2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ACATLAN

"SISTEMA INTEGRAL DE CONTABILIDAD EN UNIX"

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

JAVIER DE LA ROSA MONDRAGON

ASESOR: RUBEN ROMERO RUIZ

UNAM
CAMPUS ACATLÁN

ACATLAN, EDO. DE MEX. 1999

A mi madre ella es
baluarte de mi existencia
por su eje que ha guiado
mis pasos por los desvaríos
de mi vida.

Gracias madre.

Javier De La Rosa Mondragón.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

La contabilidad de un sistema es la manera de proporcionar y clasificar la información del consumo de los recursos generados por la utilización del sistema.

En cualquier sistema operativo, las herramientas de contabilidad forman parte importante en la optimización del rendimiento del sistema, en el caso particular de UNIX, éste ha sido criticado muchas veces por tener un débil sistema contable, si bien UNIX se desarrolló en ambientes académicos informales con pequeños sistemas, en donde no era tan importante llevar un control estricto de la contabilidad, en la actualidad es utilizado también en ambientes más formales y/o compañías burocráticas donde el cobro y la contabilidad del tiempo de conexión del sistema son más importantes.

Cualquiera que este interesado en el rendimiento del sistema necesita saber exactamente lo que el sistema esta haciendo y cómo se *están usando los recursos. Además se debe conocer* qué aplicaciones corren los usuarios, cuánto tiempo del sistema se está consumiendo al correr esas aplicaciones, cuánto tiempo del sistema se esta consumiendo corriendo las utilerías generales, así como, qué usuarios consumen más recursos

La contabilidad de una máquina, finalmente la podemos leer de los archivos de texto generados por esta, quizá si se trata de una sola máquina, no tengamos un gran volumen de archivos por analizar, pero la situación se complica para aquellos laboratorios de cómputo que tienen varias máquinas trabajando con muchos usuarios durante todo el día

Como una solución a lo anterior se creó el "Sistema integral de contabilidad en UNIX" que evalúa el rendimiento de los sistemas UNIX, *permitiendo monitorear y generar reportes* de contabilidad más amigables, del uso de disco, memoria, programas, comandos, por cada uno de los usuarios que utilizan las máquinas integradas

al sistema. La tarea de analizar archivos de texto es remplazada por un análisis de gráficas representativas de los datos de los archivos de texto antes mencionados; por lo que se refiere al problema de una conexión para cada máquina, lo que se propuso fue crear un nuevo servicio de red, el cual estará dedicado exclusivamente a atender solicitudes de requerimientos de archivos de contabilidad, para transferirlos a la máquina donde se este realizando el trabajo. La comunicación entre las máquinas esta basado en el modelo cliente/servidor conteniendo además una interfaz gráfica en Perl/TK, que facilita al Administrador el uso de este sistema.

Este sistema ayudará a la recolección y manipulación de la información generada por la contabilidad de cada una de las máquinas del Laboratorio de Visualización y del Departamento de Supercómputo de la Dirección General de Servicios de Cómputo Académico (DGSCA) de la Universidad Nacional Autónoma de México (UNAM).

Objetivos

El objetivo de la tesis es diseñar un programa que permita analizar de una forma fácil y en conjunto los archivos de la contabilidad en UNIX.

Los objetivos específicos son:

- Terminar SICU en su primera versión.
- Instalarlo y usarlo como parte de las herramientas desarrolladas en el Departamento de Administración de Supercómputo de la DGSCA, UNAM.

Hipótesis

Se instaló el SICU en las máquinas del Departamento de Supercómputo y Laboratorio de Visualización, entonces se incrementó la revisión de la contabilidad, con la ventaja de interpretar de una manera más rápida el conjunto de datos proporcionados por los reportes de las máquinas.

Variable dependiente

SICU, herramienta de análisis de reportes de contabilidad. Básicamente, el sistema está compuesto de dos módulos instalados en sus correspondientes entidades cliente/servidor.

Variable independiente

- Mejoramiento en la medición de los recursos de sistemas UNIX. Debido a la facilidad de interpretación de los reportes gráficos de uso del sistema.
- Optimización de los recursos humanos dedicados a la administración del sistema. La optimización en el análisis de los reportes de contabilidad da lugar a ocupar tiempo en otras tareas de administración igualmente importantes.
- Control de las actividades relacionadas con la administración. En un esquema general de trabajo para la administración de los sistemas, cada tarea realizada sirve de referencia para encontrar el rendimiento óptimo de un sistema, lo cual refleja una buena administración.

Estructura del documento

La tesis consta de cinco capítulos y comienza con la introducción, donde se da una breve razón de por qué es importante contar con herramientas que ayuden a la administración de los sistemas. En el capítulo I se describirá la historia del sistema operativo UNIX, su estructura, así como las características generales más importantes de este sistema que lo hacen portable a una gran variedad de máquinas que van desde computadoras personales hasta supercomputadoras. En el capítulo II se describe la manera de activar la contabilidad y los diferentes archivos generados por esta contabilidad, teniendo como referencia el tipo de sistema operativo UNIX al que se refiere (BSD o System V). En el capítulo III se describen las principales características del Modelo Cliente/Servidor, el cual es el modelo ideal para desarrollar el proyecto SICU (Sistema Integral de Contabilidad en UNIX), teniendo en mente que una máquina proporcionará

la información necesaria que solicite otra. En el capítulo IV, se describirán las necesidades que cubrirá este sistema además de sus ventajas como un sistema que pueda proporcionar información de contabilidad de varias máquinas a través de la red. En el capítulo V se describe el diseño y la implementación del SICU, utilizando las herramientas de programación adecuadas, como los lenguajes PERL/TK que nos proporcionan las librerías, para crear interfacez gráficas adecuadas para las estaciones de trabajo; la implementación y uso del sistema se describirán en este capítulo, como una parte fundamental del desarrollo del trabajo para tener un punto de vista objetivo sobre las conclusiones generales del mismo.

Este trabajo esta dirigido preferentemente para aquellas personas que tienen un conocimiento de computación con nivel de estudios superiores y para aquellas personas que deseen introducirse al sistema UNIX y su contabilidad.

Agradecimientos

A mi madre y mis hermanos, Ofe, Paty, Pablo y Martha, por todo su amor.
A Elia por darle sentido a mi vida
A mis amigas, Yola,Susanita y Paly.
A todos mis amigos, que comparten conmigo tantos recuerdos.
A las autoridades de DGSCA, que con su apoyo y facilidades prestadas han hecho posible la finalización de la primera etapa de este proyecto
A los Ingenieros Ruben Romero, Alejandro Viruega y Beatriz Oropeza, por su tiempo y comentarios para el término de esta tesis.
A todos gracias!.

Convenciones Tipográficas

Al escribir esta tesis se han usado un número de convenciones tipográficas para marcar comandos de unix, variables, etc. mismas que se detallan a continuación.

Fuente Itálica	Se usa para nuevos conceptos y avisos. También se usa para *enfatizar* texto.
Fuente Negrita	Se usa para marcar nombres de maquinas, nombres de archivos, comandos UNIX, para representar interacción con la pantalla.
$	Denota el prompt de Bourne shell.
#	Denota el prompt de superusuario.
%	Denota el prompt de C shell.
$HOME	Es el directorio de trabajo de cada usuario.
~ada	Es el directorio de trabajo de la cuenta de administración.

Capítulo 1

ESTRUCTURA DEL SISTEMA OPERATIVO UNIX.

Este capítulo menciona algunas características importantes del sistema operativo UNIX, que ponen de manifiesto el ambiente de trabajo de este sistema.

1.1 Historia del Sistema Operativo UNIX

UNIX es un sistema operativo desarrollado en los Laboratorios Bell, en New Jersey, Estados Unidos. Los creadores principales fueron Ken Thompson y Dennis Ritchie. En los 60's las compañías AT&T, Honeywell, General Electric y MIT inician un proyecto de información distribuído llamado **Multics**. El objetivo era diseñar un gran sistema multiusuario para proporcionar servicios de cómputo las 24 horas, los 365 días del año y con una arquitectura fácilmente extendible.

En 1969 AT&T decide salirse del proyecto, entonces Ken Thompson y Dennis Ritchie, de está compañía, continuaron con el desarrollo de algunas ideas, dando como resultado la primera versión de UNIX, escrita en el lenguaje ensamblador[1] de una minicomputadora PDP-7, Brian Kernighan sugiere el nombre de UNIX (como un juego de palabras de Multics) para el nuevo sistema.

[1] Lenguaje de bajo nivel

En 1971 Thompson y Ritchie reescriben UNIX para una nueva computadora llamada PDP-11, por otro lado Ritchie también se dedicó a desarrollar el compilador para el lenguaje de programación que acababa de diseñar y al cual lo llamó lenguaje C[2].

En 1973, Ken Thompson y Dennis Ritchie escriben el núcleo de UNIX en el lenguaje C, cambiando así la tradición de escribir sistemas operativos en el lenguaje ensamblador, con ello se logró que UNIX fuera más portátil y fácil de modificar. Tiempo después se concedió el permiso para que algunas instituciones no lucrativas, tuvieran acceso a UNIX en su versión de la PDP-11, que ya era muy popular en universidades e institutos de investigación por lo que marcó una rápida difusión del sistema en todo el mundo. Se realizaron algunas variantes al sistema UNIX hasta llegar a UNIX System V.

En 1977 la Universidad de Berkeley California, comienza a distribuir UNIX con las modificaciones que ellos le habían hecho llamándola "Berkeley Software Distribution", o BSD.

En la actualidad UNIX se considera un estándar para computadoras multiusuario, y ha sido adoptado por una gran cantidad de máquinas. Existen varias versiones comerciales del sistema, como lo son: UNIX III, UNIX V, UNIX BSD, XENIX, etc, pero todas tienen mucho en común. La utilización de UNIX crece día a día, ya no solo en los ambientes científicos y de investigación, sino en aplicaciones comerciales, militares, de manejo de bases de datos, etc.

1.2 Características Generales del Sistema Operativo UNIX

- Es un sistema multiusuario y de multiproceso, cada usuario puede ejecutar varios procesos simultáneamente.
- El sistema UNIX está escrito en un lenguaje de alto nivel, facilitando con esto, el leer, entender, modificar y portar a diferentes tipos de máquinas.
- Tiene una interfaz de usuario simple que tiene el poder de brindar los servicios que los usuarios requieren.

[2]Uno de los lenguajes de programación de medio nivel más usados y populares actualmente

- Proporciona facilidades para construir programas complejos a partir de programas simples.
- Usa un sistema de archivos jerárquico permitiendo un mantenimiento fácil de los archivos y una implementación eficiente.
- Tiene facilidades para redireccionamiento de entrada/salida.
- Proporciona una interfaz consistente a dispositivos periféricos.
- Permite comunicación entre procesos.
- Esconde al usuario la arquitectura de la máquina, haciendo más fácil escribir programas que se ejecuten en diferentes implementaciones de hardware.

1.3 Niveles del Sistema Operativo UNIX

La estructura del sistema operativo UNIX se entiende a través de un diagrama que se compone de diferentes niveles (figura 1.1).

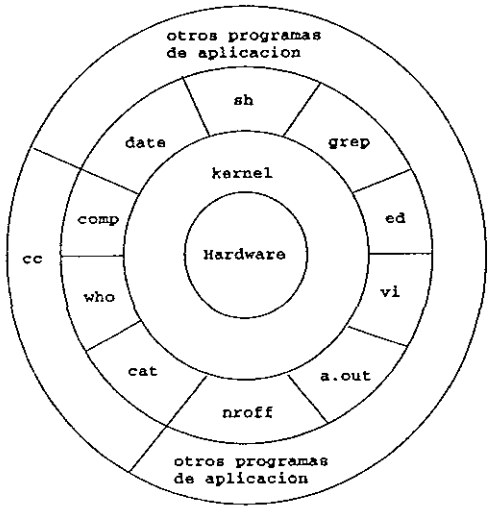

Figura 1.1: Capas del sistema operativo UNIX

El nivel interior es el hardware, este proporciona al sistema los servicios básicos para los programas, interactuando directamente

con la siguiente capa donde se encuentra el sistema operativo. El sistema operativo también es llamado comunmente Kernel y debido a que los programas son independientes del hardware en donde se encuentran funcionando, es relativamente fácil mover estos. Los programas como el *shell* (el programa que nos permite interactuar con el sistema operativo) y editores propios del sistema como *ed* o *vi*[3], se encuentran en el siguiente nivel, estos interactuan con el Kernel por medio de invocaciones definidas como *llamadas al sistema.*

Las llamadas al sistema son funciones escritas en lenguje C que le indican al Kernel qué operaciones requiere el programa que lo invoca e intercambia datos entre el Kernel y él. Varios programas estándar dentro de la configuración son conocidos como comandos[4], pero también los programas creados por los usuarios se encuentran en este nivel, estos programas son llamados *a.out*, que es el nombre estándar de los archivos ejecutables producidos por el compilador C. Otros programas de aplicación pueden construirse en un nivel más alto, como por ejemplo el compilador de C o el compilador de Fortran.

Muchos subsistemas de aplicaciones y programas como el *shell* y los editores, comunmente son llamados como sinónimos del "sistema operativo UNIX", sin embargo, todos ellos utilizan los servicios del Kernel, y podemos disponer de estos, por medio de las llamadas al sistema.

1.4 Modelo del Sistema Operativo UNIX

Los principales componentes del Kernel son el *subsistema de archivos* y el *subsistema de control de procesos.* Por lo que los dos conceptos importantes en el modelo del sistema operativo UNIX son: los archivos y los procesos.

El subsistema de archivos proporciona control sobre los archivos, entre otras cosas reserva espacio para los archivos; administrando el espacio libre, controlando el acceso y recobrando datos para usuarios. El subsistema de archivos accesa a los periféricos usando un mecanismo llamado *buffering*[5]. Este mecanismo regula el flujo de los datos entre el Kernel y dispositivos secundarios de almacena-

[3] Editores de texto en UNIX
[4] Instrucciones para el sistema operativo
[5] Almacenamiento momentaneo

miento. El mecanismo buffering interactua con los manejadores de dispositivos de entrada/salida de bloques, para inicializar transferencias de datos con el Kernel. Los manejadores de dispositivos, son módulos del Kernel que controlan la operación de dispositivos de entrada/salida de bloques.

El subsistema de control de procesos es responsable, por ejemplo, de sincronizar los procesos, de la comunicación entre procesos, del manejo de memoria y el tiempo que asigna el *scheduler*[6] a cada proceso.

El modulo scheduler distribuye el tiempo de CPU[7] para todos los procesos, los cuales se ejecutan uno a la vez, hasta que dejan de consumir CPU mientras esperan un recurso, o cuando exceden el tiempo determinado para ejecutarse. El scheduler entonces elige otro proceso que tenga prioridad mayor entre los procesos que estan listos para ejecutarse, el proceso interrumpido se ejecutará nuevamente, cuando este disponible para hacerlo y su prioridad lo permita

El módulo de memoria maneja y controla la distribución de memoria: si en cualquier momento el sistema no tiene suficiente memoria física para todos los procesos, el Kernel intercambia estos entre la memoria principal y la secundaria, para que todos los procesos obtengan el tiempo razonable para ejecutarse A este tipo de memoria auxiliar se lo conoce como memoria *swap*.

1.5 Perspectiva del Usuario

1.5.1 El Sistema de Archivos

Una de las funciones más importantes de un sistema operativo es proporcionar un sistema de archivos para manejo de información. El sistema de archivos de UNIX se caracteriza por lo siguiente:

- Tiene una estructura jerárquica.
- Manejo de archivos consistente.
- Habilidad de crear y borrar archivos.
- Crecimiento dinámico de archivos.

[6] Despachador
[7] Unidad Central de Proceso

- Protección de archivos.
- Los periféricos (terminales, unidades de cinta, disco) son manejados como archivos.

El sistema de archivos esta organizado como un árbol (figura 1.2).

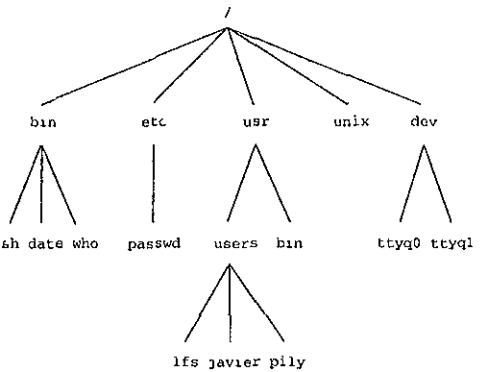

Figura 1.2: Estructura de archivos

El árbol contiene un nodo llamado *root*[8] (/) que es el directorio inicial y a partir de éste se encuentran otros nodos que pueden ser tanto archivos como directorios y estos últimos pueden contener a su vez directorios o subdirectorios. Debido a esta filosofía, el manejo del sistema es por medio de pocas órdenes.

Los diferentes tipos de archivos que el sistema reconoce son:

Archivos Normales	Son archivos que contienen datos. Incluyen archivos de texto, programas fuente, ejecutables, imágenes etc.
Directorios	Contienen referencias a los archivos que se encuentran dentro de ellos.
Dispositivos de caracteres	Todo tipo de periféricos que intercambian datos un *byte* a la vez. Se incluyen terminales, impresoras, lectores ópticos etc.

[8] *Nodo raiz*

Dispositivos de bloques — Son periféricos que intercambian datos de varios bytes, normalmente 512. Se incluyen discos duros, discos flexibles, unidades de cinta, etc.

