la información diaria con el comando:

su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sadate +%d"

III.3.1.3 El comando sal.

Sintaxis:

/usr/lib/sa/sal [t n]

El comando sal, una variante de **sadc**, es usado para recolectar y guardar información en el archivo binario **/usr/adm/sa/sadd** , donde **dd** es el día actual. Los argumentos t y n provocan que los

registros sean escritos n veces a intervalos de t segundos, o una vez si son omitidos.

III.3.1.4 El comando sa2.

Sintaxis:

/usr/lib/sa/sa2 [-ubdycwqvmprDSAC] [-s time] [-e time] [-i sec]

El comando sa2, una variante del comando **sar**, escribe diariamente un reporte en el archivo **/usr/adm/sa/sardd** , donde dd es el dia del reporte. La estructura del archivo binario generado diariamente es la siguiente :

```
struct sa {
     struct sysinfo si;  /*see /usr/include/sys/sysinfo.h*/
     struct minfo mi;    /*defined in sys/sysinfo.h*/
     struct dinfo di;    /*RFS info  (sys/sysinfo.h)*/
     struct rcinfo rc;   /*Client cache info (sys/sysinfo.h)*/
     struct bpbinfo bi;  /*Co-processor info (sys/sysinfo.h)*/
     int bpb_utilize;    /*Co-processor utilize flag*/
     int minserve,maxserve;/*RFS server low/high water marks*/
     int szinode;    /*current size of inode table*/
     int szfile;     /*current size of file  table*/
     int szproc;     /*current size of proc  table*/
     int szlckf;     /*current size of file rec. header table*/
     int szlckr;     /*current size of file rec. lock   table*/
     int mszinode;  /*size of inode table*/
     int mszfile;    /*size of file  table*/
     int mszproc;    /*size of proc  table*/
     int mszlckf;    /*maximum size of file rec. header table*/
     int mszlckr;    /*maximum size of file rec. lock   table*/
     long inodeovf; /*cumulative overflows of inode    table*/
     long fileovf;  /*cumulative overflows of file     table*/
     long procovf;  /*cumulative overflows of proc     table*/
     time_t ts;      /*time stamp, seconds*/
     long devio[NDEVS][4];    /*device unit information*/
#define IO_OPS 0     /*cumulative I/O requests*/
#define IO_BCNT 1    /*cumulative blocks transferred*/
#define IO_ACT 2     /*cumulative drive busy time in ticks*/
#define IO_RESP 3    /*cumulative I/O resp time in ticks*/);
```

La información generada por **sar** puede, entonces, dividirse en cuatro grandes áreas (por opción):

A r e a s	o p c i o n e s
Actividad de CPU	-c, -q, -u
Utilización de Memoria	-p, -r, -v, -w
Sistema de E/S	-a, -b, -d, -w, -y
Compartimiento de Archivos Remotos (RFS)	-D, -S, -C

Actividad de CPU.

Cuando el sistema esta ocupado, el procesador puede estar ya sea en modo usuario o en modo sistema. Modo Usuario (**%usr**) es el porcentaje de tiempo de CPU gastado en ejecutar programas de usuario. Modo Sistema (**%sys**) es el porcentaje de tiempo utilizado en realizar llamadas del Sistema u otros procesos propios del Sistema Operativo. Cuando el Sistema no está ocupado, el procesador puede estar ya sea ocioso (**%idle**) o puede estar esperando por terminación de E/S (**%wio**). Si el campo **%wio** es alto, puede significar que el sistema está en un cuello de botella esperando por E/S. Si este es el caso, el Administrador puede considerar la opción de mover sistemas de archivos a diferentes discos buscando repartir la actividad de E/S en varios dispositivos de disco. Los porcentajes de tiempo de Sistema (**%sys**) y tiempo de Usuario (**%usr**) variarán de sistema a sistema dependiendo de las aplicaciones que se ejecutan en cada sistema. Si los porcentajes de Usuario o de Sistema cambian dramáticamente en un corto período de tiempo y no hay nuevas aplicaciones en ejecución, el Administrador puede buscar

la causa auxiliándose del comando **ps** (process status) o verificando alguno de los archivos de contabilidad del uso del sistema.

El sistema puede desempeñarse más eficientemente si se ejecutan procesos grandes no interactivos cuando el tiempo de ocio del sistema es alto.

Utilización de Memoria.

Si la cantidad de intercambio (**swpin, swpot**) se incrementa dramáticamente, puede ser señal de que el demonio de demanda de páginas (**pageout**) no está pudiendo satisfacer la demanda de memoria. Incrementar la cantidad de memoria en el sistema ayudaría a resolver esta situación. Si el número de conmutación de procesos por segundo (**pswch/s**) se incrementa drásticamente, los parámetros configurables del despachador de procesos (process scheduling) pueden ser verificados, o puede ejecutarse el comando **ps** para revisar las prioridades de los procesos. Si ocurre desbordamiento (**overflow**) al ubicar estructuras de datos del sistema, el Administrador del sistema podrá necesitar incrementar los parámetros configurables correspondientes. Los parámetros asociados con **inode** y **proc** son : **NPROC, NINODE, S5NINODE** y **UPSINODE.** Si una condición de desbordamiento ocurre por archivos, el sistema puede llegar a necesitar más memoria.

Si el número promedio de páginas para procesos de usuario (**freemem**) es consistentemente bajo, el sistema puede requerir más

memoria. Si el número de bloques disponibles para intercambio de procesos (**freeswp**) es bajo, nuevas o más grandes particiones de swap deberían ser agregadas al sistema.

Sistema de E/S.

Idealmente la actividad entre discos, como se determina por el porcentaje de tiempo que un dispositivo estuvo ocupado atendiendo requerimientos (**%busy**), debería estar tan balanceado como fuera posible.

El valor máximo de **%busy** debería ser siempre menor que 50 %. Valores mayores de 50 % indican un cuello de botella en disco. Los valores de **avwait** y **avserv** pueden decrementarse después de ejecutar el programa **dcopy** en sistemas de archivos del Sistema V. dcopy reubica inodos, directorios, y bloques de datos para que la información almacenada en el disco pueda ser accesada con mayor eficiencia. La razón de "cache hit" (**%rcache**) es el número de veces que el sistema encuentra información en los buffers en memoria cache. Si la información no está en los buffers, el sistema debe buscar en el disco antes de leer la información a memoria. Generalmente los campos reportados por **sar** en este aspecto que son más importantes son **%rcache**, el cual debería ser mayor que 90 % en sistemas ocupados, y **%wcache**, que debería estar arriba del 60 %. Incrementando el valor del parámetro configurable **BUFHWN** puede incrementar el rendimiento en el manejo de buffers de E/S.

III.3.2 El comando ps

El comando ps puede ser utilizado para monitorear los procesos que están actualmente corriendo. Si la información reportada por **sar** muestra algo no usual, el Administrador puede determinar si un proceso esta utilizando demasiados recursos. El comando ps cuenta con muchas opciones que permiten mostrar diferentes características de procesos en ejecución. Entre las más comunes se encuentran:

ps -f : Proporciona un listado completo de los procesos.

Su salida es parecida a lo que se muestra a continuación :

```
UID   PID   PPID  C    STIME      TTY   TIME COMMAND
abc   132   100   0    13:30:07   tty1  0:01 /usr/bin/vi qq
abc   133   100   0    13:31:02   tty1  0:02 /usr/bin/sh
abc   134   133   2    13:32:05   tty1  0:01 ls -l
```

Donde,

```
UID       = Usuario
PID       = Identificador del proceso
PPID      = Identificador del proceso padre
C         = Indice de utilización del procesador
STIME     = Tiempo de inicio del proceso
TTY       = Identificación de terminal
TIME      = Tiempo acumulado de CPU utilizado
COMMAND   = Comando en ejecución
```

ps -l : Proporciona una forma larga del comando ps. Una salida de este comando puede ser :

```
 F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY  TIME COMD
10 S abc 132  100 0  30 20 4036 57  10d9 tty1 0:01 /usr/bin/vi
90 O abc 133  100 0   3 20 1c40 32  2a34 tty1 0:02 /usr/bin/sh
```

Donde :

F	=	Características del proceso
S	=	Estado actual del proceso
PRI	=	Prioridad del proceso
NI	=	Valor "nice" del proceso
ADDR	=	Dirección inicial en memoria del proceso
SZ	=	Tamaño en páginas del proceso en memoria

III.3.3 El comando timex

El comando **timex** mide el tiempo de ejecución de un comando y reporta las actividades del sistema que ocurrieron durante el tiempo que el proceso estuvo en ejecución. El consumo del sistema puede ser recolectado para cada programa de aplicación y ser utilizado para afinación de los recursos mas pesadamente cargados. Por ejemplo,

timex -s nombre-de-programa

La opción **-s** reporta la actividad total del sistema para la duración de la ejecución del comando. Cuando la ejecución termina, la información de la actividad es impresa en la salida estándar.

III.4 Relación de parámetros del sistema operativo contra carga de trabajo.

El sistema operativo incluye una serie de herramientas para medir el desempeño del equipo. Estas herramientas pueden ser usadas para localizar el área que está provocando la degradación del sistema y son :

sar : (System Activity Reporter) muestra el estado del sistema y genera reportes con las diferentes actividades del sistema.

timex : Reporta la actividad por proceso del sistema durante la ejecución de un comando o programa.

La actividad es medida por contadores que se encuentran en el núcleo del sistema. Cada vez que una operación es ejecutada el contador asociado a ella es incrementado. Los reportes son generados en base al contenido de dichos contadores, y pueden ser usados para diagnosticar problemas en el sistema.

Las áreas más criticas para monitorear son el uso de la memoria y el comportamiento del procesador.

III.4.1.1 Actividad de buffers (sar -b)

La actividad de los buffers se reporta de la siguiente manera:

bread/s : Promedio del número de bloques físicos leídos del sistemas de buffers del disco duro por segundo.

lread/ : Promedio del número de bloques lógicos leídos del sistema de buffers por segundo.

%rcach : Porcentaje de lecturas lógicas encontradas en el buffer cache (100% - breads/lreads)

bwrit/ : Promedio del número de bloques físicos escritos del sistema de buffers al disco por segundo.

lwrit/s : Promedio del número de bloques lógicos escritos al sistema de buffers por segundo.

%wcache : Porcentaje de registros lógicos encontrados en el buffer cache (100% - bwrrit/writs).

pread/s : Porcentaje del número de lecturas físicas requeridas por segundo.

pwrit/s : Promedio del número de escrituras físicas requeridas por segundo.

De los parámetros mencionados los más importantes para el diagnóstico de la eficiencia del sistema son %rcahe y %wcache; los cuales miden la efectividad del sistema de buffers. Si %rcache cae abajo de 90, o %wcache cae abajo de 65, implica que el desempeño del sistema es pobre y solo será posible incrementarlo si se aumenta el número de buffers.

```
unix unix 3.2 2 386i     02/18/89

16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07       3      39      93       1      16      91       0       0
16:33:17       4      40      90       2      16      87       0       0
16:33:27       4      41      90       3       7      64       0       0
Average        4      40      91       2      13      84       0       0
```

Ejemplo tabla **sar -b** de salida

III.4.1.2 Actividad del Name Cache (sar -n)

Los nombres (namei) de los archivos usados mas recientemente son guardados en memoria cache con el fin de incrementar el desempeño de E/S en el disco duro. Esto puede reducir el tiempo que el sistema pierde en ligar el nombre de un archivo con su contenido, incrementando así la eficiencia del sistema si los archivos son accesados múltiples veces. Los archivos ejecutables o de datos son tratados de la misma manera.

Los parámetros reportados son:

c-hits : Número de aciertos

cmisses : Número de fallas

hit% : Porcentaje de la relación de aciertos con fallas.

El porcentaje normal de aciertos es del 95% (i.e. el 95% de los requerimientos para un archivo dado fué satisfecho buscándolo en el name cache en lugar de buscarlo en el disco en el buffer cache) si el porcentaje decrece a menos de 90% los parámetros s5cacheents, s5hashqs y s5ofbias deben ser incrementados.

III.4.1.3 Cola de procesos (sar -q).

Mediante los siguientes parámetros se reporta el promedio de la longitud de la cola de procesos y el porcentaje de ocupación de la misma:

runq-sz : Es la cola de ejecución de procesos en memoria; típicamente debe ser menor a 2. Valores altos significan que el CPU esta limitado (CPU-bound).

%runocq : Porcentaje de tiempo que la cola de ejecución esta ocupada; mientras sea mayor es mejor.

swpq-sz : Cola de swap de procesos que serán enviados a memoria; mientras menor sea el número es mejor.

%swpocc : Porcentaje de tiempo que la cola de swap esta ocupada; mientras más pequeño sea será mejor.

```
unix unix 3.2 2 386i     02/18/89

11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07     1.7     98     1.5      36
11:01:17     1.0     63     1.0      31
11:01:27     1.0     58     1.0      49
Average      1.3     74     1.2      39
```

Ejemplo tabla **sar -q**

En este ejemplo, la utilización del procesador (%runocc) varia entre 58% y 98%, a la vez que la fracción de tiempo que está vacía la cola de swap (%swpocc) es 31% y 49%.

Esto significa que la memoria no está causando un cuello de botella en el sistema pero más memoria ayudaría a reducir la actividad de swaping/paginación.

Si %runocc es mayor que 90 y runq-z es mayor que 2, el CPU esta sobrecargado y es el responsable de la degradación; en este caso se requiere adicionar capacidad al CPU para obtener una respuesta aceptable del sistema.

Si %swpocc es mayor que 20, será necesario incrementar memoria o aumentar el tamaño de algunos buffers para reducir la actividad de swaping/paginación.

III.4.1.4 Utilización del CPU (sar -u).

En cualquier sistema el CPU pude estar en alguno de los dos estados siguientes: esta ocupado o desocupado.

Cuando esta ocupado, el procesador esta realizando ejecuciones en modo usuario o en modo sistema. Cuando se encuentra desocupado, esta esperando por una terminación de entrada/salida o no tiene nada que hacer. Los parámetros reportados indican el porcentaje de tiempo que el procesador esta en modo sistema (%sys), en modo usuario (%user), esperando por entrada/salida (%wio), y desocupado (%idle).

En un sistema típico de tiempo compartido, %sys y %usr tienen aproximadamente el mismo valor. En aplicaciones especiales algunos de ellos puede ser mayor que el otro sin que esto quiera decir que algo esta mal. Un %wio alto significa generalmente un cuello de botella en disco.

Si %idle es alto con un tiempo de respuesta degradado, puede ser que la memoria esta limitada; el tiempo empleado por esperar memoria se atribuye a %idle.

El porcentaje de desocupación (%idle) puede proporcionar alguna señal sobre el desempeño del equipo. El valor varía normalmente del 40% al 100%, aun con un gran número de usuarios activos. Cuando este valor baja constantemente de 30%, la competencia principal de recursos no es del todo por memoria, sino que el CPU es el recurso crítico.

III.4.1.5 Tablas del sistema (sar -v).

Monitoreando las tablas del sistema podemos analizar el estado de los proceso, los inodos, los archivos, los registros de memoria compartida, y la tabla de archivos en memoria compartida (shared-memory) por medio de los siguientes parámetros:

proc-sz : Número de entradas registradas en la tabla de procesos que están siendo usadas/asignadas al núcleo (kernel).

inodo-sz : Número de entradas registradas en la tabla de inodos que están siendo usadas/asignados al kernel.

file-sz : Número de entradas registradas en la tabla de archivos que están siendo usados/asignados al kernel.

ov : Número de veces que ha ocurrido un overflow (una columna para cada uno de los parámetros anteriores).

lock-sz : Número de entradas registradas en la tabla de memoria compartida que están siendo usadas/asignados al kernel.

Los valores están dados como nivel/tamaño de tabla, lo que nos quiere decir el número de entradas usadas y cual es el tamaño asignado a cada tabla. El tamaño asignado a cada tabla puede ser reducido, para ahorrar memoria principal, al máximo valor registrado.

III.4.1.6 Actividad de swap (sar -w).

La actividad de swaping (intercambio) puede ser analizada por los siguientes parámetros:

swpin/s : Número de transferencia en memoria por segundo.

bswin/s : Número de bloques (512 bytes) transferidos

de memoria a disco, incluyendo la carga inicial de algunos programas por segundo.

swpot/s : Número de transferencias de memoria al área de swap por segundo. Si es mayor a 9 implica que se debe incrementar memoria o decrementar los buffers.

bswot/s : Número de bloques transferidos del área de swap del disco a memoria por segundo.

pswch/s : Intercambio de procesos por segundo. Este valor debe de ser de 30 a 50 en un sistema ocupado por 4 ó 6 usuarios.

III.4.2 Actividad por proceso

La actividad del Sistema puede monitorearse por proceso utilizando la utileria "timex". Si no hay otros programas ejecutándose timex nos da una buena idea de que recursos esta ocupando dicho programa. La utilización de los recursos del equipo de cómputo puede ser la unión de los recursos usados para cada programa y ésta unión puede ser usada para sintonizar el sistema.

IV. SINTONIZACION DEL SISTEMA OPERATIVO

IV.1 Parámetros Configurables

IV.1.1 ¿Que son los parámetros configurables?

Los **tunable parameters** (parámeters configurables) son valores que un administrador puede ajustar para que el sistema trabaje eficientemente y a toda su capacidad. Estos parámetros son usados para fijar el tamaño de varias tablas del sistema y las acotaciones para el trabajo de los usuarios, pudiendo dar mejor servicio cuando se presenta una carga fuerte de trabajo.

También son muy útiles cuando se presentan problemas en la operación del sistema, por ejemplo, el exceder las acotaciones de los usuarios o exceder los límites de memoria o procesamiento, los cuales pueden ser resueltos cambiando los valores que estos contienen.

Los parámetros configurables del sistema, están ubicados en archivos de configuración. A la acción de cambiar estos valores, reconfigurar el sistema, y reinicializarlo se le llama sintonizar ("**tuning**") el sistema.

IV.1.2 Tipos de Valores

Existen, básicamente, dos tipos de valores en los parámetros configurables:

- Valores por default. Inicialmente, las máquinas que tienen UNIX system V son entregadas con ciertos valores fijados previamente y grabados en los discos de instalación.

Estos valores son, generalmente apropiados para configuraciones con cargas pequeñas de trabajo de la computadora, y con procesos no muy grandes.

- Valores auto-configurables. Son valores que se calculan en la memoria de la máquina cuando el sistema esta siendo inicializado y pierden su valor al dar de baja el equipo. Estos valores están, dinámicamente cambiando, cada vez que el equipo en inicializado.

La mayoría de los valores tienen un valor máximo y un valor mínimo, como un rango posible de valores sugeridos. Los efectos de exceder estos valores, ya sea hacia arriba o hacia abajo, puede causar serios problemas con el sistema.

IV.1.3 Los archivos de configuración "mtune" y "stune"

El sistema UNIX, se caracteriza siempre por tener dos archivo donde son almacenados todos los valores de los parámetros que pueden ser configurados. Los nombres de los archivos, así como la ubicación del directorio donde estos son almacenado, varían de un sistema a otro, pero la mayoría coincide en llamarlos "mtune" y "stune".

- El archivo "mtune". El archivo contiene normalmente el valor mínimo que puede ser utilizado, el valor máximo y un valor por default que es el que se fija al instalar el sistema.

Es importante aclarar que este archivo normalmente NO es modificado por el administrador, ya que los valores que contiene son los que el fabricante garantiza darán un funcionamiento óptimo al equipo y evitaran problemas con el sistema operativo, y sólo puede ser modificado por indicación del fabricante.

En la tabla IV.1.1 se muestra el contenido típico de un archivo "mtune:

***	Default	Min	Max	***
***	General Kernel Parameters			***
NCALL	60	30	250	
NPROC	200	50	400	
MAXUP	30	15	60	
NHBUF	64	32	1024	
NPBUF	20	20	40	
NAUTOUP	60	0	120	
FDFLUSHHR	1	1	1	
NBUF	100	100	3000	
BUFHWM	200	5	400	

***	File System Parameters		***
NINODE	400	100	1300
UFSNINODE	30	100	1300
NDQUOT	200	100	400
***	Paging parameters		***
GPGSLO	25	0	25
***	Resource Limit Parameters		***
SCPULIM	0x7FFFFFFF	60	0X7FFFFFFF
HCPULIM	0X7FFFFFFF	60	0X7FFFFFFF
SFSZLIM	0X200000	0X100000	0X7FFFFFFF
HFSZLIM	0X200000	0X100000	0X7FFFFFFF
SFNOLIM	0X400	0X20	0X400
HFNOLIM	0X400	0X20	0X400

Tabla IV.1.1 Archivo "mtune" típico

- El archivo **"stune"**. El archivo almacena los parámetros configurables cuyos valores han sido cambiados de los valores que por default el fabricante propone al instalar el sistema.

Para decidir que parámetro será agregado en el archivo, se utiliza columna que tiene los valores por default en el archivo "mtune". Este archivo es totalmente modificable por el operador de hecho es el más utilizado en la sintonización del problema.

La Tabla IV.1.2 muestra el contenido típico de un archivo "stune":

```
*** ident "@(#) /etc/conf/cf.d/stune.s1 1.1 4.0   03/10/92  ***
MAXUP     40
NPROC     340
NINODE    600
NDQUOT    400
```

Tabla IV.1.2 Archivo "stune" típico

IV.1.4 Parámetros configurables mas comúnmente cambiados.

Existen dentro de los archivos mencionados anteriormente un número muy grande de parámetros para configurar, pero existen algunos de ellos, que por su importancia y el efecto que tienen en el funcionamiento del Sistema Operativo, son cambiados con una frecuencia muy alta. Estos valores cambiados frecuentemente, pueden ser clasificados en cinco categorías:

- Parámetros Generales del Kernel
- Cantidad de Memoria Asignada (Tablas y Buffers)
- Límites de los Usuarios
- Límites del Sistema
- Parámetros de Rendimiento

IV.1.5 Parámetros Generales del Kernel

Los siguientes parámetros configurables pueden ser cambiados para reflejar las características generales del sistema:

NCALL Especifica el número de posibles lugares en la tabla de llamadas de salida para dispositivos. La tabla de llamadas de salida es usada por los manejadores de los dispositivos periféricos para hacer un barrido de atención a los dispositivos, o para hacer una rutina de **"pooling"**. En el caso de que el número fijado en la tabla llegara a ser excedido, el sistema mandara una señal de **"PANIC"**, y el valor en la tabla tendrá que ser incrementado. El valor por default es de 60 y cada dato en la tabla ocupa 16 bytes.

NPROC Especifica el número máximo de procesos permitidos en el sistema. Para hacer un cálculo adecuado de este parámetro es necesario considerar cuantos usuarios utilizan el sistema, y cuantos procesos corre típicamente cada uno de ellos. El valor por default es de 200, y cada valor ocupa 212 bytes.

MAXUP Especifica el número máximo de procesos por usuario, tomando en cuenta el número de identificación del usuario para la definición. Es necesario hacer la aclaración de que el Super Usuario (root: uid 0), no esta sujeto a

ninguna restricción. El valor por default es de 30.

IV.1.6 Cantidad de Memoria Asignada

Los siguientes parámetros pueden ser modificados para afectar el manejo del encabezado(header) de buffer del sistema, así como para afectar el algoritmo de paginación por demanda:

NBUF Establece el número de "buffer headers" para almacenamiento, para ser asignados dinámicamente según sea requerido. Cada bloque de información de un requerimiento de Entrada-Salida utiliza un "buffer header" para su almacenamiento. Los "buffer headers" son asignados como van siendo demandados y son puestos en la memoria disponible del sistema. Si el valor de NBUF es grande, hace menor el número de localidades de memoria necesarias en el sistema. Cada conjunto de "buffer headers" puede no ser usado para otros propósitos, por lo que si es configurado un valor muy grande, se estará desperdiciando mucha de la memoria disponible. El valor de NBUF por default es de 100 y cada elemento requiere 88 bytes.

BUFHWM Indica el número de kilobytes de memoria que puede ser usado por los buffers de entrada y salida. Si la taza de datos en el buffer es baja, incrementando NBUF o BUFHWM

puede incrementarse esta taza. La frecuencia de datos en el buffer puede ser determinada usando el comando "sar". El valor por default es de 200.

FSFLUSHR Este valor determina el número de segundos en que el contenido de la memoria es grabado en disco (flush). Este proceso corre cada FSFLUSHR segundos para escribir los buffers de los archivos que han sido modificados, el superbloque del sistema de archivos, y los inodos. Incrementando este valor se decrementa la frecuencia de escritura al disco incrementando el rendimiento del sistema. Si el valor es decrementado aumenta la frecuencia de escrituras lo cual hace disminuir la probabilidad de una corrupción de datos a causa de un mal funcionamiento del sistema. El valor por default es de un segundo.

NAUTOUP Especifica el tiempo en segundos en que un buffer modificado puede mantener un dato antes de escribirlo al disco. Incrementando este valor, se incrementa también el rendimiento del sistema. Decrementandolo se incrementa la confiabilidad. El valor por default es de 60 segundos.

GPGSLO Especifica el número de páginas libres antes de invocar al proceso de "swap", el cual regresa páginas de procesos a la lista de páginas libres. El valor por default es de

25 y debe ser mayor o igual a cero. La mayoría de los parámetros de paginación por demanda es calculado automáticamente por el sistema y depende de la cantidad de memoria del mismo.

IV.1.7 Límites de los Usuarios

En versiones anteriores al R4, los límites en los tamaños de archivos y los archivos abiertos eran típicamente sintonizados por el administrador. Los procesos no tenían control sobre estos límites. Una vez que el límite era alcanzado el proceso fallaba, reportando un error, al tratar de asignar un recurso particular.

En el R4, los límites de los procesos son más flexibles. Un proceso tiene control sobre el tamaño actual del archivo y el límite de archivos abiertos. Los límites del usuario normalmente tienen un "soft limit" y un "hard limit" asociado con ellos. Un proceso de usuario falla si el "soft limit" es alcanzado, pero este límite puede ser incrementado por el mismo usuario, hasta alcanzar el "hard limit".

Los siguientes parámetros son utilizados para cambiar estos límites:

SFSZLIM Determina el valor de "soft limit" en el tamaño de archivo. Si el usuario excede este límite al escribir un archivo, una señal de error es regresada al proceso y la

escritura falla. El valor por default es de 2MB.

HFSZLIM Determina el valor de "hard limit" en el tamaño del archivo. Este es el valor máximo que el "soft limit" puede alcanzar. Su valor por default es de 2MB.

SFNOLIM Es el valor de "soft limit" en el número de archivos abiertos por proceso. Si un usuario excede este límite fallara la apertura del archivo. El default es 64.

HFNOLIM Es el valor de "hard limit" en el número de archivos abiertos por proceso. Es el valor máximo que el "soft limit" puede alcanzar para abrir archivos. El default es 1024.

IV.1.8 Parámetros configurables

Los siguientes parámetros están asociados con el sistema de archivos:

NINODE Este valor especifica cuantos elementos en la tabla de inodos puede ser asignado. Cada archivo que es abierto, tiene un inodo asociado con el que apunta a la localidad de disco donde esta almacenado e identifica sus atributos. Cuando un archivo o directorio es accesado, una copia de su inodo esta contenida en la tabla de

inodos del sistema. Cada elemento en esta tabla representa un inodo en la memoria principal asociado conun archivo activo. El número de elementos depende del numero de archivos activos. Si el valor es sobre pasado, un mensaje de error se reporta a la consola del sistema. El valor por default es de 400 y cada elemento utiliza 132 bytes.

La comunicación entre los procesos permite al administrador del sistema sincronizar e intercambiar información entre ellos. Esta comunicación consiste en semáforos, mensajes y memoria compartida.

SEMÁFOROS Permiten la sincronización de procesos haciendo un bloqueo de uno de ellos mientras que el proceso esta esperando por la disponibilidad de un recurso.

MENSAJES Permite el intercambio de información entre procesos.

MEMORIA COMPARTIDA

Permite que dos procesos accesen la misma área de memoria para poder utilizar los mismos datos.

Todos estos parámetros están localizados en el archivo **/etc/conf/cf.d/stune**, del que ya hemos hablado anteriormente.

IV.2 Interrelación entre parámetros.

El computador es capaz de soportar una gran variedad del uso de sus recurso, es decir, un sistema con varios usuarios activos que, están editando archivos pequeños consumen diferentes recursos de los que consumiría un sólo usuario explotando una gran base de datos. El ajuste de parámetros del kernel se realiza con el objeto de permitirle una operación más eficiente, obviamente esto implica el incremento en el tamaño de las estructuras datos utilizados por el sistema operativo, por lo tanto, el incremento de ciertos parámetros innecesarios provocará un desempeño deficiente del sistema. Por ejemplo, si se incrementa el parámetro NPROC, el sistema será capaz de mantener una lista más larga de procesos activos (PROCess table), esto podrá tener un efecto negativo en el planificador del núcleo ya que para decidir cual es el próximo proceso a ejecutar lo tendrá que seleccionar de una lista más grande; además el espacio requerido por el núcleo será mayor ya que el tamaño de la tabla fué incrementado; provocando que el espacio disponible de memoria se reduzca para los procesos de usuario generando una posible reducción en el rendimiento.

IV.2.1 Discos y buffers.

NBUF: Número de buffers del sistema generados en tiempo de arranque. Normalmente está entre 100 y 600, cada uno contiene 1076 bytes.

NHBUF: Especifica el total de colas a generar por cada buffer de 1k.El valor de NHBUF debe de seleccionarse de tal manera que NBUF dividido entre NHBUF sea aproximadamente 4.

IV.2.2. Sistema de archivos, nodos, archivos.

NINODE: Especifica el tamaño de tablas de inodos a generarse, el tamaño usado depende del número de archivos abiertos, normalmente es de 100 a 400, siendo el default de 300. El valor asignado atañe directamente el valor de NFILE ya que NINODE será mayor o igual que el valor de NFILE. NINODE debe de ser siempre menor o igual que NS5INODE, porque si es mayor el sistema degenerará a tal grado que no será posible utilizarse. Cuando la tabla de inodos se desborda, el sistema desplegará el siguiente mensaje:

WARNING: i-node table overflow.

NFILE: Especifica el número de registros que podrá tener la tabla de archivos abiertos, cada registro representará un archivo abierto; el valor por default es de 200, pero puede variar de 100 a 600 y cada registro ocupa 12 bytes. NFILE está relacionado con NINODE de tal manera que NFILE es menor o igual que NINODE. La estructura de control de NFILE opera de la misma manera que la estructura de NINODE. Cuando se desborda la tabla de archivos el sistema desplegará el siguiente mensaje:

NOTICE: file table overflow.

Como recordatorio se mencionará que este parámetro no afecta el número de archivos abiertos por proceso (NOFILES).

NOFILES: Determina el número máximo de archivos abiertos por proceso, el default es de 60 y a menos que una aplicación recomendará modificarlo no debe hacerse.

El shell utiliza tres registros en la tabla de archivos abiertos: entrada standard, salida standard y salida de error standard (0,1 y 2; que normalmente son reservados para stdin, stdout y stderr respectivamente). Esto deja al valor de NOFILE-3 como el número de archivos abiertos por proceso, en el caso de que un proceso requiera tres archivos abiertos más, deberán de cerrarse los archivos standard, lo cual no es recomendable y debe de hacerse con cautela. Si el valor de NOFILES es mayor que el máximo (100) o menor que el mínimo (60), el valor a configurarse será el de default (60) enviándose un mensaje a la consola.

S5CACHENTS: Número de nombres que utilizará el name cache. Debe de tener un valor entre 1 y 1024, siendo el default de 256. El valor recomendado para diversos grupos de trabajo es hacerlo grande, aproximadamente 3 veces el valor de NINODE.

S5OFBIAS: Determina la prioridad para mantener juntos los archivos en memoria cache, el valor de S5OFBIAS tiene un rango de 0 a 256, siendo el default de 8. Un valor de 0 significa que los

archivos abiertos no tienen prioridad especial; Si el valor es igual a S5CACHEENTS, los nombres de los archivos abiertos permanecerán siempre en memoria cache.

IV.2.3 Procesos, manejo de memoria e intercambio.

NPROC: Especifica el número de procesos activos que puede registrar la tabla de procesos; el swaper es siempre el primer proceso e init el segundo. El valor de NPROC depende del número de terminales disponibles y del número de intercambios de procesos por cada usuario. El promedio por usuario varia de 2 a 5 (véase MAXUP cuyo valor de default es el de 25). El valor de NPROC varia de 50 a 400, siendo el default de 100; Cuando la tabla de procesos se llene el sistema envía el error EAGAIN.

MAXUP: Determina el número de procesos concurrentes por usuario (no superusuario) permitidos; el valor varía normalmente de 15 a 60 sin exceder el valor de NPROC (NPROC es al menos 10% mayor que MAXUP). Este valor es por cada número de identificación de usuario, no por terminal.

VHNDFRAC: Determina el valor de la variable del sistema VHANDL, la cual contiene la memoria de usuario máxima disponible dividida entre VHNDFRAC o por el valor de GPGSHI. El valor de VHANDL determina cuando arranca el demonio de paginación vhand. La cantidad de memoria disponible libre es comparada con el valor de

VHANDL y si es menor el demonio vhand es despertado.

El valor de VHNDFRAC es 16, decrementarlo hace al demonio más activo, o si lo incrementamos será menos activo; el valor debe de estar entre 0 y 25 por ciento de la memoria disponible.

GPG5HI: Determina el punto más alto de memoria libre en páginas para vhand y detener el robo de páginas por proceso. El default es de 40, si es mayor el demonio es más activo, si es menor será menos activo. El valor es un entero mayor que 0 y que GPGSLO, y menor al 25% del número de páginas disponibles de memoria.

IV.3. Efecto de ajuste de parámetros

Para lograr la reconfiguración del núcleo se deben modificar los valores de los parámetros configurables, el alterar una tabla del sistema tiene casi siempre repercusiones sobre otras, debido a la interrelación entre parámetros.

Es por esto que al modificar el valor de algún parámetro, especialmente los del KERNEL, se deben de considerar algunas restricciones que son impuestas por el propio sistema. Se debe de ser muy cuidadoso al modificar un parámetro, ya que se pueden tener reacciones insospechadas por parte del sistema operativo. Para tener una visión más clara de los efectos de ajuste de parámetros, observemos lo siguiente:

PARAMETRO	DEFAULT	MINIMO	MAXIMO	RELACION CON	REGLA DE SINTONIZACION
NCALL	30	30	250	--	
NPROC	100	50	400	MAXUP	Debe ser >= MAXUP + MAXUP(0.10)
NBUF	100	100	600	NHBUF	Deber ser aproximadamente 4 veces NHBUF
MAUTOUP	60	0	120	--	
GPGSLO	25	0	25	GPGSHI	Debe ser >= 0 y < GPGSHI
NINODE	300	100	300	NFILE	Debe ser >= NFILE
NHBUF	64	32	1024	NBUF	NBUF/NHBUR Aproximadamente= 4
NFILE	200	100	600	NINODE	Deber ser <= NINODE
SSCACHENTS	256	1	1024	NINODE	Debe ser Aproximadamente 3 veces NINODE
MAXUP	--	15	60	NPROC	Debe ser < NPROC
VMMDFRAC	16	--	--	GPGSHI	Debe estar entre 0 y 25% del número de páginas de memoria
GPGSHI	40	--	--	GPGSLO	Debe ser > 0 y < número de páginas de memoria
MAXBUF	--	--	--	NBUF	Si NBUF = 0, el KERNEL lo configura automáticamente

Tabla IV.3.1 Relación de parámetros configurables

IV.3.1 Reglas de sintonización

En base a la información mostrada el en cuadro anterior, se pueden definir algunas de las más importantes **Reglas de sintonización.** Estas reglas establecen los lineamientos a seguir en cuanto a la modificación de los parámetros configurables, ayudando a que el cambio en el valor de algún parámetro no tenga impactos negativos en el rendimiento del sistema.