Cada archivo en el sistema UNIX tiene un *inodo* único, que básicamente es un registro por medio del cúal el sistema de archivos identifica a todos los archivos existentes. El inodo contiene la información necesaria para que un proceso pueda accesar al archivo, permisos de acceso, tamaño del archivo, y la localización de los datos en el sistema de archivos. Los procesos accesan a los archivos por un grupo bien definido de *llamadas al sistema*[9]. El nombre de un archivo esta dado por una dirección llamada *trayectoria*[10], está describe en donde se encuentra localizado el archivo dentro de la jerarquía del sistema de archivos. Una *trayectoria* es una secuencia de nombres separados por el caracter slash (/) Cada nombre unicamente especifica a un archivo, y el Kernel convierte el nombre de la trayectoria en el inodo del archivo.

Los sistemas multiusuario necesitan ofrecer protección para impedir que algunos usuarios lean y/o borren archivos de otros usuarios sin la debida autorización. UNIX tiene un esquema elaborado de propietarios de archivos y permisos, a través del cual los usuarios y los grupos de usuarios pueden proteger sus archivos. Los permisos para accesar un archivo esta controlado por el *permiso de acceso* esté puede controlar independientemente los permisos de lectura, escritura o ejecución (r,w,x) de un archivo, para las tres clases de usuarios: el dueño, el grupo, y los demás (u,g,o).

El sistema operativo UNIX accesa a los periféricos de igual forma como lo hace con los archivo regulares. Esto es debido a que los nombres de los periféricos y los nombres de los archivos regulares son similares, además las condiciones de trabajo también son similares, muchos programas no tienen porque conocer internamente el tipo de archivo o periférico que ellos manipulan.

1.5.2 El Ambiente de Procesamiento

Un programa es un archivo ejecutable y un proceso es una instancia de un programa en ejecución. Los procesos pueden ejecutar-

[9] Comunicaciones directas con el Kernel
[10] El nombre

se simultáneamente sobre el sistema operativo UNIX (esta característica es llamada multitarea) muchas instancias pueden existir simultáneamente en el sistema. Varias llamadas al sistema como por ejemplo; el *fork()* o el *exec()*[11] permiten a los procesos crear nuevos procesos, terminar procesos, sincronizar estados de ejecución de procesos y controlar determinada reacción a diferentes eventos, los procesos se ejecutan independientemente uno del otro.

Generalmente, las llamadas al sistema permiten al usuario escribir programas que hacen sofisticadas operaciones y como resultado, el Kernel de UNIX no contiene demasiadas funciones, que son parte del "Kernel" de otros sistemas. El principal ejemplo de tales programas que através de las llamadas al sistema se comunican con el Kernel, es el *shell*, este programa interpreta como un comando la primera palabra introducida en la línea de comandos. El *shell* permite tres tipos de comandos.

- Primero, un comando puede ser un archivo ejecutable que contiene código objeto, producido por la compilación de código fuente (un programa en C por ejemplo).
- Segundo, un comando puede ser un archivo ejecutable que contiene una secuencia de líneas de comandos.
- Finalmente un comando, puede ser un *script shell* interno, incluyendo comandos para ciclos (`for -- done`, `while -- done`) y comandos para condiciones (`if--then--else--fi`).

El *shell* usualmente ejecuta un comando síncronamente, esperando que este comando termine para poder leer el siguiente, sin embargo, también nos permite ejecución asincrona, esto es, mientras se lee el siguiente comando, se ejecuta el anterior y no espera a que termine el otro. Los comandos ejecutados asíncronamente se dice que se ejecutan en *background*[12]. Una manera de mandar ejecutar algún comando en background es utilzando el caracter & después del comando, por ejemplo, tecleando el comando:

% who &

El sistema ejecuta el programa */bin/who*[13] en background y el *shell* se encuentra listo para aceptar otro comando. A causa de que

[11] Llamadas al sistema para ejecutar otros procesos

[12] También se dice que se ejecuta en forma no interactiva

[13] El directorio "/bin" contiene muchos de los comandos usuales y esté es incluido en la secuencia de busquedas del shell (PATH)

el *shell* es un programa, y no una parte del Kernel, éste es fácilmente modificable, y se puede adaptar a cualquier ambiente en particular.

1.5.3 Construcción de Primitivas de Bloque

La filosofía de UNIX proporciona primitivas de operación del sistema, permitiendo a los usuarios escribir pequeños programas modulares que pueden ser usados como bloques, para la construcción de programas más complejos. Una de estas primitivas visibles para los usuarios del *shell* es la capacidad de redireccionamiento de entrada y salida (E/S). Los procesos convencionalmente leen del archivo de *entrada estándar* (por default el teclado), escriben en el archivo de *salida estándar* (por default la pantalla) y escriben los mensajes de error al archivo *estándar de errores* (por default la pantalla). La ejecución de los procesos en una terminal, típicamente utiliza estos tres archivos, pero cada uno puede "redireccionarse" independientemente, con los siguientes caracteres .

">" salida "<" entrada "2>" error[14]

La segunda primitiva de construcción de bloques, es el *pipe* (|). Este es un mecanismo que permite a los datos ser transmitidos entre el proceso que lee y el proceso que escribe. Los procesos pueden redireccionar su salida a algún *pipe*. Los datos que el primer proceso escribe dentro del pipe, es la entrada para el segundo proceso. El segundo proceso también puede redireccionar su salida y así varias veces, dependiendo de las necesidades del programador. El proceso no necesita conocer el tipo de archivo de salida.

1.5.4 Funciones del Sistema Operativo UNIX

El Kernel realiza varias operaciones a favor de los procesos de los usuarios. Entre los servicios proporcionados por el Kernel están:

- Control de la ejecución de los procesos para permitir su creación, terminación, suspensión y comunicación
- Programa los tiempos de procesos para su ejecución sobre el CPU. Los procesos comparten el CPU de la siguiente manera: El CPU ejecuta un proceso, el Kernel suspende éste cuando su tiempo se agota, y el Kernel programa otro proceso listo

[14] salida de error en Bourne shell

para ser ejecutado, después el Kernel reprograma el proceso suspendido.

- Asigna memoria principal para la ejecución de un proceso. El Kernel libera memoria principal escribiendo temporalmente un proceso en memoria secundaria llamada *swap*. Si el Kernel escribe procesos completamente en el swap, la implementación de UNIX es llamada un sistema *swapping*.
- Permite a los procesos acceso controlado a los dispositivos de periféricos (terminales, manejadores de cinta, manejadores de disco, dispositivos de red). [Bac86]

1.6 Acceso al Sistema Operativo UNIX

En UNIX, existe una cuenta de acceso privilegiada, llamada *root*, esta cuenta es otorgada generalmente, a una persona que es responsable del sistema y generalmente se le nombra *administrador*.

La persona encargada del sistema, crea las claves de los usuarios, otorgandoles un nombre (*login name*), es decir cada usuario tiene un nombre único.

Lo primero que se debe hacer es entrar al sistema operativo UNIX. El propósito de entrar al sistema se resume en dos aspectos: dejar que el sistema verifique el permiso del usuario para que lo pueda usar y dejarle definir su ambiente de trabajo. El primer signo que la terminal exhibe para indicar que espera una entrada de usuario al sistema es el indicador:

```
login:
```

El usuario debe digitar su clave de entrada (login name), después el sistema pide una contraseña por medio del indicador:

```
passwd:
```

Por motivos de seguridad, la contraseña no aparece desplegada en la pantalla cuando se teclea. UNIX comprueba si la clave *login* y el *passwd* son correctos, para permitir la entrada. De lo contrario, marca un error y nuevamente aparece el indicador *login:*. Para salir del sistema es necesario teclear el comando *logout* o *exit*.

Dentro del sistema cada usuario esta registrado en el archivo */etc/passwd* y cada línea del archivo esta separada por siete campos,

siendo los dos puntos (:) el separador de cada uno de ellos e indican lo siguiente:

nombre	El *login name* asignado al usuario. Generalmente son nombres que pueden recordarse fácilmente o que identifiquen plenamente al usuario.
passwd-codificado	El *passwd* del usuario codificado. Esto es por razones de seguridad, ya que de no ser así, todos los passwd se podrían leer de este archivo.
UID	Es el número de identificación del usuario. Cada usuario tiene un UID único, en realidad este es el identificador que utiliza el sistema para distinguir a los usuarios.
GID	Determina a que grupo primario pertenece el usuario. Este número es usualmente el número identificador de grupo asignado en el archivo */etc/group* (expuesto más adelante).
información-usuario	Usualmente contiene el nombre completo del usuario, su oficina y teléfono.
directorio-de trabajo	El directorio personal (HOME) del usuario. Cuando ingresa el usuario al sistema este es su directorio inicial de trabajo
shell	El programa que UNIX usará como interprete para el usuario. UNIX automáticamente ejecuta este programa. Normalmente se usa alguno como el */bin/sh* (Bourne Shell), */bin/csh* (C shell) o */bin/ksh* (Korn shell).

Por ejemplo, parte del contenido del archivo */etc/passwd* podría ser:

```
root m34zHTkGk6LHU:0:1:System PRIVILEGED Account:/:/usr/local/bin/csh
adm x:5:3:Accounting Files Owner:/var/adm:/bin/sh
```

Los grupos en UNIX se definen para proporcionar la capacidad de que un conjunto de usuarios compartan archivos y otros recursos del sistema, así cada usuario tiene definido un grupo (GID) al que pertenece. Los diferentes grupos del sistema están definidos en el

archivo */etc/group*, cada línea del archivo está separada por cuatro campos, siendo los dos puntos (:) el separador de campos de cada uno de ellos e indican lo siguiente:

nombre-grupo — Es el nombre que identifica al grupo. Cada nombre puede ser representativo para las tareas que realice el grupo.

El segundo campo facilita la compatibilidad de las versiones UNIX. Este campo anteriormente se utilizaba y ahora sólo contiene un asterisco.

GID — Es el número identificador de grupo.

usuarios-adicionales — Este campo tiene una lista de los usuarios y grupos secundarios que podrían tener acceso a los archivos del grupo, adicionalmente a aquellos inicializados en el grupo en virtud del archivo */etc/passwd*. Los nombres de los usuarios en la lista deben estar separados por una (,).

Por ejemplo, parte del contenido del archivo */etc/group* podría ser:

```
system:*:0:root
daemon:*:1:daemon
cray:*:303:
cursos:*:304:
```

Como se mencionó anteriormente, parte de las funciones del sistema UNIX, es administrar los recursos de la máquina, de tal forma que varios usuarios puedan compartirla. Para hacer esto, el sistema UNIX mantiene un entorno separado para cada usuario. Cada usuario que entra al sistema tendrá definidas las variables de ambiente del sistema, éstas son una parte importante del entorno de UNIX , algunas de ellas son:

PATH — Define las trayectorias de búsqueda de los comandos de los usuarios.

HOME — Contiene el nombre del directorio de trabajo del usuario, dentro del *csh* también el caracter (~) se refiere a esté directorio. por ejemplo *~adm* se refiere al directorio personal del administrador del sistema.

TERM Contiene el nombre del tipo de terminal que se esta utilizando. [JPL97]

Capítulo 2

CONTABILIDAD EN SISTEMAS UNIX

Este capítulo explica la manera en que UNIX prepara los diferentes tipos de reportes de contabilidad, describe los pasos para activar y desactivar la misma y hace notar las diferencias entre la contabilidad de System V y BSD

2.1 ¿Por qué Contabilizar los Recursos?

En general, UNIX proporciona procesos de contabilidad basados en los usuarios, la operación del sistema proporciona datos de su consumo a través de registros estadísticos de cada proceso que se ejecuta, incluyendo el UID del usuario, de cada uno de los procesos Los registros almacenan la imagen (nombre) de los comandos que fueron ejecutados por el proceso y los recursos del sistema (tales como memoria, tiempo de CPU y operaciones de E/S) que fueron utilizados

El sistema contable esta diseñado para llevar un control del uso de los recursos del sistema y tiene una aplicación práctica principalmente en aquellos usuarios que están pagando por dichos recursos. La recolección de datos también puede ser utilizada por algún tipo de sistema que se encargue de monitorear el rendimiento.

Una vez conociendo qué programas estan consumiendo más tiempo y memoria del sistema, se puede comenzar a desarrollar estrategias al respecto por ejemplo:

- Si los programas más consumibles fueron desarrollados localmente, se puede sugerir a los desarrolladores que optimicen sus algoritmos.

- Si los programas más importantes usan gran parte de la memoria, se pueden tomar medidas para procurar disponibilidad de memoria.

- Si algún sistema ejecuta muchos programas que cargan el sistema de Entrada/Salida, se pueden buscar soluciones para optimizar el disco.

Cuando se trata con problemas globales es difícil resolverlos, sin la ayuda de buenos resúmenes globales.

2.2 Archivos de Contabilidad

Los sistemas de contabilidad bajo las versiones de BSD y System V son diferentes, pero ambos estan basadas en datos iniciales similares, por lo tanto, los tipos de información, que se puedan generar de ellos son esencialmente iguales, aunque los mecanismos de salida y formato sean diferentes.

En la versión BSD la contabilidad esta habilitada en los nuevos sistemas pero se puede deshabilitar si se desea. Bajo la versión de System V, inicialmente la contabilidad esta deshabilitada y debe ser activada por el administrador del sistema.

Cuando esta activo el sistema de contabilidad, el Kernel de UNIX escribe un registro a un archivo binario por cada proceso que termina. Estos registros (con variaciones pequeñas entre las dos versiones) son almacenados en un archivo con el nombre de **acct** o **pacct** de acuerdo a la tabla 2.1 donde se puede observar la trayectoria completa de la ubicación del archivo, dentro del directorio HOME del usuario *adm* (convencionalmente UID 4)[Fri95], por ejemplo:

BSD	
BSD	*/usr/adm/acct*
SunOs	*/var/adm/acct*
System V	
V.3	*/usr/adm/pacct*
V.4	*/var/adm/pacct*
AIX 3.1	*/usr/adm/pacct*
XENIX	*/usr/adm/pacct*
IRIX	*/var/adm/pacct*

Tabla 2.1: Archivos de registro de los procesos

Los registros escritos por el sistema de contabilidad contienen los siguientes datos de los procesos:

- Nombre de la imagen.
- Tiempo de CPU.
- Tiempo transcurrido
- Tiempo de inicio del proceso.
- Memoria utilizada.
- Número de caracteres leídos y escritos.
- Número de bloques de E/S leídos y escritos en disco.
- Terminal inicial TTY.
- Varias banderas asociadas con los procesos, incluyendo estado de salida

Otros datos de contabilidad están almacenados en los siguientes archivos

/etc/utmp Un archivo binario de bitácora que contiene datos sobre cada usuario conectado.

/adm/wtmp Un archivo binario de bitácora que registra cada entrada (*login*) y salida (*logout*) que se efectúa en el sistema.

/adm/lastlog Un base de datos que contiene los datos del último acceso de cada usuario.

2.2.1 Contabilidad Estilo BSD

2.2.2 Archivos

La contabilidad de BSD utiliza los siguientes archivos globales adicionales:

~*adm/savacct* Archivo global estándar. Los registros de contabilidad se unen dentro de este archivo por medio de la utilería de contabilidad `sa`.

~*adm/usracct* Archivo global basado en los usuarios. Este archivo es actualizado y accesado por medio de la utilería de contabilidad `sa -m`.

2.2.3 Habilitar la Contabilidad

En los sistemas basados en BSD, por default, la contabilidad esta habilitada, esto significa que ambas de las siguientes condiciones se pueden llevar a cabo.

- El comando `accton` puede ser ejecutado desde que el sistema fue inicializado por última vez.
- El archivo de contabilidad especificado por el comando `accton` existe.

El comando `accton` está comunmente incluido en el archivo de inicialización `/etc/rc`, que se utiliza para integrar los servicios necesarios del sistema. Este especifica al archivo `~/adm/acct` como el archivo de contabilidad. Las líneas relevantes del archivo `/etc/rc` para la contabilidad son:

```
if [ -f /usr/adm/acct ]; then
    accton  /usr/adm/acct; echo -n 'accounting'  >/dev/console
```

El comando `accton` es activado cuando se especifica un archivo contable como su argumento. Una vez que el comando es ejecutado, la contabilidad se registrará automáticamente sobre el archivo de contabilidad.

2.2.4 Opciones de Administración de la Contabilidad

Dada por default la instalación de la contabilidad, los administradores del sistema, tienen varias opciones.

- *Dejar las cosas como están.* En muchos lugares las instalaciones por default se adaptan a sus necesidades perfectamente. La contabilidad en estos casos, estará activada y mantenida automáticamente sin la intervención de nadie, pero el procesamiento y crecimiento del archivo es responsabilidad del administrador.

- *Deshabilitar la contabilidad.* Para realizar esto se ejecuta el comando `accton` sin argumentos.

```
# accton
```

  La contabilidad ahora cesa, y no se escriben más registros en el archivo de contabilidad, hasta que éste sea reiniciado. En consecuencia se necesita comentar lo referente al comando `accton` en el archivo `/etc/rc`. Si se desea, el archivo contable puede ser borrado después de deshabilitar la contabilidad, sin embargo hay dos puntos importantes referente a esto, para tomarse en cuenta.