R E G L A S:

1. Al modificar el valor de NPROC, este deberá ser mayor o igual al valor de MAXUP más el 10% de MAXUP.
2. Al modificar NBUF se debe considerar la relación de que sea 4 veces más grande que el valor de NHBUF.
3. El valor de GPGSLO debe ser >=0 y < al valor de GPPGSHI.
4. NINODE debe ser siempre >= NFILE.
5. El valor de S5CACHENTS debe ser aproximadamente 3 veces mayor que el valor de NINODE.
6. El valor de MMAXUP debe ser siempre < que el valor de NPROC.
7. El valor de VHDFRAC debe estar entre el 0% y el 25% de la memoria disponible.
8. El valor de GPGSHI debe ser > 0 y que el valor de GPGSLO y < al 25% del número de páginas disponibles de memoria.

PARAMETRO	MENSAJE DE ERROR ENVIADO POR EL KERNEL
NINODE	Se excede la capacidad de la tabla de inodos
NCALL	Se excede la capacidad de la tabla de llamadas de salida para dispositivos
NFILE	Se excede la capacidad de la tabla de archivos
NOFILES	El valor configurado en NOFILES (*) es menor al mínimo permitido (o es más grande que el máximo permitido)

Tabla IV.3.2 Mensajes de error asociados a los parámetros configurables

(*) Representa el valor actual del parámetro y se despliega en pantalla.

MODIFICACION DESEADA	PARAMETROS A MODIFICAR
Mejoramiento del performance instalando memoria adicional	NBUF, NHBUF
Otros parámetros relacionados al mejoramiento del performance	NAUTOUP, MAXSLICE, BDFLUSHR
Incrementar los límites del sistema cuando se ha incrementado la memoria.	NCALL, NINODE, NFILE, NPROC, NREGIONS
Usuarios necesitan crear archivos más grandes.	ULIMIT
Cada usuario necesita abrir más archivos	NOFILES
Cada usuario necesita correr más procesos	MAXUP

Tabla IV.3.3 Casos especiales de sintonización

IV.4 Reconstrucción del núcleo.

El sistema UNIX se puede optimizar para utilizarlo, con una variedad se configuraciones de hardware y como una plataforma de muchas aplicaciones. El kernel (núcleo), el cual se encuentra en el corazón del sistema operativo, controla un número de recursos que constantemente se utilizan, se actualizan y se reciclan. Estos recursos incluyen:

- **buffers:** Son unidades de almacenamiento cache dentro de la memoria que contienen información usada recientemente (los buffers crecen o aumentan eficientemente al guardar ésta información, para tenerla a la mano y disminuyen al leer la información del disco).
- **table entries** (entradas de tabla): Son un espacio en las tablas del sistema que el kernel usa para mantener la pista de las tareas comunes, recursos y eventos.
- **otros recursos:** Estos son otros valores definibles o configurables que gobiernan los recursos especiales (tales como el número de multipantallas disponibles o la cantidad de semáforos).

El uso de éstos recursos está definido por ciertos límites conocidos como parámetros configurables del kernel. Estos límites se pueden disminuir o aumentar, algunas veces a expensas de otros recursos. Cada recurso o límite está representado por un parámetro separado, del kernel. El decidir cómo optimizar de la mejor manera el uso de éstos recursos, se conoce como **sintonización del desempeño** o bien, como **sintonización del kernel.**

IV.4.1 Parámetros del kernel

Los parámetros del kernel son valores contenidos en el mismo kernel del sistema UNIX, el cuál es el núcleo o corazón del sistema operativo. Cada vez que se cambian éstos parámetros ajustables, sus nuevos valores son relígados (o recompilados) dentro del kernel, para que los nuevos límites se pongan en vigor. La utilería configure(ADM) cambia el valor de los parámetros del kernel.

La sintonización del desempeño (o funcionamiento) es una actividad importante que debe revisarse cuando se instala por primera vez el sistema UNIX.

Cuando se sube el sistema por primera vez, el sistema automáticamente toma una configuración básica, que es satisfactoria para la mayoría de las situaciones. Esta configuración, sin embargo, no puede tomar en cuenta los patrones de uso y el comportamiento de las aplicaciones particulares.

Por ésta razón, la estructura del sistema permite reconfigurarlo para mejorar el desempeño (o funcionamiento) de las aplicaciones particulares de cada máquina además de la configuración estándar. (No es recomendable ajustar los parámetros del kernel si no hay necesidad aparente para hacerlo).

Existen varias razones para reconstruir los recursos del sistema, cuando:

- se instala hardware para tener memoria adicional, y por lo tanto para tener más recursos de memoria para almacenar.
- se presentan mensajes de error persistentes, indicando que ciertos recursos están agotados, tales como los inodos o las entradas de tabla.
- el tiempo de respuesta del sistema es constantemente lento, indicando que otros recursos están muy restringidos para la operación eficiente del sistema (como cuando se cuenta con muy poca memoria instalada)
- las necesidades de uso de los recursos que se deben ajustar para satisfacer las necesidades de una aplicación particular.

Además, es importante determinar cuáles son los recursos que se están desperdiciando o están ineficientemente distribuidos. Ciertos parámetros configurables son normalmente ajustados a valores mayores, cuando se instala memoria adicional para permitir que el sistema soporte más usuarios.

Sin embargo, para una computadora usada como una computadora personal de alta potencia o con procesador dedicado, no debe ser necesario aumentar los parámetros configurables del kernel, cuando se instala memoria adicional. De hecho, ajustar ciertos parámetros normalmente asociados con el aumento de memoria para soportar más usuarios, realmente puede disminuir el desepeño (o funcionamiento)

total del sistema. Esto es porque éstos parámetros aumentan los requerimientos de espacio para la información en el kernel, por lo tanto disminuye la nueva memoria disponible para los procesos de los usuarios.

El establecer simplemente, el uso deseado que se le dará a la computadora y las observaciones de qué tan bien esta funcionado, podrían ser una guía para determinar la necesidad de ajustar los parámetros configurables.

Las aplicaciones especializadas frecuentemente requieren que se relocalicen los **recursos llave** del sistema para un óptimo funcionamiento.

Por ejemplo, los usuarios con bases de datos grandes pueden necesitar, que se junten más archivos simultáneamente, de los que permiten las asignaciones comunes, y los usuarios quienes no necesitan las herramientas especializadas tales como el manejo de mensajes, pueden obtener un aumento en el desempeño, al desasignar estas herramientas.

IV.4.2 Reasignación de los recursos del Kernel con el comando configure (configurar)

La utilieria **configure** es un programa de direccionamiento que por medio de un menú, presenta cada recurso y las peticiones de

entrada para su modificación. Después de modificar los parámetros del kernel, se debe:

- **link** (ligar) el kernel con la función **link_unix(ADM)**,
- **copiar** el kernel al directorio raíz,
- **boot** (inicializar o apagar y encender) la máquina y
- **probar** el nuevo kernel.

Para cambiar cualquier parámetro del kernel, se deben seguir los siguientes pasos:

1. **Boot** (apagar y encender) la máquina y entrar en modo usuario único, (de mantenimiento)
2. Usar la función **custom(ADM)** para determinar si el paquete del conjunto de enlace (**LINK**) está instalado. Si no es así, usar el mismo comando **custom** para instalarlo.
3. Después de verificar que el conjunto de enlace (**link-kit**) está instalado, dar los siguientes comandos:

```
cd /etc/conf/cf.d
./configure
```

Ó **sysadmsh** users select: System|Configure|kernel|Parameters

4. El menú de configure se desplegará en seguida:

```
1. Disk and Buffers
2. Character Buffers
3. Files, Inodes and Filesystems
4. Process, Memory managment and
   Swapping
5. Clock
6. MultiScreens
7. Message Queues
8. Semaphores
9. Shared Data
10. System Name
11. Streams Data
12. Event Queus and Devices
13. Hardware Dependent Parameters
14. Remote file sharing Parameters

Seleccionar una categoría de parámetros para configurar, tecleando
un numero entre 1 y 14 o teclear "q" para salir:____
```

Los parámetros están agrupados por categorías, y para localizar un parámetro, se debe consultar la parte de Tunable System Parameters Descriptions (Descripción de los Parámetros Configurables del Sistema), al final de la System Administrator's Guide (Guía del Aministrador del Sistema). Elija una categoría, con el número que le corresponde en el menú. Los recursos de la categoría seleccionada, se desplegarán, uno por uno, cada uno con su valor actual. Escriba el nuevo valor para cada recurso o para mantener el mismo valor, simplemente oprima <Return>. Después de

que todos los recursos en la categoría se hayan desplegado, el comando **configure** regresa al menú inicial de categorías. Elija otra categoría para reconfigurar o bien, para salir de éste menú, oprima **q**.

5. Al terminar de cambiar los parámetros, se deben link (ligar) dentro del nuevo kernel, dando el siguiente comando:

   ./link_unix

   ⌂ **sysadmsh** users select: System|Configure|Kernel|Rebuild

Esto ensambla cada uno de los módulos del kernel dentro del nuevo kernel, que debe ser instalado a continuación.

6. **Boot** (inicializar) el nuevo kernel con el siguiente comando:

   /etc/shutdown

   ⌂ **sysadmsh** users select: System|Terminate

Un **prompt** (petición de entrada) del **boot** (inicialización) aparece. Cuando se presiona <Return> para reinicializar el sistema, el nuevo kernel es cargado y corrido.

Si existe algún problema con el nuevo kernel, se debe reinicializar el sistema con el kernel anterior, esto se hace dando: **unix.old** en el prompt del boot.

IV.4.2.1 Para usar el comando en línea configure

configure también tiene una interfase de comando en línea adecuado para los desarrolladores de aplicaciones. Por ejemplo, una persona que desea **lock** (cerrar) simultáneamente setenta archivos en lugar de cincuenta, puede construir un shell script para desarrollar la reconfiguración de los parámetros. Para consultar el valor actual de cualquier recurso configure, usando la interfase de comandos en línea, se debe dar:

```
./configure -y RECURSO
```

donde **RECURSO** es el nombre del parámetro **configure** (en mayúsculas). Para cambiar el valor de cualquier recurso con el comando en línea, dar:

```
./configure RECURSO=valor
```

Esta interfase es igual que la interactiva; por lo tanto, los mismos recursos se pueden configurar con ambas interfases.

IV.4.2.2 Exceder los límites en la configuración

La utilería **configure** hace incapié en ciertos límites sobre los valores de los parámetros del kernel. Esto es porque cuando ciertos valores se exceden, el funcionamiento del sistema puede sufrir dramáticamente, pues si se ha sobrepasado un valor de un parámetro se debe usar la opción "-o" (como se describe en la página referente a **configure(ADM)** del manual). La opción para

corregir el exceso, sólo funciona si se especifica un parámetro en la línea de comandos, porque no se puede usar con el menú.

IV.4.3 Reconfigurar debido a los mensajes de error persistentes

Hay casos en que el sistema operativo avisa que los límites se han excedido, éstos mensajes se despliegan en la consola. Algunos de éstos mensajes solo son avisos. Otros anteponen un **system panic** (pánico del sistema) en el que se presentan mensajes de diagnóstico adicionales, de acuerdo a cada caso en particular y el sistema se cae, requiriendo que se reinicialice.

El kernel no debe reconfigurarse debido a un solo mensaje de error del kernel que se recibe, aún un par de veces, sino cuando un solo mensaje persiste entre las sesiones del sistema.

Si se presentan cualesquiera de los mensajes listados en la tabla I.4.1, se deben revisar las descripciones para los parámetros apropiados en la sección Tunable System Parameters Descriptions (Descripción de los Parámetros Configurables del Sistema), al final de la System Administrator's Guide (Guía del Aministrador del Sistema).

Si es necesario ajustar un recurso, primero se ha de tratar de incrementar el valor con una cantidad pequeña, si el problema persiste, aumentarlo al 100% o más de su valor original. Si aún así

el problema no se soluciona es necesario investigar más detalladamente para localizar el programa y las secuencias exáctas que causan el error.

IV.4.4 Reconfigurar para un mejor desempeño

El sistema está configurado, de manera que las grandes cantidades de recursos del kernel, están asignadas a las tareas más comunes (tales como leer y escribir en el disco), sin ignorar las herramientas más especializadas (tales como la comunicación de interprocesos). Este balance puede ser cambiado para conformar los requerimientos individuales.

Mensajes de Error del kernel	Parámetro
iget inode table overflow	NINODE
Timeout table overflow	NFILE
mfree map overflow n(*)	SPTMAP
Region table overflow	NREGION
Configured value of NOFILES n(*) is less than minimum (grater than maximum).	NOFILES
stropen: out of streams	NSTREAM
swapdel - too few free pages	MINASHEM
stropen: out of stream	NSTREAM
stropen: out of queues	NQUEUE

(*) El valor n indica el valor actual desplegado en el mensaje de error

Tabla IV.4.1 Mensajes de error y Parámetros configurables asociados

IV.4.4.1 Precauciones en la sintonización del kernel

Los sistemas pueden soportar muchas aplicaciones diferentes de los recursos. Un sistema que soporta varios usuarios editando archivos pequeños, consume diferentes recursos a los de un sistema de un solo usuario que corre una base de datos grande. Los parámetros se ajustan para permitir que el kernel opere más eficientemente.

Esto frecuentemente aumenta el tamaño de las estructuras de datos del kernel. Además, ésto puede hacer más eficiente un aspecto de la operación del sistema, el kernel toma más tiempo para examinar grandes estructuras. Esto implica que el incremento inecesario de ciertos parámetros, puede realmente alentar el sistema.

Por ejemplo, aumentar el parámetro **NPROC** permite al sistema mantener una larga lista de procesos activos (**process table**). Esto puede tener un efecto contrario en el scheduler de kernel porque debe examinar esta larga tabla repetidamente, cada vez que necesite saber qué proceso es el siguiente que se ha de correr. Además, cuando se aumenta el tamaño de las tablas, los requerimientos de espacio de información del kernel aumenta y entonces hay menos espacio de memoria disponible para los procesos de los usuarios y también puede bajar sobre todo el desempeño (o funcionamiento) del sistema.

IV.4.4.2 Necesidades de recursos comunes

Frecuentemente el uso del sistema presenta la necesidad de sintonizar ciertos parámetros en circunstancias particulares, una necesidad común es la habilidad de crear archivos muy grandes.

Esto puede estar acompañado al ser super-usuario y modificar el **ulimit** para un proceso shell particular que se está corriendo en modo super-usuario. Una solución alternativa es modificar el **ULIMIT** del sistema para todos los usuarios.

Este parámetro y otros límites encontrados se resumen en la tabla IV.4.2.

(Referirse a la sección Tunable System Parameters Descriptions (Descripción de los Parámetros Configurables del Sistema), al final de la System Administrator's Guide (Guía del Aministrador del Sistema), para conocer los detalles de cada parámetro antes de hacer cualquier cambio).

Mejoramiento deseado	Parámetro
Mejorar el funcionamiento del sistema cuando se instala memoria adicional	NBUF, NHBUF (ver 'Medir el Buffer cache)
Otros parámetros del sistema relacionados con el desempeño (o funcionamiento)	NAUTOUP,MAXSLICE BDFLUSHR, AGEINTERVAL
Aumentar los límites del sistema cuando se instala memoria adicional (para soportar más usuarios y reducir las oportunidades de problemas del sistema en ocaciones de carga pesada, est.)	NCALL,NINODE, NSINODE,NFILE, NPROC,NREGIONS, NCLIST (ver también mensages, semáforos, y parámetros de memoria compartida
Cuando los usuarios necesitan crear archivos más grandes.	ULIMIT
Cada necesidad del usuario para abrir más archivos.	NOFILES
Cada necesidad del usuario para correr más procesos.	MAXUP
Otros límites del sistema que pueden se encontrados	SHLBMAX,FLCKREC, SPTMAP,NUMXT, NUMSXT,PRFMAX (Ver también parámetros STREAMS)

Tabla IV.4.2 Necesidades de casos especiales configurables

V. IMPLEMENTACION DEL SISTEMA

V.1 Selección del software.

Antes de poder elegir la herramienta en que vamos a trabajar, es necesario definir las diferentes posibilidades que contemplamos, así como sus ventajas y desventajas.

Para elegir las herramientas, de acuerdo a los objetivos generales del sistema (Monitoreo y Sintonización del sistema operativo), y la forma en que deseamos que interactue con el usuario, buscamos que dichas herramientas tuvieran características tales como facilidad en el manejo de las tablas del sistema operativo, graficación de los datos, para una mejor visualización del comportamiento del sistema, y una presentación amigable para el usuario. Por otra parte, debemos aprovechar las herramientas con que contamos, sin tener que recurrir a la adquisición de alguna de éstas. Dentro de las herramientas posibles que consideramos, están las siguientes:

- **FOX BASE**
- **INFORMIX/SQL**
- **INFORMIX/4GL**
- **SHELL**
- **LENGUAJE "C"**

A continuación describimos las ventajas y desventajas de cada una de las posibles opciones.

V.1.1 FOX BASE

Aunque "Sistema de Bases de Datos" es un término informativo, también se puede aplicar a las distintas formas en que se clasifica, almacena y utiliza la información. En el corazón de todo sistema de información existe una base de datos.

El manejador de bases de datos FOX BASE es un conjunto de ordenes que permiten accesar la información contenida en la base de datos. Las ordenes del lenguaje de programación ofrecen multitud de posibilidades de manipulación de la información.

El FOX BASE puede manejar 1,000 millones de registros y hasta 128 campos de registro. Cada registro puede contener hasta 4,000 caracteres. Puede ordenar varios campos al mismo tiempo y trabajar con 10 archivos. Estas características tan poderosas son más impresionantes que reales.

Unos simples cálculos indican que si una base de datos contiene 1,000 millones de registros de 128 campos cada uno, a un sistema le llevaría más de un mes en leer una sola vez la base de datos y dicha base ocuparía 30 millones de megabytes (Mb). Pero por otro lado, esta herramienta ordena los datos muy rápido.

El FOX BASE permite el acceso de múltiples usuarios al programa cuando se trabaja en modo multiusuario. Las limitaciones

de FOX BASE son pocas pero existen y una de ellas es que debe leer la información del disco, lo que hace disminuir la velocidad de sus programas.

V.1.2 INFORMIX/SQL

SQL (Structured Query Language) es un lenguaje de consulta estructurado para bases de datos relacionales que maneja información como conjuntos lógicos llamados "relaciones" (tablas). Es un lenguaje que puede ser utilizado interactivamente o a través de una aplicación. SQL provee un pequeño y consistente conjunto de comandos que permiten definir, desplegar y actualizar información en tablas. La importancia real del SQL consiste en el hecho de que se ha convertido en el estándar de lenguajes para el manejo de bases de datos y no en la habilidad o capacidad de servir como un lenguaje de consulta para el usuario final. En este sentido existen algunas utilerías o programas de servicio que permiten seleccionar información y condiciones por medio de un menú.

SQL fundamenta su operación en la teoría de conjuntos, lo que lo convierte en un lenguaje muy interesante y útil. SQL está constituído de tres partes fundamentales:

1) Lenguaje de manejo de datos (DML)
2) Lenguaje de definición de datos (DDL)
3) Lenguaje de control de datos (DCL)

La sección de DML es utilizada para obtener, actualizar, insertar y borrar renglones en una tabla. La sección de DDL es usada para crear objetos de bases de datos, como índices. La sección DCL proporciona la seguridad de las bases de datos.

Dentro de los componentes del SQL existe otra sección (secundaria) llamada "Sistema de Catálogo", que almacena toda la información sobre los objetos de la base de datos creados, usando el DDL. Las tablas de catálogo pueden inclusive, ser modificadas utilizando el DML.

SQL no es un lenguaje de procedimientos con las características de interfase, sino que provee únicamente primitivas para la estandarización de la salida de información, basándose en otros productos para tales propósitos.

V.1.3 INFORMIX/4GL

Es el lenguaje de cuarta generación más popular en el ambiente UNIX. Comprende las herramientas para el manejo de pantallas, elaboración de reportes y los poderosos comandos de SQL, todo ello dentro de un fácil lenguaje de programación, que combina postulados procedurales y no procedurales, proporcionando la facilidad de poder de programación, sin necesidad de tener que recurrir a lenguajes de tercera generación.

El software de INFORMIX para el manejo de bases de datos relacionales se encuentra disponible para más de 450 distintos equipos, provenientes de más de 85 fabricantes y soportan prácticamente todas las normas de comunicación para redes.

La arquitectura de INFORMIX combina una estructura eficiente, para la transferencia de datos, el poder de SQL (para la definición, el manejo y el control de datos), y la velocidad de "C-ISAM" para el acceso de datos. INFORMIX/4GL es un lenguaje de cuarta generación no procedural que permite describir qué es lo que se quiere sin tener que escribir una larga lista de pasos para obtenerlo. En el diseño de INFORMIX/4GL se introdujeron facilidades que lo hacen simple en la creación de poderosas aplicaciones con un pequeño conjunto de instrucciones amigables, con los cuales se pueden ejecutar las siguientes funciones:

- Uso de ventanas
- Creación de menúes
- Utilizar formas de pantalla para el acceso de datos
- Uso de SQL para la manipulación de la base de datos
- Llamada a pantallas de ayuda
- Creación de reportes
- Acceso de varios renglones de datos de una simple forma, con scroll
- Proporciona formas para consulta por ejemplos
- Control de teclas
- Dar de alta atributos de pantalla condicionales

- Acceso a herramientas de depuración
- Llamadas a funciones de librerías de "C" o "4GL".

INFORMIX/4GL preprocesa el código fuente 4GL mediante tres fases para llegar a obtener el código ejecutable, utilizando el compilador de "C" para INFORMIX/4GL. La capacidad y operación de esta herramienta está limitada por las propias características del Hardware en que se instala.

V.1.4 SHELL

El sistema UNIX está dividido lógicamente en dos partes: el kernel y las utilerías.

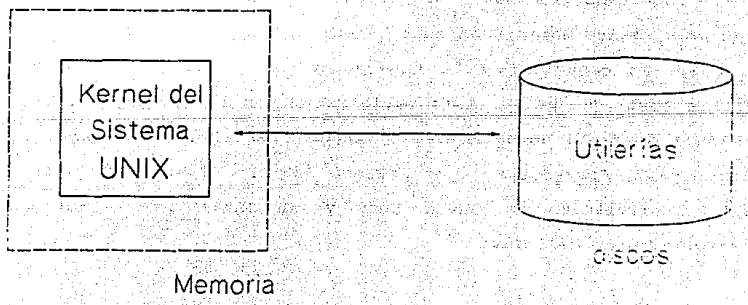


Fig. V.1.1 El sistema UNIX

El kernel es el corazón del sistema UNIX y reside en la memoria de la computadora desde el momento en que la computadora es encendida e booted (inicializada) hasta que es apagada.

Las utilerías, por otra parte, residen en el disco de la computadora y sólo son traídas a la memoria cuando es necesario. Virtualmente todos los comandos del sistema UNIX son clasificados como una utility (utilería); por lo tanto, los programas permanecen en el disco y son llevados a la memoria sólo cuando el comando es ejecutado.

El shell, también es un programa de utilería y es cargado en la memoria para ejecutarse siempre que uno entre al sistema.

Es importante reconocer que el shell es sólo un programa. No tiene privilegios especiales en el sistema, por lo que cualquiera que tenga la capacidad y la dedicación puede crear su propio programa shell. De hecho, ésta es la razón por la que hay varias versiones de shell actualmente, incluyendo el shell estándar de Bourne **"sh"**, desarrollado por Stephen Bourne; el shell de Korn **"ksh"** desarrollado por David Korn y el shell de C **"csh"**, desarrollado por Bill Joy.

El shell es el que analiza cada línea tecleada e inicializa la ejecución del comando seleccionado, además tiene otras responsabilidades, que se muestran en la figura V.1.2.

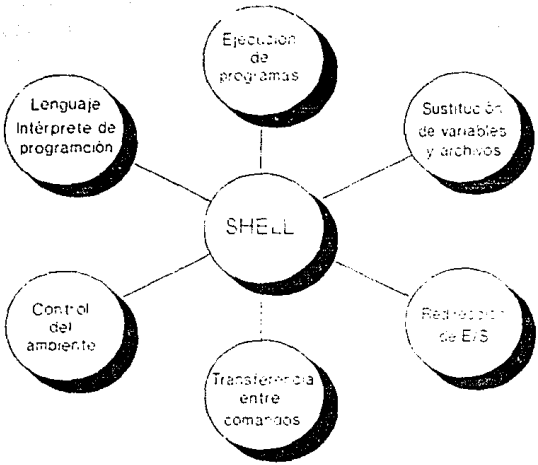


Fig. V.1.2 Las responsabilidades de shell

Ejecución de Programas

El shell es responsable de la ejecución de todos los programas que se llaman desde la terminal. Cada vez que se teclea una línea para el shell, éste analiza la línea y determina qué hacer.

Substitución de nombres de variables y de archivos

Como cualquier otro lenguaje de programación, el shell permite que se asignen valores a las variables. Siempre que se especifiquen

una de ésas variables en la línea de comandos, precedida del signo "$", el shell la substituye por el valor que le fué asignado anteriormente. El shell también permite sustituir el nombre del archivo en la línea de comandos, esto es, se puede tomar todo un conjunto de nombres de archivos como argumento de un programa especificando el patrón de los nombres; el shell encontrará los nombres de los archivos que igualen dicho patrón.

Redireccionamiento de E/S

Es responsabilidad del shell cuidar el redireccionamiento de entrada y salida en la línea de comandos. El shell examina la línea de comandos para detectar la existencia de caracteres especiales de redireccionamiento: "<", ">", o ">>", "<<". Esto permite que la salida de cualquier programa se dirija hacia un archivo determinado, en lugar de dirigirla hacia una terminal; se puede hacer que la entrada provenga también de un archivo y no de la terminal. La entrada y la salida incluso pueden conectáse a otros programas.

Pipeline hookup (Transferencia entre comandos)

Así como el shell revisa la línea de comandos para buscar caracteres de redireccionamiento, también busca el caracter de pipe (tubo) "¦" (o bien "^"). Para cada caracter pipe que encuentra, conecta la salida estándar del comando que precede el"¦", a la entrada estándar del comando que está después del "¦". Entonces inicializa la ejecución de ambos programas.

Control del ambiente

El shell provee ciertos comandos que permiten personalizar el ambiente. El ambiente incluye el **home directory** (directorio principal), los caracteres que el shell despliega, y una lista de los directorios en los que busca siempre que se requiera ejecutar un programa. Con todo esto el usuario puede definir sus propios comandos y abreviaturas.

Lenguaje intérprete de programación

El shell tiene su propio lenguaje de programación **built-in** (de construcción). Este es un lenguaje intérprete, esto quiere decir que el shell analiza las sentencias de lenguaje en cada línea, y las va ejecutando al mismo tiempo . Esto lo hace diferente de lenguajes de programación como C, y Fortran, en los cuales las sentencias de programación son compiladas en un ejecutable, antes de ser ejecutadas. Los programas desarrollados en lenguajes intérpretes de programación son más fáciles de analizar y modificar que los compilados. Sin embargo, generalmente tardan más en ejecutarse que sus equivalentes compilados.

El lenguaje de programación shell, provee características que se encuentran en la mayoría de los otros lenguajes de programación; cuenta con estructuras de ciclos, proposiciones de decisión, y es procedure-oriented (orientado a procesos). El uso del shell permite hacer muchas cosas con poco trabajo, sin tener que programar en un lenguaje ordinario como C. Sin embargo el shell estándar le faltan

varias caracteristicas (estructuras) de programación, tales como arreglos, tipos de datos, y operaciones aritméticas especiales.

V.1.5 LENGUAJE "C"

C es un lenguaje de programación de empleo general, caracterizado por su concisión y por poseer un moderno flujo de control y estructuras de datos, así como un rico conjunto de operadores. No es un lenguaje de "muy alto nivel" ni "grande", no estando especializado en ningún área particular de aplicación. En cambio, su carencia de restricciones y su generalidad lo hacen más eficaz y conveniente para muchas tareas que otros lenguajes, supuestamente más potentes.

C fué diseñado originalmente para el sistema operativo UNIX en la PDP-11 de la DEC (Digital Electronics Co.) e implantado en ella por Dennis Ritchie. Por ésta razón ha sido estrechamente asociado con el sistema operativo UNIX, pues su desarrollo se realizó en este sistema y debido a que tanto UNIX como sus software fueron escritos en C.

Sin embargo, no está ligado a ningún hardware o sistema operativo en concreto, siendo fácil de escribir programas que ejecuten tareas sin ningún cambio en cualquier máquina que soporte a C. Aunque se le ha llamado "lenguaje de programación de sistemas" a causa de su utilidad en la escritura de sistemas operativos, ha

sido utilizado con el mismo éxito para escribir programas numéricos, programas de procesamiento de textos y bases de datos.

C es un lenguaje de relativo "bajo nivel". Esta tipificación no es peyorativa; simplemente significa que C trabaja con la misma clase de objetos que la mayoría de las computadoras: caracteres, números y direcciones, que pueden ser combinados con los operadores aritméticos y lógicos, utilizados normalmente en las máquinas. C no cuenta con operaciones de entrada/salida: no existen proposiciones READ o WRITE, ni métodos propios para el acceso a archivos. Todos éstos mecanismos de alto nivel deben ser aportados por funciones llamadas explícitamente. De la misma manera, C sólo ofrece proposiciones (sentencias) de control de flujo sencillas, secuenciales, de selección, de iteración, bloques y subprogramas, pero no multiprogramación, paralelismo, sincronización o corrutinas.

Los programas en C tienden a ser lo suficientemente eficientes como para que no haya necesidad de escribirlos en lenguaje ensamblador. El ejemplo más claro es el sistema operativo UNIX, escrito en su mayor parte en lenguaje C.

Finalmente C, igual que cualquier otro lenguaje, tiene sus inconvenientes. Algunos de los operadores tienen precedencia equivocada, y algunas partes de la sintaxis podrían ser mejores. Existen varias versiones del lenguaje, aunque difieren en poco.

A pesar de todo, C ha demostrado ser un lenguaje extremadamente eficaz y expresivo para una extensa gama de aplicaciones de programación.

V.I.6 Alternativa seleccionada

De los paquetes disponibles analizados, tanto para el manejo de base de datos, así como los lenguaje de tercera y cuarta generación, creemos que utilizar **INFORMIX/4GL** y algunas rutinas escritas en **SHELL**, por las características mencionadas anteriormente, cumplen con los requerimientos necesario para el desarrollo de nuestro sistema, por lo que las rutinas para obtener información de las tablas del sistemas, así como las que reconfiguran los parámetros del kernel y el kernel en sí, serán hechas en **SHELL**, y la interfase visual, el sistema de graficación y el manejo de la información será desarrollado utilizando **INFORMIX/4GL**.

V.2 ANALISIS FUNCIONAL Y DE INFORMACION

V.2.1 Objetivo del Sistema

El objetivo del Sistema de Evaluación de Rendimiento y Sintonización para el Sistema Operativo UNIX SV es proveer al Administrador del equipo con una herramienta fácil de usar que permita :

- Consultar la configuración física del equipo.

- Monitorear el comportamiento del Sistema Operativo.
- Detectar posibles causas de baja de rendimiento del Sistema Operativo.
- Sugerir posibles correcciones de configuración del Sistema Operativo.
- Para el caso de correcciones de parámetros de configuración, efectuarlas bajo consentimiento del Administrador del Sistema (usuario).

V.2.2 Alcances del Sistema

Debido a la complejidad en la sintonización del Sistema Operativo, la realización de esta tarea requiere de la participación del Administrador en cuanto al conocimiento del Sistema Operativo para entender la posible repercusión en la sintonización de parámetros. Cuando sea detectado un problema por carencia de recursos del sistema, tales como cantidad de memoria, velocidad del CPU, tiempos de acceso a discos, etc. , el sistema podrá, en su caso, diagnosticarlo y sugerir posibles incrementos de dichos recursos. Para la implementación del Sistema SERSUNIX se ha seleccionado el sistema SCO UNIX SV (Santa Cruz Operation), en el entendido de que podrán surgir incompatibilidades para otras implementaciones de UNIX SV. El diagnóstico de la configuración física se limitará a aquellos elementos que influyan de manera directa en la sintonización del Sistema Operativo.

El monitoreo del comportamiento del Sistema Operativo estará enfocado a intervalos de tiempo seleccionados para una carga de trabajo independiente del Sistema que será establecida por el Administrador del Sistema. Cabe hacer notar que este no es un Sistema que proporcione necesariamente medidas de comparación entre diferentes instalaciones (benchmark). El monitoreo deberá poderse realizar en forma continua por requerimiento del Administrador generando, en su caso, gráficos o reportes estadísticos que muestren el comportamiento de el o los recursos de interés.

En cuanto a la consulta de la configuración lógica, se centrará el interés en los elementos lógicos críticos, así como en parámetros del kernel susceptibles de modificación. Se presupone un equipo multiusuario que no esté conectado en ningún tipo de red, considerándose únicamente como un servidor local.

La detección de posibles causas de baja de rendimiento será únicamente bajo cargas de trabajo reales, que deberán ser manejadas por el Administrador del sistema acordes a los requerimientos de la instalación. En la medida de lo posible se detectarán estas situaciones, obviando en su caso procesos que afecten de manera considerable el rendimiento del sistema.

En cuanto a correcciones de parámetros configurables, se sugerirán cuales cambios podrían realizarse y la razón para hacerlo, quedando al Administrador del Sistema cualquier ajuste

adicional de los mismos y la decisión de que el Sistema SERSUNIX lleve a cabo dichos cambios.