  Primero, el comando `accton` sin ningún argumento es solamente el camino para deshabilitar la contabilidad. Si se borra el archivo de contabilidad sin deshabiltar la misma, entonces el sistema todavía intentará guardar y escribir los registros correspondientes, pero no tendrá el archivo donde se escriban éstos.

  Segundo, cuando la contabilidad se activa, el archivo contable necesita ser creado a mano, ya que el software de contabilidad no lo genera automáticamente si éste no existe. Se puede utilizar el comando `touch`[14] para crear el archivo antes de integrar la contabilidad.

```
# touch /usr/adm/acct
# accton /usr/adm/acct
```

[14] Comando de UNIX para crear archivos regulares

- *Cambiar el archivo de contabilidad.* Esto se puede hacer deshabilitando la contabilidad y después reestableciendola con un archivo contable diferente, por ejemplo:

```
# accton
# touch /sysadmin/accounts
# accton /sysadmin/accounts
```

  En el ejemplo anterior los comandos reemplazarán el archivo de contabilidad por default por un archivo que el administardor elija.

- *Poner todo o parte del sistema de contabilidad, sobre una base manual.* El administrador puede decidir manejar todas o algunas de las funciones contables manualmente. Si esta opción es integrada, entonces las líneas relevantes en `/etc/rc` deberán ser comentadas[15], y el administrador será responsable de inicializar la contabilidad cuando lo deseé.

- *Automatizar completamente la contabilidad.* Se pueden escribir scripts sencillos, para procesar los datos iniciales de la contabilidad dentro de archivos globales y aún para producir archivos de contabilidad deseados.

2.2.5 Reporte del Consumo de Recursos con Base en su Imagen

La utilería `sa` produce reportes basados en el proceso que fue ejecutado. Este comando entre otras cosas, organiza y presenta estadísticas por cada nombre de imagen. Sin ninguna opción `sa` produce algo similar a:

```
# sa
  405    42168.17re    30489.35cp       105avio     16376k   pine
   88    41993.63re      262.81cp       196avio       421k   Xwst3d
 1208     1031.63re      210.92cp        69avio        72k   virtex
  341    27445.43re       90.49cp        34avio       161k   dxterm
```

En la salida por default, el nombre de la imagen, aparece en la última columna. Los campos numéricos en la salida de `sa`, son identificados por sus sufijos de acuerdo a la tabla 2.2.

[15] Para comentar una línea se inserta el caracter # al inicio de la línea deseada

SUFIJO	DATO
ninguno	Número de veces en que fue ejecutado.
cp,cpu	Tiempo de CPU (sistema + usuario) en minutos
re	Tiempo transcurrido en minutos
avio	Promedio de operaciones E/S por ejecución.
k	Promedio de memoria usada sobre tiempo de CPU.
k*sec	Almacenaje integral de CPU en kilo-core segundos.
tio	Total de operaciones E/S por toda la ejecución
s	Tiempo de CPU del sistema en minutos.
u	Tiempo de CPU de usuario en minutos.

Tabla 2.2. Sufijos para la salida del comando sa

No todos los datos aparecen en cada reporte, los primeros cinco datos aparecen en la salida por default y los otros datos se muestran en reportes generados con diferentes opciones.

La salida de sa puede ser ordenada con base en las siguientes opciones.

OPCION	ORDENADO POR
-b	Promedio total de tiempo CPU por ejecución.
-d	Promedio de número de operaciones de E/S de disco.
-D	Número total de operaciones de E/S de disco.
-k	Promedio de tiempo de CPU de memoria usada.
-K	Almacenamiento integral CPU.
-n	Número de ejecuciones.

Tabla 2.3: Opciones de ordenamiento del comando sa

La opción *-D* produce un reporte que contiene el total de operaciones de E/S usadas por los comandos, las líneas están ordenadas de acuerdo a sus totales, por ejemplo:

```
# sa -D
  43132    6826.17re      78.12cp    406131tio        44k  sendmail
   1402  100072.55re      57.21cp    242804tio        47k  elm
   4351    2679.13re      13.69cp    120989tio        11k  mail
   1214    1035.72re     213.81cp     83508tio        73k  virtex
```

Alternativamente, el archivo global puede especificarse con las opciones -S y -U ambas podrían ser seguidas por una dirección. -S indica una alternativa para `/usr/adm/savacct`, y -U especifica una alternativa para el archivo global por-usuario `/usr/adm/ursacct`.

2.2.6 Reporte Tiempo de Conexión: ac

La utilización de ac reporta el tiempo de conexión de los usuarios. Este comando obtiene información del archivo `/usr/adm/wtmp`, el cual contiene registros de las entradas y salidas al sistema por todos y de cada uno de los usuarios. Sin ninguna opción **ac** despliega el total del tiempo de conexión (en horas, por todos los usuarios) por ejemplo:

```
# ac

 total  5501.06
```

El comando puede ser llamado para uno o más nombres de usuarios, desplegando el total de todos ellos, por ejemplo:

```
# ac chavez  wang  fine

      total   1588.65
```

La opción -p muestra el tiempo de conexión (en horas) por usuario, por ejemplo:

```
# ac -p chavez wang fine
       chavez    685.25
       wang      170.77
       fine      732.78
       total    1588.79

#  ac -p muestra todos los usuarios.
```

2.3 Contabilidad en System V

"La contabilidad de System V es mucho más elaborada que en las versiones de BSD. Es un sistema complejo de comandos, scripts

de shell y programas en C, llamados en grandes secuencias, todo diseñado para ser totalmente automático y requerir pequeñas intervenciones o incluso ninguna".[Fri95]

2.3.1 Directorios y Archivos de Contabilidad

El Kernel escribe un registro por cada proceso que termina, en el archivo llamado `/var/adm/pacct`, este archivo contiene un registro por proceso terminado, organizado de acuerdo al formato definido en `/usr/include/sys/acct.h`

El directorio `/usr/lib/acct` contiene los programas y shell scripts necesarios para ejecutar el sistema de contabilidad. Los procesos utilizados por el sistema de contabilidad utilizan la cuenta *adm* para realizar ciertas tareas

El directorio HOME del usuario *adm* contiene un subdirectorio llamado `acct`, el cual contine tres subdirectorios que son:

adm/acct/fiscal Contiene archivos globales periódicos creados por medio del comando `monacct`

adm/acct/nite Archivos binarios globales, registros de contabilidad de los procesos, registros de contabilidad de disco y estado, bitácora de errores. Estos archivos son actualizados diariamenmte por el comando `runacct`.

adm/acct/sum Archivos globales diarios actualizados por el comando `runacct`.

Adicionalmente al archivo `pacct` descrito anteriormente, existen varios archivos iniciales de datos

adm/wtmp Registro de las conexiones realizadas al sistema.

adm/acct/nite/diskacct Uso de disco.

adm/fee Con el comando `chargefee` el administrador puede adicionar a cada usuario, registros por servicios especiales no contemplados por el sistema, los registros adicionales serán automáticamente incorporados dentro del sistema contable.

El flujo de la información, en el sistema de contabilidad de System V, (figura 2.1), comienza con los archivos de contabilidad iniciales, cuyos datos se generan a través del sistema operativo y el uso de comandos, después estos archivos son procesados por una serie de utilerías, produciendo varios archivos globales binarios y finalmente se generan los reportes de tipo ASCII, apropiados para el uso del administrador del sistema.

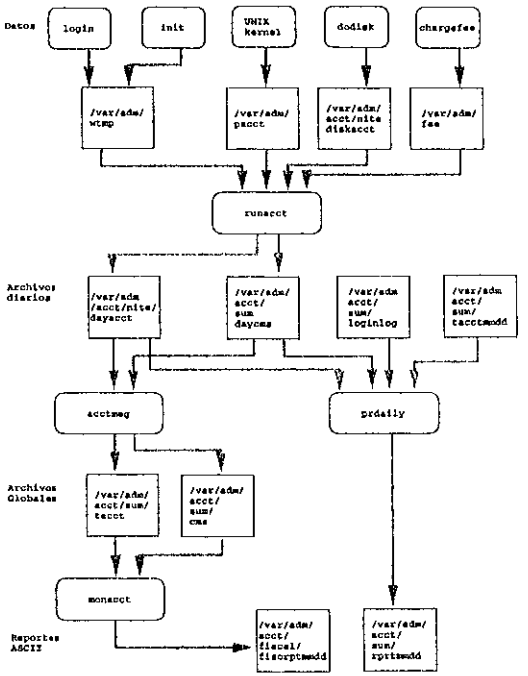

Figura 2.1: Contabilidad en System V

2.3.2 Activando la Contabilidad

La contabilidad inicialmente esta inactiva en los sistemas System V. Los siguientes pasos son necesarios para habilitarla. [Nem95]

- Entrar al sistema con la cuenta de *root*

- Utilizar el comando:

```
# chconfig acct on
```

- Utilizar el comando:

```
# /usr/lib/acct/startup
```

Una vez que se tiene habilitada la opción del sistema de contabilidad, todos los archivos y líneas de comandos para su implementación, tienen que estar correctamente. Algunos puntos importantes son los siguientes:

1. En el archivo `/etc/init.d/acct` debe contener entre otras líneas lo siguiente:

```
/usr/lib/acct/startup
/usr/lib/acct/shutacct
```

   La primera línea inicia la contabilidad durante el proceso de inicializar el sistema, la segunda línea detiene la contabilidad antes de finalizar el sistema.

2. Las siguientes líneas (o algunas similares) deben existir en el archivo `/var/spool/cron/crontabs/adm/`:

```
# controlar el tama\~no del archivo  pacct  (cada 5 minutos)
5 * * * 1-6       /usr/lib/acct/ckpacct
# procesos de contabilidad iniciales (4:00 am diariamente)
0 4 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
```

   El comando `ckpacct` verifica el tamaño de `/var/adm/pacct`, si este archivo crece más de 1000 bloques (cada bloque es igual a 512 kb). `ckpacct` ejecuta el comando `turnacct` con el argumento `switch`. Esto hace que `turnacct` respalde el archivo `/var/adm/pacct` e inicie un nuevo archivo `pacct` vacio.

   El comando `runacct`, es el script principal de procesamiento de contabilidad diaria. Este normalmente es ejecutado a través del *cron*[16] de *adm* en horas de poco uso del sistema, `runacct`

[16] Sistema para ejecutar procesos en un tiempo específico

procesa archivos de contabilidad de conexión, carga, disco y procesos, también realiza archivos diarios y globales acumulativos para ser usados por ejemplo por el comando `prdaily` (para imprimir archivos de contabilidad en la impresora).

3. Debe existir la siguiente línea en el archivo `/var/spool/cron/crontabs/root`:

```
# generar el uso del disco
0  2  * * 4  /usr/lib/acct/dodisk
```

El comando `dodisk`, proporciona un reporte sobre el uso del disco, escribiendo registros en `/var/adm/acct/nite/disktacct` de acuerdo con el formato descrito en `tacct.h`.

4. Para agrupar mensualmente los datos contables, la siguiente línea en `/usr/spool/cron/crontabs/adm`, permite al comando `monacct`, depurar los reportes diarios y los incluye en un reporte mensual, dentro del directorio `/var/adm/acct/fiscal` sería:

```
# generar reportes mensuales (5:00 am en el primer dia de mes)
0  5 1 * *  /usr/lib/acct/monacct
```

Se utiliza la fecha del mes para el sufijo del nombre del archivo.

5. Verificar que el usuario *adm* tenga permisos sobre los archivos que utiliza el sistema de contabilidad, también verificar que la variable de ambiente PATH en el archivo `/var/adm/.profile` contenga:

```
PATH=/usr/lib/acct:/bin:/usr/bin .
```

2.3.3 Desactivando la Contabilidad

Los siguientes pasos son necesarios para deshabilitarla.

- Entrar al sistema con la cuenta de *root*
- Utilizar el comando:

```
# chconfig acct off
```

- Utilizar el comando:

```
# /usr/lib/acct/shutacct
```

Esto hace que el Kernel no escriba más información en el archivo `/var/adm/pacct`

2.3.4 Reportes de Contabilidad

El Reporte diario y mensual de comandos muestra el consumo de los recursos del sistema hecho por cada comando, incluyendo el tiempo transcurrido por cada comando que fue utilizado, el total de tiempo de CPU, memoria utilizada y consumo de transacciones de E/S. El Reporte de actividad por terminal muestra el porcentaje del tiempo que estuvo cada terminal en uso del total de tiempo en el periodo de contabilidad, el total de tiempo acumulado de cada terminal y el número de distintas sesiones. La última parte del reporte muestra la fecha del último acceso de cada UID definido en el archivo `/etc/passwd`.

Diariamente los reportes de contabilidad son almacenados en archivos llamados:

```
~adm/acct/sum/rprtmmdd
```

donde *mm* y *dd* son el mes y el día respectivamente. Cada archivo consta de cinco reportes referentes a lo siguiente.

- Uso por usuario.
- Ultima conexión por cada usuario.
- Uso de comandos para el día y mes previo.
- Actividad por terminal.

Aquí tenemos parte de un reporte que nos muestra el consumo de los usuarios:

```
Jan 23 23:00 1997  DAILY USAGE REPORT FOR pulque Page 1

     LOGIN    CPU (MINS)    KCORE-MINS    CONNECT (MINS)   DISK    #OF   #OF  # DISK  FEE
UID  NAME     PRIME NPRIME PRIME NPRIME  PRIME  NPRIME   BLOCKS PROCS SESS SAMPLES
0    TOTAL    33    414    17098 216293  11890    7981   885632 27888  13     28     0
0    root     17    28       482   2126      0       0   672903  1713   0      1     0
110  martha   0     0          0      0      0       0     6001     0   0      1     0
```

A continuación se explica el significado de cada columna:

UID — El UID del usuario

LOGIN NAME — Clave del usuario.

CPU	El total de minutos, que los procesos del usuario utilizaron la Unidad Central de Procesamiento.
KCORE-MINS	Una medida acumulativa de la cantidad de memoria que un proceso utiliza mientras se ejecuta. Es una indicación de cuánta memoria consumen los procesos de cada usuario.
CONNECT	Tiempo total de conexión
DISK BLOCKS	Promedio total de uso del espacio en disco por el usuario.
#PROCS	Total de procesos pertenecientes al usuario.
#SESS	Número distintivo de sesión.
#DISK SAMPLES	Número de veces que fue utilizado `dodisk` durante un periodo de contabilidad.
FEE	Cantidad adicional de cosumo del sistema, introducidas con el comando `chargefee`.

Las horas de uso correspondientes a PRIME (días hábiles) y NPRIME (días no hábiles), están definidas en el archivo llamado `/usr/lib/acct/holidays`, que el administrador puede configurar, dependiendo de las necesidades del lugar de trabajo. Lo siguiente es un ejemplo de lo que puede contener este archivo:

```
* Prime/Nonprime Table for  Accounting System
* Curr  Prime   Non-Prime
* Year  Start   Start
  1997  1000    2200
* Day of        Calendar        Company
* Year          Date            Holiday

  1/1           Jan 1           Vacaciones UNAM
  1/2           Jan 2           Vacaciones UNAM
  1/3           Jan 3           Vacaciones UNAM
  1/4           Jan 4           Vacaciones UNAM
  1/5           Jan 5           Vacaciones UNAM
  2/5           Feb 5           Const. de 1917
  3/21          Mar 21          Nat. Benito Juarez
```

Las líneas que inician con un asterisco son comentarios, la línea que nos describe el año en curso, también contiene el periodo de horas de trabajo durante el día, las siguientes líneas son los días que el calendario marca como no laborables.

Los recursos usados durante las horas hábiles y no hábiles, son totalmente separadas por el sistema de contabilidad.

La parte correspondiente al reporte del último acceso al sistema por cada usuario, es similar a:

```
Oct 15 04:00 1997  LAST LOGIN Page 1


00-00-00  adm          00-00-00  noaccess     00-00-00  server
00-00-00  aplica       00-00-00  nobody       00-00-00  smtp
00-00-00  bin          00-00-00  nobody4      00-00-00  sus
00-00-00  daemon       00-00-00  nuucp        00-00-00  sys
00-00-00  lfs          00-00-00  raul         97-10-15  diana
00-00-00  listen       00-00-00  root         97-10-15  jrm
```

Donde se muestra la clave de usuario una fecha correspondiente al último acceso realizado.

A continuación se muestra la parte correspondiente al reporte del uso del sistema por cada comando procesado.

```
                                    TOTAL COMMAND SUMMARY
COMMAND    NUMBER    TOTAL     TOTAL     TOTAL     MEAN   MEAN     HOG    CHARS     BLOCKS
NAME        CMDS   KCOREMIN  CPU-MIN   REAL-MIN  SIZE-K CPU-MIN  FACTOR TRNSFD    READ

TOTALS     10549  743855.12   155.27  45738.59  4790.83   0.01    0.00  119576064  111513
netscape       8  574508.75    71.47    734.23  8038.42   8.93    0.10  717425664    8867
java         240  102349.48    22.64     69.29  4520.37   0.09    0.33   46210304   18243
medidas       38   15639.54    25.16    211.17   621.63   0.66    0.12  416966400      57
webmagic       5    7539.75     2.22    595.41  3399.35   0.44    0.00   16911360     821
```

Cada columna se explica a continuación:

COMMAND NAME Es el nombre del comando que fue ejecutado.

NUMBER CMDS El número total de veces que se utilizó el comando.

TOTAL KCOREMIN El total acumulado de la cantidad de segmentos de kilobytes de memoria utilizada por el proceso en cada minuto de ejecución.