V.2.3 Documentación generada

A continuación se presenta la documentación generada del análisis estructurado de SERSUNIX, que contiene los siguientes puntos:

V.2.3.1. Modelo Lógico

Diagrama de Contexto

Diagramas de Flujos de Datos

- DFD 0 SERSUNIX
- DFD 1 MONITOREA Y EVALUA S.O.
- DFD 1.1 MONITOREA S.O.
- DFD 2 SINTONIZA S.O.
- DFD 2.1 DIGANOSTICA KERNEL

Diagrama de descomposición

V.2.3.2. Modelo de Datos

Diccionario de datos

- Datos almacenados
- Flujos de datos
- Datos elementales

V.2.3.3 Miniespecificaciones

Módulo de monitoreo

- 1.1.1 Maneja requerimiento monitor
- 1.1.2 Monitorea CPU
- 1.1.3 Monitorea E/S
- 1.1.4 Monitorea Memoria
- 1.1.5 Concentra actividad en S.O.
- 1.2 Muestra configuración equipo

Módulo de sintonización

- 2.1.1 Detecta ajustes en parámetros
- 2.1.2 Genera recomendación
- 2.2 Reconfigura kernel
- 2.3 Obtiene parámetros kernel

DIAGRAMA DE CONTEXTO

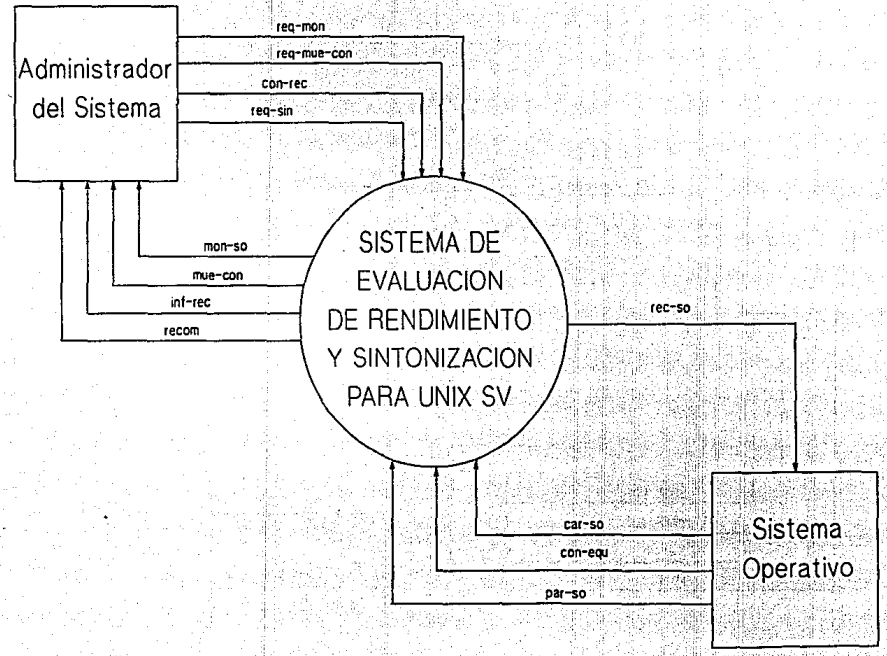


DIAGRAMA DE FLUJO DE DATOS 0

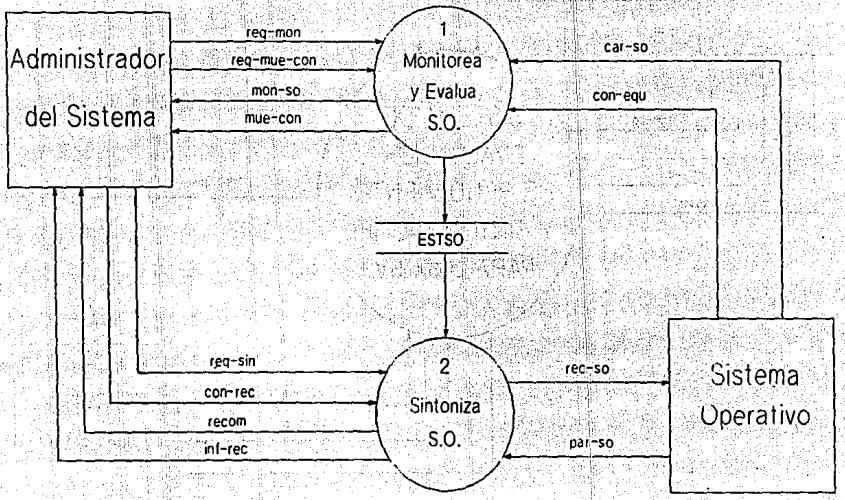


DIAGRAMA DE FLUJO DE DATOS 1

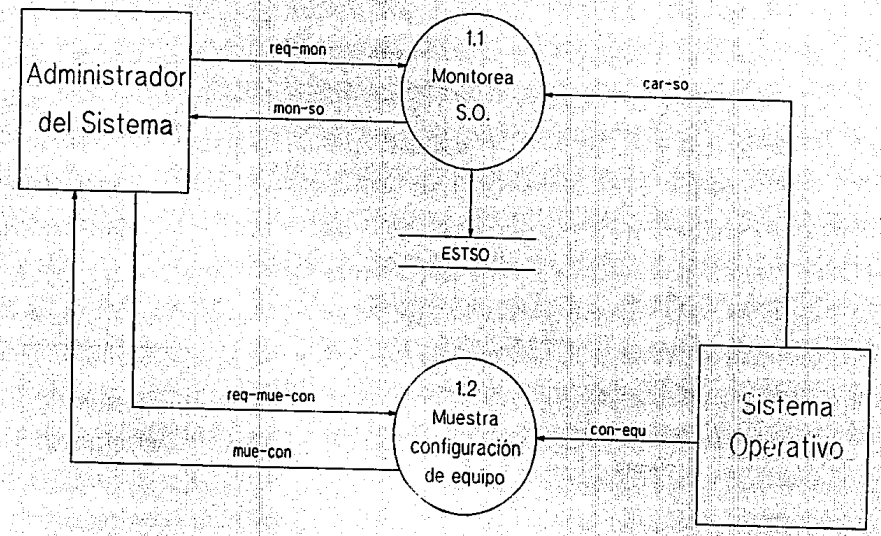


DIAGRAMA DE FLUJO DE DATOS 1.1

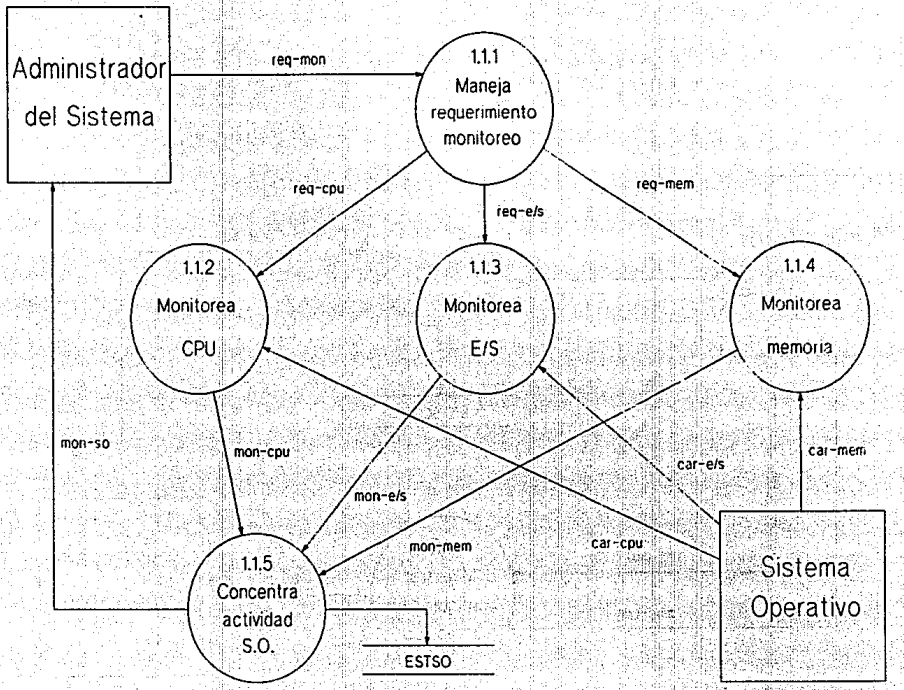


DIAGRAMA DE FLUJO DE DATOS 2

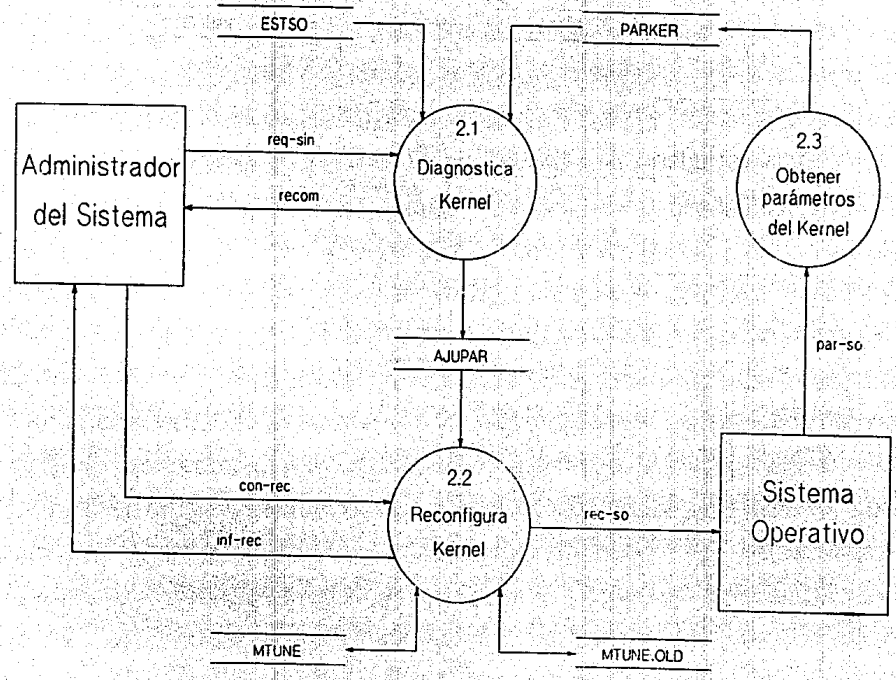


DIAGRAMA DE FLUJO DE DATOS 2.1

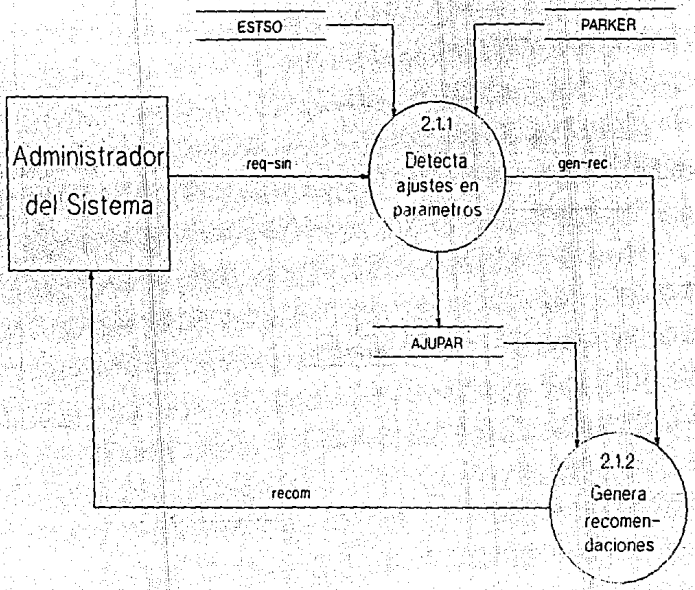
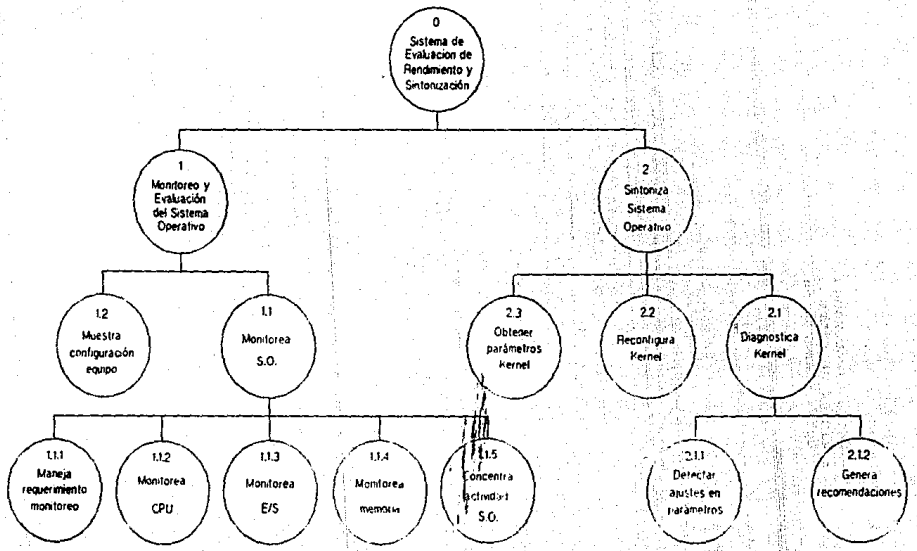


DIAGRAMA DE DESCOMPOSICION



.Diccionario de datos

- Datos almacenados
- Flujos de datos
- Datos elementales

Datos almacenados

AJUPAR
ESTSO
MTUNE
MTUNE.OLD
PARKER

Datos almacenados	AJUPAR
Descripción	Contiene los información de los posibles ajustes a los valores de los parámetros del kernel.
Flujos de entrada	aju-par
Flujos de salida	aju-par

Datos almacenados	ESTSO
Descripción	Contiene las estadísticas de los diferentes parámetros del kernel, generadas por el S.O. a partir del monitoreo del mismo.
Flujos de entrada	est-so
Flujos de salida	est-so

Datos almacenados	MTUNE
Descripción	Contiene los valores de los parámetros configurables del kernel, que se usarán para la reconfiguración del S.O.
Flujos de entrada	rec-so
Flujos de salida	rec-so

Datos almacenados	MTUNE.OLD
Descripción	Contiene los valores actuales de los parámetros configurables del kernel, que se usarán como respaldo en caso de cancelar la reconfiguración del kernel.
Flujos de entrada	rec-so-ant
Flujos de salida	rec-so-ant

Datos almacenados	PARKER
Descripción	Contiene los valores actuales de los parámetros configurables del kernel, obtenidos directamente del S.O., para generar la recomendación de reconfiguración.
Flujos de entrada	par-ker
Flujos de salida	par-ker

Flujos de datos

car-cpu	car-e/s	car-mem	car-so
con-equ	con-rec	gen-rec	inf-rec
mon-cpu	mon-e/s	mon-mem	mon-so
mue-con	par-so	rec-so	recom
req-cpu	req-e/s	req-mem	req-mon
req-mue-con	req-sin		

Flujo de datos	car-cpu
Descripción	Información proporcionada por el Sistema Operativo que describe la utilización de CPU.
Fuente	Sistema Operativo
Destino	1.1.2 MONITOREA CPU
Tipo	Estructura de Datos
Composición	1{%usr + %sys + %wio + %idle + proc-sz + inod-sz + file-sz + proc-ov + inod-ov + file-ov + runq-sz + %runocc }

Flujo de datos	car-e/s
Descripción	Información proporcionada por el Sistema Operativo que describe la utilización de Entrada/Salida.
Fuente	Sistema Operativo
Destino	1.1.3 MONITOREA E/S
Tipo	Estructura de Datos
Composición	1(rawch/s + canch/s + outch/s + rcvin/s + xmtin/s + mdmin/s)

Flujo de datos	car-mem
Descripción	Información proporcionada por el Sistema Operativo que describe la utilización de memoria.
Fuente	Sistema Operativo
Destino	1.1.4 MONITOREA MEMORIA
Tipo	Estructura de Datos
Composición	1(freemem + freeswap + bread/s + bwrit/s + lread/s + lwrit/s + %rcache + %wcache + pread/s + pwrit/s)

Flujo de datos	con-equ
Descripción	Información proporcionada por el Sistema Operativo que describe la configuración física actual del equipo.
Fuente	SISTEMA OPERATIVO
Destino	1.2 MUESTRA CONFIGURACION EQUIPO
Tipo	Estructura de datos
Composición	confi-cpu + confi-fpu + confi-seria + confi-paral + confi-memor + confi-hd + confi-fd + confi-cinta + confi-conso + confi-adapt

Flujo de datos	con-rec
Descripción	Instrucciones proporcionadas por el Administrador del Sistema Operativo referentes a la selección de parámetros a reconfigurar del kernel.
Fuente	ADMINISTRADOR DEL SISTEMA
Destino	2.2 RECONFIGURA KERNEL
Tipo	Estructura de datos
Composición	1(par-sin-nom + par-sin-val)

Flujo de datos	gen-rec
Descripción	Instrucciones para generar recomendación de posible sintonización
Fuente	2.1.1 DETECTA AJUSTES EN PARAMETROS
Destino	2.1.2 GENERA RECOMENDACION
Tipo	Dato elemental

Flujo de datos	inf-rec
Descripción	Información proporcionada al Administrador del Sistema Operativo para describir la acción de sintonización efectuada por el sistema y el resultado de su realización
Fuente	2.2 RECONFIGURA KERNEL
Destino	ADMINISTRADOR DEL SISTEMA
Tipo	Dato elemental

Flujo de datos	mon-cpu
Descripción	Información del monitoreo del CPU
Fuente	1.1.2 MONITOREA CPU
Destino	1.1.5 CONCENTRA ACTIVIDAD SO
Tipo	Estructura de datos
Composición	1(%usr + %sys + %idle + %wio)

Flujo de datos	mon-e/s
Descripción	Información del monitoreo de E/S
Fuente	1.1.3 MONITOREA E/S
Destino	1.1.5 CONCENTRA ACTIVIDAD SO
Tipo	Estructura de datos
Composición	1(rawch/s + canch/s + outch/s)

Flujo de datos	mon-mem
Descripción	Información del monitoreo de la memoria
Fuente	1.1.4 MONITOREA MEMORIA
Destino	1.1.5 CONCENTRA ACTIVIDAD SO
Tipo	Estructura de datos
Composición	1(freemem + freeswp + lread/s + %rcache + bread/s + bwrit/s + lwrit/s + %wcache + pread/s + pwrit/s + swpin/s + swpot/s + bswin/s + bswot/s + pswch/s)

Flujo de datos	mon-so
Descripción	Información proporcionada al Administrador del Sistema que muestra el comportamiento del mismo bajo una carga de trabajo real, en términos de CPU, E/S y Memoria.
Fuente	1.1.5 CONCENTRA ACTIVIDAD SO
Destino	ADMINISTRADOR DEL SISTEMA (AS)
Tipo	Grupo
Composición	mon-cpu + mon-e/s + mon-mem

Flujo de datos	mue-con
Descripción	Despliegue de la configuración física del equipo
Fuente	1.2 MUESTRA CONFIGURACION EQUIPO
Destino	ADMINISTRADOR SISTEMA OPERATIVO
Tipo	Estructura de datos
Composición	confi-cpu + confi-fpu + confi-seria + confi-paral + confi-memor + confi-hd + confi-fd + confi-cinta + confi-conso + confi-adapt

Flujo de datos	par-so
Descripción	Parámetros actuales del kernel del Sistema Operativo y sus valores asociados
Fuente	SISTEMA OPERATIVO
Destino	2.3 OBTIENE PARAMETROS KERNEL
Tipo	Estructura de datos
Composición	1(par-nom + par-val + par-min + par-max)

Flujo de datos	rec-so
Descripción	Instrucciones de reconfiguración de parámetros del kernel del Sistema Operativo
Fuente	2.2 RECONFIGURA KERNEL MTUNE
Destino	MTUNE SISTEMA OPERATIVO
Tipo	Estructura de datos
Composición	0(par-so + par-val-act + par-val-nVo)

Flujo de datos	recom
Descripción	Recomendación al Administrador del Sistema acerca del comportamiento del Sistema Operativo y posibles ajustes a realizar
Fuente	2.1.2 GENERA RECOMENDACION
Destino	ADMINISTRADOR DEL SISTEMA
Tipo	Estructura de datos
Composición	0{par-diag-nom + par-diag-val} + 1{msg-recomen}

Flujo de datos	req-cpu
Descripción	Requerimiento de monitoreo de utilización de CPU
Fuente	1.1.1 MANEJA REQUERIMIENTO MONITOR
Destino	1.1.2 MONITOREA CPU
Tipo	Dato elemental

Flujo de datos	req-e/s
Descripción	Requerimiento de monitoreo de utilización de E/S
Fuente	1.1.1 MANEJA REQUERIMIENTO MONITOR
Destino	1.1.3 MONITOREA E/S
Tipo	Dato elemental

Flujo de datos	req-mem
Descripción	Requerimiento de monitoreo de utilización de memoria
Fuente	1.1.1 MANEJA REQUERIMIENTO MONITOR
Destino	1.1.4 MONITOREA E/S
Tipo	Dato elemental

Flujo de datos	req-mon
Descripción	Instrucciones que provee el Administrador de Sistema para monitorear la actividad del CPU, E/S y Memoria.
Fuente	ADMINISTRADOR DEL SISTEMA
Destino	1.1.1 MANEJA REQUERIMIENTO MONITOR
Tipo	Estructura de datos
Composición	req-cpu + req-e/s + req-mem

Flujo de datos	req-mue-con
Descripción	Requerimiento para mostrar configuración física de los componentes del equipo
Fuente	ADMINISTRADOR SISTEMA OPERATIVO
Destino	1.2 MUESTRA CONFIGURACION EQUIPO
Tipo	Dato elemental

Flujo de datos	req-sin
Descripción	Petición proporcionada por el Administrador del Sistema Operativo para diagnosticar la configuración del kernel
Fuente	ADMINISTRADOR DEL SISTEMA
Destino	2.1.1 DETECTAR AJUSTES EN PARAMETROS
Tipo	Dato elemental

Datos Elementales

Dato elemental	Descripción	Tipo	Long	Flujo/ Estructura
req-cpu	Requerimiento monitoreo CPU	A	1	req-mon
req-e/s	Requerimiento monitoreo E/S	A	1	req-mon
req-mem	Requerimiento monitoreo memoria	A	1	req-mon
cpu-type	Tipo de CPU	A	5	confi-cpu
cpu-speed	Velocidad CPU	N	2	confi-cpu
fpu-vec	Vector Int. FPU	N	2	confi-fpu
fpu-dma	DMA FPU	N	2	confi-fpu
fpu-type	Tipo FPU	N	5	confi-fpu
seria-vec	Vector Int. serial	N	2	confi-seria
seria-base	Dir. base serial	A	5	confi-seria
seria-offset	Offset serial	N	3	confi-seria
seria-dma	DMA serial	N	2	confi-seria
seria-unit	Unidad serial	N	1	confi-seria
seria-type	Tipo serial	A	8	confi-seria
seria-nports	Num. ptos. seriales	N	1	confi-seria
paral-base	Dir. base paralelo	A	5	confi-paral
paral-offset	Offset paralelo	N	3	confi-paral
paral-vec	Vector Int. paralelo	N	2	confi-paral
paral-dma	DMA paralelo	N	2	confi-paral
paral-unit	Unidad paralelo	N	1	confi-paral
memor-main	Memoria principal	N	6	confi-memor
memor-cache	Memoria cache	N	6	confi-memor
hd-vec	Vector Int. disco d.	N	2	confi-hd
hd-dma	DMA disco duro	N	2	confi-hd
hd-type	Tipo disco duro	A	1	confi-hd

Dato elemental	Descripción	Tipo	Long	Flujo/ Estructura
hd-cyls	Cilindros disco duro	N	3	confi-hd
hd-ha	Host adapter disco d.	N	1	confi-hd
hd-id	Identificador disco d	N	1	confi-hd
fd-base	Dir. base floppy d.	A	5	confi-fd
fd-offset	Offset floppy d.	A	3	confi-fd
fd-vec	Vector Int. floppy d.	N	2	confi-fd
fd-dma	DMA floppy disk	N	1	confi-fd
fd-unit	Unidad floppy disk	N	1	confi-fd
fd-type	Tipo floppy disk	A	6	confi-fd
cinta-vec	Vector Int. cinta m.	N	2	confi-cinta
cinta-dma	DMA cinta magnética	N	1	confi-cinta
cinta-type	Tipo cinta m.	A	1	confi-cinta
conso-vec	Vector Int. consola	N	2	confi-conso
conso-dma	DMA consola	N	1	confi-conso
conso-unit	Unidad consola	N	3	confi-conso
conso-type	Tipo consola	N	1	confi-conso
conso-screen	Pantalla consola	N	3	confi-conso
conso-numter	Núm. terminales	N	2	confi-conso
conso-base	Dir. base adaptador	A	5	confi-adapt
adapt-offset	Offset adaptador	A	3	confi-adapt
adapt-vec	Vector Int. adaptador	N	2	confi-adapt
adapt-dma	DMA adaptador	N	1	confi-adapt

Dato elemental	Descripción	Tipo	Long	Flujo/ Estructura
%usr	% tiempo modo usuario	N	3	mon-cpu car-cpu
%sys	% tiempo modo sistema	N	3	mon-cpu car-cpu
%idle	% tiempo ocioso	N	3	mon-cpu car-cpu
%wio	% tiempo ocioso e/s	N	3	mon-cpu car-cpu
rawch/s	Tasa de lectura de caracteres raw	N	2	mon-e/s car-e/s
canch/s	Tasa de lectura de caracteres por canon	N	1	mon-e/s car-e/s
outch/s	Tasa de escritura de caracteres	N	4	mon-e/s car-e/s
rcvin/s	Tasa de int./recep.	N	2	car-e/s
xmtin/s	Tasa de int./trans.	N	2	car-e/s
mdmin/s	Tasa de int./modem	N	2	car-e/s
freemem	Memoria libre	N	6	mon-mem car-mem
freeswap	Swap libre	N	6	mon-mem car-mem
bread/s	Transf. de datos/seg de disco a buffers	N	3	mon-mem car-mem
bwrit/s	Transf. de datos/seg de buffers a disco	N	3	mon-mem car-mem
lread/s	Lectura de buffers	N	3	mon-mem car-mem
lwrit/s	Escritura de buffers	N	3	mon-mem car-mem
%rcache	Tasa de "hits" en lectura de cache	N	3	mon-mem car-mem
%wcache	Tasa de "hits" en escritura de cache	N	3	mon-mem car-mem

Dato elemental	Descripción	Tipo	Long	Flujo/ Estructura
pread/s	Lecturas vía disp. físicos	N	3	mon-mem car-mem
pwrit/s	Escrituras vía disp. físicos	N	3	mon-mem car-mem
par-diag-nom	Nombre parámetro S.O.	A	12	recom
par-diag-val	Valor parámetros S.O.	N	8	recom
msg-recomen	Mensaje recomendación	A	60	recom
par-sin-nom	Nombre parámetro sintonizar	A	12	con-rec
par-sin-val	Valor parámetro a sintonizar	N	8	con-rec
req-sin	Requerimiento de sintonización	A	1	req-sin
inf-rec	Informe de reconfiguración	A	60	inf-rec
par-nom	Nombre de parámetro	A	12	par-so
par-val	Valor de parámetro	N	8	par-so
par-min	Mínimo valor param.	N	8	par-so
par-max	Máximo valor param.	N	8	par-so
req-mue-con	Requerimiento de mostrar configuración	A	1	req-mue-con
gen-rec	Genera recomendación	A	1	gen-rec
par-so	Parámetro S.O. a reconfigurar	A	12	rec-so
par-val-act	Valor actual de par.	N	8	rec-so
par-val-nvo	Valor nuevo de par.	N	8	rec-so

Dato elemental	Descripción	Tipo	Long	Flujo/ Estructura
proc-sz	Tamaño tabla procesos	A	8	car-cpu
inod-sz	Tamaño tabla inodos	A	8	car-cpu
file-sz	Tamaño tabla archivos	A	8	car-cpu
proc-ov	Overflow tabla proc.	N	2	car-cpu
inod-ov	Overflow tabla inodos	N	2	car-cpu
file-ov	Overflow tabla arch.	N	2	car-cpu
swpin/s	swap in por segundo	N	5	car-mem
swpot/s	swap out por segundo	N	5	car-mem
bswin/s	swap in buffers/s	N	5	car-mem
bswot/s	swap out buffers/s	N	5	car-mem
pswch/s	conmutación de prc/s	N	3	car-mem
runq-sz	tamaño prom. de cola procesos p/ejecución	N	5	car-mem
%runocc	% ocupación cola procesos p/ejecución	N	3	car-mem

Miniespecificaciones

Procesos

```
      0. SERSUNIX
           1. Monitorea y Evalua S.O.
                1.1 Monitorea S.O.
*                    1.1.1 Maneja requerimiento monitoreo
*                    1.1.2 Monitorea CPU
*                    1.1.3 Monitorea E/S
*                    1.1.4 Monitorea Memoria
*                    1.1.5 Concentra actividad S.O.
*               1.2 Muestra configuración equipo
           2. Sintoniza S.O.
                2.1 Diagnostica kernel
*                    2.1.1 Detecta ajustes en parámetros
*                    2.1.2 Genera reconmendación
*               2.2 Reconfigura kernel
*               2.3 Obtener parametros kernel
```

Modulo de monitoreo

Proceso	1.1.1 MANEJA REQUERIMIENTO MONITOR
Descripción	Para cada <requerimiento> SI <requerimiento> es <req-cpu> Ejecutar <MONITOREA CPU> Mostrar <valores reportados> SI <requerimiento> es <req-e/s> Ejecutar <MONITOREA E/S> Mostrar <valores reportados> SI <requerimiento> es <req-mem> Ejecutar <MONITOREA MEM> Mostrar <valores reportados> OTRO Mostrar <error 1> fin
Flujos de entrada	req-mon
Flujos de salida	req-mon-cpu req-mon-e/s req-mon-mem
Archivos de entrada	-ninguno-
Archivos de salida	-ninguno-

Proceso	1.1.2 MONITOREA CPU
Descripción	Para un periodo de tiempo dado Ejecutar <rutinas del SO SAR CPU> Obtener <valores reportados SAR CPU> Regresar <valores reportados SAR CPU>
Flujos de entrada	req-mon-cpu car-so
Flujos de salida	mon-cpu
Archivos de entrada	-ninguno-
Archivos de salida	-ninguno-

Proceso	1.1.3 MONITOREA E/S
Descripción	Para un periodo de tiempo dado Ejecutar <rutinas del SO SAR E/S> Obtener <valores reportados SAR E/S> Regresar <valores reportados SAR E/S>
Flujos de entrada	req-mon-e/s car-so
Flujos de salida	mon-e/s
Archivos de entrada	-ninguno-
Archivos de salida	-ninguno-

Proceso	1.1.4 MONITOREA MEMORIA
Descripción	Para un periodo de tiempo dado Ejecutar <rutinas del SO SAR MEM> Obtener <valores reportados SAR MEM> Regresar <valores reportados SAR MEM>
Flujos de entrada	req-mon-mem car-so
Flujos de salida	mon-mem
Archivos de entrada	-ninguno-
Archivos de salida	-ninguno-

Proceso	1.1.5 CONCENTRA ACTIVIDAD SO
Descripción	Reporta monitoreo de <Utilización de CPU> y/o <Utilización de E/S> y/o <Utilización de MEM> Genera <estadistica de CPU, E/S, MEMORIA>
Flujos de entrada	mon-cpu mon-e/s mon-mem
Flujos de salida	mon-so
Archivos de entrada	-ninguno-
Archivos de salida	ESTSO

Proceso	1.2 MUESTRA CONFIGURACION EQUIPO
Descripción	Para cada <elemento del hardware> Obtener <configuración del elemento> Mostrar <configuración del elemento>
Flujos de entrada	con-equ req-mue-con
Flujos de salida	mue-con
Archivos de entrada	-ninguno-
Archivos de salida	-ninguno-

Modulo de Sintonización

Proceso	2.1.1 DETECTA AJUSTES EN PARAMETROS
Descripción	Para cada <parámetro involucrado> Aplicar <reglas de sintonización> Comparar con <parámetros actuales> Decidir conforme a <estadísticas> posibles <cambios en parámetros> Registrar posibles <cambios en parámetros>
Flujos de entrada	req-sin
Flujos de salida	gen-rec
Archivos de entrada	PARKER ESTSO
Archivos de salida	AJUPAR

Proceso	2.1.2 GENERA RECOMENDACION
Descripción	Interpretar lo registrado en <AJUPAR> Mostrar <recomendación ajustes>
Flujos de entrada	gen-rec
Flujos de salida	recom
Archivos de entrada	AJUPAR
Archivos de salida	-ninguno-

Proceso	2.2 RECONFIGURA KERNEL
Descripción	Guardar <configuración actual> Sintonizar <parámetros recomendados> Informar <resultado de sintonización>
Flujos de entrada	con-req
Flujos de salida	inf-rec rec-so
Archivos de entrada	AJUPAR MTUNE
Archivos de salida	MTUNE.OLD MTUNE

Proceso	2.3 OBTIENE PARAMETROS KERNEL
Descripción	Para cada <parámetro del kernel> Guardar <nombre y valor> en <PARKER>
Flujos de entrada	par-so
Flujos de salida	-ninguno-
Archivos de entrada	-ninguno-
Archivos de salida	PARKER

V.3 DISEÑO Y PROGRAMACION DEL SOFTWARE

Con base en el análisis presentado en el capítulo V.2, se plantea el diseño del sistema. A continuación se presenta el Diagrama de Bloques de las funciones de SERSUNIX.

V.3.1 Interfase de usuario

Se plantea una interfase de usuario :

- Fácil de utilizar

  El uso del sistema deberá ser homogéneo en todas las pantallas y menúes, esto es, la selección de opciones de menú se realizarán de la misma manera, el tipo de presentación será uniforme para despliegues de información con las mismas características, etc.

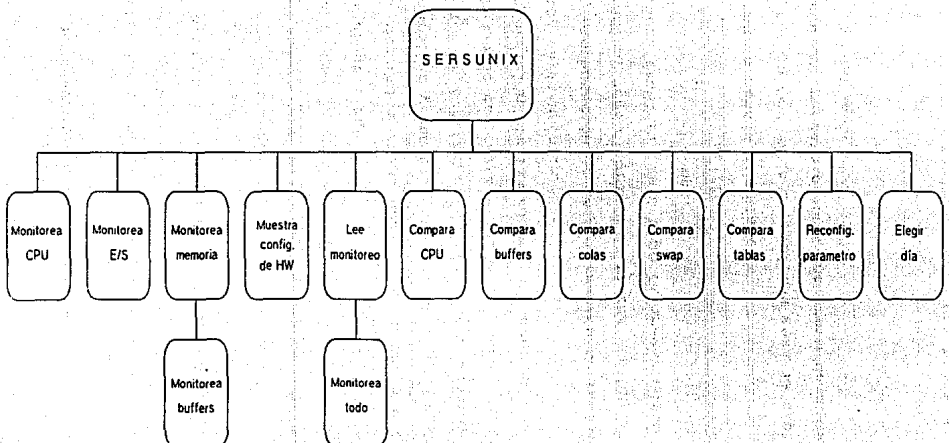
- Gráfica en su mayoría

  La presentación de la información se realizará en forma gráfica, como diagramas de barras, en lugar de hacerse de manera tabular. Este tipo de presentación da una mejor apreciación del comportamiento de la información mostrada al usuario del sistema.

- Orientada a menúes circulares

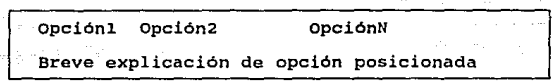
  El manejo de menúes de opciones será de tipo horizontal-circular, proporcionando una breve descripción de cada opción del menú.

DIAGRAMA DE BLOQUES



V.3.1.1 Formato de menúes

Para todo menú, su formato será :

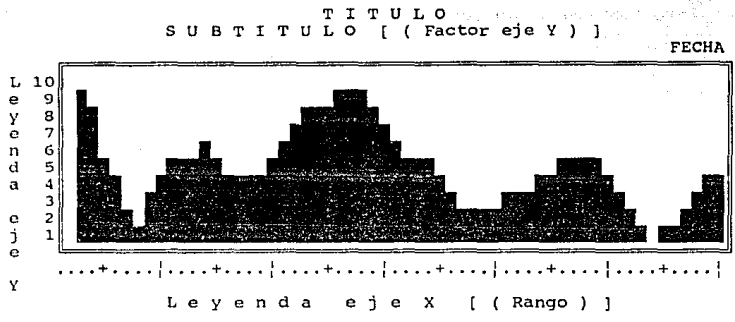


donde cada opción comenzará con una letra única. Moviendo las flechas del teclado, se distinguirá con video inverso la opción a la cual corresponde la explicación de la línea inferior, sin ser por esto aún seleccionada.

La selección de una opción se podrá llevar a cabo posicionándose en una opción y oprimiendo la tecla <ENTER>, o proporcionando la primera letra de la opción deseada.

V.3.2.2 Formato de pantallas

Para toda pantalla de despliegue gráfico de monitoreo su formato será como se muestra a continuación :



<F1> Ayuda <F4> Imprime

Donde:

TITULO:

Título general del despliegue. En caso de que el despliegue forme parte de un grupo de despliegues, este título se mantendrá.

SUBTITULO:

Subtítulo particular del despliegue. Diferencia al despliegue de otros del mismo grupo.

Factor eje Y:

Factor con el cual afectar la escala de 1-10 mostrada en el eje Y para derivar los valores mostrados.

Leyenda eje X:

Texto descriptivo del dominio de valores en el eje X.

Rango:

Rango (valor inicial, valor final) de valores mostrados en el eje X.

Leyenda eje Y:

Texto descriptivo del dominio de valores en el eje Y.

FECHA:

Fecha (MMM DD) de la cual se muestran los valores en la gráfica.

V.3.2 Diseño de Base de Datos

Archivo	Campos / Tipo / Longitud	
cpu	num_con	smallint
	usr	smallint
	sys	smallint
	wio	smallint
	idle	smallint
mem	num_con	smallint
	mem	integer
	swp	integer
es	num_con	smallint
	raw	integer
	can	integer
	out	integer
buf	num_con	smallint
	bre	smallint
	lre	smallint
	rca	smallint
	bwr	smallint
	lwr	smallint
	wca	smallint
	pre	smallint
	pwr	smallint
tab	num_con	smallint
	pro	smallint
	pov	smallint
	ino	smallint
	iov	smallint
	fil	smallint
	fov	smallint
swp	num_con	smallint
	swpi	decimal(5,2)
	bswi	decimal(5,2)
	swpo	decimal(5,2)
	bswo	decimal(5,2)
	pswc	smallint
col	num_con	smallint
	runq	decimal(5,2)
	runo	smallint
aju_cpu	des	char(9)
	act	smallint
	rec	smallint

Archivo	Campos / Tipo / Longitud	
aju_buf	des act rec	char(12) smallint smallint
aju_tab	des act rec	char(12) smallint smallint
aju_col	des act rec	char(12) smallint samllint
aju_swp	des act rec	char(12) smallint smallint

V.4 Integración, pruebas y ajustes

Los módulos básicos, que son los encargados de obtener la información contenida en las tablas del sistemas, fueron elaborados utilizando el lenguaje de comandos de UNIX (SHELL), especificamente el comando "sar" y "ps", con sus diferentes opciones, para monitorear el comportamiento del procesador, la memoria, los buffers, el area de swap, la entrada y la salida.

Así mismo se implementaron rutinas para formatear esa información y almacenarla en tablas de la base de datos para su posterior procesamiento.

Los módulos programados en 4GL, se encargaron de la interfase visual, por medio de menues, para hacer el sistema amigable, así como de procesar la información contenida en las tablas, y

desplegarla de manera gráfica, tanto hacia monitor de la computadora como hacia la impresora. Como se muestra en la Fig. V.4.1.

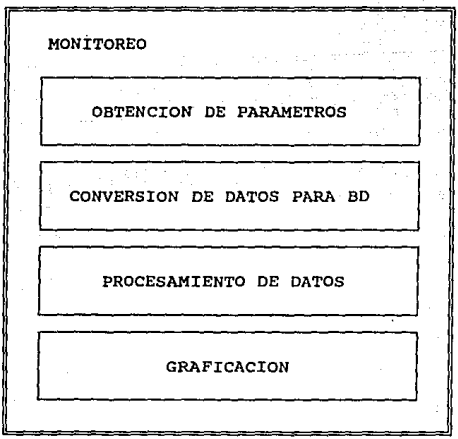


Fig. V.4.1 Esquema funcional del módulo de monitoreo

Un último módulo se programó para interactuar con el SHELL y poder hacer recomendaciones de los valores de los parámetros, fijar esos valores en los archivos de configuración de UNIX y generar un kernel nuevo con los ajustes recomendados. Existe también la opción de cancelar los cambios hechos en el kernel y regresar los parámetros a sus valores originales.

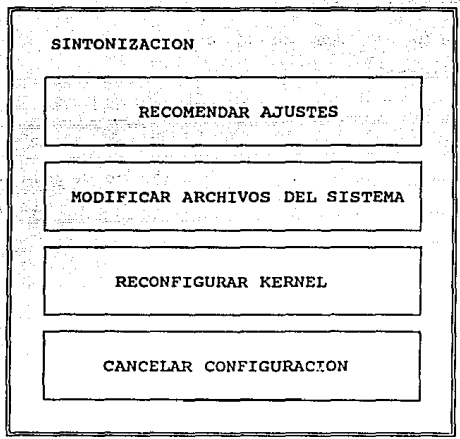


Fig. V.4.2 Esquema funcional del módulo de sintonización

El sistema fué desarrollado en las oficinas de la Secretaria de Hacienda y Crédito Público, en donde la máquina que se utilizar da servicio a 24 usuarios, quienes desarrollan aplicaciones en diferentes herramientas tales como Lotus 123, Word de Microsoft e INFORMIX/4GL, principalmente.

SERSUNIX fué probado con carga de trabajo real, tanto en horas pico como en horas de poco trabajo, lo que aporto los ajustes necesarios para su funcionamiento óptimo. Así mismo, se elaboraron

programas para simular cargas de trabajo en el procesador, la memoria, y el área de swap con el objeto de verificar el funcionamiento tanto del monitoreo, como de los ajustes a la configuración del kernel.

Basados en las pruebas anteriores, el sistema original sufrió algunos ajustes tales como elegir los intervalos de toma de datos y el día de monitoreo, obtener la impresión de las gráficas de rendimiento, y la cancelar la nueva configuración.

CONCLUSIONES

CONCLUSIONES

El desarrollo del presente trabajo nos ha permitido obtener, reforzar y aplicar toda una serie de conocimientos, que junto con las experiencias que compartimos nos han llevado a obtener las siguientes conclusiones:

Por una parte en el aspecto interpersonal hemos aprendido que repartir, aceptar, y realizar tareas encaminadas a alcanzar un objetivo común, lo que ayudo a cubrir en el menor tiempo el trabajo de la tesis. Gracias a esto se logró una total integración del equipo y una mejor dinámica de trabajo.

Por otro lado hemos generado la oportunidad de aplicar parte de los conocimientos adquiridos durante nuestra formación académica, a una situación real, proponiendo para un problema real una solución adecuada.

Hemos entendido mejor la importancia que tienen los sistemas operativos y el importante papel que tienen dentro del desarrollo de sistemas.

Por nuestra experiencia en este sistema, podemos recomendar que antes de desarrollar cualquier aplicación es necesario conocer el sistema operativo en el que trabajaremos, y las herramientas que éste nos ofrece, ya que en ocaciones podemos disponer de toda una serie de herramientas que nos harán más natural la forma de llevar

a cabo nuestro trabajo, y obtendremos sistemas más confiables, eficientes e implementados en menos tiempo.

Gracias al desarrollo de SERSUNIX nos hemos dado cuenta de las capacidades que un UNIX System V ofrece. Así mismo hemos obtenido un mayor y más profundo conocimiento del mismo, que como mencionamos en este trabajo, fué diseñado para desarrolladores de sistemas y provee las herramientas necesarias para su monitoreo, la presentación de la información y el almacenamiento de esta, utilizando el lenguaje de comandos (shell).

Dicho sistema, UNIX, esta considerado dentro de los "Sistemas Abiertos", por cumplir con los estándares fijados por la industria, por la portabilidad de los programas contruidos en el lenguaje de comandos del sistema (shell), y por la variedad de dispositivos periféricos que pueden ser conectados en un equipo.

SERSUNIX contribuye a los sistemas abiertos, ya que permite monitorear el rendimiento del sistema y ejecutar las tareas de sintonización del sistema operativo UNIX System V, las cuales requieren invertir un tiempo considerable por parte de los administradores del sistema en ambientes de producción.

BIBLIOGRAFIA

Bibliografía

- Sistemas Operativos. Diseño e Implantación
  Tanenbaum Andrw S.
  Prentice - Hall, 1988

- Open Desktop Administrator's Guide
  The Santa Cruz Operation, Inc., 1991

- Unix Sistema V. Versión 4
  Kenneth H. Rosen
  Osborne McGraw - Hill, 1991

- System Administrator's Reference
  SCO UNIX System V/386
  The Santa Cruz Operation, Inc., 1988

- Device Driver Writer's Guide
  SCO UNIX System V/386 Development System
  The Santa Cruz Operation, Inc., 1989

- Introducción a los Sistemas Operativos
  Harvey M. Deitel
  Sitesa y Addison - Wesley Iberoamericana, 1989

- Fundamentos de los Sistemas Operativos
  A. M. Lister
  Gustavo Gil, 1986

- Sistemas Operativos, Conceptos Fundamentales
  J. L. Peterson y A. Siberschatz
  Reverté, 1989

- Unix System Architechture
  Andleigh, Prabhatk
  Prentice Hall, 1991

- Sistemas Operativos para Micros, Minis y Macrocomputadoras
  Barron David
  McGraw - Hill, 1986

- Data Processing In Unix
  Ramkrishna S. Tare
  McGraw - Hill International Editions, 1989

- Introducción a los Sistemas Operativos
  Alcalde Lancharro Eduardo/
  Morera Pascual Juan/
  Perez - Campanero Juan A., 1992

- Sistemas Operativos
  Juanes Baza David
  Paraninfo, S.A., 1991

- El Entorno de la Programación Unix
  Kernighan, Brian W./
  Pike Rob, 1987

- Operating System Administrator's Guide
  Cap. 18 "Tuning System Performance"
  The Santa Cruz Operation, Inc. 1990

- Administrator's Guide
  Santa Cruz Operations
  Open Desktop, 1989

- Unix Administration, Guide For System V
  Rebecca Thomas y Rik Farrow
  Ed. Prentice Hall

- Unix System V Release 4. Administration AT&t
  1991

- SHELL: UNIX SHELL PROGRAMMING
  Stephen G. Kochan y Patrick H. Wood
  Hayden Books, 1990

- C
  El Lenguaje de Programación C
  Brian W. Kernighan y Dennis M. Ritchie
  Prentice Hall, 1985

- UNIX
  Understanding UNIX a conceptual guide
  James R. Groff y Paul N. Weinberg
  QueCo., 1988

APENDICE A

MANUAL DEL USUARIO

Sistema de Evaluación de Rendimiento y Sintonización UNIX SV

SERSUNIX

MANUAL DE USUARIO

Universidad Nacional Autónoma de México
Facultad de Ingeniería

Sistema de Evaluación de Rendimiento y Sintonización UNIX SV

SERSUNIX

MANUAL DE USUARIO

Universidad Nacional Autónoma de México
Facultad de Ingenieria

DEFINICION

El Sistema de Evaluación de Rendimiento y Sintonización para UNIX System V (SERSUNIX) ha sido desarrollado conscientes de la complejidad en la sintonización del Sistema Operativo.

SERSUNIX proporciona al Administrador del Sistema Operativo una herramienta para facilitar dicha tarea, mediante un conjunto de despliegues gráficos que muestran el comportamiento del uso de los recursos principales del sistema, a saber: CPU, E/S, memoria, así como rutinas de diagnóstico que comparan la configuración actual del kernel del sistema con la utilización de recursos, basandose en reglas de sintonización recomendadas por el proveedor del Sistema Operativo. Con base en las recomendaciones que generan las rutinas de diagnóstico, el Administrador tiene la opción de indicar a SERSUNIX que reconfigure el kernel de acuerdo a la recomendación propuesta.

OBJETIVOS

- Consultar la configuración física del equipo.
- Monitorear el comportamiento del Sistema Operativo.
- Detectar posibles causas de baja de rendimiento del Sistema Operativo.
- Sugerir posibles correciones de configuración del Sistema Operativo.
- Reconfigurar el kernel del Sistema Operativo, bajo consentimiento del Administrador del Sistema.

CONTENIDO

El presente manual describe la forma de operar el sistema SERSUNIX. Se describe la forma de navegar en las diferentes opciones de menues, así como la función de cada una de ellas.

Cada menú esta acompañado por el diagrama de flujo de datos correspondiente que muestra en forma gráfica el flujo de la información, así como las funciones asociadas.

Cada transacción es detallada en su forma de uso, opciones de proceso y utilización de almacenamientos de datos.

CONSIDERACIONES

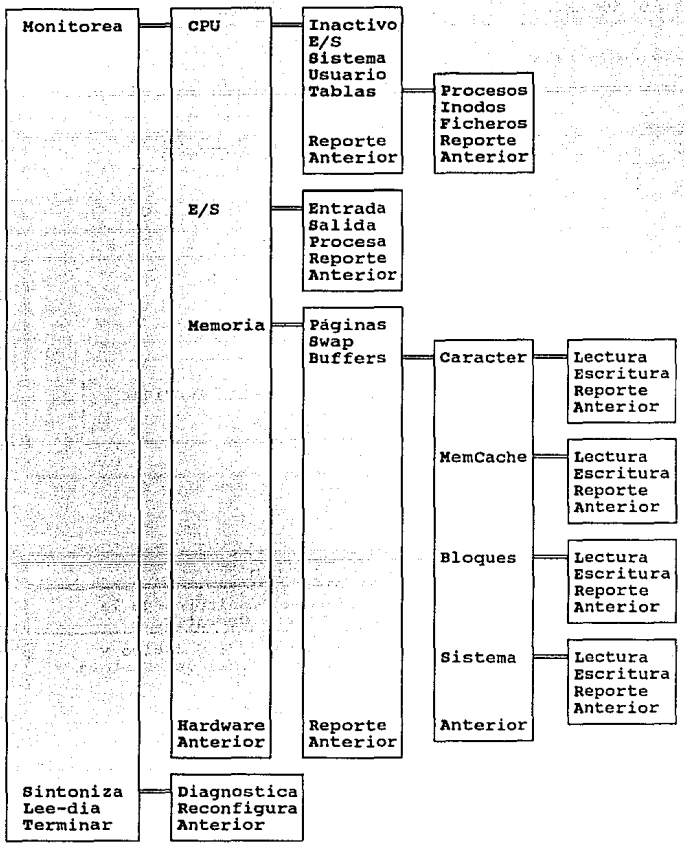
- El sistema puede ser operado únicamente por el superusuario de UNIX System V.
- La decisión de reconfiguración del kernel queda a consideración del Administrador.
- Se asume que el equipo utiliza el Sistema Operativo UNIX de Santa Cruz Operation.
- Se requiere de INFORMIX SE (Standard Engine) instalado en el equipo para ejecutar SERSUNIX.
- Se requiere una consola de despliegue con emulación ANSI.
- Para todo menú, se cuenta con la opción oculta de ejecutar comandos del shell mediante un signo de admiración (!) seguido opcionalmente de un comando.
- Para todo menú, se podrá seleccionar una opción mediante el movimiento del cursor con las flechas del teclado y seleccionando la opción deseada con <ENTER> o digitando la primer letra de la opción deseada.

ACCESO AL SISTEMA Y SALIDA

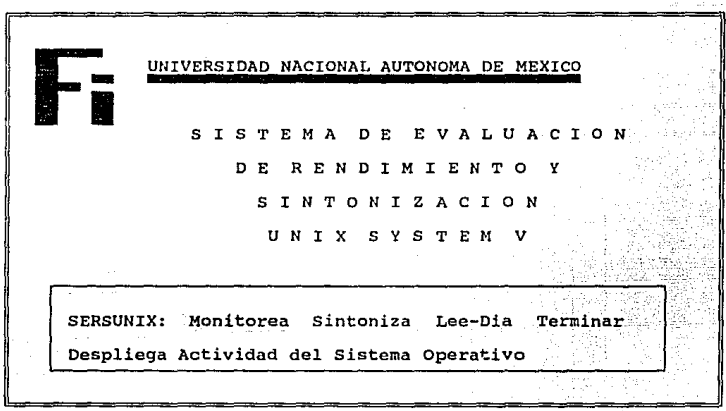
Habiendo accesado el sistema como super-usuario, teclear sersunix, ej:

```
# sersunix
```

Para salir del sistema, seleccione la opción "Terminar" del menú principal.


Monitorea
CPU
Inactivo
E/S
Sistema
Usuario
Tablas
Reporte
Anterior
Procesos
Inodos
Ficheros
Reporte
Anterior
E/S
Entrada
Salida
Procesa
Reporte
Anterior
Memoria
Páginas
Swap
Buffers
Caracter
Lectura
Escritura
Reporte
Anterior
MemCache
Lectura
Escritura
Reporte
Anterior
Bloques
Lectura
Escritura
Reporte
Anterior
Sistema
Lectura
Escritura
Reporte
Anterior
Hardware
Anterior
Reporte
Anterior
Anterior
Sintoniza
Lee-dia
Terminar
Diagnostica
Reconfigura
Anterior

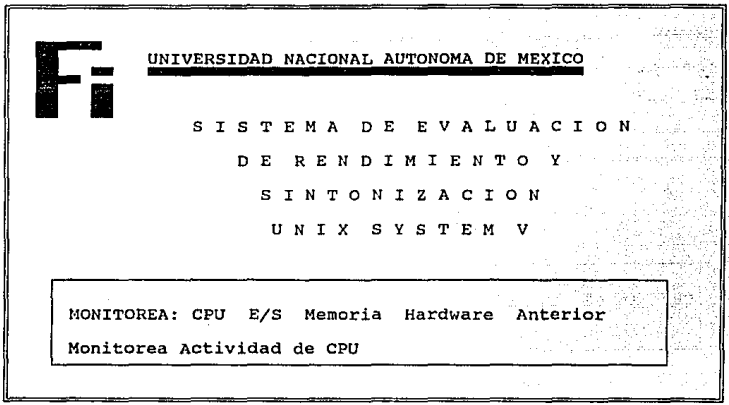
Formato de pantalla



Opciones del menú

Nivel	Opción	D e s c r i p c i ó n
1	Monitorea	Accesa el submenú de monitoreo de la actividad del Sistema Operativo.
2	Sintoniza	Accesa el submenú para diagnóstico y reconfiguración del kernel del Sistema Operativo.
3	Lee-Dia	Selecciona dia a monitorear
4	Terminar	Terminar la ejecución de SERSUNIX y regresar al Sistema Operativo.

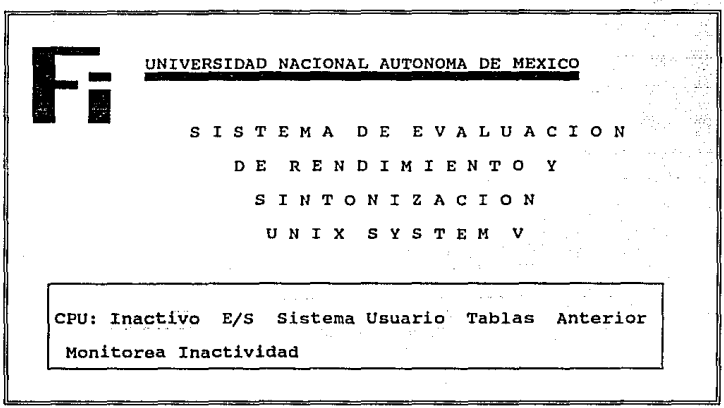
Formato de pantalla



Opciones del menú

Nivel	Opción	D e s c r i p c i ó n
1	CPU	Accesa el submenú de monitoreo de CPU.
2	E/S	Accesa el submenú de monitoreo de Entrada/Salida.
3	Memoria	Accesa el submenú de monitoreo de memoria.
4	Hardware	Mediante esta opción se despliega la configuración física actual del equipo.
5	Anterior	Con esta opción regresa al menú anterior.

Formato de pantalla



Opciones del menú

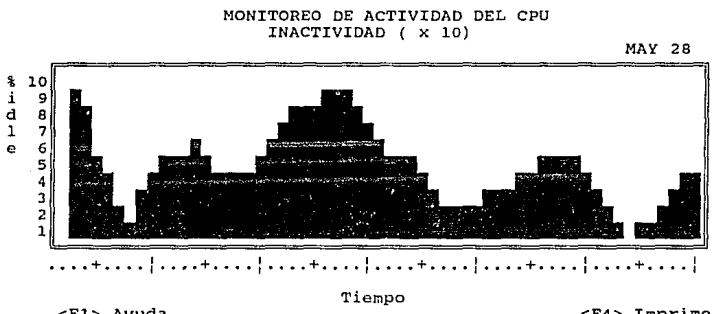
Nivel	Opción	D e s c r i p c i ó n
1	Inactivo	Despliega gráficamente el porcentaje de inactividad del CPU.
2	E/S	Despliega el porcentaje de inactividad del CPU por espera de operaciones de Entrada/Salida.
3	Sistema	Despliega el porcentaje de utilización del CPU por procesos en modo Sistema.
4	Usuario	Despliega el porcentaje de utilización del CPU por procesos de usuario.
5	Tablas	Accesa el submenú de monitoreo de tablas.
6	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica la inactividad del CPU en el lapso monitoreado por la utileria SAR (%idle). Este desplegado muestra el porcentaje de inactividad en el tiempo que el CPU no realizo proceso alguno por no haber sido requerido. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%idle).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : cpu

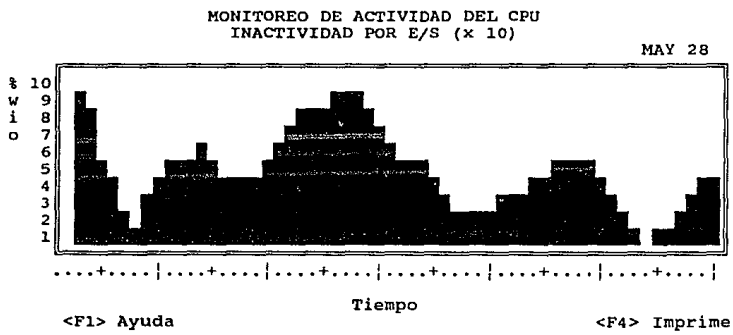
Salida : cpu

Objetivo

Mostrar de forma gráfica la inactividad del CPU por espera de operaciones de entrada/salida en el lapso monitoreado por la utileria SAR (%wio). Este desplegado muestra el porcentaje de inactividad en el tiempo que el CPU no realizó proceso alguno por encontrarse esperando la terminación de tareas de entrada/salida. Valores altos de %wio denotan probables cuellos de botella por E/S. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%wio).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : cpu

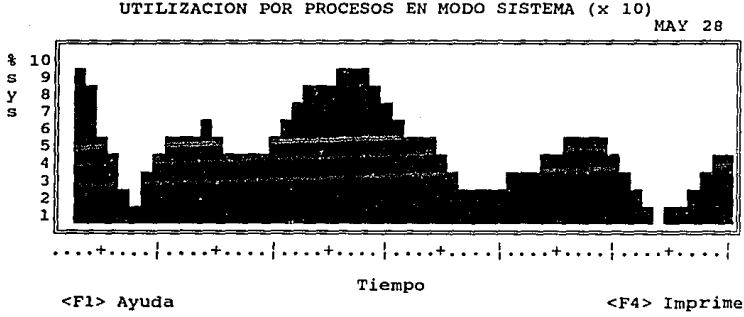
Salida : cpu

Objetivo

Mostrar de forma gráfica la actividad del CPU en modo sistema en el lapso monitoreado por la utileria SAR (%sys). Este desplegado muestra el porcentaje de actividad en el tiempo que el CPU dedicó a procesar tareas del sistema. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%sys).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : cpu

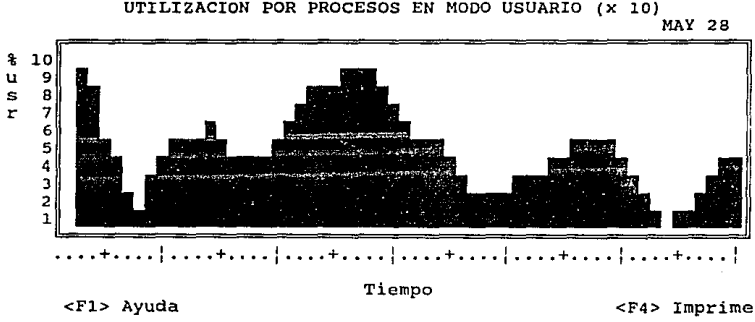
Salida : cpu

Objetivo

Mostrar de forma gráfica la actividad del CPU en modo usuario en el lapso monitoreado por la utileria SAR (%usr). Este desplegado muestra el porcentaje de actividad en el tiempo que el CPU dedicó a procesar tareas de usuarios. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%usr).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

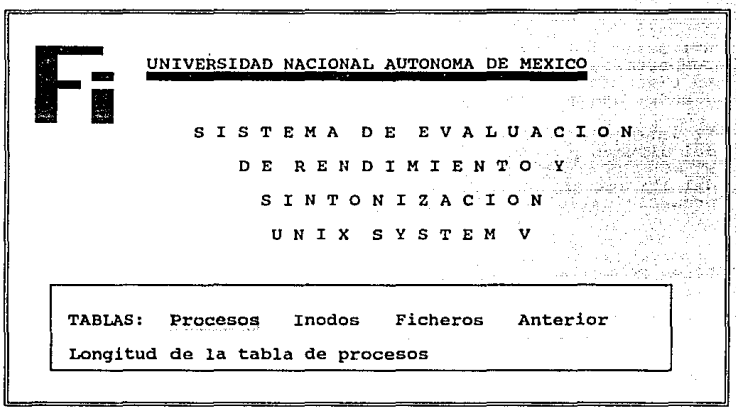
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : cpu

Salida : cpu

Formato de pantalla



Opciones del menú

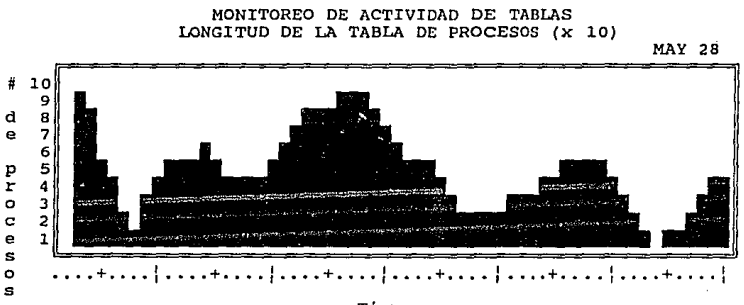
Nivel	Opción	D e s c r i p c i ó n
1	Procesos	Despliega el comportamiento de la longitud de la Tabla de Procesos.
2	Inodos	Despliega el comportamiento de la longitud de la tabla de Inodos.
3	Ficheros	Despliega el comportamiento de la longitud de la tabla de Archivos.
4	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de la tabla de procesos en cuanto a entradas en la misma, monitoreado por la utileria SAR (proc-sz). Este desplegado muestra el comportamiento de la tabla afectada por el número de procesos en ejecución en el sistema. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (proc-sz).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : tab

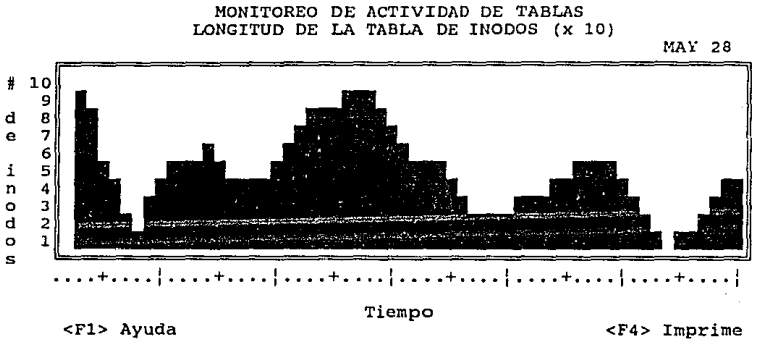
Salida : tab

Objetivo

Mostrar de forma gráfica el comportamiento de la tabla de inodos en cuanto a entradas en la misma, monitoreado por la utileria SAR (inod-sz). Este desplegado muestra el comportamiento de la tabla afectada por el número de archivos abiertos en el sistema. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (inod-sz).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : tab

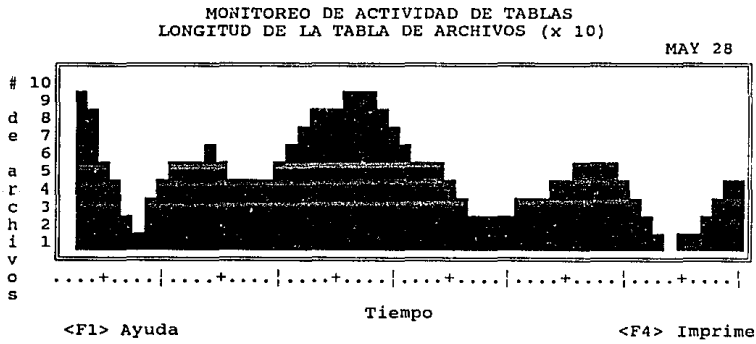
Salida : tab

Objetivo

Mostrar de forma gráfica el comportamiento de la tabla de archivos en cuanto a entradas en la misma, monitoreado por la utileria SAR (file-sz). Este desplegado muestra el comportamiento de la tabla afectada por el tamaño de los archivos abiertos en el sistema. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (inod-sz).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

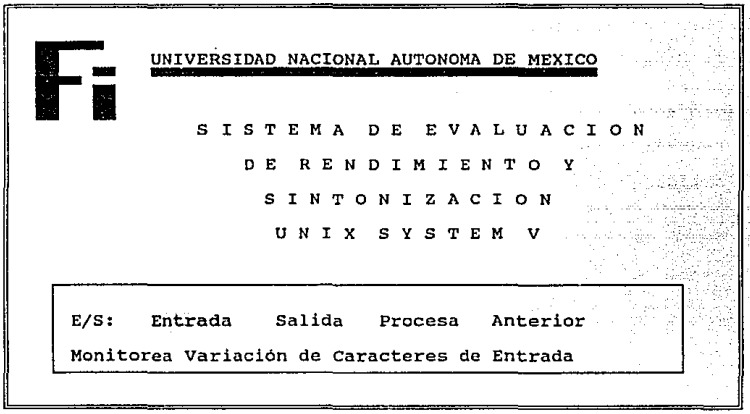
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : tab

Salida : tab

Formato de pantalla



Opciones del menú

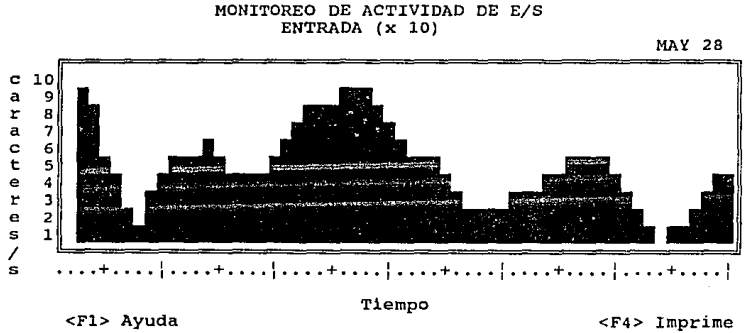
Nivel	Opción	D e s c r i p c i ó n
1	Entrada	Despliega la variación de caracteres de entrada en el sistema.
2	Salida	Despliega la variación de caracteres de salida en el sistema.
3	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de transferencia de caracteres de terminales (tty) al sistema, monitoreado por la utileria SAR (rawch/s). Este desplegado muestra la tasa de caracteres recibidos por segundo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (rawch/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : es

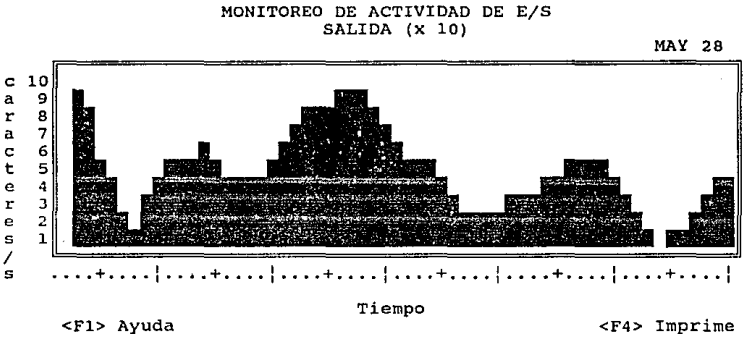
Salida : es

Objetivo

Mostrar de forma gráfica el comportamiento de transferencia de caracteres del sistema hacia terminales (tty), monitoreado por la utileria SAR (outch/s). Este desplegado muestra la tasa de caracteres transmitdos por segundo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (outch/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

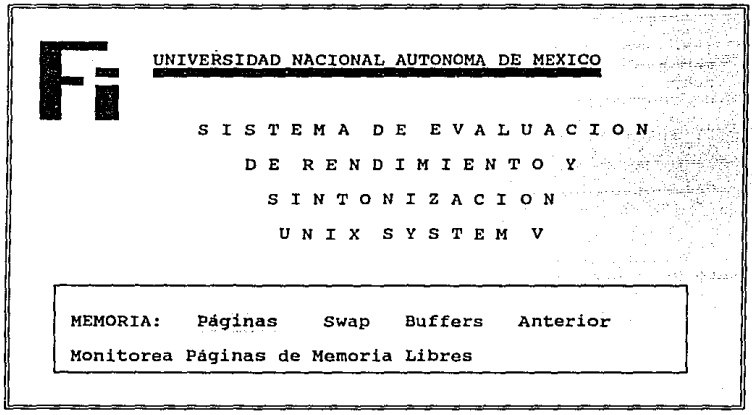
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : es

Salida : es

Formato de pantalla



Opciones del menú

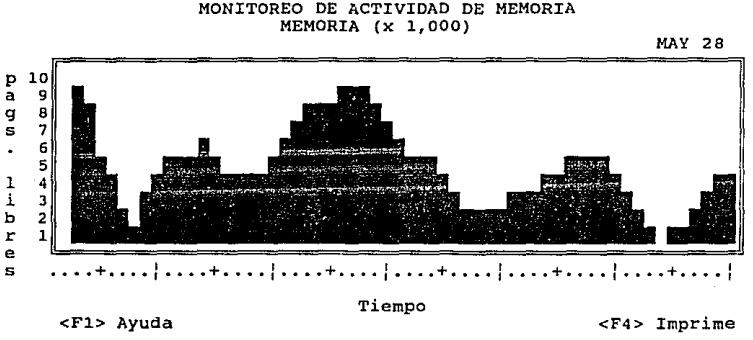
Nivel	Opción	D e s c r i p c i ó n
1	Páginas	Despliega comportamiento de páginas de memoria libres en el sistema.
2	Swap	Despliega comportamiento de páginas de Intercambio (swap) libres.
3	Buffers	Accesa submenú de buffers.
4	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de páginas de memoria libres en el sistema, monitoreado por la utileria SAR (freemem). Este desplegado muestra la cantidad de páginas de memoria disponibles para procesos en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (freemem).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : mem

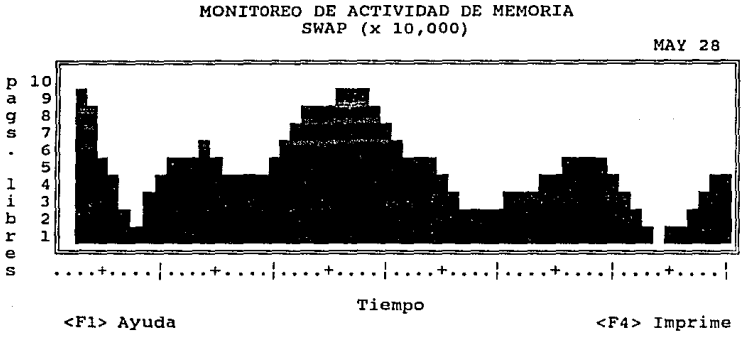
Salida : mem

Objetivo

Mostrar de forma gráfica el comportamiento de disponibilidad de bloques de disco para procesos de swap (intercambio), monitoreado por la utileria SAR (freeswap). Este desplegado muestra la cantidad de bloques de disco disponibles para ser usados por procesos de intercambio de memoria a disco y viceversa en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (freeswap).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

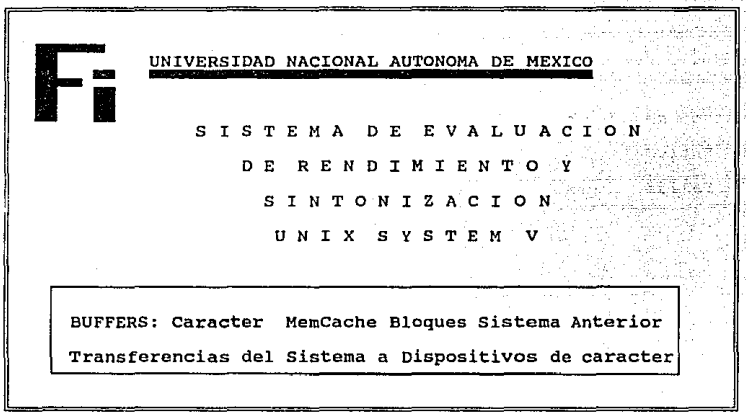
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : mem

Salida : mem

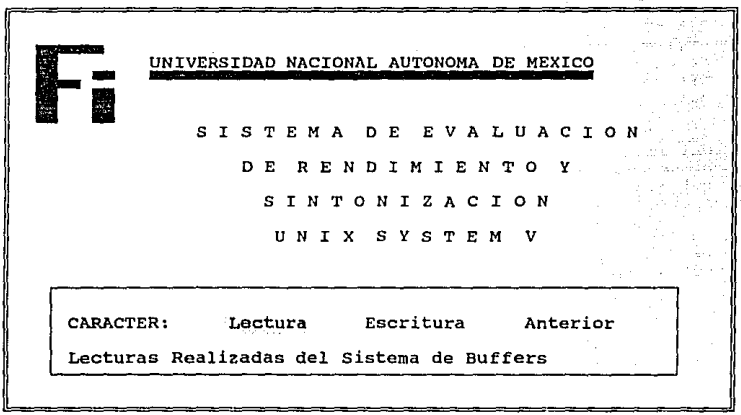
Formato de pantalla



Opciones del menú

Nivel	Opción	D e s c r i p c i ó n
1	Caracter	Accesa submenú Caracter.
2	MemCache	Accesa submenú MemCache.
3	Bloques	Accesa submenú Bloques.
4	Sistema	Accesa submenú Sistema.
5	Anterior	Con esta opción regresa al menú anterior.