TOTAL CPU-MIN El total de minutos de procesamiento que el programa tiene acumulado.

TOTAL REAL-MIN El total de minutos que este programa tiene acumulado, es el total de tiempo que un programa esta inclusive esperando a que el usuario interactúe con el.

MEAN SIZE-K Es la media de TOTAL KCOREMIN sobre el número de veces que se ejecutó el comando.

MEAN CPU-MIN Es la media entre el NUMBER CMDS y el TOTAL CPU-MIN.

HOG FACTOR Es una medida relativa del tiempo total disponible de CPU consumidos por los procesos durante su ejecución.

CHARS TRNSFD Es el total del número de caracteres transferidos por las llamadas al sistema *read write*.

BLOCKS READ Es el número total de bloques físicos que son leídos y escritos por los procesos llevados a cabo.

El reporte para cada terminal nos muestra la actividad por cada una de ellas, similar a lo que a continuación se muestra:

```
TOTAL DURATION IS 448 MINUTES
LINE          MINUTES  PERCENT  # SESS  # ON   # OFF
/dev/pts/13   0        0        0       0      0
/dev/pts/14   0        0        0       0      0
console       1440     100      1       1      1
ftp10764      0        0        1       1      1
pts/1         1440     100      1       1      1
pts/10        1440     100      1       1      1
```

Las columnas nos describen lo siguiente:

LINE La terminal en línea o puerto accesado.

MINUTES El total de minutos que la terminal estuvo en uso durante el periodo de contabilidad.

PERCENT El total de minutos que la terminal estuvo en uso, dividido entre el total de minutos del periodo de contabilidad.

SESS El número de veces que el puerto fue accesado por una sesión.

# ON	El número de veces que el puerto fue utilizado por algún usuario al conectarse.
# OFF	El número de veces que algún usuario dejó de estar dentro del sistema y también cualquier interrupción que ocurra en la línea.

El reporte de contabilidad mensual se guarda en los archivos llamados:

```
/adm/acct/fiscal/fiscrptnn
```

donde *nn* es el mes. Estos reportes son muy similares a los reportes de contabilidad diarios.

Capítulo 3

MODELO CLIENTE/SERVIDOR

Este capítulo describe las principales características del Modelo Cliente/Servidor, las cuales son utilizadas para el desarrollo de este trabajo

3.1 Redes de Computadoras

"Una red de computadoras es un sistema de interconexión entre terminales ligadas conjuntamente con el propósito de intercambiar datos de una máquina a otra."[Cra90]

Las redes permiten a las computadoras compartir archivos, periféricos, recursos en general como los son impresoras y plotters, y otros dispositivos de cómputo. Las redes permiten que se puedan compartir recursos de cada una de las diferentes computadoras conectadas, además los usuarios pueden comunicarse y enviar mensajes *a través de la red*.

Al *software* que se encarga de administrar los recursos que se estarán compartiendo (discos duros, impresora etc.) se le conoce como Sistema Operativo de Red (NOS del inglés Network Operating System).

En realidad una red de computadoras es muy compleja a causa del tipo y número de computadoras ligadas a través de ella, los diferentes medios usados para conectar las computadoras y la distancia geográfica entre éstas

La siguiente lista explica cómo la composición de una red afecta sus funciones.

1. Una red de computadoras esta compuesta de al menos dos y posiblemente cientos de computadoras y dispositivos periféricos (impresoras, terminales, discos etc.) Cada punto que pueda conectarse a la red es llamado nodo.

   Cada computadora sobre la red es llamada *host*, la computadora en la cual se origina un comando es llamada host local y las otras computadoras sobre la red son llamadas hosts remotos.

2. Medios de la red para ligar físicamente las computadoras.

   Existen diversos medios que se usan para transmitir datos en una red de computadoras, los cuales varían en costo, velocidad de transmisión y en las diferentes técnicas aplicables a transmisiones y distancias. A continuación se enuncian algunos ejemplos.

   El cable coaxial esta compuesto por un alambre (un conductor) cubierto de una placa que actúa como tierra. El conductor y la tierra están separados por un aislante. Proporciona un buen aislamiento de las interferencias eléctricas. Las ventajas principales del cable coaxial son las siguientes:

   - Transmite voz, video y datos.
   - Instalación relativamente sencilla.
   - Tecnología fácil de entender.
   - Gran disponibilidad en instalaciones ya existentes.

   El cable de fibra óptica consiste de una fibra muy delgada hecha de dos tipos de vidrio, uno para la parte interior y otro para la exterior, con diferentes índices de refracción. Esta combinación previene que la luz penetre en una parte de la fibra hasta la parte exterior. La fibra por si misma está protegida por una placa para darle mayor integridad estructural. Algunas de las ventajas de la fibra óptica son los siguientes:

   - Aplicaciones de alta velocidad.
   - No genera señales eléctricas o magnéticas.
   - Inmune a interferencia y corrosión.

- Puede propagar una señal sin la necesidad de un amplificador, a distancias muy largas.

Es una de las mejores alternativas, el costo es elevado pero la tecnología con la que se fabrica esta avanzando y el precio decrementa cada vez más.

3. Un método general de clasificación de las redes es por medio de la distancia geográfica entre las computadoras conectadas al sistema.

Las redes entran en una de las siguientes categorias:

- Red de área local (LAN).
- Red de área metropolitana (MAN)
- Red de área amplia (WAN).

Una red de área local (LAN del inglés Local Area Network) consiste en uns sistema de computadoras que están localizadas a poca distancia entre ellas, como una oficina, un edificio o en un campus universitario.

Las redes de computadoras en área local, en los últimos años, han experimentado un crecimiento explosivo. Las redes se pueden conectar a otras redes, para crear una extensa área de redes, las cuales en su momento utilizan conexiones internacionales para crear redes de área global.

Una red de área metropolitana (MAN del inglés Metropolitan Area Network) son redes que cubren el territorio geográfico de una ciudad que a su vez pueden tener conectadas redes de área local

Las redes MAN pueden tener interconexión con redes LAN y WAN mediante algún dispositivo de interconectividad (gateways, routers etc.)

Una red de área amplia WAN (del inglés Wide Area Network) generalmente es interpretada como una red internacional y son sistemas que enlazan computadoras y otros dispositivos a otras computadoras o redes en localidades remotas. Las redes de área amplia comunican áreas geográficas muy grandes; como diferentes ciudades, diferentes estados, diferentes países e incluso diferentes continentes.

Un ejemplo del tipo de redes WAN es la ARPANET, la cual fue desarrollada por una Agencia del Departamento de Defensa de los Estados Unidos, y liga sistemas computacionales académicos, del gobierno, y la industria alrededor del mundo.

4. Clasificación de las redes por el tipo de conmutación.

   Una red conmutada consiste de un conjunto de nodos interconectados en los cuales los datos se transmiten de la fuente al destino y son enrutados a través de los nodos de la red.

   Las redes entran en una de las siguientes categorías:

   - Redes de Conmutación de Circuitos.
   - Redes de Conmutación de Mensajes.
   - Redes de Conmutación de Paquetes.

   La comunicación vía circuitos conmutados, implica que se tiene una trayectoria de comunicación dedicada temporalmente al enlace entre dos estaciones. La trayectoria se establece por medio de una secuencia de conexiones entre nodos de manera que, en cada enlace, hay un canal de comunicación dedicado a esta comunicación.

   La comunicación vía mensajes, intercambia unidades de información denominados mensajes, con la conmutación de mensajes no es necesario establecer una trayectoria dedicada al enlace entre dos estaciones. En lugar de ello, la estación de origen le agrega una dirección de destino al mensaje y lo envía. El mensaje pasa por la red de nodo en nodo, en cada nodo se recibe el mensaje, se almacena temporalmente y luego se envía al siguiente nodo.

   La conmutación de paquetes es similar a la conmutación de mensajes, su principal diferencia es la longitud de unidades que maneja uno y otro sistema. En los sistemas de conmutación se usan unidades de datos de 1000 bits típicamente. En la estación deben fragmentarse los mensajes en paquetes de longitud determinada, para poder transmitirse a través de la red.

3.2 Protocolos

Adicionalmente a las conexiones físicas que permiten a las computadoras de una red la comunicación entre ellas, debe de existir un medio para especificar las reglas de comunicación que permitan a los *hosts* la comunicación con algún otro. La comunicación sería imposible sin algún tipo de lenguaje o código, en las redes de computadoras estos lenguajes son llamados colectivamente *protocolos*. Los protocolos usados por las redes de computadoras no son sino las reglas estrictas para el intercambio de mensajes entre dos o más nodos Dos de los más importantes protocolos son, el OSI y el TCP/IP

3.2.1 OSI

El sistema de interconexión abierto (OSI del inglés Open Systems Interconnection), define un protocolo de siete capas (figura 3.1), que resumen la manera en la cual se pueden comunicar las máquinas.

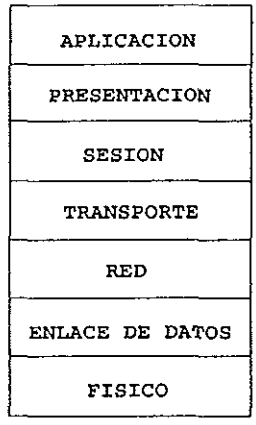

Figura 3.1: Estructura de OSI

- Capa física. Es el nivel más bajo en el modelo OSI y define las características físicas y eléctricas que existen en el hardware de los sistemas conectados: Por ejemplo, cable coaxial, cable par trenzado, F O., satélites etc.

- Capa de enlace de datos. Es el nivel que se encarga de enviar los bits a través del nivel físico. La capa de enlace de datos detecta errores y corrige estos, por requerimiento de retransmisión de paquetes o mensajes mal transmitidos. Esta capa esta dividida en dos subcapas: El control del medio de acceso (MAC del inglés media acces control) y el control de enlace lógico (LLC del inglés logical link control). La subcapa MAC tiene la responsabilidad del acceso de red. La subcapa LLC opera a través de la MAC enviando y recibiendo los paquetes de datos y mensajes.

- Capa de red. Esta capa es responsable de cambiar y proporcionar la ruta adecuada de los mensajes a los destinatarios apropiados. Esta coordina los medios para el direccionamiento y entrega de mensajes.

- Capa de transporte. Cuando un mensaje contiene más de un paquete, la capa de transporte organiza secuencialmente los paquetes de mensajes y regula el flujo de tráfico. Esta capa es responsable de detectar y descartar paquetes duplicados.

- Capa de sesión. Organiza y sincroniza el intercambio de datos entre los procesos de la aplicación. La capa de sesión está involucrada en la coordinación de las comunicaciones entre diferentes aplicaciones y le permite a cada una conocimiento del estado de la otra. Una *sesión* es un intercambio de mensajes, un diálogo entre dos procesadores.

- Capa de presentación. Para que las aplicaciones que están cooperando compartan datos deben estar de acuerdo en como los datos son representados, esta capa de OSI proporciona las rutinas estándar para la presentación de los datos.

- Capa de aplicaciones. La tarea asignada a esta capa de la aplicación, es desplegar información recibida y enviar los nuevos datos del usuario a las capas inferiores. Los mensajes dentro del protocolo OSI en este nivel bajan a través de las otras capas hacia la capa física y a través de la red hacia otra capa física y nuevamente suben a través de las capas de otro procesador hasta el nivel de aplicación y sus programas.

3.2.2 TCP/IP

El protocolo de comunicación TCP/IP tiene sus orígenes en el año 1969, en un proyecto de investigación fundado en los Estados Unidos por el DARPA (Defense Research Agency, Agencia de Proyectos Avanzados de la Investigación de Defensa), en 1983 fue adoptado como un estándar del nuevo conjunto de protocolos TCP/IP y todos los nodos de la red pasaron a utilizarlo. Ahora en casi todos los sistemas UNIX se utiliza el protocolo TCP/IP. Para ocultar la diversidad de *hardware* que se puede usar en una red, TCP/IP define una *Interfaz* a través de la cual se accesa al *hardware*. Esta interfaz ofrece un conjunto de operaciones para enviar y recibir paquetes.

Los dos componentes de TCP/IP el Protocolo de Control de Transmisión (*Transmission Control Protocol*) y Protocolo Internet (*Internet Protocol*) se muestran en forma de niveles (figura 3.2).

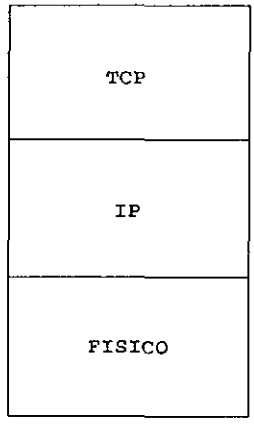

Figura 3.2 Estructura de TCP/IP

- TCP hace las conexiones lógicas entre *hosts* y asegura que la transmisión sea exacta entre ellos. Esta también arregla el flujo de datos entre *hosts*.

- IP rutea datos entre *hosts*, y transmite datos en la red después de determinar la mejor ruta disponible.

3.3 El Modelo Cliente/Servidor

Los programas escritos para usar las facilidades proporcionados por las redes usualmente siguen y soportan algunos modelos.

El modelo más popular para el software de red es el modelo cliente/servidor (figura 3.3). Este modelo consiste en tener procesos servidores en una máquina para proporcionar servicios a procesos de otra máquina, llamadas consumidores o clientes. Un proceso cliente podría contactar un proceso servidor y requerir un servicio particular, entonces el servidor podría ejecutar el servicio a favor del cliente. Por ejemplo, en una aplicación donde se copian archivos entre máquinas sobre la red, es dividida en dos partes. La parte del cliente es llamada por el usuario sobre la máquina local para iniciar la transferencia, la parte del servidor reside sobre la máquina remota que asiste en la transmisión del archivo.

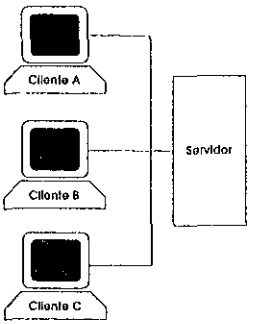

Figura 3.3: Modelo Cliente/Servidor

El ambiente es típicamente heterogéneo. La plataforma del hardware y el sistema operativo del cliente y el servidor usualmente no son el mismo. Ellos se comunican através de un conjunto definido de estándares, programas de interfaces de aplicación (API's) y llamadas remotas (Remote Procedure Calls o RPC's).

3.4 El Servidor

" Un servidor es uno o más procesadores multiusuarios con memoria compartida, que proporcionan cálculos, conectividad, ser-

vicios de bases de datos e interfaces para las necesidades de las aplicaciones".[Pat92]

El servidor no requiere un hardware especial; por lo tanto la plataforma del hardware puede ser seleccionada con base en las demandas de la aplicación y economía. Para una aplicación basada en el modelo cliente/servidor, el servidor trabaja mejor cuando existe una configuración con sistema operativo que soporta prioridad de multitareas; un sistema operativo con prioridad de multitareas, tiene la habilidad de permitir que una tarea con prioridad alta obtenga el control del procesador sobre una tarea que se esta ejecutando con una prioridad menor. El servidor suministra y controla accesos compartidos a los recursos que posee. Las aplicaciones sobre un servidor pueden ser aisladas una de otra así que el error de una no puede dañar alguna otra. La prioridad dentro de un sistema multitarea asegura que una simple tarea no pueda tomar todos los recursos del servidor e impedir que otras tareas sean suministradas por el servicio.

El servidor proporciona recursos como son, aplicaciones, archivos, bases de datos, impresoras, fax, imágenes, comunicaciones, seguridad, manejadores de servicios de sistemas y redes.

3.5 El Cliente

Un cliente es una estación de trabajo que proporciona la presentación de los servicios y la apropiada conectividad a bases de datos y apropiadas interfaces gráficas para las necesidades de las aplicaciones.

Las estaciones de trabajo son computadoras interconectadas, estas comparten recursos del servidor.

El proceso de datos de una red es distribuído, por lo tanto, el desempeño de la estación de trabajo se debe definir en función a la aplicación que se estará manejando en ella. Analizar el tipo de aplicaciones que estarán manejando en la red es de suma importancia para lograr que la estación sea adecuada.

Cualquier estación de trabajo puede usarse como un cliente, si la misma estación de trabajo puede compartir simultáneamente memoria a varios usuarios, ésta también puede usarse como servidor. Cuando una estación de trabajo cliente es conectada a una LAN, ésta puede accesar a los servicios proporcionados por el sistema

operativo de red, adicionalmente a aquellos proporcionados por la estación de trabajo servidor.

El modelo cliente/servidor presenta una clara separación de funciones, basadas en la idea de cuándo actúa el servidor y cómo proporciona servicios respondiendo a las peticiones de los clientes.

La plataforma ideal cliente/servidor es un ambiente de sistema abierto, usando una disciplina requerimiento–servidor que esta basada en los estándares normales. Esto permite que múltiples plataformas de software y hadware interactúen cuando el estándar requerimiento–servidor es implementado a los servidores que posiblemente crezcan y cambien su sistema operativo y plataformas de hardware sin cambiar las aplicaciones del cliente.

3.6 Comunicaciones entre Procesos

La comunicación entre procesos (IPC del inglés Interprocess Communication) permite a los sistemas UNIX la comunicación de dos o más procesos con algún otro.