Formato de pantalla



Opciones del menú

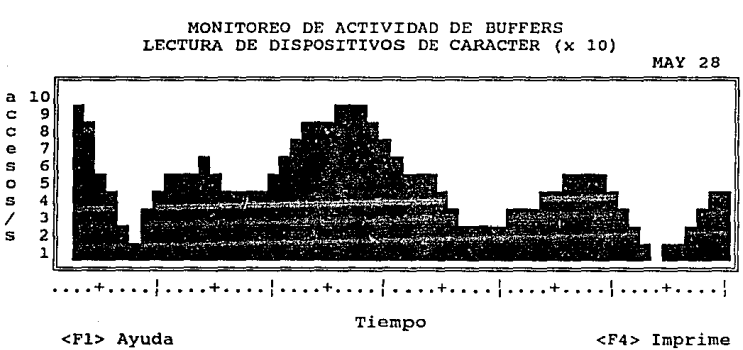
Nivel	Opción	D e s c r i p c i ó n
1	Lectura	Despliega lecturas realizadas de dispositivos de caracter.
2	Escritura	Despliega escrituras realizadas a dispositivos de caracter.
3	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de lecturas de dispositivos de caracteres (raw devices), monitoreado por la utileria SAR (pread/s). Este desplegado muestra la cantidad de caracteres leídos de dispositivos de tipo caracter en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (pread/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

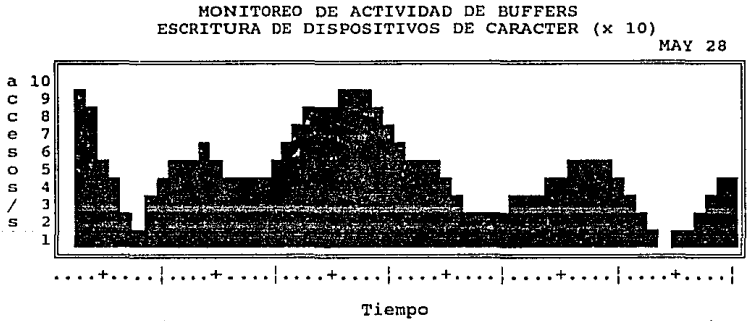
Salida : buf

Objetivo

Mostrar de forma gráfica el comportamiento de escrituras a dispositivos de caracteres (raw devices), monitoreado por la utileria SAR (pwrit/s). Este desplegado muestra la cantidad de caracteres escritos a dispositivos de tipo caracter en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (pwrit/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

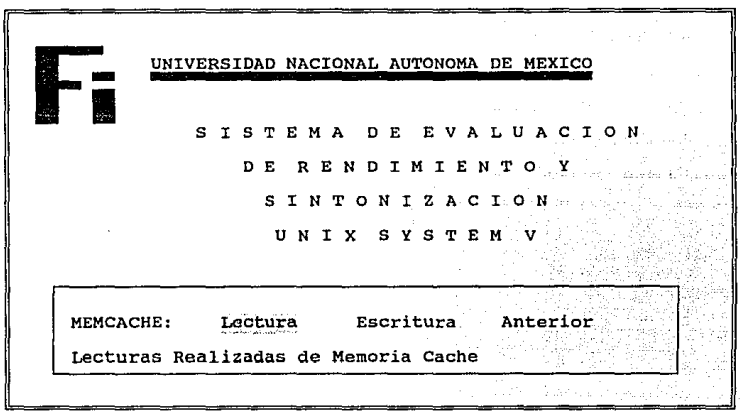
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

Salida : buf

Formato de pantalla



Opciones del menú

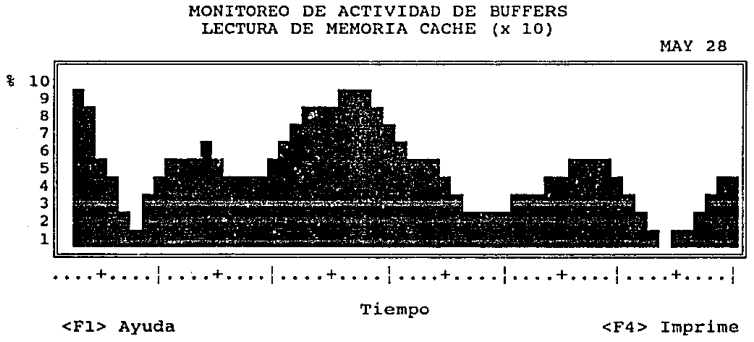
Nivel	Opción	D e s c r i p c i ó n
1	Lectura	Despliega el comportamiento de lecturas de memoria cache.
2	Escritura	Despliega el comportamiento de escrituras a memoria cache.
3	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de la tasa de aciertos (hit ratio) de lectura de memoria cache, monitoreado por la utileria SAR (%rcache). Este desplegado muestra la tasa de aciertos con que el S.O. encontró información en memoria cache. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%rcache).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

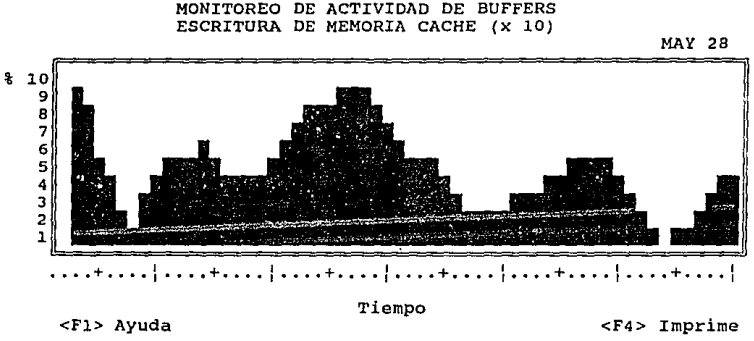
Salida : buf

Objetivo

Mostrar de forma gráfica el comportamiento de la tasa de escritura a memoria cache, monitoreado por la utileria SAR (%wcache). Este desplegado muestra la tasa de escrituras realizadas a memoria cache. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (%wcache).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

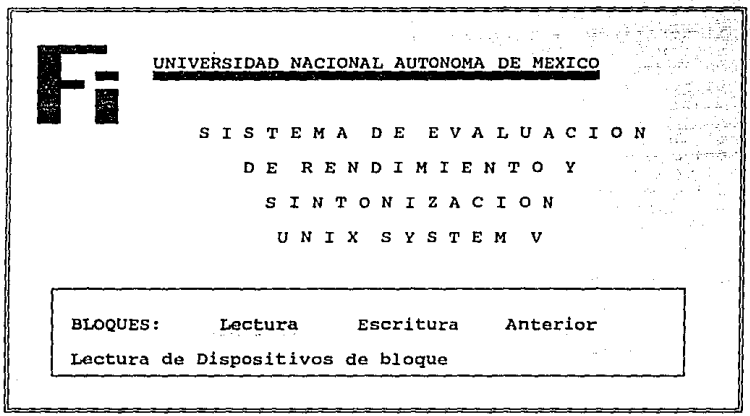
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

Salida : buf

Formato de pantalla



Opciones del menú

Nivel	Opción	D e s c r i p c i ó n
1	Lectura	Despliega el comportamiento de lecturas de dispositivos de bloque.
2	Escritura	Despliega el comportamiento de escrituras a dispositivos de bloque.
3	Anterior	Con esta opción regresa al menú anterior.

Objetivo

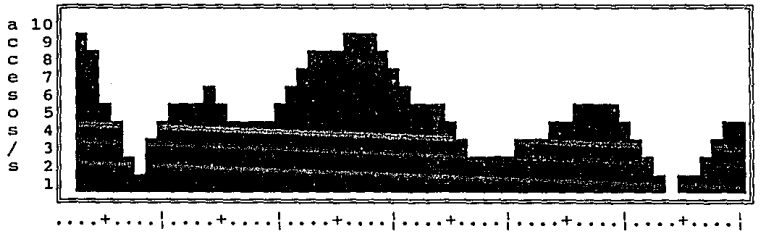
Mostrar de forma gráfica el comportamiento de lecturas de dispositivos de bloque (block devices), monitoreado por la utileria SAR (bread/s). Este desplegado muestra la cantidad de caracteres leidos de dispositivos de tipo bloque en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (bread/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla

MONITOREO DE ACTIVIDAD DE BUFFERS
LECTURA DE DISPOSITIVOS DE BLOQUE (x 10)

MAY 28



<F1> Ayuda <F4> Imprime

Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

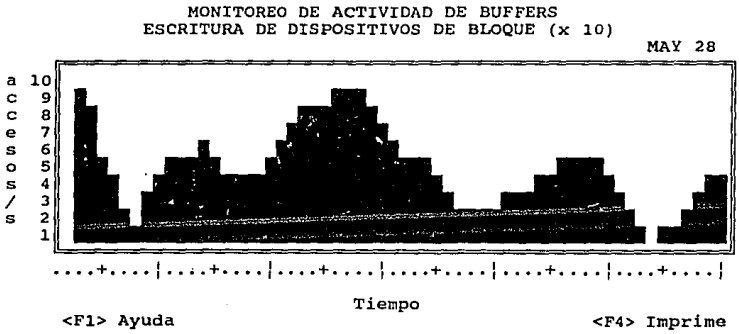
Salida : buf

Objetivo

Mostrar de forma gráfica el comportamiento de escrituras a dispositivos de bloque (block devices), monitoreado por la utileria SAR (bwrit/s). Este desplegado muestra la cantidad de caracteres escritos a dispositivos de tipo bloque en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (bwrit/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del día actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

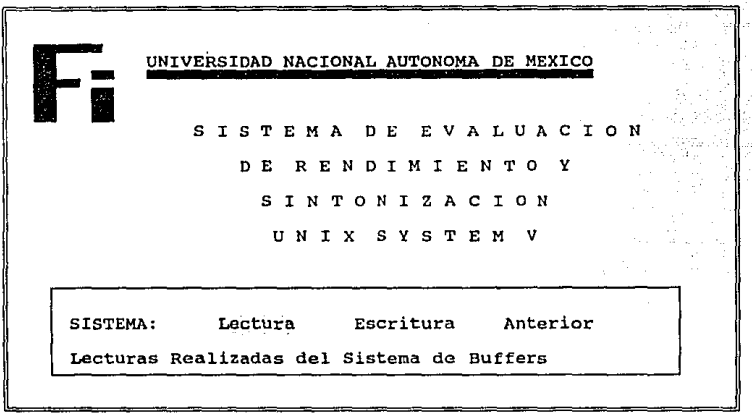
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

Salida : buf

Formato de pantalla



Opciones del menú

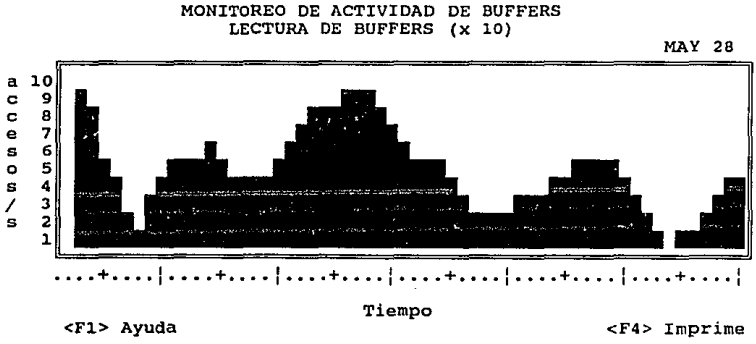
Nivel	Opción	D e s c r i p c i ó n
1	Lectura	Despliega el comportamiento de lecturas realizadas de buffers del sistema.
2	Escritura	Despliega el comportamiento de escrituras a buffers del sistema.
3	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Mostrar de forma gráfica el comportamiento de lecturas de buffers del sistema, monitoreado por la utileria SAR (lread/s). Este desplegado muestra la cantidad de buffers del sistema leídos en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (lread/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



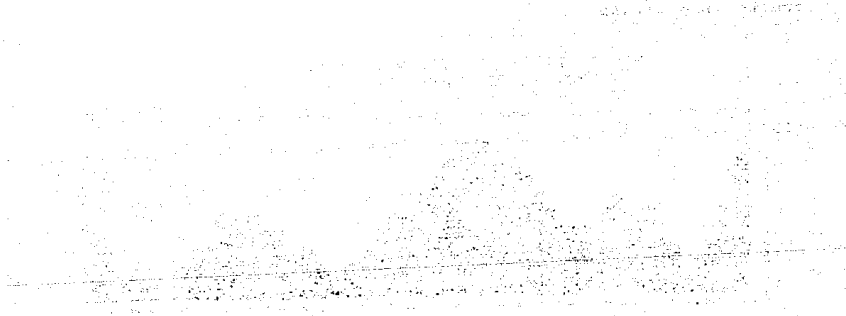
Opciones de uso

Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

Salida : buf

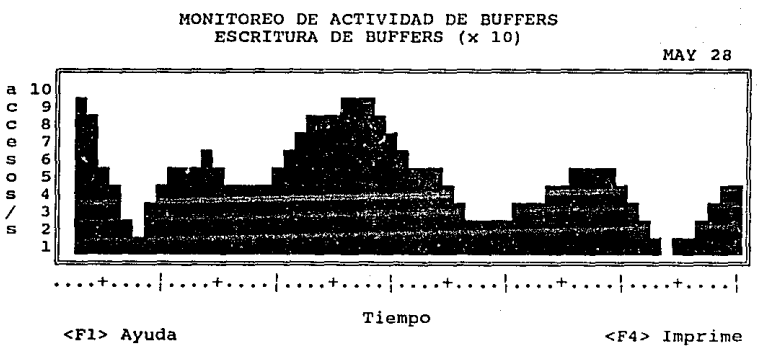


Objetivo

Mostrar de forma gráfica el comportamiento de escrituras de buffers del sistema, monitoreado por la utileria SAR (lwrit/s). Este desplegado muestra la cantidad de buffers del sistema escritos en el tiempo. En el eje X se muestra el tiempo y en el eje Y se muestra el valor correspondiente (lwrit/s).

El intervalo de tiempo se inicia con la puesta en operación del sistema y termina al finalizar las operaciones del mismo, si se está monitoreando una fecha atrasada; si el monitereo es del dia actual, el intervalo final de la gráfica representa el promedio del ciclo en el momento del monitoreo.

Formato de pantalla



Opciones de uso

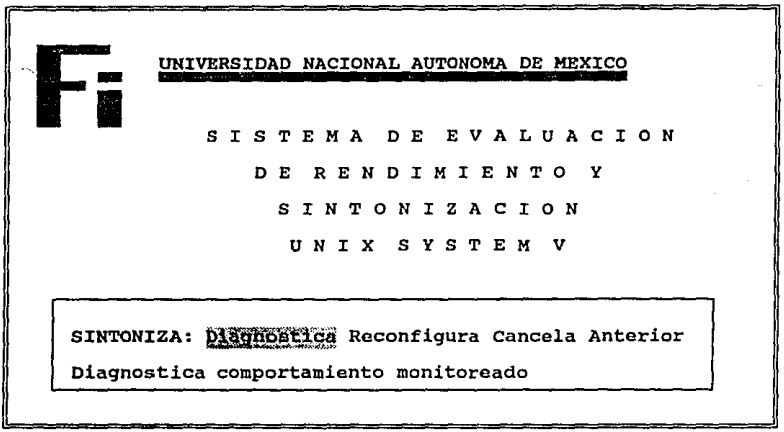
Observar el comportamiento mostrado en la gráfica.
F1 Despliega el objetivo de la gráfica
F4 Manda a imprimir la gráfica

Archivos de entrada/salida

Entrada : buf

Salida : buf

Formato de pantalla



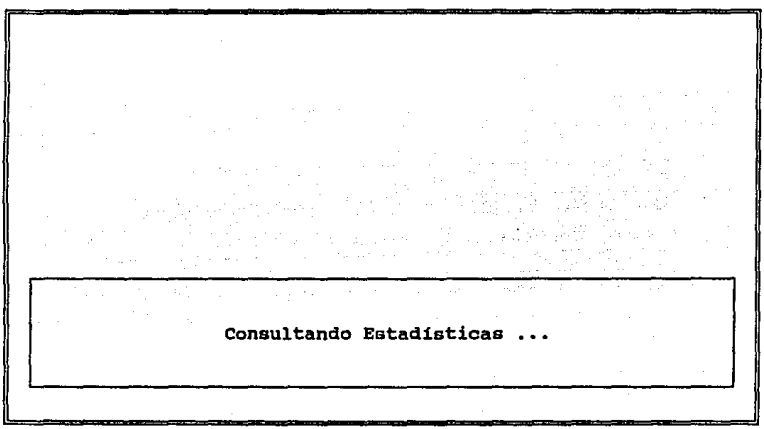
Opciones del menú

Nivel	Opción	D e s c r i p c i ó n
1	Diagnostica	Genera una recomendación a partir de las estadísticas generadas en el monitoreo y conforme a reglas de configuración.
2	Reconfigura	Realiza la reconstrucción del kernel con los parámetros recomendados.
3	Cancela	Realiza la reconstrucción del kernel con los parámetros originales.
4	Anterior	Con esta opción regresa al menú anterior.

Objetivo

Consultar las estadísticas generadas anteriormente, en la opción de monitoreo de los diferentes parámetros configurables del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de consultar los archivos de de recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

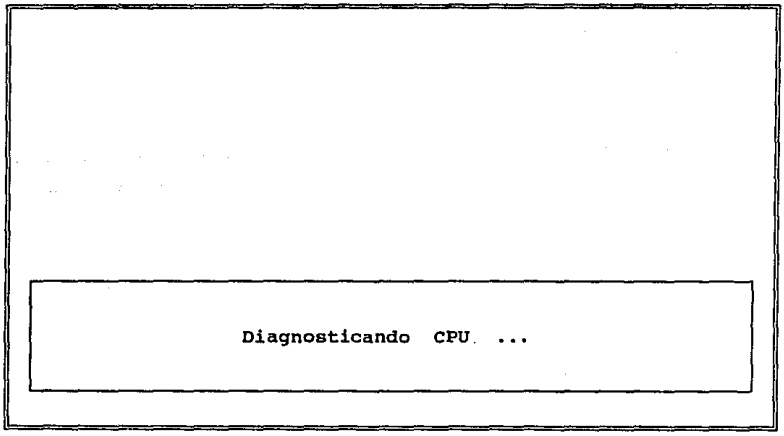
Entrada : cpu
tab
es
mem
buf

Salida : ninguno

Objetivo

Hacer el diagnóstico del CPU, en base a las estadísticas.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el CPU y que aparezca la siguiente pantalla.

Archivos de entrada/salida

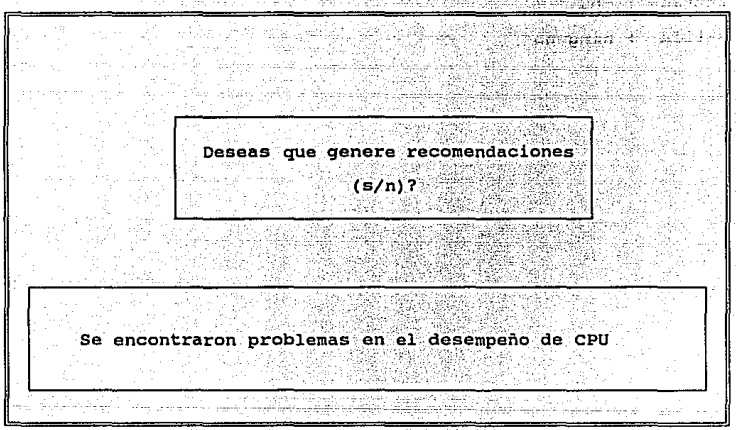
Entrada : cpu

Salida : ninguno

Objetivo

En caso de que encuentre problemas en el CPU, da la opción para generar o no las recomendaciones, para dichos problemas.

Formato de pantalla



Opciones de uso

Si se desea obtener las recomendaciones para el diagnóstico de CPU, se debe responde con una "s", y si no se debe teclear una "n".

Archivos de entrada/salida

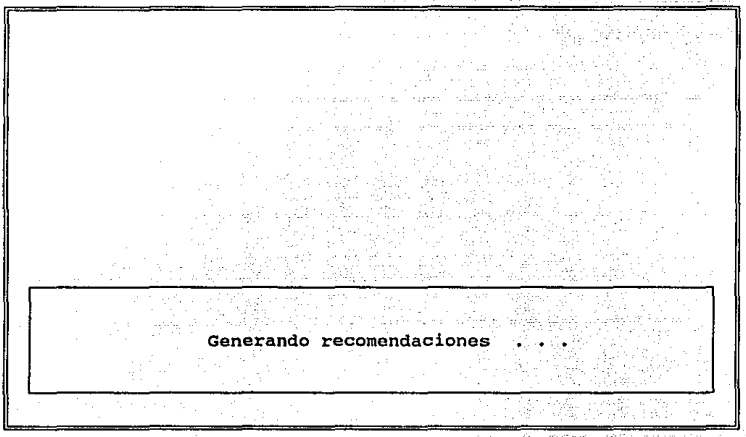
Entrada : ninguno

Salida : ninguno

Objetivo

Si se aceptan las recomendaciones del diagnóstico del CPU, las genera para presentarlas en la siguiente pantalla.

Formato de pantalla



Opciones de uso

Esperar a que termine de generar las recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

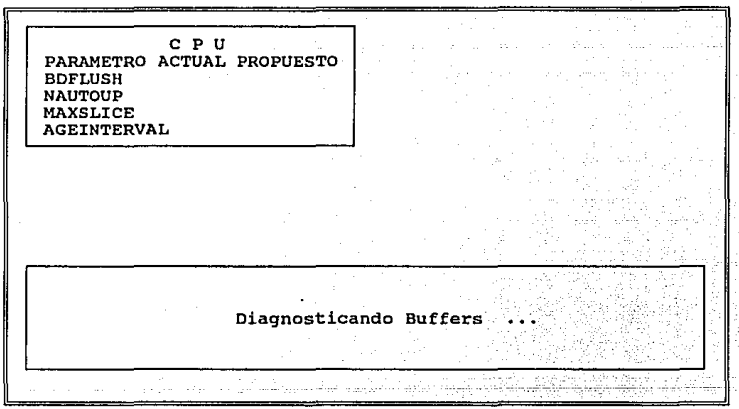
Entrada : ninguno

Salida : ninguno

Objetivo

Presenta las recomendaciones generadas a partir del diagnóstico del CPU, y realiza el diagnóstico del sistema de Buffers.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el sistema de bufferes y que aparezca la siguiente pantalla.

Archivos de entrada/salida

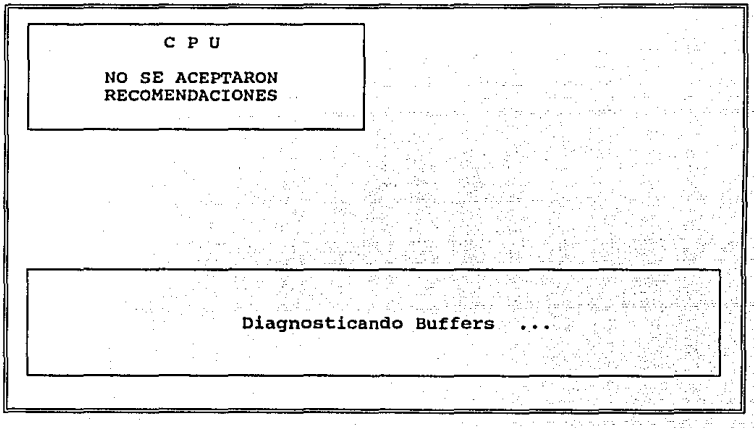
Entrada : buf

Salida : ninguno

Objetivo

Si no se aceptaron las recomendaciones del diagnóstico del CPU, realiza el diagnóstico del sistema de Buffers.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el sistema de bufferes y que aparezca la siguiente pantalla.

Archivos de entrada/salida

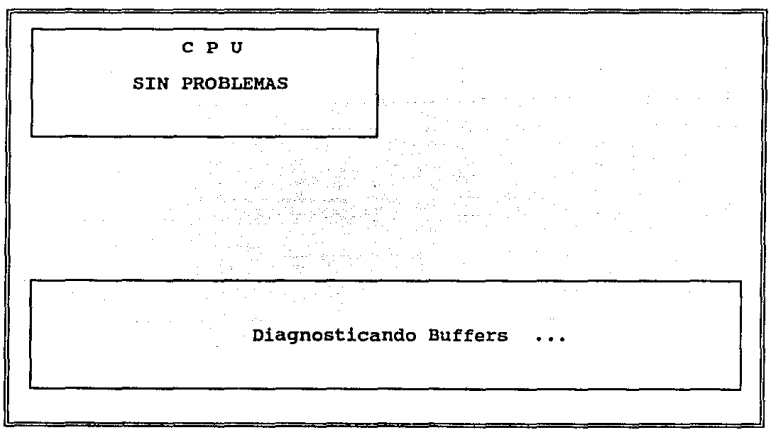
Entrada : buf

Salida : ninguno

Objetivo

Si no se encuentran problemas en el CPU, realiza el diagnóstico del sistema de Buffers.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el sistema de bufferes y que aparezca la siguiente pantalla.

Archivos de entrada/salida

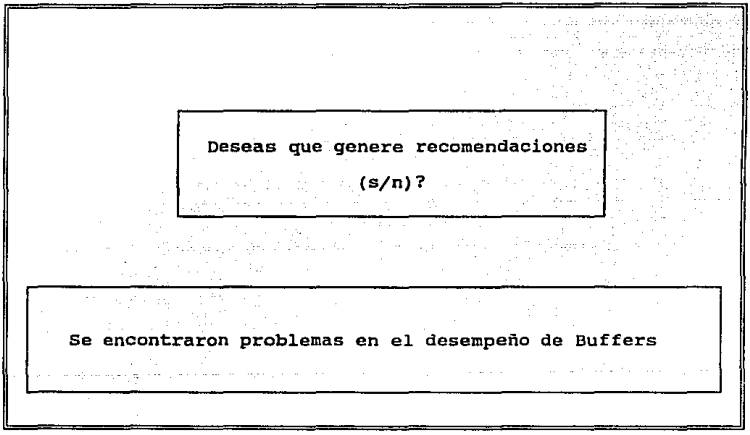
Entrada : buf

Salida : ninguno

Objetivo

En caso de que encuentre problemas en el sistema de Buffers, da la opción para generar o no las recomendaciones, para dichos problemas.

Formato de pantalla



Opciones de uso

Si se desea obtener las recomendaciones para el diagnóstico de Buffers, se debe responde con una "s", y si no se debe teclear una "n".

Archivos de entrada/salida

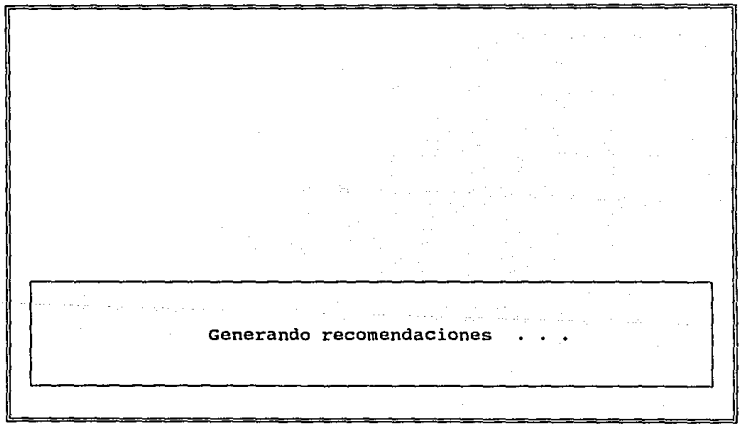
Entrada : ninguno

Salida : ninguno

Objetivo

Si se aceptan las recomendaciones del diagnóstico del sistema de Buffers, las genera para presentarlas en la siguiente pantalla.

Formato de pantalla



Opciones de uso

Esperar a que termine de generar las recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

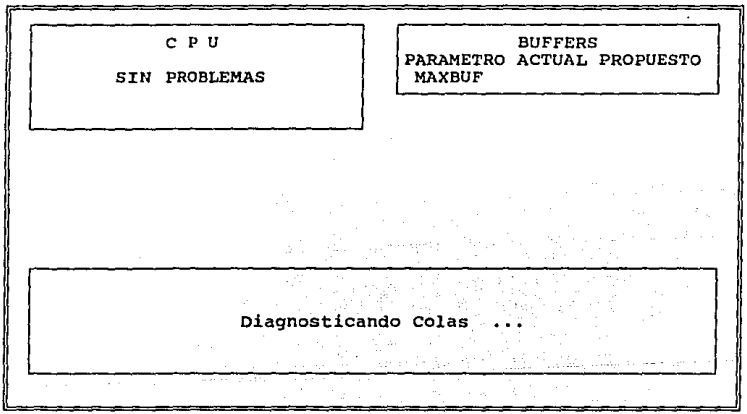
Entrada : ninguno

Salida : ninguno

Objetivo

Presenta las recomendaciones generadas a partir del diagnóstico del sistema de Buffers, y realiza el diagnóstico de las colas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las colas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

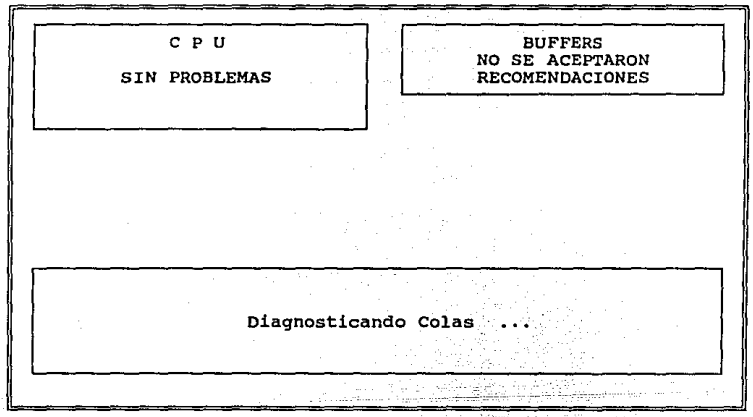
Entrada : buf

Salida : ninguno

Objetivo

Si no se aceptaron las recomendaciones del diagnóstico del sistema de Buffers, realiza el diagnóstico de las colas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las colas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

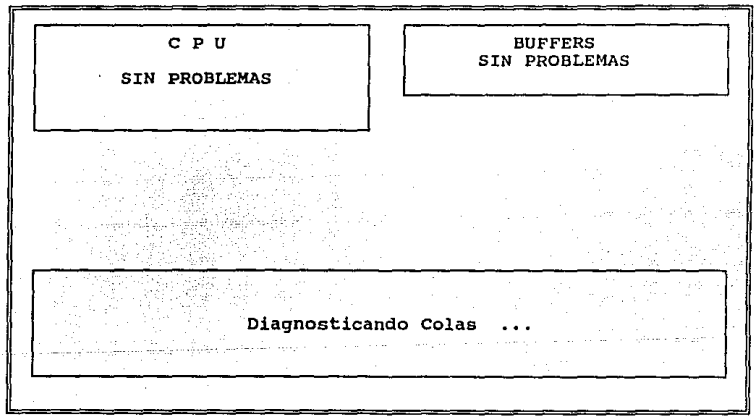
Entrada : buf

Salida : ninguno

Objetivo

Si no se encuentran problemas en el sistema de Buffers, realiza el diagnóstico de las colas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las colas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

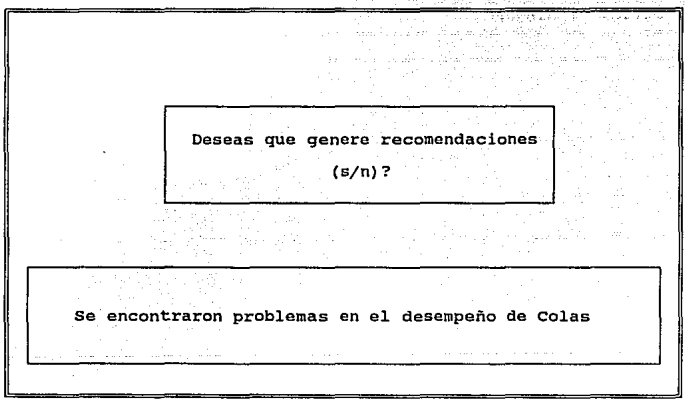
Entrada : buf

Salida : ninguno

Objetivo

En caso de que encuentre problemas en las colas del sistema, da la opción para generar o no las recomendaciones, para dichos problemas.

Formato de pantalla



Opciones de uso

Si se desea obtener las recomendaciones para el diagnóstico de las colas del sistema, se debe responde con una "s", y si no se debe teclear una "n".

Archivos de entrada/salida

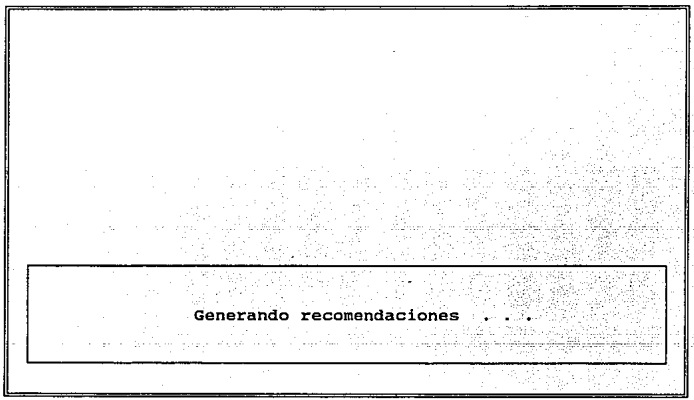
Entrada : buf

Salida : ninguno

Objetivo

Si se aceptan las recomendaciones del diagnóstico de las colas del sistema, las genera para presentarlas en la siguiente pantalla.

Formato de pantalla



Opciones de uso

Esperar a que termine de generar las recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

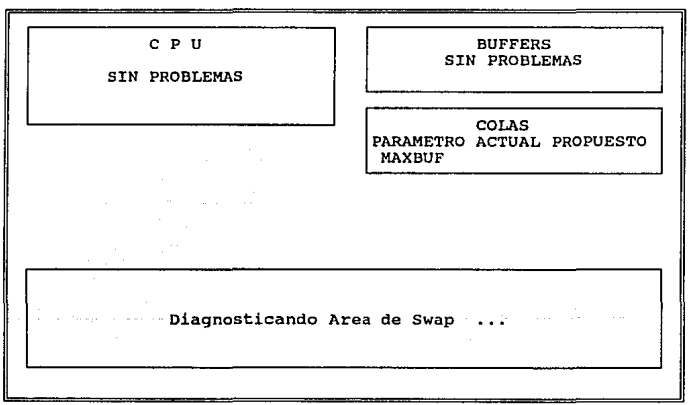
Entrada : cpu
tab
es
mem
buf

Salida : ninguno

Objetivo

Presenta las recomendaciones generadas a partir del diagnóstico de las colas del sistema, y realiza el diagnóstico del área de swap.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el area de swap del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

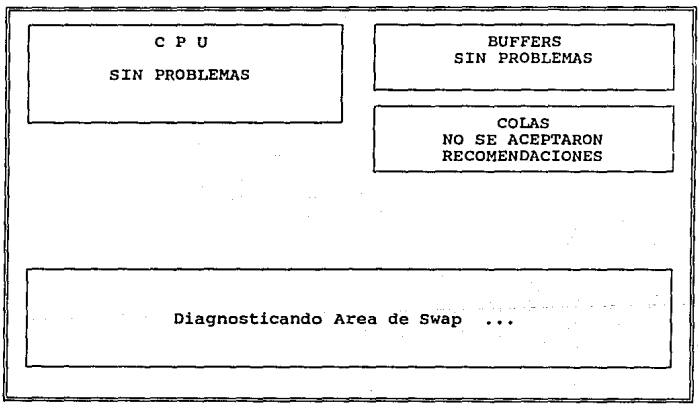
Entrada : buf

Salida : ninguno

Objetivo

Si no se aceptaron las recomendaciones del diagnóstico de las colas del sistema, realiza el diagnóstico del area de swap.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el area de swap del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

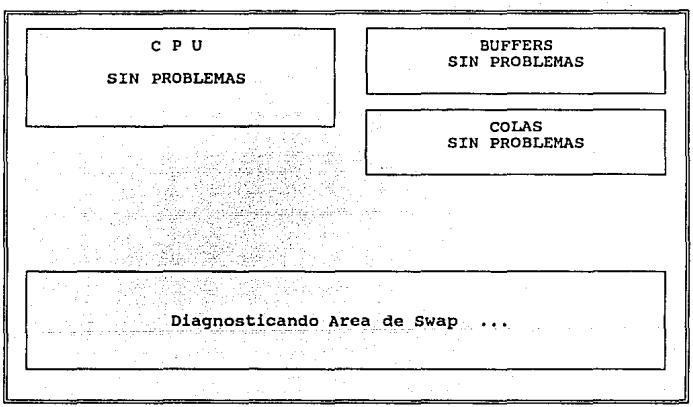
Entrada : buf

Salida : ninguno

Objetivo

Si no se encuentran problemas en las colas del sistema, realiza el diagnóstico del area de swap.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar el area de swap del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

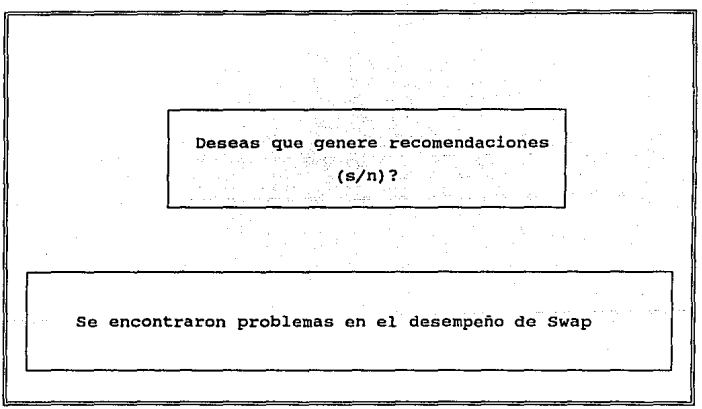
Entrada : buf

Salida : ninguno

Objetivo

En caso de que encuentre problemas en el área de swap, da la opción para generar o no las recomendaciones, para dichos problemas.

Formato de pantalla



Opciones de uso

Si se desea obtener las recomendaciones para el diagnóstico del area de swap, se debe responde con una "s", y si no se debe teclear una "n".

Archivos de entrada/salida

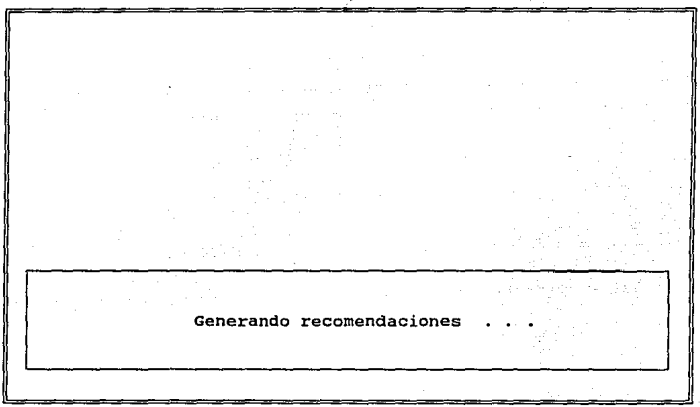
Entrada : buf

Salida : ninguno

Objetivo

Si se aceptan las recomendaciones del diagnóstico del area de swap, las genera para presentarlas en la siguiente pantalla.

Formato de pantalla



Opciones de uso

Esperar a que termine de generar las recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

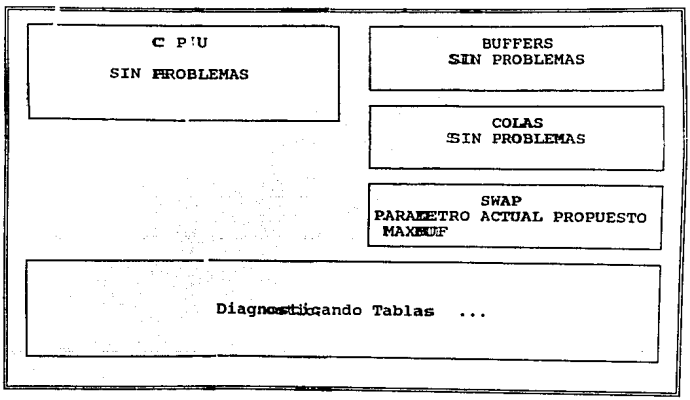
Entrada : buf

Salida : ninguno

Objetivo

Presenta las recomendaciones generadas a partir del diagnóstico del area de swap, y realiza el diagnóstico de las tablas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las tablas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

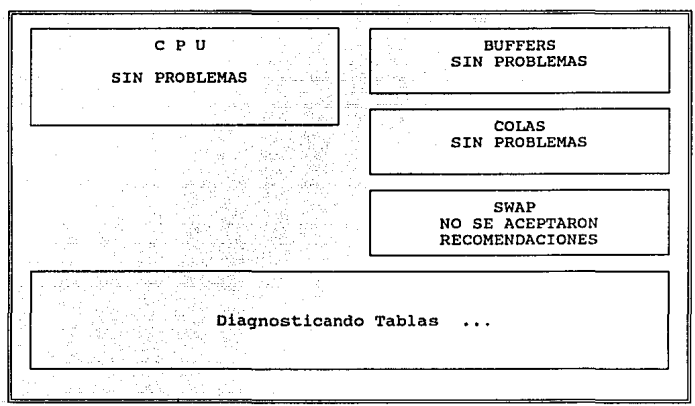
Entrada : buf

Salida : ninguno

Objetivo

Si no se aceptaron las recomendaciones del diagnóstico del area de swap, realiza el diagnóstico de las tablas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las tablas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

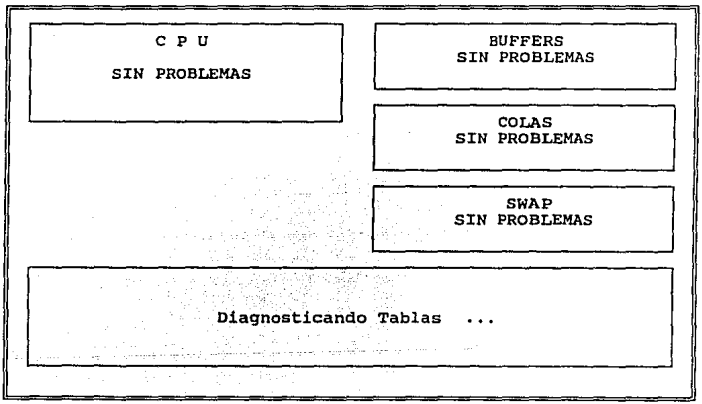
Entrada : buf

Salida : ninguno

Objetivo

Si no se encuentran problemas en el area de swap, realiza el diagnóstico de las tablas del sistema.

Formato de pantalla



Opciones de uso

Esperar a que termine de diagnosticar las tablas del sistema y que aparezca la siguiente pantalla.

Archivos de entrada/salida

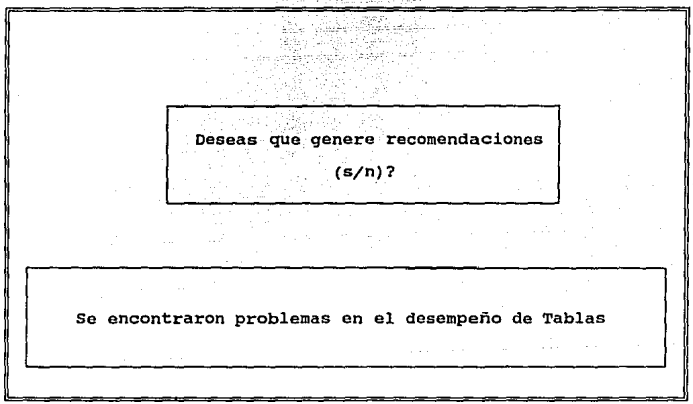
Entrada : buf

Salida : ninguno

Objetivo

En caso de que encuentre problemas en las tablas del sistema, da la opción para generar o no las recomendaciones, para dichos problemas.

Formato de pantalla



Opciones de uso

Si se desea obtener las recomendaciones para el diagnóstico de las tablas del sistema, se debe responde con una "s", y si no se debe teclear una "n".

Archivos de entrada/salida

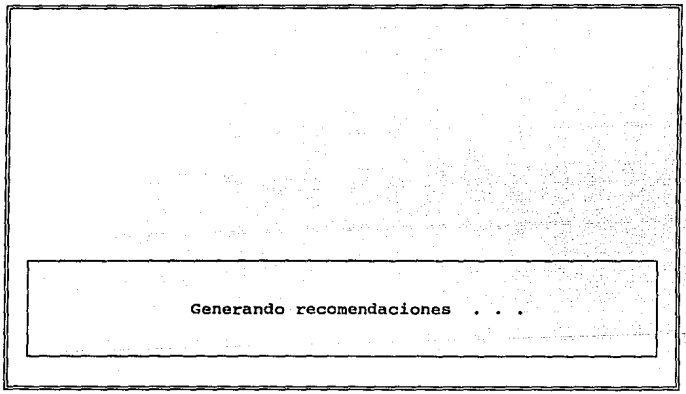
Entrada : buf

Salida : ninguno

Objetivo

Si se aceptan las recomendaciones del diagnóstico de las tablas del sistema, las genera para presentarlas en la siguiente pantalla.

Formato de pantalla



Opciones de uso

Esperar a que termine de generar las recomendaciones y que aparezca la siguiente pantalla.

Archivos de entrada/salida

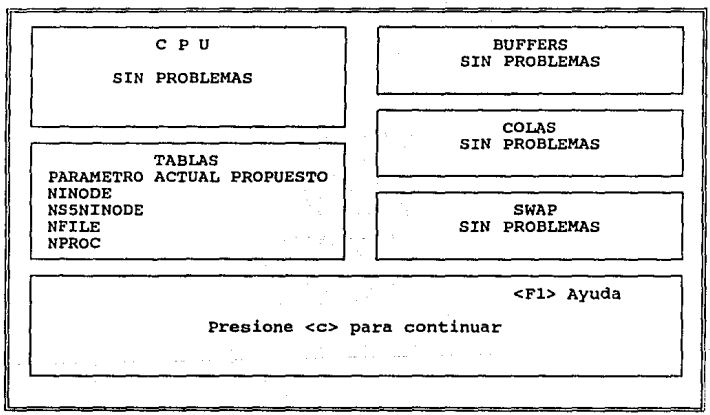
Entrada : buf

Salida : ninguno

Objetivo

Presenta las recomendaciones generadas a partir del diagnóstico de las tablas del sistema, termina con la opción "Diagnostica", y regresa al menú principal.

Formato de pantalla



Opciones de uso

Revisar las recomendaciones que se presentaron y oprimir "c" para regresar al menú de **sintoniza.**
F1 Despliega una breve explicación de lo que significa cada parámetro configurable, presentado en la pantalla.

Archivos de entrada/salida

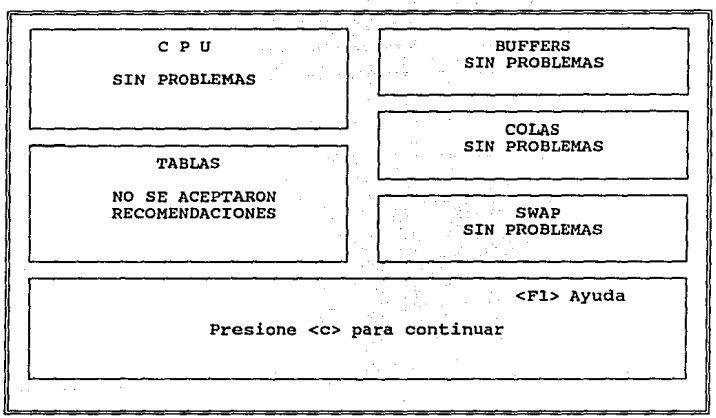
Entrada : ninguno

Salida : ninguno

Objetivo

Si no se aceptaron las recomendaciones del diagnóstico de las tablas del sistema, termina con la opción "Diagnostica", y regresa al menú principal.

Formato de pantalla



Opciones de uso

Revisar las recomendaciones que se presentaron y oprimir "c" para regresar al menú de **sintoniza**.
F1 Despliega una breve explicación de lo que significa cada parámetro configurable, presentado en la pantalla.

Archivos de entrada/salida

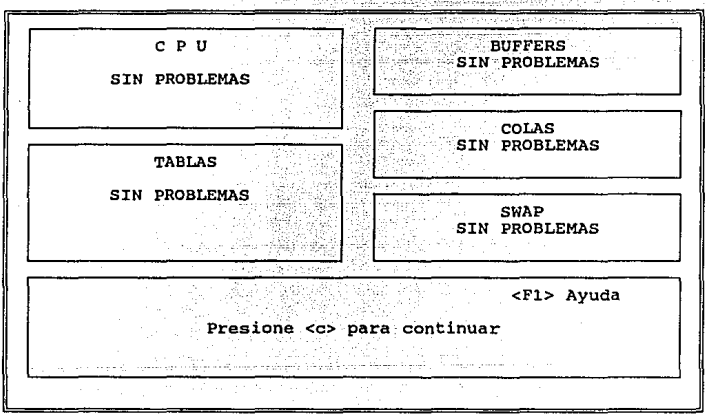
Entrada : ninguno

Salida : ninguno

Objetivo

Si no se encuentran problemas en el diagnóstico de las tablas del sistema, termina con la opción "Diagnostica", y regresa al menú principal.

Formato de pantalla



Opciones de uso

Revisar las recomendaciones que se presentaron y oprimir "c" para regresar al menú de **sintoniza.**
F1 Despliega una breve explicación de lo que significa cada parámetro configurable, presentado en la pantalla.

Archivos de entrada/salida

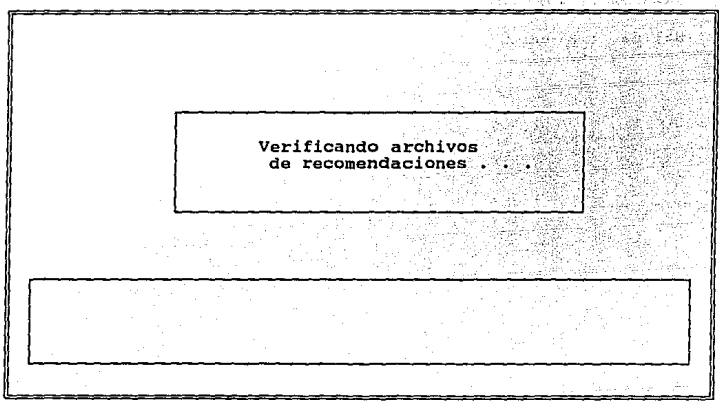
Entrada : ninguno

Salida : ninguno

Objetivo

Consultar las estadísticas generadas anteriormente, en la opción de sintoniza de los diferentes parámetros configurables del sistema, para proceder a la reconfiguración del kernel.

Formato de pantalla



Opciones de uso

Esperar a que termine de consultar las estadisticas anteriores y que aparezca la siguiente pantalla.

Archivos de entrada/salida

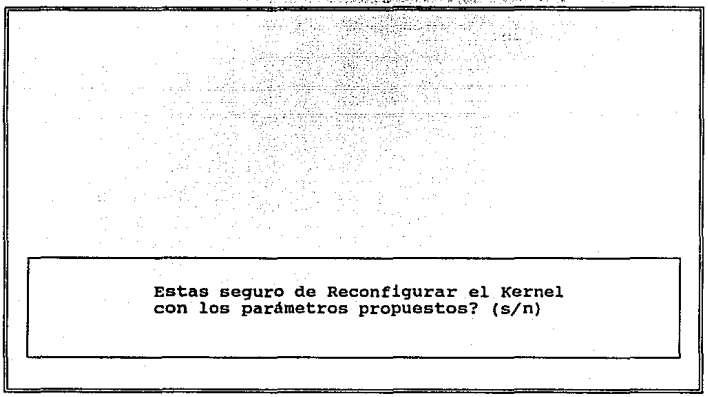
Entrada : buf y/o cpu y/o tab y/o es y/o mem

Salida : ninguno

Objetivo

Confirmar la decisión de reconfigurar el kernel, como medida de seguridad.

Formato de pantalla



Opciones de uso

Si se desea proseguir con el proceso de reconfiguración del kernel, se debe responde con una "s", y esperar a que aparezca la siguiente pantalla; y si no se desea continuar con la reconfiguración del kernel, se debe teclear una "n", y se regresará a la pantalla de **sintoniza**.

Archivos de entrada/salida

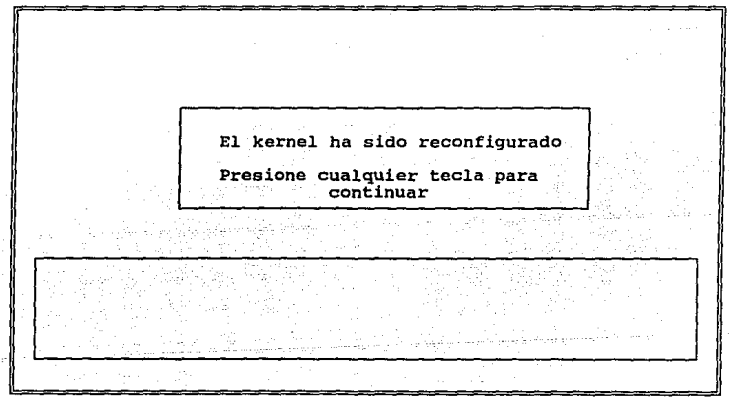
Entrada : ninguno

Salida : ninguno

Objetivo

Informar que la reconfiguración del kernel, con los valores de los parámetros recomendados anteriormente, fué satisfactoria y ha concluido.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de **sintoniza.**

Archivos de entrada/salida

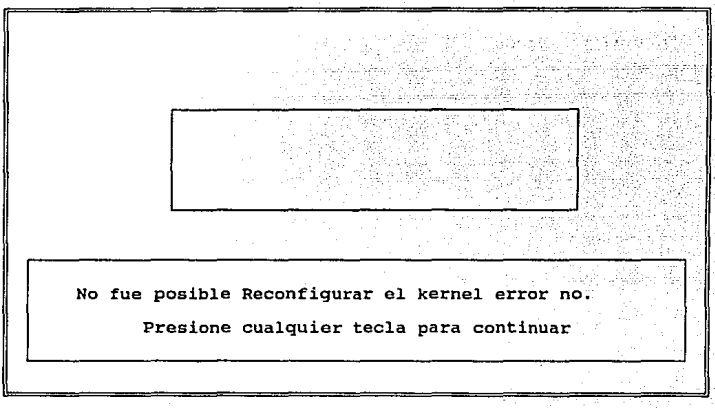
Entrada : buf y/o cpu y/o tab y/o es y/o mem

Salida : STUNE.OLD, MTUNE.OLD

Objetivo

Informar que la reconfiguración del kernel, no se realizó debido al número de error que se presenta en la pantalla.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de sintoniza.

Archivos de entrada/salida

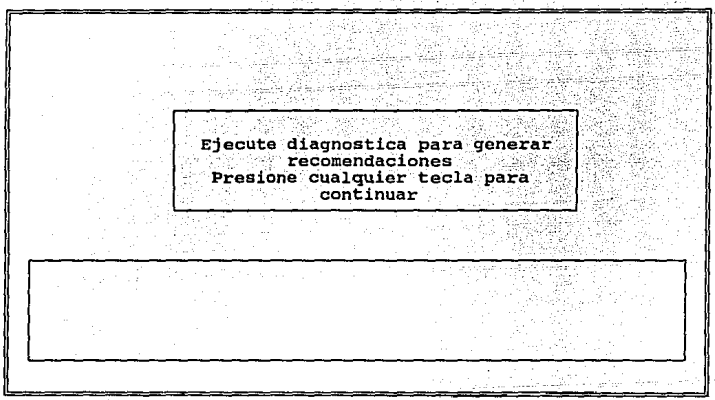
Entrada : ninguno

Salida : ninguno

Objetivo

Indicar el proceso para reconfigurar el kernel: Primero diagnosticar los parámetros configurables, para poder generar las recomendaciones, (sobre las cuales se reconfigura el kernel), y así despues poder reconfigurar.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de sintoniza.

Archivos de entrada/salida

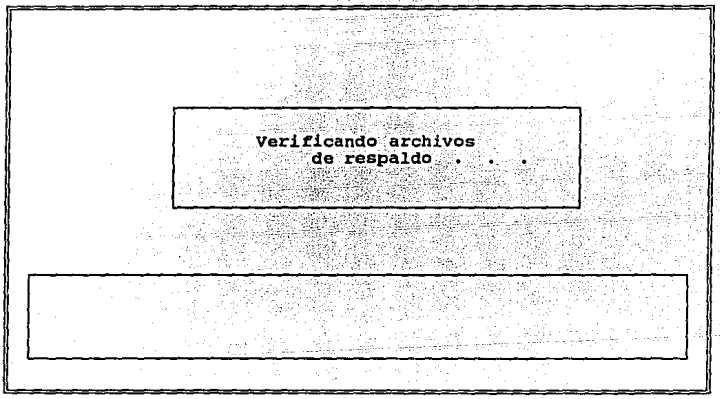
Entrada : ninguno

Salida : ninguno

Objetivo

Consultar las estadísticas de respaldo generadas anteriormente, en la opción de sintoniza de los diferentes parámetros configurables del sistema, para cancelar la reconfiguración anterior.

Formato de pantalla



Opciones de uso

Esperar a que termine de consultar las estadísticas de respaldo y que aparezca la siguiente pantalla.

Archivos de entrada/salida

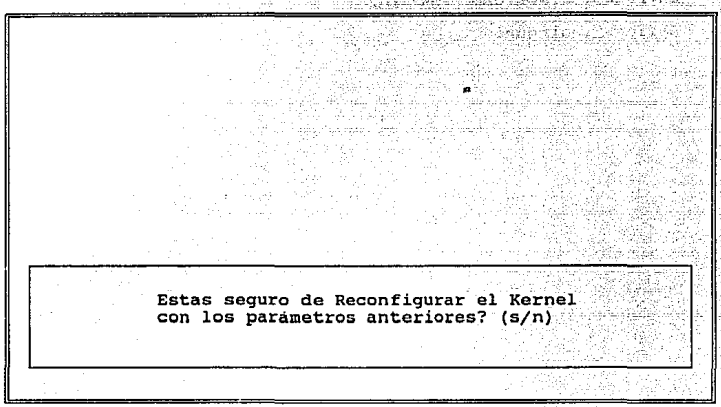
Entrada : STNUE.OLD MTUNE.OLD

Salida : ninguno

Objetivo

Confirmar la decisión de reconfigurar el kernel, como medida de seguridad.

Formato de pantalla



Opciones de uso

Si se desea proseguir con la cancelación (del proceso de reconfiguración anterior), se debe responde con una "**s**", y esperar a que aparezca la siguiente pantalla; y si no se desea continuar con la cancelación, se debe teclear una "**n**", y se regresará a la pantalla de **sintoniza**.

Archivos de entrada/salida

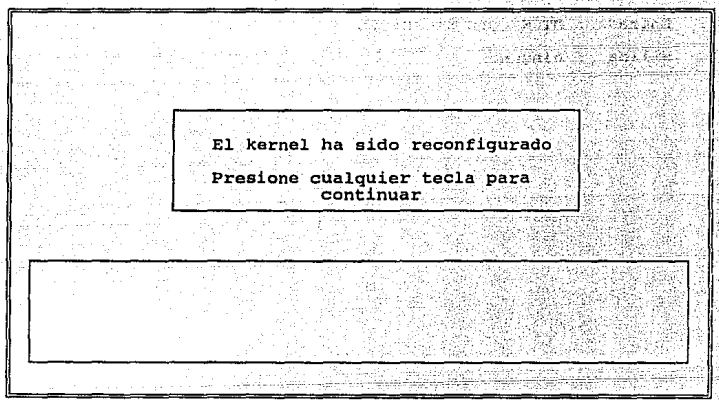
Entrada : MTUNE.OLD STUNE.OLD

Salida : ninguno

Objetivo

Informar que la reconfiguración del kernel, con los valores de respaldo (anteriores a los recomendados en el diagnóstico anterior), fué satisfactoria y ha concluido.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de sintoniza.

Archivos de entrada/salida

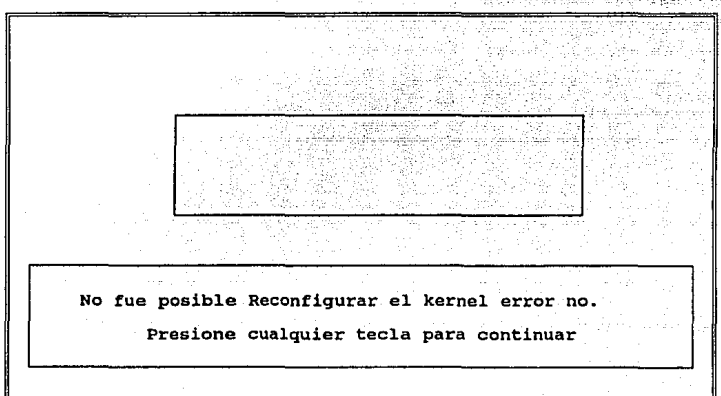
Entrada : ninguno

Salida : ninguno

Objetivo

Informar que la reconfiguración del kernel, con los valores de respaldo (esto es, la cancelación) no se realizó debido al número de error que se presenta en la pantalla.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de sintoniza.

Archivos de entrada/salida

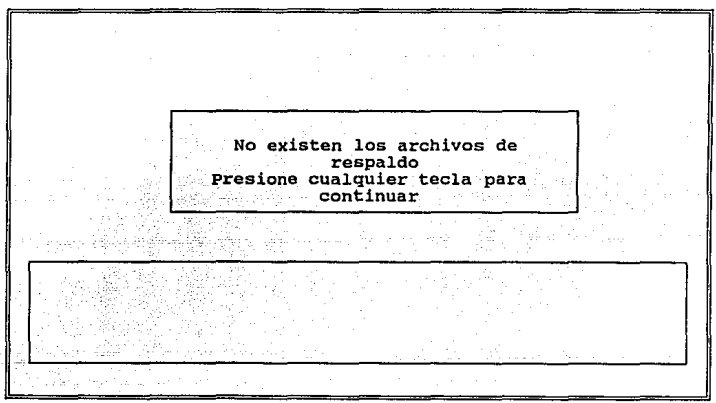
Entrada : ninguno

Salida : ninguno

Objetivo

Indicar el proceso para reconfigurar el kernel: Primero diagnosticar los parámetros configurables, para poder generar las recomendaciones, (sobre las cuales se reconfigura el kernel), y así despues poder reconfigurar.

Formato de pantalla



Opciones de uso

Oprimir cualquier tecla para regresar al menú de **sintoniza.**

Archivos de entrada/salida

Entrada : ninguno

Salida : ninguno

APENDICE B

MANUAL TECNICO

M A N U A L T E C N I C O

CONTENIDO

I. OBJETIVO DEL MANUAL.

El presente manual tiene como función dar un soporte al administrador del sistema, para la adecuada operación del sistema SERSUNIX; tomando en cuenta las consideraciones técnicas del sistema.

I.1 DESCRIPCION

En este manual se encuentra toda la información técnica detallada que se requiere para tener una mejor comprensión del sistema SERSUNIX, en términos de cómo está construido el sistema. Además se contemplan todos los aspectos necesarios para poder ejecutar el sistema. Finalmente se cuenta con una lista de los programas y rutinas (incluyendo nombre y objetivo) que conforman y hacen posible el funcionamiento del sistema SERSUNIX.

II. AMBIENTE DE OPERACION.

Para la instalación y operación del sistema SERSUNIX se debe contar con el ambiente computacional mencionado a continuación:

- Disponer de una computadora que tenga el sistema operativo UNIX instalado.
- Se requiere del sistema operativo UNIX System V/versión 3.2 de SCO (Santa Cruz Operation).

- Se necesita el manejador de base de datos INFORMIX 4GL versión 1.1 instalado.

III. INFRAESTRUCTURA DE DESARROLLO.

La base del desarrollo del sistema SERSUNIX está soportado en dos herramientas que UNIX posee: SHELL e INFORMIX. Estas dos utilerías conforman la infraestructura que hace posible la construcción del sistema.

RUTINAS-SHELL. El SHELL es empleado para generar las rutinas que ejecutan algunos de los comandos propios de UNIX, las funciones básicas de las rutinas en SHELL son las siguientes:

- Generar una interfase entre el sistema operativo UNIX y el sistema SERSUNIX.
- Obtener datos estadísticos de la operación del sistema.
- Obtener configuración del equipo.
- Obtener el valor actual de los parámetros configurables
- Modificar los valores de los parámetros configurables
- Respaldar la configuración actual del sistema
- Recompilar el Kernel del sistema.

PROGRAMAS-INFORMIX. El manejador de base de datos INFORMIX es empleado para la construcción de todos

los programas del sistema, estos programas realizan llamados con paso de parámetros a las rutinas Shell. Algunas de las funciones que tienen a su cargo estos programas son:

- Permitir la definición y el fácil manejo de las tablas de la base de datos.
- Realizar la mayor parte de los procesos del sistema
- Realizar la mayor parte de las validaciones del sistema
- Generar la interfase visual del sistema.
- Generar las gráficas del sistema
- Permitir el manejo de menúes circulares
- Ofrecer pantallas de ayuda
- Generar programas en código C, de fácil comunicación con UNIX.