En el modelo cliente/servidor siempre existe la necesidad de la comunicación entre procesos. Los clientes y servidores siempre deben comunicarse con peticiones y respuestas. En realidad, una gran parte de aplicaciones cliente/servidor estan implicadas con *IPC* (comunicaciones entre procesos).

El uso de comunicación entre los procesos es inherente en ambientes de sistemas multitarea. Las tareas activas operan independientemente y reciben requisiciones y envían respuestas vía los IPC.

Para una implementación efectiva de cualquier aplicación cliente/servidor, las IPCs son usadas en aquellas operaciones que implican comunicación entre los procesos en una simple máquina o a través de diferentes máquinas de una red.

Algunos de los servicios que proporcionan las IPC son los siguientes:

- Protocolos para coordinar cuando se envían y reciben los datos entre los procesos.
- Mecanismos de espera que permiten a los procesos ejecutarse asíncronamente.
- Soporte para intercambios de datos de muchos a uno (un servidor tratando con muchos clientes).

- Soporte de procedimientos remotos para solicitar un servicio remoto

3.7 Puertos

Cada computadora que utiliza TCP/IP permite a los programas conectarse a los diferentes puertos de alguna otra; cada puerto es la entrada de algún servicio de red: *ftp, mail, finger, who*, etc. Los puertos se pueden ver como puntos de anclaje para conexiones de red. por ejemplo, si una aplicación quiere ofrecer un cierto servicio, se le asigna un determinado puerto en donde espera las peticiones de los clientes (a esto también se le llama escuchar en un puerto). Un cliente que quiera usar este servicio consigue un puerto libre en su nodo local y se conecta al puerto del servidor en el nodo remoto. Una propiedad importante de los puertos es que una vez que se ha establecido una conexión entre el cliente y el servidor, otra copia del servidor tiene la posibilidad de responder en el mismo puerto a peticiones de más clientes. TCP es capaz de distinguir unas conexiones de otras, ya que todas ellas provienen de diferentes puertos o nodos. Por ejemplo, si accede dos veces a la estación de trabajo llamada **mira** de la estación de trabajo **polaris**, entonces el primer cliente *telnet*[18] usará el puerto local 1023, y el segundo usará el puerto 1022, sin embargo, ambos se conectarán al mismo puerto de **mira**

3.8 Sockets

Dos computadoras se comunican de una a otra, vía un *circuito* de red. cada circuito es identificado por una combinación de dos números llamados *socket*. Básicamente un socket es la dirección IP de una máquina más el número de puerto usado por el software TCP. Los sockets API se desarrollaron en la universidad de California en Berkeley como parte de la versión BSD 4.1c de UNIX.

Los sockets fueron diseñados para proporcionar una consistente y fácil comunicación para los procesos, sin hacer caso de si son o no procesos de la misma máquina. Además los sockets fueron diseñados para esconder detalles de la manera de llevar a cabo la comunicación.

[18] Comando UNIX para conectarse a una estación de trabajo

La programación de sockets es basada en el modelo cliente/servidor, como ya se mencionó el cliente es el programa que hace requerimientos al servidor, el servidor es simplemente otro programa que responde a estos requerimientos.

La biblioteca de sockets, proporciona socket basados en TCP/IP, para comunicaciones entre diferentes máquinas (los sockets AF_INET) y otros que manejan comunicaciones locales a la máquina (la clase AF_UNIX).

Exiten sockets en ambas máquinas, la que envía y la que recibe. Los clientes envían sobre sus sockets y el servidor escucha sobre sus sockets y acepta las conexiones cuando es necesario.

3.8.1 El Modelo de Socket

1. El proceso de servidor crea un socket genérico.
2. El servidor asocia el socket a una dirección agregada.
3. El servidor notifica al sistema que esta listo para la conexión.
4. El servidor está esperando para la primera conexión.
5. El proceso de cliente crea un socket genérico.
6. El cliente asocia el socket para la dirección de cualquier sistema seleccionado.
7. Una vez asociado, el cliente conecta su socket al socket del servidor usando la dirección destino; esto establece la conexión.
8. El servidor es notificado de la nueva conexión.
9. Una copia del programa servidor sigue esperando a más clientes.
10. Otra copia del programa servidor atiende la conexión.
11. La conexión lee los datos del socket que han sido enviados por el cliente. El servidor escribe datos al socket

Capítulo 4

SISTEMA INTEGRAL DE CONTABILIDAD UNIX (SICU)

En este capítulo se describe al Sistema Integral de Contabilidad en UNIX (SICU), así como los antecedentes y las necesidades que dieron origen a su desarrollo.

4.1 ¿Que es el Sistema Integral de Contabilidad UNIX?

El Sistema Integral de Contabilidad en UNIX (SICU) es una herramienta de administración para revisar la contabilidad de los sistemas UNIX basados en UNIX System V. El SICU se compone de una serie de programas desarrollados en *Perl*[19] y su función general es la de transferir archivos de contabilidad a través de la red, para filtrarlos y transformarlos en información gráfica que permita al administrador tener una visión más rápida y general del uso o abuso de los recursos del sistema.

4.2 Antecedentes

La idea original para el desarrollo de este programa nace en el Departamento de Administración de Supercómputo de la Dirección

[19] El lenguaje de programación Perl significa Practical Extraction and Report Language

General de Servicios de Cómputo Académico (DGSCA) en la Universidad Nacional Autónoma de México (UNAM).

En este departamento se llevan a cabo las labores de administración de las supercomputadoras CRAY YMP 4/464 y CRAY–SGI Origin2000, las cuales son utilizadas en proyectos de investigación. Así mismo cuenta con equipo para el desarrollo de herramientas y servicios para la comunidad universitaria relacionados con las labores de administración de sistemas, además se realiza la administración de los equipos del Laboratorio de Visualización.

El Laboratorio de Visualización, cuenta con estaciones de trabajo con excelentes características de manejo de imágenes y graficación, las cuales son necesarias para la obtención visual de los resultados numéricos proporcionados por las supercómputadoras.

De esta forma las máquinas cuya administración es responsabilidad del Departamento de Supercómputo son:

Supercómputo	2 Silicon Graphics Indy.
	4 SPARCstation Classic.
	2 SPARC 4.
Visualización	1 Silicon Graphics Octane.
	1 Silicon Graphics Onyx Reality Engine 2.
	1 Silicon Graphics Indigo 2 Extreme.
	1 Silicon Graphics Indigo 2 XZ.
	2 Silicon Graphics Indi.
	1 Silicon Graphics O2.
	1 Silicon Graphics 4D/35TG
	1 Silicon Graphics 4D/420 VGX
	1 Sun SPARC 1000

Las claves para trabajar en las máquinas mencionadas anteriormente son asignadas dependiendo de las características del usuario; una máquina tiene un gran número de usuarios, que le dan uso continuo al sistema con propósitos de investigación, desarrollo o administración. Cada usuario tiene que utilizar su cuenta sólo para tareas asignadas, ya que el abusar de los recursos del sistema perjudica al

resto de los usuarios, por otro lado algunos usuarios desconocen que debido a una mala codificación de sus programas éstos consumen recursos inescesarios del sistema.

Con el análisis de la contabilidad podemos definir un mal uso (consciente o inconsciente) de los recursos del sistema, pero considerando que se tiene un gran número de máquinas surge la necesidad de contar con una herramienta optimizada que ayude al administrador de sistemas a realizar este análisis.

El Sistema Integral de Contabilidad en UNIX (SICU), haciendo uso de los reportes de contabilidad, facilita la labor y rápidez de analizar el consumo de recursos de un conjunto de máquinas, generando reportes gráficos fáciles de interpretar.

4.2.1 Problemática

¿Que sucede con toda la información generada por el sistema de contabilidad de cada máquina con la que se cuenta ?

Cada máquina genera un reporte de contabilidad diario y lo almacena en un archivo, el administrador del sistema a su vez realiza el análisis de la información de cada archivo, lo cual le puede indicar abusos por parte de algún usuario o indicio de que algún comando de usuario o de sistema esta consumiendo demasiados recursos. El administrador debe recolectar toda la información y analizarla en forma manual, lo cual genera una serie de problemas.

- Cada máquina genera un archivo diario de contabilidad, considerando que se cuenta con varias máquinas esto genera gran cantidad de archivos.

- Para tener una visión más general de la situación de la contabilidad del sistema el administrador tiene que revisar una serie de archivos que tienen lugar durante un periodo de tiempo prolongado

- Los archivos son de formato muy ancho por lo que es un problema seguir los renglones de estos.

- Los editores básicos de unix dividen en partes los renglones muy largos como los de la contabilidad perdiendo legibilidad.

- El análisis de los archivos tiene que ser hecho a mano por parte del administrador, es una tarea tediosa que no a todos les gusta realizar.
- Debido que es una tarea tardada y generalmente el administrador o las necesidades de los usuarios le exigen al administrador cumpla alguna otra tarea más inmediata que la revisión de la contabilidad y se va dejando relegada esta tarea también muy importante.

Todo lo anterior puede redundar en una subutilización del sistema de contabilidad ya que esta va acumulando los reportes que de pronto son demasiados y ya nadie utiliza.

4.2.2 Alternativa de Solución

La solución a esta problemática es hacer que se entienda el significado de los reportes de la contabilidad, generando una representación gráfica sencilla de asimilar, tangible para el administrador y mediante la cual se resuelvan rápidamente situaciones del uso de los recursos, además que disponga de una sola vez al acceso de los diferentes archivos de las máquinas por analizar.

4.3 Descripción General

El SICU es un sistema que facilita la búsqueda de información relevante de la contabilidad y ayuda a resolver los problemas antes descritos. El SICU está desarrollado con base en el modelo cliente/servidor, donde el cliente solicita archivos de contabilidad del servidor y este servidor atiende la petición enviando el archivo solicitado (figura 4.1).

4.3.1 Servidor–SICU

Cada una de las máquinas que funcione como servidor resuelve peticiones de algún cliente. Atento a la petición por el puerto asignado para esta tarea, se encuentra un programa que resuelve si la petición es válida y envía la solicitud. Este programa siempre estará listo para el número de peticiones que lleguen, ya que por cada petición ejecutará un proceso nuevo para atender la misma y al mismo

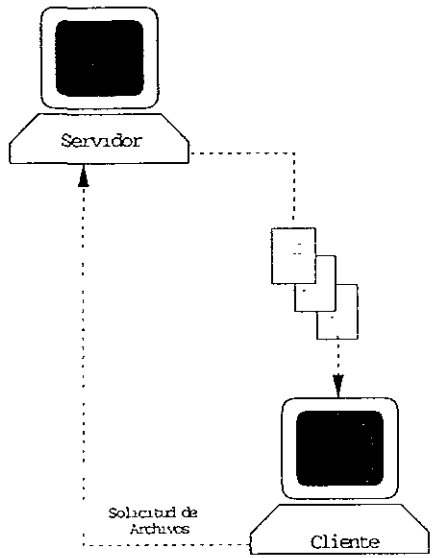

Figura 4.1: Solicitud del Cliente al Servidor

tiempo el programa seguirá atento a que llegue una nueva petición, aprovechando de esta manera la característica de multiprocesamiento que tiene UNIX (figura 4.2).

4.3.2 Cliente–SICU

La parte cliente del SICU, es una interfaz gráfica, de las llamadas *interfaces orientadas a ventanas con opción de señalar y elegir*[20]. "La interfaz de usuario es el mecanismo a través del cual se establece un diálogo entre el programa y el humano"[Pre93]. Aquí podemos hacer la solicitud a cada uno de los servidores disponibles por medio de botones, para realizar alguna transferencia de sus archivos de contabilidad. Una vez realizado esto se lleva a cabo el filtrado de información de los mismos para finalmente convertir los datos en imágenes que nos representan el uso de los recursos de los sistemas.

La interfaz gráfica proporciona la facilidad visual para llevar una

[20] También denominadas "WIMP"(del inglés, ventanas, iconos, menús y dispositivos de señalización)

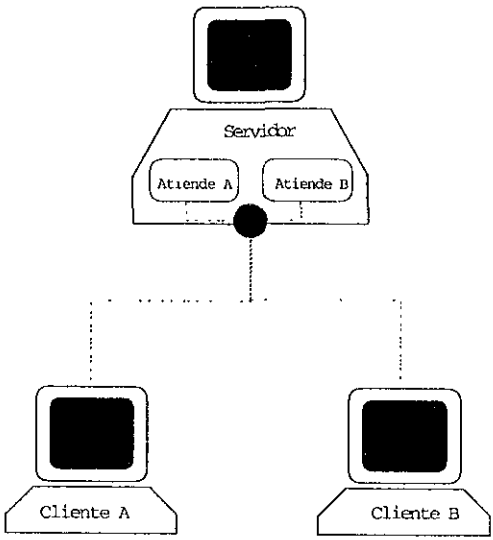

Figura 4.2: Atención a más de un Cliente al mismo tiempo

tarea de elección de máquinas (servidores) que requieren de algún análisis en sus reportes de contabilidad y es en la misma interfaz donde podemos ver las gráficas resultantes (barra o pie). Se incluyen dentro de la interfaz menús con funciones propicias para la interacción Hombre–Máquina.

La cuarta generación de interfaces gráficas[Pre93] es aprovechada para trabajar con el SICU ya que en todos las máquinas donde se puede ejecutar tienen la habilidad para realizar diferentes tareas simultáneas y es posible realizar en otras ventanas otras tareas rutinarias (figura 4.3).

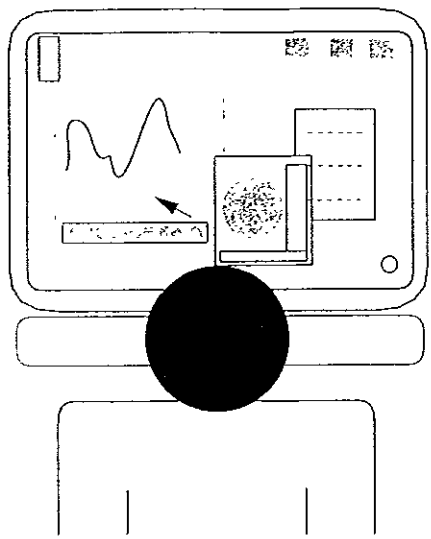

Figura 4.3: Cuarta generación de interfaces gráficas

4.4 Ventajas del SICU

- Los archivos de contabilidad serán más fáciles de revisar
- Desde una máquina podemos solicitar a otra una petición de requerimiento de algún archivo de contabilidad.
- Las máquinas proporcionan un servicio exclusivo para transferir los archivos sin necesidad de tener una conexión directa a estas máquinas, es decir, no necesitamos entrar a estas máquinas para revisar la contabilidad.
- Se sustituye un servicio de red de uso más general como lo es el de de tranferencias de archivos llamado *ftp*[21] por uno que este dedicado a transferir archivos de contabilidad.

[21] Comando de UNIX para transferir archivos de una máquina a otra

- Al evitar la necesidad de entrar al sistema, evita también que viaje demasiado el passwd del usuario que esté revisando los archivos de contabilidad a través de la red.

Capítulo 5

DISEÑO E IMPLEMENTACIÓN DEL SICU

En este capítulo se presenta el diseño e implementación del SICU, el programa fuente que corresponde a cada uno de los módulos se explica el funcionamiento de dicho programa destacando los puntos más importantes.

5.1 Diseño Orientado a Flujo de Datos

El diseño es de gran importancia en la fase de desarrollo de cualquier sistema. se puede definir como: "El proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física".[Pre93]

En la estructuración del diseño de sistemas se empieza construyendo un modelo lógico, usando diagramas de flujo de datos como una herramienta. Durante el diseño del sistema este modelo lógico se convierte en forma física. El diseño orientado al flujo de datos es una metodología que utiliza las características del flujo de información para generar la estructura del programa. La información puede representarse como un flujo continuo que sufre una serie de transformaciones conforme pasa de la entrada a la salida

El diseño orientado al flujo de datos define varias representaciones que transforman el flujo de la información en la estructura del pro-

grama. A medida que la información se mueve a través del sistema, es modificado por una serie de transformaciones.

5.1.1 Flujo de Transformación

La información que entra en el sistema por algún medio que transforma datos externos a una forma interna se identifica como *flujo entrante.*

Los datos que se transforman en una nueva forma externa moviéndose por diferentes medios son llamados *flujo saliente.*

5.1.2 Diagrama de Flujo de Datos

El *Diagrama de Flujo de datos* (DFD) es una técnica gráfica que representa el flujo de la información y transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida.

El rectángulo se usa para representar una *entidad externa*, es decir un elemento del sistema. Un círculo representa un *proceso o transformación* que se aplica a los datos. Las flechas indican la dirección del flujo de los datos. La línea doble representa un depósito de datos que se guardan para ser usados posteriormente.

Se utiliza el diagrama de flujo de datos para representar el sistema de 4 diferentes niveles de abstracción. Se muestran a continuación.

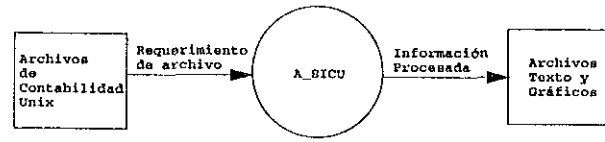

Figura 5.1: Diagrama de Flujo de Datos Nivel 0

En el DFD nivel 0 (figura 5.1), también llamado *modelo fundamental del sistema*, se observa que las dos entidades principales son: los archivos de contabilidad unix y los archivos de salida. Los archivos de contabilidad producen la información que es transformada y los archivos de salida son la información generada por el sistema. Las flechas nos indican los elementos de datos compuestos como las dos órdenes principales que son: requerir y transformar.