IV. Listados de las rutinas en shell

```
# Rutina que borra el interior de marco

echo
"\033[2;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[3;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[4;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[5;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[6;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[7;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[8;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[9;2H\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
```

```
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\0
40\040\040\040\040\c"
echo
"\033[10;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[11;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[12;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[13;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[14;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[15;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[16;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
```

```
echo
"\033[17;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[18;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[19;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[20;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[21;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[22;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[23;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040"
```

```
# modifica el valor de MAXBUF de acuerdo a sar -b
#

if grep MAXBUF /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 100'  -lt  $2 ]
   then
      val='expr $1 + 100'
      echo "/etc/conf/bin/idtune -f MAXBUF $val" >> ejecuta
      echo MAXBUF:$1:$val: > ajupar.buf
   else
      echo "/etc/conf/bin/idtune -f MAXBUF $2" >> ejecuta
      echo MAXBUF:$1:$2: > ajupar.buf
   fi
fi
/usr/informix/bin/dbload -d tesis -c dbl_aju.buf -l err_aju
-e 10 > /dev/null
unset val
rm ./tmp ./err_aju ./ajupar.buf
```

```
# modifica el valor de MAXBUF de acuerdo a sar -q
# esta funcion disminuye el numero de procesos

if grep MAXBUF /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,3'
   if [ 'expr $1 - 100'  -gt  $2 ]
   then
      val='expr $1 - 100'
      echo "/etc/conf/bin/idtune -f MAXBUF $val" >> ejecuta
      echo MAXBUF:$1:$val: >> ajupar.col
   else
      echo "/etc/conf/bin/idtune -f MAXBUF $2" >> ejecuta
      echo MAXBUF:$1:$2: >> ajupar.col
   fi
fi
/usr/informix/bin//dbload -d tesis -c dbl_aju.col -l err_aju
-e 10 > /dev/null
unset val
rm -r ./tmp ./err_aju ./ajupar.col 2> /dev/null
```

```
# modifica el valor de BDFLUSHR de acuerdo a sar -u

rm ./ejecuta 2>&-
if grep BDFLUSHR /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 30'  -lt  $2 ]
   then
      val='expr $1 + 30'
      echo "/etc/conf/bin/idtune -f BDFLUSHR $val" >>
ejecuta
      echo "BDFLUSH:$1:$val:" >> ajupar.cpu
   else
      echo "/etc/conf/bin/idtune -f BDFLUSHR $2" >> ejecuta
      echo "BDFLUSH:$1:$2:" >> ajupar.cpu
   fi
fi


# modifica el valor de NAUTOUP de acuerdo a sar -u

if grep NAUTOUP /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 10'  -lt  $2 ]
   then
      val='expr $1 + 10'
      echo "/etc/conf/bin/idtune -f NAUTOUP $val" >> ejecuta
      echo "NAUTOUP:$1:$val:" >> ajupar.cpu
   else
      echo "/etc/conf/bin/idtune -f NAUTOUP $2" >> ejecuta
      echo "NAUTOUP:$1:$2:" >> ajupar.cpu
   fi
fi


# modifica el valor de MAXSLICE de acuerdo a sar -u

if grep MAXSLICE /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,3'
   if [ 'expr $1 - 5'  -gt  $2 ]
   then
      val='expr $1 - 5'
      echo "/etc/conf/bin/idtune -f MAXSLICE $val" >>
ejecuta
      echo "MAXSLICE:$1:$val:" >> ajupar.cpu
   else
      echo "/etc/conf/bin/idtune -f MAXSLICE $2" >> ejecuta
      echo "MAXSLICE:$1:$2:" >> ajupar.cpu
   fi
fi
```

```
# modifica el valor de AGEINTERVAL de acuerdo a sar -u

if grep AGEINTERVAL /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 5'  -lt  $2 ]
   then
      val='expr $1 + 5'
      echo "/etc/conf/bin/idtune -f AGEINTERVAL $val" >>
ejecuta
      echo "AGEINTERVAL:$1:$val:" >> ajupar.cpu
   else
      echo "/etc/conf/bin/idtune -f AGEINTERVAL $2" >>
ejecuta
      echo "AGEINTERVAL:$1:$2:" >> ajupar.cpu
   fi
fi
/usr/informix/bin/dbload -d tesis -c dbl_aju.cpu -l err_aju
-e 10 > /dev/null
unset val
rm tmp err_aju ./ajupar.* 2>&-
```

```
# modifica el valor de MAXBUF de acuerdo a sar -w
# esta funcion disminuye el numero de procesos

if grep MAXBUF /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,3'
   if [ 'expr $1 - 100'  -gt  $2 ]
   then
      val='expr $1 - 100'
      echo "/etc/conf/bin/idtune -f MAXBUF $val" >> ejecuta
      echo MAXBUF:$1:$val: >> ajupar.swp
   else
      echo "/etc/conf/bin/idtune -f MAXBUF $2" >> ejecuta
      echo MAXBUF:$1:$2: >> ajupar.swp
   fi
fi
/usr/informix/bin/dbload -d tesis -c dbl_aju.swp -l err_aju
-e 10 > /dev/null
unset val
rm -r ./tmp ./err_aju i./ajupar.swp
```

```
# modifica el valor de NINODE, NFILE y NS5INODE de acuerdo a
sar -v

if grep NINODE /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 50'  -lt  $2 ]
   then
      iant=$1
      iact='expr $1 + 50'
      if grep NFILE /etc/conf/cf.d/mtune > tmp
      then
         set 'cat tmp | cut -f 2'
         fant=$1
         fact='expr $1 + 50'
         if [ $fact -gt $iact ]
         then
         fact=$iact
         fi
         echo "/etc/conf/bin/idtune -f NINODE $iact" >>
ejecuta
         echo "/etc/conf/bin/idtune -f NS5INODE $iact" >>
ejecuta
         echo "/etc/conf/bin/idtune -f NFILE $fact" >>
ejecuta
         echo "NINODE:$iant:$iact:" >> ajupar.tab
         echo "NS5INODE:$iant:$iact:" >> ajupar.tab
         echo "NFILE:$fant:$fact:" >> ajupar.tab
      fi
   else
      echo "/etc/conf/bin/idtune -f NINODE $2" >> ejecuta
      echo "/etc/conf/bin/idtune -f NFILE 'expr $2 - 50'" >>
ejecuta
      echo "/etc/conf/bin/idtune -f NS5INODE $2" >> ejecuta
      echo "NINODE:$1:$2:" >> ajupar.tab
      echo "NS5INODE:$1:$2:" >> ajupar.tab
      grep NFILE /etc/conf/cf.d/mtune > tmp
      set 'cat tmp | cut -f 2'
      echo "NFILE:$1:$2:" >> ajupar.tab
   fi
fi
/usr/informix/bin/dbload -d tesis -c dbl_aju.tab -l err_aju
-e 10 > /dev/null
unset iant iact fant fact
rm -r tmp ./err_aju ./ajupar.tab 2> /dev/null
```

```
# modifica el valor de NPROC de acuerdo a sar -v

if grep NPROC /etc/conf/cf.d/mtune > tmp
then
   set 'cat tmp | cut -f 2,4'
   if [ 'expr $1 + 50'  -lt  $2 ]
   then
      val='expr $1 + 50'
      echo "/etc/conf/bin/idtune -f NPROC $val" >> ejecuta
      echo "NPROC:$1:$val:" > ajupar.tab
   else
      echo "/etc/conf/bin/idtune -f NPROC $2" >> ejecuta
      echo "NPROC:$1:$2:" > ajupar.tab
   fi
fi
/usr/informix/bin/dbload -d tesis -c dbl_aju.tab -l err_aju
-e 10  > /dev/null
unset val
rm  ./tmp ./err_aju
```

```
# shell para mover los archivos mtune.old y stune.old
#su ruta original y reconstruir el kernel

#mv mtune.old /etc/conf/cf.d
#mv stune.old /etc/conf/cf.d
#/etc/conf/bin/idbuild
exit 0
```

```
if hwconfig 1>conf.hw 2>/dev/null
then
 esc='\033['
 h='H'
 q=';'
 echo "\033[3;15H \033[7m C O N F I G U R A C I O N  D E
H A R D W A R E \033[m"

# obtiene la memoria
 echo '\033[6;3HMemoria Total:\033[1m' 'memsize'
'\033[mbytes'

# modulo para la Unidad de Punto Flotante
 if grep fpu conf.hw > conf.dat
 then
  echo '\033[8;3HUnidad de Punto Flotante:\033[1m
Presente\033[m'
  echo '\033[9;5HVector de Interrupcion:\033[1m' 'cut -d" "
-f2 conf.dat | cut -d"=" -f2'"\033[m"
  echo '\033[10;3HDir. de Acceso a Memoria:\033[1m' 'cut -d"
" -f3 conf.dat | cut -d"=" -f2'"\033[m"
  echo '\033[11;23HTipo:\033[1m' 'cut -d" " -f4 conf.dat |
cut -d"=" -f2'"\033[m"
 else
  echo '\033[8;3HUnidad de Punto Flotante:\033[1m No
Presente\033[m'
 fi

# modulo para la consola
 if grep console conf.hw > conf.dat
 then
  echo '\033[6;49HConsola de Operacion:\033[1m
Presente\033[m'
  echo '\033[7;47HVector de Interrupcion:\033[1m' 'cut -d" "
-f2 conf.dat | cut -d"=" -f2'"\033[m"
  echo '\033[8;45HDir. de Acceso a Memoria:\033[1m' 'cut -d"
" -f3 conf.dat | cut -d"=" -f2'"\033[m"
  echo '\033[9;63HUnidad:\033[1m' 'cut -d" " -f4 conf.dat |
cut -d"=" -f2'"\033[m"
  echo '\033[10;65HTipo:\033[1m' 'cut -d" " -f5 conf.dat |
cut -d"=" -f2'"\033[m"
  echo '\033[11;53HMemoria de Video:\033[1m' 'cut -d" " -f7
conf.dat | cut -d"=" -f2'"\033[m"
 else
  echo '\033[6;46HConsola de Operacion:\033[1m No
Presente\033[m'
 fi

# modulo para puertos seriales
 if grep serial conf.hw > conf.dat
 then
  set 'wc -l conf.dat'
  c=2
  r=14
```

```
  i=0
  while :
  do
   grep "unit=$i" conf.dat > tmp
   r='expr $r + 1'
   echo "$esc$r$q$c$h      Puerto Serial Numero:\033[1m
$i\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h      Dir. Base de Memoria:\033[1m" 'cut
-d" " -f2 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h                          Offset:\033[1m" 'cut
-d" " -f3 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h    Vector de Interrupcion:\033[1m" 'cut
-d" " -f4 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h Dir. de Acceso a Memoria:\033[1m" 'cut
-d" " -f5 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h                             Tipo:\033[1m" 'cut
-d" " -f7 tmp | cut -d"=" -f2'"\033[m"
   c=44
   r=14
   i='expr $i + 1'
   if [ $i = $1 ]
   then
    break
   fi
  done
 else
  echo "\033[14;13H Puerto Serial:\033[1m No Presente\033[m"
 fi
echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
read a
./medbor.sh

# modulo para puertos paralelos
 if grep parallel conf.hw > conf.dat
 then
  set 'wc -l conf.dat'
  c=2
  r=14
  i=0
  while :
  do
   grep "unit=$i" conf.dat > tmp
   r='expr $r + 1'
   echo "$esc$r$q$c$h      Puerto Paralelo Numero:\033[1m
$i\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h       Dir. Base de Memoria:\033[1m"
'cut -d" " -f2 tmp | cut -d"=" -f2'"\033[m"
```

```
   r='expr $r + 1'
   echo "$esc$r$q$c$h                          Offset:\033[1m"
'cut -d" " -f3 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h     Vector de Interrupcion:\033[1m"
'cut -d" " -f4 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h   Dir. de Acceso a Memoria:\033[1m"
'cut -d" " -f5 tmp | cut -d"=" -f2'"\033[m"
   i='expr $i + 1'
   if [ $i = $1 ]
   then
    break
   fi
  done
 else
  echo "\033[14;10H  Puerto Paralelo:\033[1m No
Presente\033[m"
 fi
echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
read a
./medbor.sh

# modulo para flopies
 if grep floppy conf.hw > conf.dat
 then
  c=2
  r=14
  set 'wc -l conf.dat'
  i=0
  while :
  do
   grep "unit=$i" conf.dat > tmp
   r='expr $r + 1'
   echo "$esc$r$q$c$h Unidad de Disco Flexible #:\033[1m
$i\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h  Direccion Base de Memoria:\033[1m"
'cut -d" " -f2 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h                          Offset:\033[1m"
'cut -d" " -f3 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h     Vector de Interrupcion:\033[1m"
'cut -d" " -f4 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h   Dir. de Acceso a Memoria:\033[1m"
'cut -d" " -f5 tmp | cut -d"=" -f2'"\033[m"
   r='expr $r + 1'
   echo "$esc$r$q$c$h                          Tipo:\033[1m"
'cut -d" " -f7 tmp | cut -d"=" -f2'"\033[m"
   i='expr $i + 1'
   if grep "unit=$i" conf.dat > tmp
```

```
    then
     if [ $i = "1" ] || [ $i = "3" ]
     then
     c=44
     r=15
      r='expr $r + 1'
      echo "$esc$r$q$c$h Unidad de Disco Flexible #:\033[1m
$i\033[m"
      r='expr $r + 1'
      echo "$esc$r$q$c$h      Vector de Interrupcion:\033[1m"
'cut -d" " -f2 tmp | cut -d"=" -f2'"\033[m"
      r='expr $r + 1'
      echo "$esc$r$q$c$h    Dir. de Acceso a Memoria:\033[1m"
'cut -d" " -f3 tmp | cut -d"=" -f2'"\033[m"
      r='expr $r + 1'
      echo "$esc$r$q$c$h                                  Tipo:\033[1m"
'cut -d" " -f5 tmp | cut -d"=" -f2'"\033[m"
     fi
    fi
    i='expr $i + 1'
    if [ $i -ge $1 ]
    then
     break
    fi
  done
 else
  echo "\033[15;3HUnidad de Disco Flexible:\033[1m No
Presente\033[m"
 fi
echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
read a
./medbor.sh


# modulo para discos duros
 if grep disk conf.hw > conf.dat
 then
  set 'wc -l conf.dat'
  i=0
  while :
  do
   c=2
   r=13
   grep "id=$i" conf.dat > tmp
   tipo='cut -d" " -f4 tmp | cut -d"=" -f2'
   case $tipo in
    S)
     grep Sdsk conf.hw > tmp1
     r='expr $r + 1'
     echo "$esc$r$q$c$h      Unidad de Disco Duro #:\033[1m
$i\033[m"
      r='expr $r + 1'
```

```
     echo "$esc$r$q$c$h      Vector de Interrupcion:\033[1m"
'cut -d" " -f2 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h    Dir. de Acceso a Memoria:\033[1m"
'cut -d" " -f3 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h                          Tipo:\033[1m
SCSI\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h                  Host Adapter:\033[1m"
'cut -d" " -f5 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h               Identificador:\033[1m"
'cut -d" " -f6 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h     Numero Logico de Unidad:\033[1m"
'cut -d" " -f7 tmp | cut -d"=" -f2'"\033[m"
     r=15
     c=44
     grep -n Sdsk tmp1 > tmp
     grep 'expr $i + 1': tmp > tmp1
     r='expr $r + 1'
     echo "$esc$r$q$c$h    Cilindros:\033[1m" 'cut -d" " -f4
tmp1 | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h      Cabezas:\033[1m" 'cut -d" " -f5
tmp1 | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h    Sectores:\033[1m" 'cut -d" " -f6
tmp1 | cut -d"=" -f2'"\033[m"
     ;;
    w0)
     r='expr $r + 1'
     echo "$esc$r$q$c$h      Unidad de Disco Duro #:\033[1m
$i\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h  Direccion Base:\033[1m" 'cut -d" "
-f3 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h  Offset:\033[1m" 'cut -d" " -f4 tmp
| cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h  Vector de Interrupcion:\033[1m"
'cut -d" " -f5 tmp | cut -d"=" -f2'"\033[m"
     echo '\033[12;9HVector de Interrupcion:' 'cut -d" " -f5
tmp | cut -d"=" -f2'
     r='expr $r + 1'
     echo "$esc$r$q$c$h  Dir. de Acceso a Memoria:\033[1m"
'cut -d" " -f6 tmp | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h                          Tipo:\033[1m
ESDI\033[m"
     c=44
     r=15
```

```
     r='expr $r + 1'
     echo "$esc$r$q$c$h   Cilindros:\033[1m" 'cut -d" " -f9
tmp1 | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h     Cabezas:\033[1m" 'cut -d" " -f10
tmp1 | cut -d"=" -f2'"\033[m"
     r='expr $r + 1'
     echo "$esc$r$q$c$h    Sectores:\033[1m" 'cut -d" " -f11
tmp1 | cut -d"=" -f2'"\033[m"
     ;;
   esac
   echo "\033[22;23H \033[7m Presione <RETURN> para
continuar \033[m"
   read a
   ./medbor.sh
   i='expr $i + 1'
   if [ $i -ge $i ]
   then
    break;break
   fi
  done
 else
  echo'\033[12;9HUnidad de Disco Duro: No Presente'
 fi
 rm ./tmp ./tmp1 ./conf.dat ./conf.hw
else
 echo "\033[12;7H\033[7m Solo el Super Usuario puede obtener
la Configuración del Hardware \033[0m"
 echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
 read a
 rm ./conf.dat
fi
```

```
# shell para revisar si existe el archivo "ejecuta"
# que actualiza los parametros del kernel

if test -s  ejecuta
then
   exit 0
else
   exit 1
fi
```

```
# shell para revisar si existe los archivos "cw mtune.old y
stune.old"
# con el objeto de reconfigurar el kernel a sus parametros
anteriores

if  test -s  mtune.old | test -s stune.old
then
   exit 0
 else
  exit 1
fi
```

```
# Rutina que pinta el marco exterior
# se definen caracteres
esd2=273
esi2=311
eid2=274
eii2=310
hor2=315
ver2=272

# SE PINTA CUADRO EXTERIOR

# se pinta linea horizontal superior

echo
"\0$esi2\\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0
$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor
2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$
hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2
\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$h
or2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\
0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$ho
r2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0
$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor
2\0$hor2\0$hor2\0$esd2\c"

# se pinta lineas verticales

echo "\033[2;1H\0$ver2\033[2;80H\0$ver2\c"
echo "\033[3;1H\0$ver2\033[3;80H\0$ver2\c"
echo "\033[4;1H\0$ver2\033[4;80H\0$ver2\c"
echo "\033[5;1H\0$ver2\033[5;80H\0$ver2\c"
echo "\033[6;1H\0$ver2\033[6;80H\0$ver2\c"
echo "\033[7;1H\0$ver2\033[7;80H\0$ver2\c"
echo "\033[8;1H\0$ver2\033[8;80H\0$ver2\c"
echo "\033[9;1H\0$ver2\033[9;80H\0$ver2\c"
echo "\033[10;1H\0$ver2\033[10;80H\0$ver2\c"
echo "\033[11;1H\0$ver2\033[11;80H\0$ver2\c"
echo "\033[12;1H\0$ver2\033[12;80H\0$ver2\c"
echo "\033[13;1H\0$ver2\033[13;80H\0$ver2\c"
echo "\033[14;1H\0$ver2\033[14;80H\0$ver2\c"
echo "\033[15;1H\0$ver2\033[15;80H\0$ver2\c"
echo "\033[16;1H\0$ver2\033[16;80H\0$ver2\c"
echo "\033[17;1H\0$ver2\033[17;80H\0$ver2\c"
echo "\033[18;1H\0$ver2\033[18;80H\0$ver2\c"
echo "\033[19;1H\0$ver2\033[19;80H\0$ver2\c"
echo "\033[20;1H\0$ver2\033[20;80H\0$ver2\c"
echo "\033[21;1H\0$ver2\033[21;80H\0$ver2\c"
echo "\033[22;1H\0$ver2\033[22;80H\0$ver2\c"
echo "\033[23;1H\0$ver2\033[23;80H\0$ver2\c"

# se pinta linea horizontal inferior

echo
"\0$eii2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$
```

```
hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2
\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$h
or2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\
0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$ho
r2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0
$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor
2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$
hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2\0$hor2
\0$hor2\0$hor2\0$eid2\c"
```

```
# Rutina que borra el interior de marco

echo
"\033[13;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[14;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[15;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[16;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[17;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[18;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[19;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[20;2H\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
```

```
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[21;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[22;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\c"
echo
"\033[23;2H\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040\040\040\040\040\040\040\040\040\040\040\
040\040\040\040\040"
```

```
# moncpu.sh
dia=$1
if sar -u -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
' { print $2":"$3":"$4":"$5":" } '| grep -n : > sarcpu.dat
then
 set 'wc -l sarcpu.dat'
 n='expr $1 - 2'
 i=1
 if [ $n -gt 0 ]
 then
   /usr/informix/bin/dbload -d tesis -c dbl_cpu -l err_cpu -
e 10 > /dev/null
   # montab.sh
   sar -v -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
' { print $2":"$3":"$4":"$5":"$6":"$7":"$8":" } ' | grep -n
/ > sartab.dat
   ed - sartab.dat < cmd
   /usr/informix/bin/dbload -d tesis -c dbl_tab -l err_tab -
e 10 > /dev/null
   rm ./err_cpu ./err_tab
   c=0
  else
   echo "\033[12;28H\033[7m Aun no hay Informacion \033[0m"
   echo "\033[22;23H \033[7m Presione <RETURN> para
continuar \033[m"
   read a
   c=1
 fi
 rm ./sarcpu.dat ./sartab.dat
 exit $c
else
 ./borra.sh
 echo "\033[12;7H\033[7m Solo el Super Usuario puede obtener
las Estadisticas del Sistema \033[0m"
 echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
 read a
 exit 1
fi
```

```
# mones.sh
if sar -y -f /usr/adm/sa/sa$1 2> /dev/null | tail +6 | awk
{ print $2":"$3":"$4":" } '| grep -n : > sares.dat
then
 set 'wc -l sares.dat'
 n=`expr $1 - 2`
 i=1
 if [ $n -gt 0 ]
 then
   /usr/informix/bin/dbload -d tesis -c dbl_es -l err_es -e
10 > /dev/null
   rm ./err_es
   c=0
  else
   echo "\033[12;28H\033[7m Aun no hay Informacion \033[0m"
   echo "\033[22;23H \033[7m Presione <RETURN> para
continuar \033[m"
   read a
   c=1
 fi
 rm ./sares.dat
 exit $c
else
 ./borra.sh
 echo "\033[12;7H\033[7m Solo el Super Usuario puede obtener
las Estadisticas del Sistema \033[0m"
 echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
 read a
 rm ./sares.dat
 exit 1
fi
```

```
# monmem.sh
dia=$1
if sar -r -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
' { print $2":"$3":" } '| grep -n : >  sarmem.dat
then
 set 'wc -l sarmem.dat'
 n='expr $1 - 2'
 i=1
 if [ $n -gt 0 ]
 then
    /usr/informix/bin/dbload -d tesis -c dbl_mem -l err_mem -
e 10 > /dev/null
    rm ./err_mem
    # monbuf.sh
    sar -b -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
'{ print $2":"$3":"$4":"$5":"$6":"$7":"$8":"$9":" } '| grep
-n : >  sarbuf.dat
    /usr/informix/bin/dbload -d tesis -c dbl_buf -l err_buf -
e 10 > /dev/null
    rm ./err_buf ./sarbuf.dat
    c=0
  else
    echo "\033[12;28H\033[7m Aun no hay Informacion \033[0m"
    echo "\033[22;23H \033[7m Presione <RETURN> para
continuar \033[m"
    read a
    c=1
 fi
 rm ./sarmem.dat
 exit $c
else
 ./borra.sh
 echo "\033[12;7H\033[7m Solo el Super Usuario puede obtener
las Estadisticas del Sistema \033[0m"
 echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
 read a
 rm ./sarmem.dat
 exit 1
fi
```

```
# monswp.sh
dia=$1
if sar -w -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
' { print $2":"$3":"$4":"$5":"$6":") '| grep -n : >
sarswp.dat
then
 set 'wc -l sarswp.dat'
 n='expr $1 - 2'
 i=1
 if [ $n -gt 0 ]
 then
   /usr/informix/bin/dbload -d tesis -c dbl_swp -l err_swp -
e 10 > /dev/null
   # moncol.sh
   sar -q -f /usr/adm/sa/sa$dia 2> /dev/null | tail +6 | awk
' { print $2":"$3":" } ' |  grep -n : > sarcol.dat
   /usr/informix/bin/dbload -d tesis -c dbl_col -l err_col -
e 10 > /dev/null
   rm ./err_swp ./err_col
   c=0
  else
   echo "\033[12;28H\033[7m Aun no hay Informacion \033[0m"
   echo "\033[22;23H \033[7m Presione <RETURN> para
continuar \033[m"
   read a
   c=1
 fi
 rm ./sarswp.dat ./sarcol.dat
 exit $c
else
 ./borra.sh
 echo "\033[12;7H\033[7m Solo el Super Usuario puede obtener
las Estadisticas del Sistema \033[0m"
 echo "\033[22;23H \033[7m Presione <RETURN> para continuar
\033[m"
 read a
 exit 1
fi
```

```
# shell para modificar el archivo stune y los parametros de
kernel
# asi como reconstruirlo

#cp /etc/conf/cf.d/mtune mtune.old
#cp /etc/conf/cf.d/stune stune.old
#sh ejecuta
#/etc/conf/bin/idbuild  comentado hasta nuevo aviso
#rm ejecuta
exit 0
```

```
if ./mones.sh $1
then
 sh ./monswp.sh $1
 sh ./monmem.sh $1
 sh ./moncpu.sh $1
 exit 0
fi
exit 1
```