Expandiendo el DFD 0 a un nivel 1 (figura 5.2), y generando más información a partir del nivel 0, se ha expandido en siete procesos,

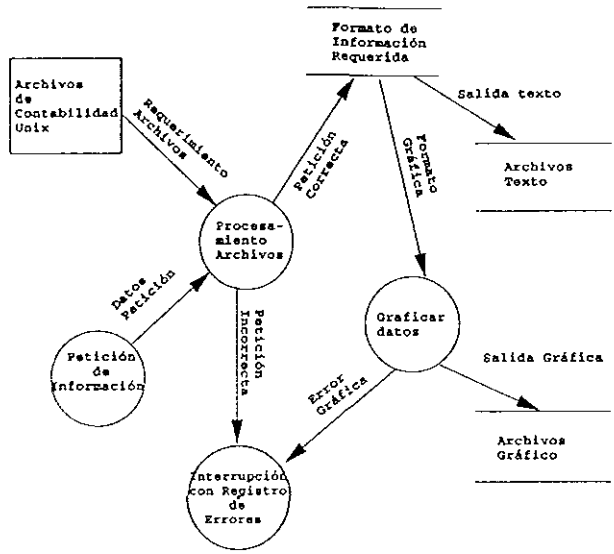

Figura 5 2: Diagrama de Flujo de Datos Nivel 1

manteniendo la continuidad del flujo de la información. Aquí podemos ver como se generan sub-burbujas, que son procesos donde llegan los datos por los diferentes flujos, dependiendo las validaciones o la petición misma.

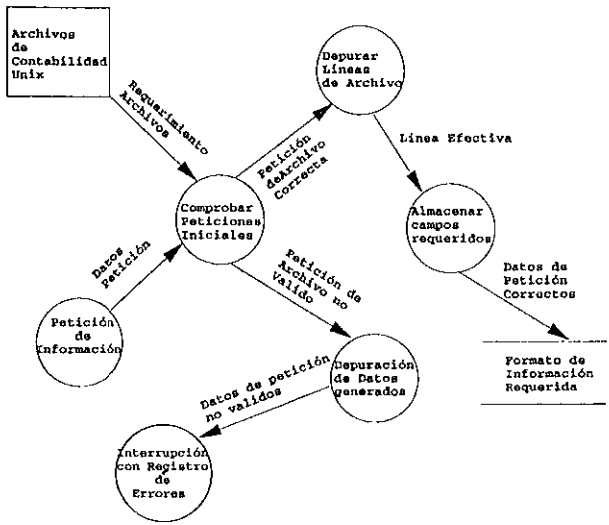

Figura 5.3: Diagrama de Flujo de Datos Nivel 2

Expandiendo el DFD 1 a un nivel 2 (figura 5.3) se generan nuevos subprocesos para detallar el flujo de datos donde se validan las peticiones, se muestra que en caso de error, se genera un nuevo subproceso que clasifica el tipo error y para el caso de validaciones correctas se almacenan los datos.

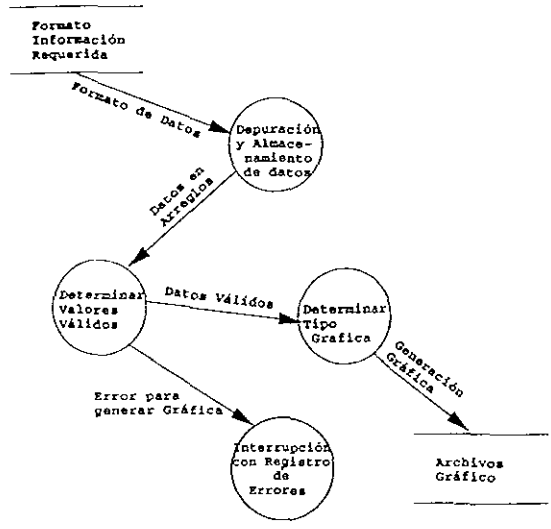

Figura 5.4 Diagrama de Flujo de Datos Nivel 3

Expandiendo el DFD 1 a un nivel 3 (figura 5.4) se generan nuevos subprocesos para detallar el flujo de datos donde se depura y determina el tipo de gráfica en caso de error, de igual manera se genera un nuevo subproceso que clasifica el tipo error.

5.2 Elección del Lenguaje de Programación

"El arte de elegir un lenguaje comienza con el propio problema planteado, al decidir cuáles son sus requisitos y su importancia relativa, ya que probablemente será imposible satisfacerlos todos por igual (con un único lenguaje)."[Pre93]

Entre los criterios que se aplican durante la evaluación son:

- Area de aplicación general.
- Complejidad algorítmica.
- Entorno en el que se ejecutará el software.
- Complejidad de las estructuras de datos.

Considerando los puntos anteriores, en la elección del lenguaje de programación para la codificación del SICU, el lenguaje de programación *Perl* cumple adecuadamente los requisitos mencionados, esto es, adecuado dentro del entorno donde se ejecuta (estaciones de trabajo con sistema operativo UNIX), el área de aplicación (administración de estaciones de trabajo) y facilita el manejo de estructuras de datos. En el mismo lenguaje se cuenta con extensiones para crear las interfaces gráficas para la manipulación del sistema.

5.2.1 Características de Perl

Este lenguaje fue creado por Larry Wall en 1986. Como administrador de sistemas Larry Wall notó que los lenguajes como *shell y AWK*[22] incluídos con el sistema operativo UNIX, no le permitían codificar de una manera fácil programas complejos para generar reportes, entonces se dedicó a escribir un lenguaje que estuviera diseñado de acuerdo, no sólo con la creación de reportes sino también con las necesidades generales de un operador de sistemas. Larry Wall observó que en la mayoría de los programas necesarios para el mantenimiento del sistema, el punto primordial es el **procesamiento y manejo de texto**, ya sea para la producción de reportes, el consolidado de información o la clasificación de archivos de datos. En Perl el procesamiento y manipulación de información como cadenas de caracteres, el manejo de archivos y la generación de salida de tipo texto (reportes) y las llamadas al sistema UNIX son de las características más importantes.[Ort96]

[22] Filtros comunes en UNIX

5.3 Módulos de Comunicación SICU

5.3.1 Módulo Controlador del Servidor

En nuestro sistema de modelo cliente/servidor el programa residente en la(s) máquina(s) que actúan como servidor(es), controla(n) las peticiones realizadas a ésta(s). Este programa se encarga que el servidor atienda a las peticiones de los clientes. La respuesta positiva de cada servidor es básicamente transferir a través de la red algún archivo de contabilidad solicitado.

Llamada al Sistema fork()

La llamada al sistema *fork()*[23] en UNIX permite la creación de un nuevo proceso que es una copia casi exacta del primero, las diferencias fundamentales son:

- El proceso hijo tiene un identificador de proceso único.
- El proceso hijo tiene un identificador de proceso padre diferente.
- El proceso hijo tiene su propia copia de los descriptores del proceso padre.
- La función *fork()* regresa 0 al hijo y al padre le regresa el identificador de proceso hijo.
- La llamada al sistema *wait()* esta ligada al *fork()* y generalmente se utiliza para hacer al proceso padre que espere por la terminación del proceso hijo.

Funciones Asociadas a un Servidor que Trabaja con Sockets

- *socket()* crea un socket genérico.
- *bind()* relaciona servidor con el socket a un puerto.
- *listen()* notifica al sistema que está listo para la conexión.
- *accept()* espera por la primera conexión.

[23] página de manual de UNIX fork(): man fork

- El servidor es notificado cuando se realiza una conexión. Típicamente el servidor realiza un fork() para que el servidor hijo maneje la conexión.
- La conexión hijo lee los datos del socket que han sido enviados por el cliente.
- El servidor padre sigue esperando más conexiones. [Sch94]

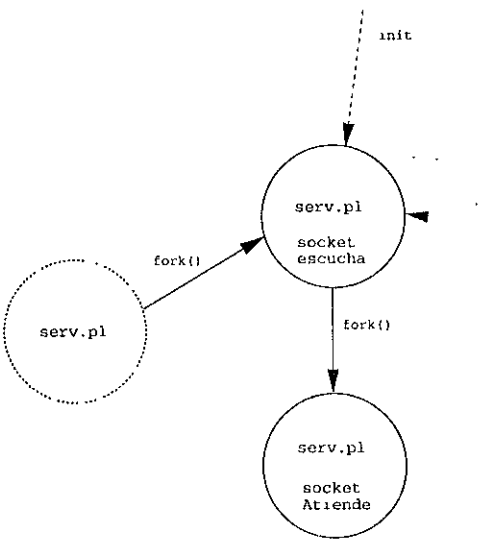

Figura 5.5: Forks en serv.pl

5.3.2 Programa Servidor del SICU:serv.pl

```
#!/usr/local/bin/perl
##////////////////////////////////////////////////////////////////
##  Programa Sevidor:serv.pl
##
##  Autor: Javier De La Rosa Mondragon
##
##  Departamento de Supercomputo
##  Direccion General de Servicios de Computo Academico
##  Universidad Nacional Autonoma de Mexico
```

```
##////////////////////////////////////////////////////////////
use strict;                     #extensiones de perl uso general
use Socket;
use FileHandle; use POSIX 'WNOHANG';
$SIG{CHLD} = sub {              # Controlar procesos generados
                my($pid),
                while (1) {
                $pid = waitpid(-1,WNOHANG);
                last if $pid < 1;
                }
                };
my $port = shift || 12345;
##------------------------------------------------------------
##    Preparacion del socket
##------------------------------------------------------------
socket (SOCKET, PF_INET, SOCK_STREAM, (getprotobyname('tcp'))[2]);
bind   (SOCKET, pack('Sna4x8', AF_INET, $port, "\0\0\0\0"))
       || die "No puedo preparar el puerto $port.";
listen (SOCKET,5 ),
my ($rbits, $pid) = (' ',0),
NEW_SOCKET->autoflush, SOCKET->autoflush;
##------------------------------------------------------------
##  Proceso hijo que se ejecuta indefinidamente
##------------------------------------------------------------
if (! ($pid = fork)) {  # si es el proceso hijo entra aqui
while(1) {
  ##----------------------------------------------------------
  ##  Espera nueva conexion
  ##----------------------------------------------------------
  accept(NEW_SOCKET, SOCKET);   # acepta conexion
  if(! ($pid = fork)) {        # si es el proceso hijo
     while(1){
        vec($rbits,fileno(NEW_SOCKET), 1) = 1;
        select($rbits, undef, undef, undef);
        $_ = <NEW_SOCKET>;
        if (/^clave1$/i) {             # valida conexion  y tipo de peticion
          print NEW_SOCKET "si\n";     # regresa cadena de aceptacion
          $_ = <NEW_SOCKET>,           # lee el nombre del arch. a transferir
          open(ARC,"/var/adm/acct/sum/rprt$_"); # habre el arch. a transferir
              while(<ARC>){
                   print NEW_SOCKET $_;   #transfiere el archvo
              }
           last,
          }
          else{
            if (/^clave2$/i) {        #valida conexion  y tipo de peticion
```

```
                print NEW_SOCKET "si\n";#regresa cadena de aceptacion
                $_ = <NEW_SOCKET>; #lee el nombre de archs. por depurar
                   chomp $_;
                   opendir (DIR,"/var/adm/acct/sum");
                   $mes=$_; my @allfiels;
                   @allfiels=sort grep(m/rprt$mes/,readdir DIR) ;
                   closedir DIR;
                               #ciclo para los archivos necesarios
                   for($i=0;$i<@allfiels;$i++){
                       open (F,"/var/adm/acct/sum/$allfiels[$i]");
                       while(<F>){
                            if(/DAILY REPORT FOR .* Page 1/){
                               print NEW_SOCKET $_;
                               $ciclo_anual++; }
                            if (/^0\s*TOTAL/){
                               print NEW_SOCKET $_;
                               $ban_total=1; }
                            if (/^TOTALS/&&$ban_total==1){
                               print NEW_SOCKET $_;
                               $ban_total=0;}
                       }
                      close(F);
                      print NEW_SOCKET "ciclo $ciclo_anual\n";
                      $ciclo_anual=0; }
                   last; }
                else{ print "no es la clave \n";last;}#conexion no valida
            }
    }
    close(ARC); close NEW_SOCKET; exit;
  } } }
##////////////////////////////////////////////////////////////
##            FIN DE PROGRAMA Sevidor:serv.pl
##////////////////////////////////////////////////////////////
```

5.3.3 Modulo Controlador del Cliente

El cliente, es el programa encargado de pedir los archivos de la contabilidad a la máquina servidor elegida, para posteriormente ser filtrados y graficados.

Funciones Asociadas a un Cliente que Trabaja con Sockets

- *socket()* crea un socket genérico.
- *bind()* Relaciona al cliente con la dirección del servidor.

- *connect()* Una vez ligado, el cliente conecta su socket al socket del servidor. Típicamente el cliente realiza un fork() para que el proceso hijo lea del socket y el proceso padre escriba al socket.

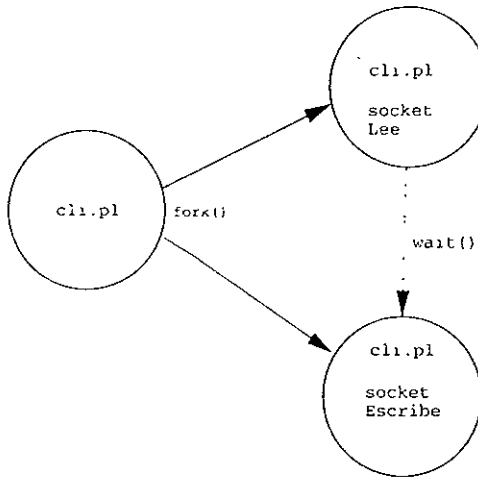

Figura 5 6: Forks en cli.pl

5.3.4 Programa Cliente del SICU:cli.pl

```
#!/opt/local/bin/perl
##////////////////////////////////////////////////////////////
##  Programa Cliente:cli.pl
##
##  Autor: Javier De La Rosa Mondragon
##
##  Departamento de Supercomputo
##  Direccion General de Servicios de Computo Academico
##  Universidad Nacional Autonoma de Mexico
##////////////////////////////////////////////////////////////
use Socket;       #Extensiones de perl uso general
use FileHandle;
if ($#ARGV==3){   #Verifica cuantos argumentos toma el programa
($port,$server,$archivo,$tipo)=@ARGV; #variables argumentos del programa
}
else{ ($port,$server,$tipo)=@ARGV; my $archivo=''; }
open(ARCR,">./$server$archivo") || die "Error no se genero el archivo $!";
##------------------------------------------------------------
##    Preparacion del socket
##------------------------------------------------------------
socket (SOCKET, PF_INET, SOCK_STREAM, (getprotobyname('tcp'))[2]),
connect (SOCKET, pack('Sna4x8',AF_INET, $port,(gethostbyname($server))[4]))
   || die "Conexion fallida\n";
SOCKET->autoflush();
$pid = fork();       # Proceso hijo para leer archivo
if($pid == 0) {
  $_ = <SOCKET>;
  if (/^si$/) {
    {print ARCR  while (<SOCKET>); }
     close(ARCR); close SOCKET; exit; }
  else { close SOCKET; exit; }
}
else {
   print SOCKET "clave\n";    # Proceso padre para escribir datos
   print SOCKET "$ARGV[2]\n";
   wait;                      # Espera a que el hijo termine
   close SOCKET; exit
}
##////////////////////////////////////////////////////////////
##             FIN DE PROGRAMA Cliente:cli.pl
##////////////////////////////////////////////////////////////
```

5.4 Filtrado y Generación de Gráficas

Para la depuración de los archivos transferidos de los cuales sólo interesan los datos referentes a las estadísticas que en su momento se solicitan, se crearon los módulos **awktk.pl,hawktk.pl**

Estos módulos trabajan para el filtrado de la información de los archivos solicitados por el cliente, utilizando *expresiones regulares*, que son la llave, para llevar a cabo un procesamiento de texto eficiente y flexible y de los cuales Perl esta altamente dotado[Fri97]. A continuación se listan expresiones regulares de perl·

.	Cualquier caracter
(..)	Agrupa una serie de elementos
^	Inicio de línea
$	Final de línea
[..]	Clase de caracteres
[^..]	Niega la clase
+	El caracter precedente duplicado una o más veces
?	El caracter precedente duplicado cero o una vez
*	El caracter precedente duplicado cero o más veces
w	Cualquier caracter alfanúmerico incluye "_"
W	Cualquier no caracter alfanumérico
s	Cualquier caracter espacio
S	Cualquier no caracter espacio
d	Cualquier caracter numérico
D	Cualquier no caracter numérico

Dejando listo en un arreglo de datos, los valores solicitados; los mismos módulos contienen las instrucciones para generar los archivos gráficos con base en los arreglos de datos generados, lo cual es posible debido al módulo llamado **GIFgraph 1.01** creado por Martien Verbruggen[24], (figura 5.7).