V. Listados de los programas

```
MAIN
 CREATE DATABASE tesis

 CREATE TABLE cpu (num_con smallint,
                   usr smallint NOT NULL,
                   sys smallint NOT NULL,
                   wio smallint NOT NULL,
                   idle smallint NOT NULL)

 CREATE TABLE mem (num_con smallint,
                   mem integer NOT NULL,
                   swp integer NOT NULL)

 CREATE TABLE es (num_con smallint,
                  raw integer NOT NULL,
                  can integer NOT NULL,
                  out integer NOT NULL)

 CREATE TABLE buf (num_con smallint,
                   bre smallint NOT NULL,
                   lre smallint NOT NULL,
                   rca smallint NOT NULL,
                   bwr smallint NOT NULL,
                   lwr smallint NOT NULL,
                   wca smallint NOT NULL,
                   pre smallint NOT NULL,
                   pwr smallint NOT NULL)

 CREATE TABLE tab (num_con smallint,
                   pro smallint NOT NULL,
                   pov smallint NOT NULL,
                   ino smallint NOT NULL,
                   iov smallint NOT NULL,
                   fil smallint NOT NULL,
                   fov smallint NOT NULL)

 CREATE TABLE swp (num_con smallint,
                   swpi decimal(5,2) NOT NULL,
                   bswi decimal(5,2) NOT NULL,
                   swpo decimal(5,2) NOT NULL,
                   bswo decimal(5,2) NOT NULL,
                   pswc smallint NOT NULL)

 CREATE TABLE col (num_con smallint,
                   runq decimal(5,2) NOT NULL,
                   runo smallint NOT NULL)

 CREATE TABLE aju_cpu (des CHAR(9),
                       act smallint NOT NULL,
                       rec smallint NOT NULL)

 CREATE TABLE aju_buf (des CHAR(12),
                       act smallint NOT NULL,
                       rec smallint NOT NULL)
```

```
CREATE TABLE aju_tab (des CHAR(12),
                      act smallint NOT NULL,
                      rec smallint NOT NULL)

CREATE TABLE aju_col (des CHAR(12),
                      act smallint NOT NULL,
                      rec smallint NOT NULL)

CREATE TABLE aju_swp (des CHAR(12),
                      act smallint NOT NULL,
                      rec smallint NOT NULL)
GRANT CONNECT TO root
END MAIN
```

```
DATABASE tesis
GLOBALS
     define mat ARRAY [70] of integer,
                v ARRAY [10,70] OF CHAR(1),
                a_mes ARRAY [12] of char(3),
                answ CHAR(1),
            titulo,subtitulo,leyendax char(80),
            cmd,leyenday char(20),
                fecha CHAR(8),
            tipo,flag,mes,dia,codigo,lectura integer,
                grafico char(1),
                paj_cpu RECORD LIKE aju_cpu.*,
                paj_buf RECORD LIKE aju_buf.*,
                paj_col RECORD LIKE aju_col.*,
                paj_swp RECORD LIKE aju_swp.*,
                paj_tab RECORD LIKE aju_tab.*,
                p_cpu RECORD LIKE cpu.*,
                p_es RECORD LIKE es.*,
                p_mem RECORD LIKE mem.*,
                p_buf RECORD LIKE buf.*,
                p_tab RECORD LIKE tab.*,
                p_swp RECORD LIKE swp.*,
                p_col RECORD LIKE col.*
END GLOBALS
```

```
GLOBALS "globales.4gl"
MAIN
DEFINE dumy INTEGER
DEFER INTERRUPT
OPTIONS
 HELP FILE "ayuda.ex",
 HELP KEY F1,
 PROMPT LINE 2
CLEAR SCREEN
CALL get_mes()
CALL marco(1,1,79,24)
CALL logo ()
LET flag = TRUE
CALL lee_dia()
LET dumy = FALSE
OPEN WINDOW win1 AT 20,10 WITH 2 ROWS, 60 COLUMNS attribute
(BORDER)
LET tipo = 100
MENU "SERSUNIX"
 COMMAND "Monitorea" "Despliega Actividad del Sistema
Operativo"
   MENU "MONITOR"
    COMMAND "CPU" "Monitorea Actividad de CPU"
     CLEAR WINDOW win1
     CALL moncpu()
     LET dumy = TRUE
    COMMAND "E/S" "Monitorea Actividad de E/S"
     CLEAR WINDOW win1
     CALL mones()
     LET dumy = TRUE
    COMMAND "Memoria" "Monitorea Actividad de Memoria"
     CLEAR WINDOW win1
     CALL monmem()
     LET dumy = TRUE
    COMMAND "Hardware" "Muestra Configuracion Fisica del
Equipo"
      OPEN WINDOW win9 AT 2,2 WITH 22 ROWS, 78 COLUMNS
attribute (BORDER)
      RUN "sh conf.sh 2> /dev/null"
      CLOSE WINDOW win9
    COMMAND KEY(F1)
     CALL monitor_help()
    COMMAND KEY(F4)
     IF dumy
      THEN
       CLEAR WINDOW win1
       CURRENT WINDOW IS win1
       DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
       CALL imp_mon()
      END IF
    COMMAND KEY("!")
     CALL bang()
    COMMAND "Anterior" "Regresa al Menu Anterior"
```

```
      IF dumy
       THEN
        CURRENT WINDOW IS screen
        CLEAR SCREEN
        CALL marco(1,1,79,24)
        CALL logo ()
        CURRENT WINDOW IS win1
        LET dumy = FALSE
      END IF
      LET tipo = 100
      EXIT MENU
    END MENU
  COMMAND "Sintoniza" "Sintoniza el Sistema Operativo"
   MENU "SINTONIZA"
    COMMAND "Diagnostica" "Diagnostica el Comportamiento
Monitoreado"
     CLEAR WINDOW win1
     CURRENT WINDOW IS SCREEN
     CALL borra(2,9,13,60)
     CURRENT WINDOW IS win1
     CALL leemon()
     CALL camcpu()
     CURRENT WINDOW is win1
     CALL cambuf()
     CURRENT WINDOW is win1
     CALL camcol()
     CURRENT WINDOW is win1
     CALL camswp()
     CURRENT WINDOW is win1
     CALL camtab()
     CURRENT WINDOW is win1
     CLEAR WINDOW win1
     WHILE TRUE
     DISPLAY " <F1> Ayuda " AT 1,48 ATTRIBUTE (REVERSE)
     PROMPT "                    Presione <c> para Continuar "
FOR char answ
      HELP 21
      IF answ MATCHES "[C,c]"
       THEN
        EXIT WHILE
      END IF
     END WHILE
     CLOSE WINDOW win2
     CLOSE WINDOW win3
     CLOSE WINDOW win4
     CLOSE WINDOW win5
     CLOSE WINDOW win6
     CURRENT WINDOW is screen
     CALL logo ()
     CURRENT WINDOW is win1
    COMMAND "Reconfigura" "Reconfigura el Kernel del Sistema
Operativo"
     CALL reconf()
```

```
    COMMAND "Cancela" "Reconstruye el Kernel con los
Parametros Anteriores"
      CALL cancel()
    COMMAND KEY("!")
      CALL bang()
    COMMAND "Anterior" "Regresa al Menu Anterior"
      EXIT MENU
   END MENU
  COMMAND "Lee-Dia" "Lee el dia a Monitorear"
    CALL lee_dia()
  COMMAND "Terminar" "Termina el Acceso al el Sistema"
   EXIT MENU
  COMMAND KEY("!")
   CALL bang()
END MENU
CLEAR SCREEN
END MAIN
```

```
GLOBALS "globales.4gl"
FUNCTION moncpu()
 DISPLAY " Generando Estadisticas .... " AT 2,16
ATTRIBUTE(REVERSE)
 DELETE FROM cpu
 DELETE FROM tab
 LET cmd ="sh ./moncpu.sh ",dia USING "&&" CLIPPED
 RUN cmd RETURNING codigo
 IF codigo = 0
  THEN
   LET titulo = "MONITOREO DE ACTIVIDAD DEL CPU"
   LET leyendax = "Tiempo"
   MENU "CPU"
    COMMAND "Inactivo" "Monitorea Inactividad"
     LET tipo = 1
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "INACTIVIDAD (x 10)"
     LET leyenday = "%idle"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM cpu
     order by 1
     CLEAR SCREEN
     FOREACH c_cur INTO p_cpu.*
      CALL monitor(p_cpu.idle)
     END FOREACH
     CLOSE c_cur
    COMMAND "E/S" "Monitorea Inactividad por Operaciones
E/S"
     LET tipo = 2
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "INACTIVIDAD POR E/S (x 10)"
     LET leyenday = "%wio"
     DECLARE c_cur2 CURSOR FOR
     SELECT * FROM cpu
     order by 1
     CLEAR SCREEN
     FOREACH c_cur2 INTO p_cpu.*
      CALL monitor(p_cpu.wio)
     END FOREACH
     CLOSE c_cur2
    COMMAND "Sistema" "Monitorea Utilizacion por Procesos en
Modo Sistema"
     LET tipo = 3
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "UTILIZACION POR PROCESOS EN MODO
SISTEMA (x 10)"
     LET leyenday = "%sys"
     DECLARE c_cur3 CURSOR FOR
     SELECT * FROM cpu
     order by 1
     CLEAR SCREEN
```

```
      FOREACH c_cur3 INTO p_cpu.*
       CALL monitor(p_cpu.sys)
      END FOREACH
      CLOSE c_cur3
     COMMAND "Usuario" "Monitorea Utilizacion por Procesos en
Modo Usuario"
      LET tipo = 4
      CALL ini_mat()
      LET lectura = 0
      LET subtitulo = "UTILIZACION POR PROCESOS EN MODO
USUARIO (x 10)"
      LET leyenday = "%usr"
      DECLARE c_cur4 CURSOR FOR
      SELECT * FROM cpu
      order by 1
      CLEAR SCREEN
      FOREACH c_cur4 INTO p_cpu.*
       CALL monitor(p_cpu.usr)
      END FOREACH
      CLOSE c_cur4
     COMMAND "Tablas" "Monitorea Utilizacion de Tablas"
      LET titulo = "MONITOREO DE ACTIVIDAD DE TABLAS"
      LET leyendax = "Tiempo"
      MENU "TABLAS"
       COMMAND "Procesos" "Longitud de la Tabla de Procesos"
        LET tipo = 5
        CALL ini_mat()
        LET lectura = 0
        LET subtitulo = "LONGITUD DE LA TABLA DE PROCESOS (x
10)"
        LET leyenday = "# de procesos"
        DECLARE c_cur5 CURSOR FOR
         SELECT * FROM tab
          order by 1
        CLEAR SCREEN
        FOREACH c_cur5 INTO p_tab.*
         CASE
          WHEN p_tab.pro > 100
           CALL monitor(100)
          OTHERWISE
           CALL monitor(p_tab.pro)
         END CASE
        END FOREACH
        CLOSE c_cur5
       COMMAND "i-Nodos" "Longitud de la Tabla de i-Nodos"
        LET tipo = 6
        CALL ini_mat()
        LET lectura = 0
        LET subtitulo = "LONGITUD DE LA TABLA DE  I-NODOS (x
10)"
        LET leyenday = "# de i-nodos"
        DECLARE c_cur6 CURSOR FOR
        SELECT * FROM tab
        order by 1
```

```
        CLEAR SCREEN
        FOREACH c_cur6 INTO p_tab.*
        CASE
          WHEN p_tab.ino/10 > 100
           CALL monitor(100)
          OTHERWISE
           CALL monitor(p_tab.ino/10)
         END CASE
        END FOREACH
        CLOSE c_cur6
       COMMAND "Ficheros" "Longitud de la Tabla de Archivos"
        LET tipo = 7
        CALL ini_mat()
        LET lectura = 0
        LET subtitulo = "LONGITUD DE LA TABLA DE ARCHIVOS (x
10)"
        LET leyenday = "# de archivos"
        DECLARE c_cur7 CURSOR FOR
        SELECT * FROM tab
         order by 1
        CLEAR SCREEN
        FOREACH c_cur7 INTO p_tab.*
         CASE
          WHEN p_tab.fil/10 > 100
           CALL monitor(100)
          OTHERWISE
           CALL monitor(p_tab.fil/10)
         END CASE
        END FOREACH
        CLOSE c_cur7
       COMMAND KEY(F1)
        CALL monitor_help()
       COMMAND KEY(F4)
        IF tipo < 100
         THEN
          CLEAR WINDOW win1
          CURRENT WINDOW IS win1
          DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
          CALL imp_mon()
        END IF
       COMMAND KEY("!")
        CALL bang()
       COMMAND "Anterior" "Regresa al Menu anterior"
        EXIT MENU
      END MENU
     COMMAND KEY(F1)
      CALL monitor_help()
     COMMAND KEY(F4)
      IF tipo < 100
       THEN
        CLEAR WINDOW win1
        CURRENT WINDOW IS win1
```

```
      DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
      CALL imp_mon()
    END IF
   COMMAND KEY("!")
    CALL bang()
   COMMAND "Anterior" "Regresa al Menu anterior"
    EXIT MENU
  END MENU
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION mones()
 DEFINE num INTEGER
 DISPLAY " Generando Estadisticas .... " AT 2,16
ATTRIBUTE(REVERSE)
 DELETE FROM es
 LET cmd = "sh ./mones.sh ",dia USING "&&" CLIPPED
 RUN cmd RETURNING codigo
 IF codigo = 0
  THEN
   LET titulo = "MONITOREO DE ACTIVIDAD DE E/S"
   LET leyendax = "Tiempo"
   MENU "E/S"
    COMMAND "Entrada" "Monitorea Variacion de Caracteres de
Entrada"
     LET tipo = 8
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "ENTRADA (x 10)"
     LET leyenday = "caracteres/s"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM es
     order by 1
     CLEAR SCREEN
     FOREACH c_cur INTO p_es.*
      CALL monitor(p_es.raw)
     END FOREACH
     CLOSE c_cur
    COMMAND "Salida" "Monitorea Variacion de Caracteres de
Salida"
     LET tipo = 9
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "SALIDA (x 10)"
     LET leyenday = "caracteres/s"
     DECLARE c_cur2 CURSOR FOR
     SELECT * FROM es
     order by 1
     CLEAR SCREEN
     FOREACH c_cur2 INTO p_es.*
      LET num = p_es.out/10
      CASE
       WHEN num > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(num)
      END CASE
     END FOREACH
     CLOSE c_cur2
    COMMAND KEY(F1)
     CALL monitor_help()
    COMMAND KEY(F4)
     IF tipo < 100
      THEN
```

```
        CLEAR WINDOW win1
        CURRENT WINDOW IS win1
        DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
        CALL imp_mon()
      END IF
     COMMAND KEY("!")
      CALL bang()
     COMMAND "Anterior" "Regresa al Menu Anterior"
      EXIT MENU
    END MENU
  END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION monmem()
 DEFINE num INTEGER
 DISPLAY " Generando Estadisticas .... " AT 2,16
ATTRIBUTE(REVERSE)
 DELETE FROM mem
 DELETE FROM buf
 LET cmd = "sh ./monmem.sh ",dia USING "&&" CLIPPED
 RUN cmd RETURNING codigo
 IF codigo = 0
  THEN
   LET titulo = "MONITOREO DE ACTIVIDAD DE MEMORIA"
   LET leyendax = "Tiempo"
   MENU "MEMORIA"
    COMMAND "Paginas" "Monitorea Paginas de Memoria Libres"
     LET tipo = 11
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "MEMORIA (x 1,000)"
     LET leyenday = "pags. libres"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM mem
     order by 1
     CLEAR SCREEN
     FOREACH c_cur INTO p_mem.*
      CALL monitor(p_mem.mem/100)
     END FOREACH
     CLOSE c_cur
    COMMAND "Swap" "Monitorea Paginas de Intercambio Libres"
     LET tipo = 12
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "SWAP (x 10,000)"
     LET leyenday = "pags. libres"
     DECLARE c_cur2 CURSOR FOR
     SELECT * FROM mem
     order by 1
     CLEAR SCREEN
     FOREACH c_cur2 INTO p_mem.*
      LET num = p_mem.swp/1000
      CASE
       WHEN num > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(num)
      END CASE
     END FOREACH
     CLOSE c_cur2
    COMMAND "Buffers" "Monitorea los Buffers del Sistema"
     CALL monbuf()
    COMMAND KEY(F1)
     CALL monitor_help()
    COMMAND KEY(F4)
     IF tipo < 100
```

```
      THEN
       CLEAR WINDOW win1
       CURRENT WINDOW IS win1
       DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
       CALL imp_mon()
      END IF
    COMMAND KEY("!")
     CALL bang()
    COMMAND "Anterior" "Regresa al Menu anterior"
     EXIT MENU
   END MENU
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION monbuf()
 DEFINE num INTEGER
 LET titulo = "MONITOREO DE ACTIVIDAD DE BUFFERS"
 LET leyendax = "Tiempo"
 MENU "BUFFERS"
  COMMAND "Caracter" "Transferencias del Sistema a
Dispositivos de Caracter"
   MENU "CARACTER"
    COMMAND "Lectura" "Lecturas Realizadas de los
Dispositivos de Caracteres"
     LET tipo = 13
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "LECTURA DE DISPOSITIVOS DE CARACTER (x
10)"
     LET leyenday = "accesos/seg"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM buf
     order by 1
     CLEAR SCREEN
     FOREACH c_cur INTO p_buf.*
      CASE
       WHEN p_buf.pre > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(p_buf.pre)
      END CASE
     END FOREACH
     CLOSE c_cur
    COMMAND "Escritura" "Escrituras Realizadas en los
Dispositivos de Carcateres"
     LET tipo = 14
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "ESCRITURA EN DISPOSITIVOS DE CARACTER
(x 10)"
     LET leyenday = "accesos/seg"
     DECLARE c_cur1 CURSOR FOR
     SELECT * FROM buf
     order by 1
     CLEAR SCREEN
     FOREACH c_cur1 INTO p_buf.*
      CASE
       WHEN p_buf.pwr > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(p_buf.pwr)
      END CASE
     END FOREACH
     CLOSE c_cur1
    COMMAND KEY(F1)
     CALL monitor_help()
    COMMAND KEY(F4)
```

```
      IF tipo < 100
       THEN
        CLEAR WINDOW win1
        CURRENT WINDOW IS win1
        DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
        CALL imp_mon()
      END IF
     COMMAND KEY("!")
      CALL bang()
     COMMAND "Anterior" "Regresa al Menu anterior"
      EXIT MENU
    END MENU
  COMMAND "MemCache" "Porcentaje de Aciertos en Memoria
Cache"
    MENU "MEMCACHE"
     COMMAND "Lectura" "Lecturas Realizadas de Memoria Cache"
      LET tipo = 15
      CALL ini_mat()
      LET lectura = 0
      LET subtitulo = "LECTURA DE MEMORIA CACHE (x 10)"
      LET leyenday = "% de lecturas"
      DECLARE c_cur3 CURSOR FOR
      SELECT * FROM buf
      order by 1
      CLEAR SCREEN
      FOREACH c_cur3 INTO p_buf.*
       CASE
        WHEN p_buf.rca > 100
         CALL monitor(100)
        OTHERWISE
         CALL monitor(p_buf.rca)
       END CASE
      END FOREACH
      CLOSE c_cur3
     COMMAND "Escritura" "Escrituras Realizadas a Memoria
Cache"
      LET tipo = 16
      CALL ini_mat()
      LET lectura = 0
      LET subtitulo = "ESCRITURA DE MEMORIA CACHE (x 10)"
      LET leyenday = "% de lecturas"
      DECLARE c_cur4 CURSOR FOR
      SELECT * FROM buf
      order by 1
      CLEAR SCREEN
      FOREACH c_cur4 INTO p_buf.*
       CASE
        WHEN p_buf.wca > 100
         CALL monitor(100)
        OTHERWISE
         CALL monitor(p_buf.wca)
       END CASE
      END FOREACH
```

```
     CLOSE c_cur4
    COMMAND KEY(F1)
     CALL monitor_help()
    COMMAND KEY(F4)
     IF tipo < 100
      THEN
      CLEAR WINDOW win1
      CURRENT WINDOW IS win1
      DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
      CALL imp_mon()
     END IF
    COMMAND KEY("!")
     CALL bang()
    COMMAND "Anterior" "Regresa al Menu anterior"
     EXIT MENU
   END MENU
  COMMAND "Bloques" "Transferencias del Sistema a
Dispositivos de Bloques"
   MENU "BLOQUES"
    COMMAND "Lectura" "Lectura de Dispositivos de Bloque"
     LET tipo = 17
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "LECTURA DE DISPOSITIVOS DE BLOQUE (x
10)"
     LET leyenday = "accesos/s "
     DECLARE c_cur5 CURSOR FOR
     SELECT * FROM buf
     order by 1
     CLEAR SCREEN
     FOREACH c_cur5 INTO p_buf.*
      CASE
       WHEN p_buf.bre > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(p_buf.bre)
      END CASE
     END FOREACH
     CLOSE c_cur5
    COMMAND "Escritura" "Escritura de Dispositivos de
Bloque"
     LET tipo = 18
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "ESCRITURA DE DISPOSITIVOS DE BLOQUE (x
10)"
     LET leyenday = "accesos/s"
     DECLARE c_cur6 CURSOR FOR
     SELECT * FROM buf
     order by 1
     CLEAR SCREEN
     FOREACH c_cur6 INTO p_buf.*
      CASE
```

```
        WHEN p_buf.bwr > 100
         CALL monitor(100)
        OTHERWISE
         CALL monitor(p_buf.bwr)
       END CASE
      END FOREACH
      CLOSE c_cur6
     COMMAND KEY(F1)
      CALL monitor_help()
     COMMAND KEY(F4)
      IF tipo < 100
       THEN
        CLEAR WINDOW win1
        CURRENT WINDOW IS win1
        DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
        CALL imp_mon()
      END IF
     COMMAND KEY("!")
      CALL bang()
     COMMAND "Anterior" "Regresa al Menu anterior"
      EXIT MENU
   END MENU
  COMMAND "Sistema" "Accesos del Sistema de Buffers"
   MENU "SISTEMA"
    COMMAND "Lectura" "Lecturas Realizadas del Sistema de
Buffers"
     LET tipo = 19
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "LECTURA DE BUFFERS (x 10)"
     LET leyenday = "accesos/seg"
     DECLARE c_cur7 CURSOR FOR
     SELECT * FROM buf
     order by 1
     CLEAR SCREEN
     FOREACH c_cur7 INTO p_buf.*
      CASE
       WHEN p_buf.lre > 100
        CALL monitor(100)
       OTHERWISE
        CALL monitor(p_buf.lre)
      END CASE
     END FOREACH
     CLOSE c_cur7
    COMMAND "Escritura" "Escrituras Realizadas al Sistema de
Buffers"
     LET tipo = 20
     CALL ini_mat()
     LET lectura = 0
     LET subtitulo = "ESCRITURA DE BUFFERS (x 10)"
     LET leyenday = "accesos/seg"
     DECLARE c_cur8 CURSOR FOR
     SELECT * FROM buf
```

```
      order by 1
      CLEAR SCREEN
      FOREACH c_cur8 INTO p_buf.*
       CASE
        WHEN p_buf.lwr > 100
         CALL monitor(100)
        OTHERWISE
         CALL monitor(p_buf.lwr)
       END CASE
      END FOREACH
      CLOSE c_cur8
     COMMAND KEY(F1)
      CALL monitor_help()
     COMMAND KEY(F4)
      IF tipo < 100
       THEN
        CLEAR WINDOW win1
        CURRENT WINDOW IS win1
        DISPLAY " Imprimiendo .... " AT 2,22
ATTRIBUTE(REVERSE)
        CALL imp_mon()
      END IF
     COMMAND KEY("!")
      CALL bang()
     COMMAND "Anterior" "Regresa al Menu anterior"
      EXIT MENU
   END MENU
  COMMAND KEY(F4)
   IF tipo < 100
    THEN
     CLEAR WINDOW win1
     CURRENT WINDOW IS win1
     DISPLAY " Imprimiendo .... " AT 2,22 ATTRIBUTE(REVERSE)
     CALL imp_mon()
   END IF
  COMMAND KEY("!")
   CALL bang()
  COMMAND "Anterior" "Regresa al Menu anterior"
   EXIT MENU
 END MENU
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION reconf()
 CLEAR WINDOW win1
 OPEN WINDOW win11 AT 12,11 WITH 3 ROWS,57 COLUMNS
ATTRIBUTE(BORDER)
 OPTIONS
  PROMPT LINE 3
 DISPLAY " Verificando Archivos de Recomendaciones ... "
  AT 2,7 ATTRIBUTE(REVERSE)
 RUN "sh existe.sh" RETURNING codigo
 IF codigo = 0
  THEN
   CLOSE WINDOW win11
   CURRENT WINDOW IS win1
   DISPLAY "Estas seguro de reconfigurar el kernel" AT 1,12
   PROMPT "                con los parametros propuestos?
(s/n)"
     FOR CHAR answ
   CLEAR WINDOW win1
   IF answ MATCHES "[S,s]"
    THEN
     RUN "sh reconf.sh" RETURNING codigo
     IF codigo = 0
      THEN
       OPEN WINDOW win11 AT 12,11 WITH 3 ROWS,57 COLUMNS
ATTRIBUTE(BORDER)
       DISPLAY " El kernel ha sido reconfigurado " AT 2,14
ATTRIBUTE(REVERSE)
       PROMPT "                    Presione cualquier tecla para
Continuar"
        FOR CHAR answ
       CLOSE WINDOW win11
      ELSE
       DISPLAY " No fue posible reconfigurar el kernel error
no. ",codigo," "
        AT 1,5 ATTRIBUTE(REVERSE)
       PROMPT "                         Presione cualquier tecla
para continuar"
        FOR CHAR answ
     END IF
   END IF
  ELSE
   CLEAR WINDOW win11
   DISPLAY " Ejecute Diagnostica para Generar
Recomendaciones " AT 2,4
    ATTRIBUTE(REVERSE)
   PROMPT
    "                   Presione cualquier tecla para
continuar" FOR CHAR answ
   CLOSE WINDOW win11
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION camcol()
 DEFINE i SMALLINT
 DELETE FROM aju_col
 CLEAR WINDOW win1
 DISPLAY " Diagnosticando Colas ... " AT 2,17
ATTRIBUTE(REVERSE)
 SLEEP 3
 SELECT * INTO p_col.* FROM col
  WHERE num_con = ( SELECT MAX (num_con) FROM col)
 IF (p_col.runo > 90) AND (p_col.runq > 2)
  THEN
   CLEAR WINDOW win1
   ERROR ""
   DISPLAY " Se Encontraron Problemas en el Desempeno de
Colas " AT 2,6 ATTRIBUTE(REVERSE)
   CALL confir() RETURNING answ
   IF answ = "S"
    THEN
     CLEAR WINDOW win1
     DISPLAY " Generando Recomendaciones para Colas " AT
2,12 ATTRIBUTE(REVERSE)
     RUN "sh ./camcol.sh"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM  aju_col
     LET i = 3
     OPEN WINDOW win4 AT 9,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " COLAS " AT 1,11 ATTRIBUTE(REVERSE)
     DISPLAY "Parametro Actual Propuesto" AT 2,2
     FOREACH c_cur INTO paj_col.*
      DISPLAY paj_col.des AT i,3
      DISPLAY paj_col.act AT i,14
      DISPLAY paj_col.rec AT i,23
      LET i = i + 1
     END FOREACH
    ELSE
     OPEN WINDOW win4 AT 9,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " COLAS " AT 1,11 ATTRIBUTE(REVERSE)
     DISPLAY "No se Acepto Recomendacion" AT 3,2
   END IF
  ELSE
   OPEN WINDOW win4 AT 9,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
   DISPLAY " COLAS " AT 1,11 ATTRIBUTE(REVERSE)
   DISPLAY " SIN PROBLEMAS " AT 3,7 ATTRIBUTE(REVERSE)
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION cambuf()
 DEFINE i SMALLINT
 DELETE FROM aju_buf
 LEt i = 1
 CLEAR WINDOW win1
 DISPLAY " Diagnosticando Buffers ... " AT 2,16
ATTRIBUTE(REVERSE)
 SLEEP 3
 SELECT * INTO p_buf.* FROM buf
  WHERE num_con = ( SELECT MAX (num_con) FROM buf)
 IF (p_buf.rca < 90) OR (p_buf.wca < 65)
  THEN
   CLEAR WINDOW win1
   ERROR ""
   DISPLAY " Se Encontraron Problemas en el Desempeno de
Buffers " AT 2,4 ATTRIBUTE(REVERSE)
   CALL confir() RETURNING answ
   IF answ = "S"
    THEN
     CLEAR WINDOW win1
     DISPLAY " Generando Recomendaciones para Buffers " AT
2,11 ATTRIBUTE(REVERSE)
     RUN "sh ./cambuf.sh"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM  aju_buf
     LET i = 3
     OPEN WINDOW win3 AT 3,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " BUFFERS " AT 1,10 ATTRIBUTE(REVERSE)
     DISPLAY "Parametro Actual Propuesto" AT 2,2
     FOREACH c_cur INTO paj_buf.*
      DISPLAY paj_buf.des AT i,3
      DISPLAY paj_buf.act AT i,14
      DISPLAY paj_buf.rec AT i,23
      LET i = i + 1
     END FOREACH
    ELSE
     OPEN WINDOW win3 AT 3,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " BUFFERS " AT 1,10 ATTRIBUTE(REVERSE)
     DISPLAY "No se Acepto Recomendacion" AT 3,2
    END IF
   ELSE
    OPEN WINDOW win3 AT 3,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
    DISPLAY " BUFFERS " AT 1,10 ATTRIBUTE(REVERSE)
    DISPLAY " SIN PROBLEMAS " AT 3,7 ATTRIBUTE(REVERSE)
  END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION camcpu()
 DEFINE i SMALLINT
 DELETE FROM aju_cpu
 CLEAR WINDOW win1
 DISPLAY " Diagnosticando CPU ... " AT 2,18
ATTRIBUTE(REVERSE)
 SLEEP 3
 SELECT * INTO p_cpu.* FROM cpu
  WHERE num_con = ( SELECT MAX (num_con) FROM cpu)
 IF p_cpu.wio > 70
  THEN
   CLEAR WINDOW win1
   ERROR ""
   DISPLAY " Se Encontraron Problemas en el Desempeno del
CPU " AT 2,6 ATTRIBUTE(REVERSE)
   CALL confir() RETURNING answ
   IF answ = "S"
    THEN
     CLEAR WINDOW win1
     DISPLAY " Generando Recomendaciones para el CPU " AT
2,12 ATTRIBUTE(REVERSE)
     RUN "sh ./camcpu.sh"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM  aju_cpu
     LET i = 3
     OPEN WINDOW win2 AT 3,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " C P U " AT 1,12 ATTRIBUTE(REVERSE)
     DISPLAY "Parametro Actual Propuesto" AT 2,2
     FOREACH c_cur INTO paj_cpu.*
      DISPLAY paj_cpu.des AT i,3
      DISPLAY paj_cpu.act AT i,14
      DISPLAY paj_cpu.rec AT i,23
      LET i = i + 1
     END FOREACH
    ELSE
     OPEN WINDOW win2 AT 3,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " C P U " AT 1,12 ATTRIBUTE(REVERSE)
     DISPLAY "No se Acepto Recomendacion" AT 4,2
   END IF
  ELSE
   OPEN WINDOW win2 AT 3,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
   DISPLAY " C P U " AT 1,12 ATTRIBUTE(REVERSE)
   DISPLAY " SIN PROBLEMAS " AT 4,8 ATTRIBUTE(REVERSE)
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION camswp()
 DEFINE i SMALLINT
 DELETE FROM aju_swp
 CLEAR WINDOW win1
 DISPLAY " Diagnosticando el Area de Swap ... " AT 2,12
ATTRIBUTE(REVERSE)
 SLEEP 3
 SELECT * INTO p_swp.* FROM swp
  WHERE num_con = ( SELECT MAX (num_con) FROM swp)
 IF p_swp.swpo > 1
  THEN
   CLEAR WINDOW win1
   ERROR ""
   DISPLAY " Se Encontraron Problemas en el Desempeno del
Swap " AT 2,5 ATTRIBUTE(REVERSE)
   CALL confir() RETURNING answ
   IF answ = "S"
    THEN
     CLEAR WINDOW win1
     DISPLAY " Generando Recomendaciones para Swap " AT 2,12
ATTRIBUTE(REVERSE)
     RUN "sh ./camswp.sh"
     DECLARE c_cur CURSOR FOR
     SELECT * FROM  aju_swp
     LET i = 3
     OPEN WINDOW win5 AT 15,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " SWAP " AT 1,11 ATTRIBUTE(REVERSE)
     DISPLAY "Parametro Actual Propuesto" AT 2,2
     FOREACH c_cur INTO paj_swp.*
      DISPLAY paj_swp.des AT i,3
      DISPLAY paj_swp.act AT i,14
      DISPLAY paj_swp.rec AT i,23
      LET i = i + 1
     END FOREACH
    ELSE
     OPEN WINDOW win5 AT 15,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " SWAP " AT 1,11 ATTRIBUTE(REVERSE)
     DISPLAY "No se Acepto Recomendacion" AT 3,2
   END IF
  ELSE
   OPEN WINDOW win5 AT 15,42 WITH 3 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
   DISPLAY " SWAP " AT 1,11 ATTRIBUTE(REVERSE)
   DISPLAY " SIN PROBLEMAS " AT 3,6 ATTRIBUTE(REVERSE)
 END IF
END FUNCTION
```

```
   END IF
 END IF
 IF (p_tab.iov > 0) OR (p_tab.fov > 0)
  THEN
   DELETE FROM aju_tab
   IF NOT flg_pro
    THEN
     CLEAR WINDOW win1
     ERROR ""
     DISPLAY " Se Encontraron Problemas en el Desempeno de
las Tablas " AT 2,3 ATTRIBUTE(REVERSE)
     CALL confir() RETURNING answ
     IF answ = "S"
      THEN
       CLEAR WINDOW win1
       DISPLAY " Generando Recomendaciones para Tablas " AT
2,12 ATTRIBUTE(REVERSE)
       LET flg_rec = TRUE
      ELSE
       OPEN WINDOW win6 AT 12,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
       DISPLAY " TABLAS " AT 1,11 ATTRIBUTE(REVERSE)
       DISPLAY "No se Acepto Recomendacion" AT 4,2
ATTRIBUTE(REVERSE)
       RETURN
     END IF
   END IF
   RUN "sh camtabi.sh"
   IF NOT flg_win
    THEN
     OPEN WINDOW win6 AT 12,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
     DISPLAY " TABLAS " AT 1,11 ATTRIBUTE(REVERSE)
     DISPLAY "Parametro Actual Propuesto" AT 2,2
   END IF
   DECLARE c_cur1 CURSOR FOR
   SELECT * FROM  aju_tab
   FOREACH c_cur1 INTO paj_tab.*
    DISPLAY paj_tab.des AT i,3
    DISPLAY paj_tab.act AT i,14
    DISPLAY paj_tab.rec AT i,22
    LET i = i + 1
   END FOREACH
   RETURN
 END IF
 OPEN WINDOW win6 AT 12,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
 DISPLAY " TABLAS " AT 1,11 ATTRIBUTE(REVERSE)
 DISPLAY " SIN PROBLEMAS " AT 4,8 ATTRIBUTE(REVERSE)
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION camtab()
 DEFINE i SMALLINT,
      flg_win,flg_pro,flg_rec INTEGER
 DELETE FROM aju_tab
 CLEAR WINDOW win1
 DISPLAY " Diagnosticando Tablas ... " AT 2,17
ATTRIBUTE(REVERSE)
 SLEEP 3
 LET flg_pro = FALSE
 LET flg_rec = FALSE
 LET flg_win = FALSE
 LET i = 3
 SELECT * INTO p_tab.* FROM tab
  WHERE num_con = ( SELECT MAX (num_con) FROM cpu)
 IF p_tab.pov > 0
  THEN
   SELECT * into p_cpu.* FROM cpu
    WHERE num_con = ( SELECT MAX (num_con) FROM cpu)
   IF p_cpu.wio < 70
    THEN
     CLEAR WINDOW win1
     ERROR ""
     DISPLAY " Se Encontraron Problemas en el Desempeno de
las Tablas " AT 2,3 ATTRIBUTE(REVERSE)
     LET flg_pro = TRUE
     CALL confir() RETURNING answ
     IF answ = "S"
      THEN
       CLEAR WINDOW win1
       DISPLAY " Generando Recomendaciones para Tablas " AT
2,12 ATTRIBUTE(REVERSE)
       LET flg_rec = TRUE
       RUN "sh camtabp.sh"
       DECLARE c_cur CURSOR FOR
       SELECT * FROM  aju_tab
       LET flg_win = TRUE
       OPEN WINDOW win6 AT 12,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
       DISPLAY " TABLAS " AT 1,11 ATTRIBUTE(REVERSE)
       DISPLAY "Parametro Actual Propuesto" AT 2,2
       FOREACH c_cur INTO paj_tab.*
        DISPLAY paj_tab.des AT i,3
        DISPLAY paj_tab.act AT i,14
        DISPLAY paj_tab.rec AT i,22
        LET i = i + 1
       END FOREACH
      ELSE
       OPEN WINDOW win6 AT 12,10 WITH 6 ROWS,28 COLUMNS
ATTRIBUTE(BORDER)
       DISPLAY " TABLAS " AT 1,11 ATTRIBUTE(REVERSE)
       DISPLAY "No se Acepto Recomendacion" AT 4,2
ATTRIBUTE(REVERSE)
     END IF
```

```
GLOBALS "globales.4gl"
FUNCTION cancel()
 CLEAR WINDOW win1
 OPEN WINDOW win12 AT 12,11 WITH 3 ROWS,57 COLUMNS
ATTRIBUTE(BORDER)
 OPTIONS
  PROMPT LINE 3
 DISPLAY " Verificando Archivos de Respaldo ... "
  AT 2,10 ATTRIBUTE(REVERSE)
 RUN "sh existen.sh" RETURNING codigo
 IF codigo = 0
  THEN
   CLOSE WINDOW win12
   CURRENT WINDOW IS win1
   DISPLAY "Estas seguro de reconfigurar el kernel" AT 1,12
   PROMPT "                  con los parametros anteriores?
(s/n)"
     FOR CHAR answ
   CLEAR WINDOW win1
   IF answ MATCHES "[S,s]"
    THEN
     RUN "sh cancel.sh" RETURNING codigo
     IF codigo = 0
      THEN
       OPEN WINDOW win12 AT 12,11 WITH 3 ROWS,57 COLUMNS
ATTRIBUTE(BORDER)
       DISPLAY " El kernel ha sido reconfigurado " AT 2,14
ATTRIBUTE(REVERSE)
       PROMPT "                  Presione cualquier tecla para
Continuar"
        FOR CHAR answ
       CLOSE WINDOW win12
      ELSE
       DISPLAY "No fue posible reconfigurar el kernel error
no. ",codigo
        AT 2,1 ATTRIBUTE(REVERSE)
       PROMPT "Presione cualquier tecla para Continuar" FOR
CHAR answ
     END IF
   END IF
  ELSE
   CLEAR WINDOW win12
   DISPLAY " No existen los Archivos de Respaldo " AT 2,11
    ATTRIBUTE(REVERSE)
   PROMPT
    "                  Presione cualquier tecla para
continuar" FOR CHAR answ
   CLOSE WINDOW win12
 END IF
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION imp_mon()
 DEFINE i,j,ren INTEGER
 FOR i = 1 TO 70
  LET ren = mat[i] / 10
  IF ren <> 0
   THEN
    LET v[ren,i] = "X"
    FOR j = ren TO 1 STEP -1
     LET v[j,i] = "X"
    END FOR
  END IF
 END FOR
 START REPORT rep_mon TO PRINTER
 FOR i = 10 TO 1 STEP -1
  OUTPUT TO REPORT rep_mon(i)
 END FOR
 FINISH REPORT rep_mon
END FUNCTION
REPORT rep_mon(i)
DEFINE i,offset INTEGER
OUTPUT
  LEFT MARGIN   1
  RIGHT MARGIN  132
  TOP MARGIN    3
  BOTTOM MARGIN 3
  PAGE LENGTH   66
FORMAT
 PAGE HEADER
    LET offset = (97 - length(titulo))/2
    PRINT COLUMN offset,titulo
    LET offset = (97 - length(subtitulo))/2
    PRINT COLUMN offset,subtitulo
    PRINT COLUMN 76,fecha
  PRINT COLUMN 11,"+-------------------------------------
-----------------------------+"
 ON EVERY ROW
  PRINT COLUMN 8,leyenday[11-i],
        COLUMN 9,i USING "##",
        COLUMN 11,"|",
        COLUMN
12,v[i,1],v[i,2],v[i,3],v[i,4],v[i,5],v[i,6],v[i,7],v[i,8],

v[i,9],v[i,10],v[i,11],v[i,12],v[i,13],v[i,14],v[i,15],

v[i,16],v[i,17],v[i,18],v[i,19],v[i,20],v[i,21],v[i,22],

v[i,23],v[i,24],v[i,25],v[i,26],v[i,27],v[i,28],v[i,29],

v[i,30],v[i,31],v[i,32],v[i,33],v[i,34],v[i,35],v[i,36],

v[i,37],v[i,38],v[i,39],v[i,40],v[i,41],v[i,42],v[i,43],

v[i,44],v[i,45],v[i,46],v[i,47],v[i,48],v[i,49],v[i,50],
```

```
v[i,51],v[i,52],v[i,53],v[i,54],v[i,55],v[i,56],v[i,57],

v[i,58],v[i,59],v[i,60],v[i,61],v[i,62],v[i,63],v[i,64],

v[i,65],v[i,66],v[i,67],v[i,68],v[i,69],v[i,70],
        COLUMN 82,"|"
 ON LAST ROW
   PRINT COLUMN 11, "+-----------------------------------------
-------------------------------+"
   PRINT COLUMN
12,"....+....|....+....|....+....|....+....|....+....|....+....|....+.
...|....+....|"
   LET offset = (97 - length(leyendax))/2
   SKIP 1 LINE
   PRINT COLUMN offset,leyendax
END REPORT
```

```
GLOBALS "globales.4gl"
FUNCTION leemon( )
 CLEAR WINDOW win1
 DISPLAY " Consultando Estadisticas .... " AT 2,15
ATTRIBUTE(REVERSE)
 DELETE FROM es
 DELETE FROM swp
 DELETE FROM col
 DELETE FROM mem
 DELETE FROM buf
 DELETE FROM cpu
 DELETE FROM tab
 LET cmd = "sh ./todos.sh ",dia USING "&&" CLIPPED
 RUN cmd RETURNING codigo
END FUNCTION
```

```
GLOBALS "globales.4gl"
FUNCTION get_mes()
    LET a_mes[1]="ENE"
    LET a_mes[2]="FEB"
    LET a_mes[3]="MAR"
    LET a_mes[4]="ABR"
    LET a_mes[5]="MAY"
    LET a_mes[6]="JUN"
    LET a_mes[7]="JUL"
    LET a_mes[8]="AGO"
    LET a_mes[9]="SEP"
    LET a_mes[10]="OCT"
    LET a_mes[11]="NOV"
    LET a_mes[12]="DIC"
END FUNCTION
FUNCTION confir()
 DEFINE ans CHAR(1)
 OPEN WINDOW w_con AT 12,20
  WITH FORM "con"
   ATTRIBUTE (BORDER,FORM LINE 2)
 INPUT BY NAME ans
  CLOSE WINDOW w_con
 RETURN ans
END FUNCTION
FUNCTION lee_dia()
 IF NOT flag
  THEN
   OPEN WINDOW w_dia AT 13,29
    WITH FORM "dia"
     ATTRIBUTE (BORDER,FORM LINE 2)
    INPUT BY NAME dia
    CLOSE WINDOW w_dia
  ELSE
   LET dia = DAY(TODAY)
   LET flag = FALSE
 END IF
 IF dia IS NULL
  THEN
   LET dia = DAY(TODAY)
 END IF
 CASE
  WHEN dia > DAY(TODAY)
   LET mes = MONTH(TODAY) - 1
   IF mes = 0
    THEN
     LET mes = 12
   END IF
  OTHERWISE
   LET mes = MONTH(TODAY)
 END CASE
 LET fecha = " ",a_mes[mes]," ",dia USING "&&"
END FUNCTION
FUNCTION ini_mat()  # funcion que inicializa la matriz de
graficacion
```

```
 DEFINE i,j SMALLINT
 FOR i = 1 to 10
  FOR j = 1 TO 70
   LET v[i,j] = " "
  END FOR
 END FOR
 FOR j = 1 TO 70
  LET mat[j] = 0
 END FOR
END FUNCTION
FUNCTION borra(x,y,l,a) # borra un area rectangular x,y pto
inicio
 DEFINE                  # l largo a ancho
       i,j,x,y,l,a SMALLINT,
       b CHAR(1)
  LET b =  " "
  FOR i = x TO x + l
   FOR j = y TO y + a
    DISPLAY b AT i,j
   END FOR
  END FOR
END FUNCTION
FUNCTION bang() # permite escapes a shell
    DEFINE cmd CHAR(80),
        x CHAR(1)
    DISPLAY "" AT 1,1
    DISPLAY "" AT 2,1
    LET x = "!"
    WHILE x = "!"
       PROMPT "!" FOR cmd
       RUN cmd
       PROMPT " Presione RETURN para continuar " FOR CHAR x
    END WHILE
END FUNCTION
FUNCTION logo ()  # dibuja el logo en pantalla
 DEFINE
       i SMALLINT,
       h,e CHAR(1)
  LET h = ASCII 223
  LET e = ASCII 144
  DISPLAY "          " AT 3,9 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 4,9 ATTRIBUTE(REVERSE)
  DISPLAY "     " AT 5,9 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 6,9 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 7,9 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 3,16 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 5,16 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 6,16 ATTRIBUTE(REVERSE)
  DISPLAY "   " AT 7,16 ATTRIBUTE(REVERSE)
  DISPLAY "UNIVERSIDAD NACIONAL AUTONOMA DE M",e,"XICO" AT
5,25
  FOR i = 25 TO 63
   DISPLAY h AT 6,i
  END FOR
```

```
  DISPLAY "S I S T E M A   D E   E V A L U A C I O N   D E L
" AT 11,15
  DISPLAY "R E N D I M I E N T O   Y   S I N T O N I Z A C I
O N " AT 13,13
  DISPLAY "U N I X   S Y S T E M   V " AT 15,28
END FUNCTION

FUNCTION marco(x,y,l,a) # dibuja un marco variable en
pantalla x,y pto inicial
 DEFINE                  # l largo a ancho
       t,i,x,y,l,a SMALLINT, # x,y=coordenadas iniciales
l=largo a=ancho
       h,v,esi,esd,eid,eii CHAR(1)
  LET h = ASCII 205
  LET v = ASCII 186
  LET esi = ASCII 201
  LET esd = ASCII 187
  LET eii = ASCII 200
  LET eid = ASCII 188
  DISPLAY esi AT x,y
  FOR i = y+1 TO y+l-2
   DISPLAY h AT x,i
  END FOR
  DISPLAY esd AT x,i
  LEt t = y + l - 1
  FOR i = x+1 TO x+a-2
   DISPLAY v AT i,y
   DISPLAY v AT i,t
  END FOR
  LEt t = x + a - 1
  DISPLAY eii AT t,y
  FOR i = y+1 TO y+l-2
   DISPLAY h AT t,i
  END FOR
  DISPLAY eid AT t,i
END FUNCTION

FUNCTION monitor(valor)
# funcion para desplegar en forma grafica el comportamiento
de los
# valores recibidos.  Presupone valores de entrada en un
rango  de
# 0 a 100 (mayormente porcentajes).  Requiere de las
siguientes
# variables globales:
#
#    mat ARRAY [70] of integer     (valores actuales)
#    titulo,                       (titulo a desplegar)
#    subtitulo,                    (subtitulo a desplegar)
#    leyendax char(80)             (leyenda en eje x)
#    leyenday char(20)             (leyenda en eje y)
#    fecha char(10)                (fecha monitoreada)
#    lectura integer               (ultimo numero de lectura)
```

```
#   grafico char(1)                (caracter ascii para
graficos)
# Cada vez que se tiene un nuevo valor a desplegar, se llama
a esta
# rutina para desplegarlo
#
************************************************************
*****
  DEFINE ren,col,largo,ancho,nummed,i,j,k,offset,dummy,valor
integer
  DEFINE resp,paso char(1)
  DEFINE marcax char(70)
  LET grafico = ascii 219
  LET ren=4                                    #coordenadas del
marco
  LET col=5
  LET largo = 72
  LET ancho = 12
  LET nummed= largo -2                         #calcula
mediciones desplegables
  LET
marcax="....+....|....+....|....+....|....+....|....+....|..
..+....|....+....|"
  if valor < 0
   then
    let valor = 0
  end if
  if lectura = 0 then
    call marco (ren,col,largo,ancho)           #dibuja marco
para despliegue
    for i = 1 to ancho-2
      LET dummy = ren+ancho-i-1
      display i at dummy, 3                    #pone marcas en
eje y
    end for
    display marcax at 16,6                     #marcas eje x
    display "<F1> Ayuda" AT 17,6
    display "<F4> Imprime" AT 17,64
    LET offset = (80 - length(titulo))/2       #calcula offset
para centrar
    display titulo at 2,offset                 #despliega
titulo
    LET offset = (80 - length(subtitulo))/2      #calcula
offset para centrar
    display subtitulo at 3,offset                #despliega
subtitulo
    LET offset = (80 - length(leyendax))/2     #calcula offset
para centrar
    display leyendax at 18,offset              #despliega
leyenda eje x
    display fecha at 3,69 attribute(reverse)#despliega la
fecha monitoreada
    for i = 1 to length(leyenday)              #despliega
leyenda eje y
```

```
      LET paso = leyenday[i,i+1] clipped
      LET dummy = i+3
      display paso at dummy, 1                    # en forma
vertical
    end for
  end if
  LET lectura = lectura + 1                        #incrementa
numero de medicion
  if lectura > nummed then                         #recorrer
monitoreo a izquierda
    for i = 1 to nummed-1
      LET mat [i] = mat [i + 1]                #recorre arreglo a la
izquierda
    end for
    LET mat [nummed] = valor                    #asigna valor
reportado
  else
    LET mat [lectura] = valor                    #asigna valor
reportado
  end if
  for i = 1 to nummed                             #despliega
grafico
    if mat[i] <> 0 then                          #si existe valor
      LET dummy = mat[i] /10                     #de 0-100% a 0-10
      LET k=i + col
      for j = ren + 1 to (ren + ancho - 2 - dummy)
        display " " at j, k
      end for
      for j = (ren + ancho - 1 - dummy) to (ren + ancho - 2)
        display grafico at j, k
      end for
    end if
  end for
END FUNCTION

FUNCTION monitor_help()
    CASE tipo
     WHEN 1 CALL showhelp(1)
     WHEN 2 CALL showhelp(2)
     WHEN 3 CALL showhelp(3)
     WHEN 4 CALL showhelp(4)
     WHEN 5 CALL showhelp(5)
     WHEN 6 CALL showhelp(6)
     WHEN 7 CALL showhelp(7)
     WHEN 8 CALL showhelp(8)
     WHEN 9 CALL showhelp(9)
     WHEN 10 CALL showhelp(10)
     WHEN 11 CALL showhelp(11)
     WHEN 12 CALL showhelp(12)
     WHEN 13 CALL showhelp(13)
     WHEN 14 CALL showhelp(14)
     WHEN 15 CALL showhelp(15)
     WHEN 16 CALL showhelp(16)
     WHEN 17 CALL showhelp(17)
```

```
      WHEN 18 CALL showhelp(18)
      WHEN 19 CALL showhelp(19)
      WHEN 20 CALL showhelp(20)
    END CASE
END FUNCTION
```