[24]Pagina http //www tep chem.tue nl/ tgtemv/perl/

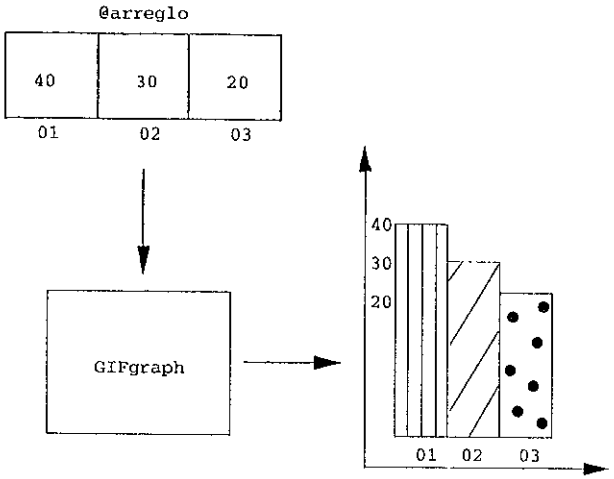

Figura 5.7: Módulo GIFgraph

5.4.1 Módulo awktk.pl

En este módulo se filtra la respectiva información para cada archivo solicitado en el rango de archivo diario de contabilidad, trabajando un archivo a la vez por cada solicitud. Posteriormente se grafican los datos obtenidos dependiendo el tipo de gráfica (pie o barra) que se solicita.

5.4.2 Programa SICU:awktk.pl

```
#!/usr/local/bin/perl
##////////////////////////////////////////////////////////////
##  Programa Graficador:awktk.pl
##
##  Autor: Javier De La Rosa Mondragon
##
##  Departamento de Supercomputo
##  Direccion General de Servicios de Computo Academico
##  Universidad Nacional Autonoma de Mexico
##////////////////////////////////////////////////////////////
use GIFgraph::bars;  # Uso de GIFgraph para graficas de barra
use GIFgraph::pie;   # Uso de GIFgraph para graficas de pie
use GD;
```

```
if ($#ARGV==6){      # Numero de argumentos
   ($Gra,$Opc,$El_archivo,$servidor, $fecha,$tipo,$anio) = @ARGV;
}
else{($Gra,$Opc,$El_archivo,$servidor,$tipo,$anio)=@ARGV;my $fecha='',
}
##-----------------------------------------------------------------
##   Declaracion de Variables
##-----------------------------------------------------------------
my @anual;
use vars qw/%cpup %kcorep %coxp %diskblock/;
use vars qw/%mencpup %menkcorep %mencoxp %mendiskblock/;
use vars qw/%kcoremin %cpumin %menkcoremin %mencpumin/;
use vars qw/$i $dia $mes $BAN $K $ROGO $suma $cambia_mes $pri_mes %meses/;
use vars qw/%ASO $Y $X @linea @renglon1 @renglon2 @renglon3/;
use vars qw/$j $k $titulo @sortcolor @DATOSsort @undef @datossort/;
use vars qw/$valor  @color @datos $cual_total/;
$dia=''; $cambia_mes=0;
if ($tipo==1){ if($fecha=~/(..)(..)/){ $mes=$1; $dia=$2; } }
else { $mes=$fecha, }
##-----------------------------------------------------------------
##   Arreglo asociativo de meses
##-----------------------------------------------------------------
%meses=(
        'Jan'=>'01', 'Feb'=>'02', 'Mar'=>'03',
        'Apr'=>'04', 'May'=>'05', 'Jun'=>'06',
        'Jul'=>'07', 'Aug'=>'08', 'Sep'=>'09',
        'Oct'=>'10', 'Nov'=>'11', 'Dec'=>'12');
##-----------------------------------------------------------------
##    Reporte  Diario por Usuarios
##-----------------------------------------------------------------
if ($tipo==1){
open(ARCHIVO,"$ENV{'PATHSICU'}/temp/$El_archivo"); #Abrir archivo
while(<ARCHIVO>){
 if(/$anio  DAILY USAGE REPORT FOR $servidor Page 1/ .. /$anio\
    DAILY COMMAND SUMMARY Page 1/){       # Expresion regular
                                          # Para rango de uso de usuarios
  if(/^[0-9]+/){                            # Expresion regular
                                            # para filtrar lineas
    @linea=split();
    $cpup{$linea[1]}=$linea[2]+$linea[3];   # Asignacion de valores a
    $kcorep{$linea[1]}=$linea[4]+$linea[5], # los arreglos correspondientes
    $coxp{$linea[1]}=$linea[6]+$linea[7];
    $diskblock{$linea[1]}=$linea[8];
  }
 }
##-----------------------------------------------------------------
```

```
##    Reporte  Diario por Comandos
##-------------------------------------------------------------------
if(/$anio  DAILY COMMAND SUMMARY Page 1/ .. /$anio\
    MONTHLY TOTAL COMMAND SUMMARY Page 1/) {  # Expresion regular
                                              # Para rango de comandos
   if(/([a-zA-Z0-9_\.\-_a-zA-Z0-9]+) +(\d+)\  # Expresion regular
      *(\d{1,11}\.()\d{2}) *(\d+\.(\d{2}))\   # para filtrar lineas
      *(\d+\.(\d{2 })) *(\d+\.(\d{2})) *(\d+\\
      .(\d{2})) *(\d+\.(\d{2})) *(\d{1,13}) *(\d{1,11})/){
      $kcoremin{$1}=$3;                        # Asignacion de valores a
      $cpumin{$1}=$5;}                  # los arreglos correspondientes
 }
}  #FIN DE WHILE DE ARCHIVO
close(ARCHIVO);
}
##-------------------------------------------------------------------
##    Reporte Periodico por Usuarios,
##-------------------------------------------------------------------
if($tipo==2){
  open(ARCHIVO,"$ENV{'PATHSICU'}/temp/$El_archivo"), # Abrir Archivo
  while(<ARCHIVO>){
      if(/DAILY REPORT/){
         @renglon1=split();
         if($cambia_mes==0){
           $pri_mes=$renglon1[0];
           $cambia_mes++; }
         if($renglon1[3]==$anio){ $BAN = 1;}
         else {$BAN=0} }
      if(/^0\s*TOTAL/ && $BAN==1 && ($Opc>=1 && $Opc<=4)){
       @renglon2=split();
      caseA:{                              # Suma de valores
      if($Opc==1){$cpup{$renglon1[1]}=$renglon2[2]+$renglon2[3];
      $mencpup{$meses{$renglon1[0]}}+=$cpup{$renglon1[1]};
      last caseA;}
      if($Opc==2){$kcorep{$renglon1[1]}=$renglon2[4]+$renglon2[5];
      $menkcorep{$meses{$renglon1[0]}}+=$kcorep{$renglon1[1]};
      last caseA;}
      if($Opc==4){$coxp{$renglon1[1]}=$renglon2[6]+$renglon2[7];
      $mencoxp{$meses{$renglon1[0]}}+=$coxp{$renglon1[1]};
      last caseA;}
      if($Opc==3){$diskblock{$renglon1[1]}=$renglon2[8];
      $mendiskblock{$meses{$renglon1[0]}}+=$diskblock{$renglon1[1]};
      last caseA;}
       }
     }
##-------------------------------------------------------------------
```

```
##    Reporte Periodico por Comandos,
##-------------------------------------------------------------------
     if(/^TOTALS/ && $BAN==1 && ($Opc==5 || $Opc==6)){
       @renglon3=split();
       caseB:{
         if($Opc==6){$kcoremin{$renglon1[1]}=$renglon3[2];
           $menkcoremin{$meses{$renglon1[0]}}+=$kcoremin{$renglon1[1]};last caseB;}
         if($Opc==5){$cpumin{$renglon1[1]}=$renglon3[3];
           $mencpumin{$meses{$renglon1[0]}}+=$cpumin{$renglon1[1]};last caseB;}
       }
     }
   } #FIN DE WHILE DE ARCHIVO
close(ARCHIVO);
}
##-------------------------------------------------------------------
##      Case es para determir el arreglo asociativo
##      a graficar, el dato se toma del segundo argumento
##      de este programa y es pasado por el programa sicu
##-------------------------------------------------------------------
 case:{
   if($Opc==1  && $tipo==1){
     %ASO=%cpup;$Y="Tiempo de CPU Minutos";$X="Usuario";
     $titulo="$servidor $anio\/$mes\/$dia";last case;}
   if($Opc==1&&($pri_mes eq $renglon1[0])){
     %ASO=%cpup;$Y="Usuarios Tiempo de CPU Minutos";$X="DIA";
     $titulo="$servidor $anio\/$mes";last case;}
   if($Opc==1&&($pri_mes ne $renglon1[0])){
     %ASO=%mencpup;$Y="Usuarios Tiempo de CPU Minutos";$X="MES";
     $titulo="$servidor   $anio";last case;}
   if($Opc==2 && $tipo==1) {
     %ASO=%kcorep ;$Y="Kb De Memoria"; $X="Usuario";
     $titulo="$servidor $anio\/$mes\/$dia";last case;}
   if($Opc==2 &&($pri_mes eq $renglon1[0])) {
     %ASO=%kcorep;$Y="Usuarios Kb De Memoria"; $X="DIA";
     $titulo="$servidor $anio\/$mes";last case;}
   if($Opc==2 &&($pri_mes ne $renglon1[0])) {
     %ASO=%menkcorep;$Y="Usuarios Kb De Memoria"; $X="MES";
     $titulo="$servidor  $anio";last case;}
   if($Opc==3 && $tipo==1) {
     %ASO=%diskblock ;$Y="Bloques De Disco"; $X="Usuario";
     $titulo="$servidor $anio\/$mes\/$dia";last case;}
   if($Opc==3 && ($pri_mes eq $renglon1[0])) {
     %ASO=%diskblock ;$Y="Usuarios Bloques De Disco"; $X="DIA";
     $titulo="$servidor $anio\/$mes";last case;}
   if($Opc==3 && ($pri_mes ne $renglon1[0])) {
     %ASO=%mendiskblock ;$Y="Usuarios Bloques De Disco"; $X="MES";
```

```
      $titulo="$servidor  $anio";last case;}
    if($Opc==4 && $tipo==1){
      %ASO=%coxp ;$Y="Tiempo De Conexion Minutos";$X="Usuario";
      $titulo="$servidor $anio\/$mes\/$dia";last case;}
    if($Opc==4 && ($pri_mes eq $renglon1[0])){
      %ASO=%coxp ;$Y="Usuarios Tiempo De Conexion Minutos";$X="DIA";
      $titulo="$servidor $anio\/$mes";last case;}
    if($Opc==4 && ($pri_mes ne $renglon1[0])){
      %ASO=%mencoxp ;$Y="Usuarios Tiempo De Conexion Minutos";$X="MES";
      $titulo="$servidor  $anio";last case;}
    if($Opc==5 && $tipo==1) {
      %ASO=%cpumin ;$Y="Tiempo De CPU Minutos"; $X="Comando";
      $titulo="$servidor $anio\/$mes\/$dia";last case;}
    if($Opc==5 && ($pri_mes eq $renglon1[0])) {
      %ASO=%cpumin ;$Y="Comandos Tiempo De CPU Minutos"; $X="DIA";
      $titulo="$servidor \/$anio\/$mes";last case;}
    if($Opc==5 && ($pri_mes ne $renglon1[0])) {
      %ASO=%mencpumin ;$Y="Comandos Tiempo De CPU Minutos"; $X="MES";
      $titulo="$servidor  $anio";last case;}
    if($Opc==6 && $tipo==1) {
      %ASO=%kcoremin ;$Y="Kb De Memoria";$X="Comando";
      $titulo="$servidor $anio\/$mes\/$dia";last case;}
    if($Opc==6 && ($pri_mes eq $renglon1[0])) {
      %ASO=%kcoremin ;$Y="Comandos Kb De Memoria";$X="DIA";
      $titulo="$servidor $anio\/$mes";last case;}
    if($Opc==6 && ($pri_mes ne $renglon1[0])) {
      %ASO=%menkcoremin ;$Y="Comandos Kb De Memoria";$X="MES";
      $titulo="$servidor $anio";last case;}
 }
##-------------------------------------------------------------------
##    While para quitar valor total, y los valores de cero
##    No representativos en graficas de Barras
##-------------------------------------------------------------------
 if($tipo==1){
   while (($i,$valor) = each (%ASO)){
      if ($i !~ /TOTALS*/){
         if ($valor > 0.0 ){
              push (@color, $i); push (@datos,$valor); } }
      else { if ($valor==0) { exit(); }
           $cual_total=$valor; }
    }
 if ($#datos==-1){ exit(); }
@datossort = reverse sort bynumber @datos;   # ordena valores
##-------------------------------------------------------------------
##  For para quitar del arreglo valores menores al 5% del mayor
##-------------------------------------------------------------------
```

```
 $K = 0; $suma = 0,
 for($ROGO=0;$datossort[$ROGO]>=($datossort[0]*0.06);$ROGO++){
     $DATOSsort[$K]=$datossort[$ROGO];
     $undef[$K]=undef; $suma=$suma+$datossort[$ROGO]; $K++;
 }
##-------------------------------------------------------------------
##  For para reasignar su valor a cada dato
##-------------------------------------------------------------------
 for ($j=0 ; $j <= $#DATOSsort; $j++){
   for ($k=0 ; $k <= $#datos; $k++){
     if  ( $DATOSsort[$j] == $datos[$k] ){
         $sortcolor[$j]=$color[$k];
         $datos[$k]=-1;
         last; } } }
}
##-------------------------------------------------------------------
##    While para asignar  valores
##    En graficas de Pie
##-------------------------------------------------------------------
 if($tipo==2){
    $cual_total=0;
    while (($i,$valor) = each (%ASO)){
              push (@color, $i); push (@datos,$valor),
              $cual_total+=$valor;
     }
    if ($#datos==-1){ exit(); }
    @sortcolor = sort bynumber @color;  # Ordenar valores
##-------------------------------------------------------------------
##  For para reasignar su valor a cada dato
##-------------------------------------------------------------------
    for ($j=0 ; $j <= $#sortcolor; $j++){
      for ($k=0 ; $k <= $#color; $k++){
          if( $sortcolor[$j] == $color[$k] ){
            $DATOSsort[$j]=$datos[$k];
            $color[$k]=-1;
            last;
           } } }
 }
##-------------------------------------------------------------------
##  Condicional para determinar el ancho de las graficas
##-------------------------------------------------------------------
 my $xgra,
 if ($#DATOSsort+1>10){ $xgra=900;}
 else { $xgra=550; }
##-------------------------------------------------------------------
##  Generar archivo auxiliar para datos historicos
```

```
##-------------------------------------------------------------------
 if($tipo==1){
   open(FF, ">>$ENV{'PATHSICU'}/temp/arch_aux") || die;
   *STDOUT = *FF;
   print "$servidor:$anio:$mes:$dia:$Opc:$cual_total\n";
  close(FF);
 }
##-------------------------------------------------------------------
##  Grafica de Barra
##-------------------------------------------------------------------
 if($Gra==1){
    my @data;
    if($tipo==1){
      $sortcolor[$#sortcolor+1]='otros';
      $DATOSsort[$#DATOSsort+1]=undef;
      $undef[$#undef+1]=$cual_total-$suma;
      push (@data,\@sortcolor); push (@data,\@DATOSsort);
      push (@data,\@undef);
    }
 if ($tipo==2){ push (@data,\@sortcolor); push (@data,\@DATOSsort); }
 my $my_graph = new GIFgraph::bars("$xgra",550);
 $my_graph->set('x_label'=>"$X", 'y_label'=>"$Y",'title'=>"$titulo",
         'y_min_value' => 0.0, 'long_ticks'=>1,'tick_length'=>10,
         'y_tick_number'=>10,'y_label_skip'=>1,dclrs=>[qw(marine dgreen)],
         'axis_space' => 18,'accentclr'='green',
         'zero_axis'=>0, 'zero_axis_only'=>0,);
  $my_graph->set_x_label_font(gdMediumBoldFont);
  $my_graph->set_y_label_font(gdMediumBoldFont);
  $my_graph->set_x_axis_font(gdMediumBoldFont);
  $my_graph->set_y_axis_font(gdMediumBoldFont);
  $my_graph->plot_to_gif("$ENV{'PATHSICU'}/temp/$El_archivo.gif",\@data );
}
##-------------------------------------------------------------------
##  Grafica de Pie
##-------------------------------------------------------------------
 else {
   if($tipo==1){
     if ( $cual_total-$suma >= $datossort[0]/25 ){
          $sortcolor[$#sortcolor+1]='otros';
          $DATOSsort[$#DATOSsort+1]=$cual_total-$suma; } }
  my $m, @color1, @datos1;
##-------------------------------------------------------------------
##  For etiqueta el  porcentaje en los datos
##-------------------------------------------------------------------
  my $sumpor;
  for ($m=0;$m<=$#sortcolor-1;$m++){
```

```
      $color1[$m]=int(($DATOSsort[$m]*100/$cual_total)+.5);
      $sumpor+=$color1[$m];
      $sortcolor[$m]=$sortcolor[$m]." ".$color1[$m]."%"; }
  $sumpor=100-$sumpor;
  $sortcolor[-1]="$sortcolor[-1] $sumpor%";
  my @dato, push (@dato,\@sortcolor); push (@dato,\@DATOSsort);
  my $My_graph = new GIFgraph::pie( 600, 500 ),
  $My_graph->set( 'title' => "$titulo", 'label' => "$Y $X",
                  dclrs => [ qw(gold lgreen lyellow lgray orange yellow
                             green dyellow dgreen lred lorange ) ],
                  'axislabelclr' => 'black' );
  $My_graph->set_label_font(gdMediumBoldFont);
  $My_graph->set_value_font(gdMediumBoldFont);
  $My_graph->plot_to_gif( "$ENV{'PATHSICU'}/temp/$El_archivo.gif", \@dato ),
 }
sub bynumber { $a <=> $b; }
##////////////////////////////////////////////////////////////////
##              FIN DE PROGRAMA awktk.pl
##////////////////////////////////////////////////////////////////
```

5.4.3 Programa SICU:hawktk.pl

El módulo hawktk.pl trabaja de una manera similar a awktk.pl con la excepción de que filtra información de un archivo *histórico*, que generan las consultas del sistema. Este archivo se encuentra en la máquina que solicita las peticiones, por lo que solicitar un reporte de tipo histórico no implica transferir archivos a través de la red. El formato de línea del archivo histórico resume los datos de todas las consultas realizadas por el sistema de la siguente manera:

```
Servidor:Anio:Mes:Dia:Opc:Total
```

Servidor	Nombre del servidor.
Anio	Año del reporte.
Mes	Mes del reporte.
Dia	Día del reporte.
Opc	Tipo de reporte, por usuario, comando, de cpu, memoria, etc.
Total	Se registra el valor total del consumo de acuerdo al tipo de reporte.

5.5 Interfaz Gráfica

El módulo para la creación de la interfaz gráfica de usuario permite, de una manera fácil, la comunicación y manipulación de todos los módulos involucrados para el funcionamiento del SICU. El usuario manipula en un ambiente de ventanas todas las órdenes disponibles a su elección.

5.5.1 Perl/Tk

Las plataformas de ventanas (Apple Mac, X window y Microsoft Windows), proporcionan un nivel de programación para crear y manejar ventanas, para reportar eventos referentes al mouse y el teclado y para dibujar elementos gráficos como líneas o círculos. El problema es que dibujar desde la forma más simple toma una cantidad considerable de código y se tienen que leer cientos de páginas de la

documentación. Los patrones más usados de código para interfaces de usuarios han sido definidos dentro de los llamados *widgets* (llamados "controles" en el mundo Microsoft). Ejemplos son los **buttons, scrollbars, listbox.**

Para la mayoría de los sistemas UNIX, **X window** es la plataforma de ventanas. Varios conjuntos de herramientas de widgets se han construido sobre X: Athena, InterViews, Motif y TK. La ventaja de Tk es que no necesita C o C++ para manipular los widgets, se construyen interfaces rápidamente y es de distribución gratuita.

Tk fue desarrollado para ser manejado por el lenguaje **Tcl**[25]. Malcolm Beattie hizo un primer intento para proporcionar una capa de Perl que internamente utilizará Tcl para utilizar las librerías Tk.

Nick Ing-Simmons quitó de Tk todo el código Tcl y generó una capa portable para añadir Tk a otros lenguajes; este esfuerzo es llamado *ptk* (Tk portable), también añadió una extensión a Perl, de esta combinación de pTk y Perl surgió el módulo Tk.pm[Sri97], el cual se utilizó para crear la interfaz gráfica de usuario del SICU. (figura 5.8).

[25] Tcl y Tk fueron desarrollados por el Dr. John Ousterhout

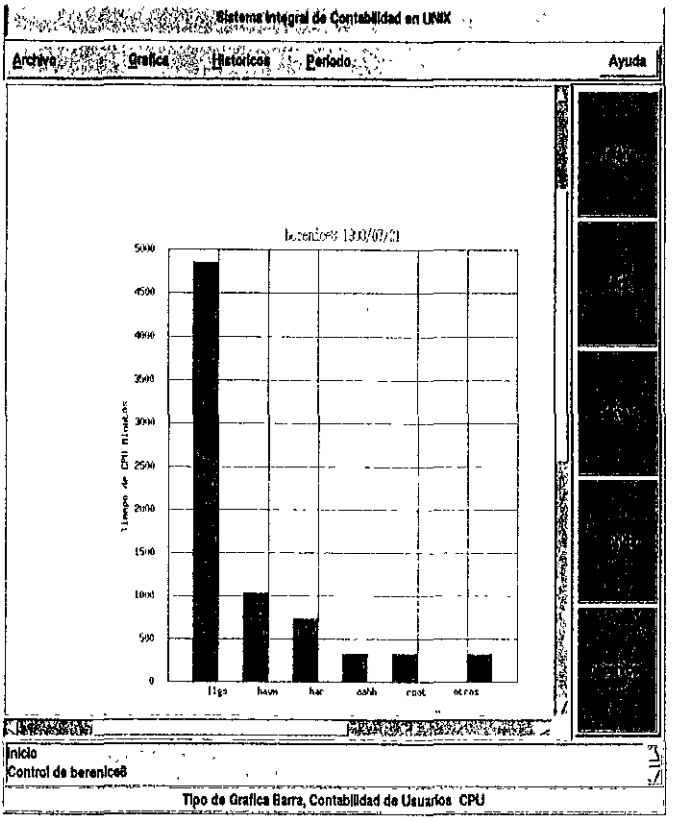

Figura 5.8: Interfaz del Sistema SICU

5.5.2 Programa SICU:sicu.pl

```
#!/opt/local/bin/perl -w
##////////////////////////////////////////////////////////////
##  Programa Interfaz:sicu.pl
##
##  Autor: Javier De La Rosa Mondragon
##
##  Departamento de Supercomputo
##  Direccion General de Servicios de Computo Academico
##  Universidad Nacional Autonoma de Mexico
##////////////////////////////////////////////////////////////
use English;
use Tk;         #  Uso de extension  Tk  (Sistema de Ventanas)
use Tk::widgets qw/Dialog ErrorDialog/;
use Tk::LabEntry;
use File::Basename;
##-----------------------------------------------------------
##  Declaracion de rutinas y variables Globales
##-----------------------------------------------------------
use subs qw/Ya_dir_img Ya_img_img do_imp/;
use subs qw/inswt invoke lsearch show_stat /;
use vars qw/$con_izq $botonser/;
use vars qw/$VP $FONT $WIDTRIB $NOM /;
use vars qw/$Imp_graph $Sav_graph $Arc_gif $chk_serv/;
use vars qw/$server $nombre @DIA $El_archivo/;
$nombre='';
use vars qw/$Gra $Opc $default/; $Gra=0; $Opc=0;
my $Barra_gra=''; my $Barra_opc=''; my $box1; $_='date';
chomp(@DIA=split());
##-----------------------------------------------------------
##  Rutina principal controladora de ventanas
##-----------------------------------------------------------
`mkdir $ENV{'PATHSICU'}/temp`; my $VP = new MainWindow;
##-----------------------------------------------------------
##  Declaracion de   variables para toplevels de servidores
##  y Baras de estado
##-----------------------------------------------------------
use vars  qw/$top_ser/; my $box0;
##-----------------------------------------------------------
```

```
##  Cuadro principal
##-----------------------------------------------------------------
$VP->title('SICU'); $VP->iconname('SICU');
my $cuad0=$VP->Frame(-relief => 'groove',
        -borderwidth => 5)->pack(-fill=>'both',-expand=>'both');
##-----------------------------------------------------------------
##  Titulo
##-----------------------------------------------------------------
my $cu1=$cuad0->Label(-text=>'Sistema Integral Contabilidad en UNIX',
            -relief => 'sunken',-borderwidth => 6,
            width => '100',height => '2',
            )->pack(-side => 'top',-fill => 'x');
##-----------------------------------------------------------------
##  Barra menu principal
##-----------------------------------------------------------------
my $w_menu=$cuad0->Frame(-relief => 'groove', -borderwidth => 5 );
   $w_menu->pack(-side => 'top', -fill => 'x');
##-----------------------------------------------------------------
##  Menu Cascada de Archivo
##-----------------------------------------------------------------
my $A=$w_menu->Menubutton(-text => 'Archivo                ',
                          -underline => 0);
    $A->command(-label => 'Abrir...', -command => sub {Abrir()},
                -accelerator => '   Alt+a',
                -underline   => 0);
    $VP->bind('<Alt-a>' => sub {Abrir()});
    $A->command(-label => 'Salvar...', -command => sub {Salvar()},
                -accelerator => '   Alt+v',
                -underline   => 3);
    $VP->bind('<Alt-v>' => sub {sub {Salvar()}});
    $A->command(-label => 'Imprimir...', -command=>sub{Imprime()},
                -accelerator => '   Alt+i',
                -underline   => 0);
    $VP->bind('<Alt-i>' => sub {Imprime()});
    $A->separator;
    $A->command(-label => 'Salir...', -command => sub {Salirse(1)},
                -accelerator => '   Alt+s', -underline   => 0);
    $VP->bind('<Alt-s>' => sub {Salirse(1)});
    $A->pack(-side => 'left'); #empaqueta Archivo
```

```
##-------------------------------------------------------------
##  Frame para arra de estado
##-------------------------------------------------------------
my $w_barra = $cuad0->Frame(-relief => 'groove', -borderwidth => 2 );
    $w_barra->pack(-side => 'bottom', -fill => 'x');
##-------------------------------------------------------------
##  Menu cascada Grafica
##-------------------------------------------------------------
my $G=$w_menu->Menubutton(-text => 'Grafica', -underline => 0);
   $G->cascade(-label => 'Tipo...' ); $G->cascade(-label => 'Opciones');
##-------------------------------------------------------------
##   Subgrafica Tipo de Grafica
##-------------------------------------------------------------
    my $Tm = $G->cget(-menu);
    my $tipo = $Tm->Menu;
    $G->entryconfigure('Tipo...', -menu => $tipo);
    $tipo->radiobutton(-label => 'Barra',
                   -variable => \$Gra, -value => 1,
                   -command => sub {BARRA()});
    $tipo->radiobutton(-label => 'Pie',
                   -variable => \$Gra, -value => 2,
                   -command => sub {BARRA()}); $tipo->invoke(1);
##-------------------------------------------------------------
##   Subgrafica Opciones de peticion
##-------------------------------------------------------------
    my $Om = $G->cget(-menu); my $opci = $Om->Menu;
    $G->entryconfigure('Opciones...', -menu => $opci);
    $opci->cascade(-label => 'Por usuario');
    $opci->cascade(-label => 'Por comando');
    my $opci1 = $opci->Menu;
    $opci->entryconfigure('Por usuario', -menu => $opci1);
    $opci1->radiobutton((-label => 'CPU',
                   -variable => \$Opc, -value => 1,
                   -command => sub {BARRA()});
    $opci1->radiobutton(-label => 'Memoria',
                   -variable => \$Opc, -value => 2,
                   -command => sub {BARRA()});
    $opci1->radiobutton(-label => 'Disco',
                   -variable => \$Opc, -value => 3,
```

```
                    -command => sub {BARRA()});
    $opci1->radiobutton(-label => 'Tiempo de Conexion',
                    -variable => \$Opc, -value => 4,
                    -command => sub {BARRA()}); $opci1->invoke(1);
    my $opci2 = $opci->Menu;
    $opci->entryconfigure('Por comando', -menu => $opci2);
    $opci2->radiobutton(-label => 'CPU',
                    -variable => \$Opc, -value => 5,
                    -command => sub {BARRA()});
    $opci2->radiobutton(-label => 'Memoria',
                    -variable => \$Opc, -value => 6,
                    -command => sub {BARRA()});
    $G->pack(-side => 'left');  #empaqueta Grafica
##-------------------------------------------------------------------
##   Menu cascada reportes Historicos
##-------------------------------------------------------------------
my $R = $w_menu->Menubutton(-text => 'Historicos .      ',
                            -underline => 0);
    $R->command(-label =>'Por Dia...',-command=>sub{TOP_H(1,1)},
                -accelerator => '   Alt+d', -underline   => 4);
    $VP->bind('<Alt-d>' =>sub {TOP_H(1,1)});
    $R->command(-label =>'Por Mes...',-command=>sub{TOP_H(2,1)},
                -accelerator => '   Alt+m', -underline   => 4);
    $VP->bind('<Alt-m>' => sub {TOP_H(2,1)});
    $R->command(-label =>'PorA#o...',-command=>sub{TOP_H(3,1)},
                -accelerator => '   Alt+o', -underline   => 6);
    $VP->bind('<Alt-o>' => sub {TOP_H(3,1)});
    $R->pack(-side => 'left');
##-------------------------------------------------------------------
##  Menu cascada reportes Periodicos
##-------------------------------------------------------------------
my  $P=$w_menu->Menubutton(-text => 'Periodo       ',
                           -underline => 0);
    $P->command(-label =>'Por Mes...',-command=>sub{TOP_H(1,2)},
                -accelerator => '   Alt+e', -underline   => 5);
    $VP->bind('<Alt-e>'=>sub {TOP_H(2,2)});
    $P->command(-label=>'Por Anio...',-command=>sub{TOP_H(2,2)},
                -accelerator => '   Alt+n', -underline   => 5);
    $VP->bind('<Alt-n>' => sub {TOP_H(3,2)});
```

```
    $P->pack(-side => 'left');
##-------------------------------------------------------------
##   Menu AYUDA
##-------------------------------------------------------------
my $Bot_Ayu = $w_menu->Button(-text => 'Ayuda',
    -command => sub {Ayuda()}); $Bot_Ayu->pack(-side => 'right');
##-------------------------------------------------------------
##  Scroll para servidores
##-------------------------------------------------------------
my $w_barra1=$cuad0->Frame(-relief =>'groove',-borderwidth=>2);
   $w_barra1->pack(-side => 'bottom', -fill => 'x');
   $box1 = $w_barra1->Listbox(-relief => 'sunken',
                              -width => -1, # Shrink to fit
                              -height => 2,
                              -setgrid => 'yes');
   $box1->insert('end','Inicio' );
   my $scroll=$w_barra1->Scrollbar(-command => ['yview', $box1],
                               -relief => 'ridge') ;
      $box1->configure(-yscrollcommand => ['set', $scroll]);
      $box1->pack(-side => 'left',-fill =>'x',-expand =>'yes');
      $scroll->pack(-side => 'left', -fill => 'y');
      $box1->see('end');
##-------------------------------------------------------------
##   Frame para Servidores
##-------------------------------------------------------------
my $conte = $cuad0->Frame(-relief => 'groove', -borderwidth => 7);
    $conte->pack(-fill => 'y' ,-side => 'right');
$conte->Button(-text => 'Polaris',-relief => 'ridge',
              -height => '7', -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'polaris')}
              )->pack(-fill => 'both',-expand  => 'both');
$conte->Button(-text => 'Mira',-relief => 'ridge',
              -height => '7', -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'mira')}
              )->pack(-fill => 'both',-expand  => 'both');
$conte->Button(-text => 'Deneb',-relief => 'ridge',
              -height => '7', -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'deneb')}
              )->pack(-fill => 'both',-expand  => 'both');
```

```
$conte->Button(-text => 'diphda',-relief => 'ridge',
              -height => '7', -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'diphda')}
              )->pack(-fill => 'both',-expand  => 'both');
$conte->Button(-text => 'berenice8',-relief => 'ridge',
              -height => '7', -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'berenice8')}
              )->pack(-fill => 'both',-expand  => 'both');
##-----------------------------------------------------------------
##   Frame para Graficas
##-----------------------------------------------------------------
my $conte1=$cuad0->Scrolled('Canvas', -relief => 'ridge',
                  -borderwidth =>3, -scrollbars => 'se',
                  -scrollregion=>['-5c','-5c','30c','30c'],
                  -background=>'white');
   $conte1->pack(-fill => 'both' ,-expand  => 'both');
MainLoop;
##-----------------------------------------------------------------
##  FIN Rutina principal controladora de ventanas
##-----------------------------------------------------------------
##-----------------------------------------------------------------
# Subrutina Ventanas  top para  Servidores
##-----------------------------------------------------------------
sub TOP {
   my ($top_serv,$server) = @_; $El_archivo = '';
   $top_serv = $VP->Toplevel; $top_serv->title($server);
   $top_serv->iconname($server);
   $top_serv->Label(-text=>"Servidor $server")->pack(-side=>'top');
   my $frame_top=$top_serv->Frame(-relief => 'flat',
   -borderwidth=>5);$frame_top->pack(-fill=>'both',-expand=>'both',
   -side =>'left');
   my $frame_top2=$top_serv->Frame(-relief => 'flat',
   -borderwidth=>5);$frame_top2->pack(-fill=>'both',-expand=>'both'
   -side =>'right');
      $frame_top->Label(-text =>'Transferir  Archivo:')
      ->pack(-side => 'top',-fill => 'x') ;
      $frame_top->Label(-text =>'Anio:')
      ->pack(-side => 'top',-fill => 'x') ;
   my $Entrada = $frame_top2->LabEntry(-width => 40,
```

```
      -labelPack => [qw/-side left -anchor w/],
      -textvariable => \$El_archivo)->pack(qw/-side top/);
   my $Enty2 = $frame_top2->LabEntry(-width => 40,
      -labelPack => [qw/-side left -anchor w/],
      -textvariable => \$DIA[-1])->pack(qw/-side top/);
   my $ejecutar=$frame_top2->Button(-text=>'Ejecutar')
      ->pack(-side =>'left');
   my $commando= sub{do_sys($server,$El_archivo,$top_serv,1,$DIA[-1])};
      $ejecutar->configure(-command => $commando);
      $Entrada->bind('<Return>' => $commando);
      $frame_top2->Button(-text => 'Salir',
      -command => sub {destroy $top_serv})
      ->pack(-side => 'right');
      $box1->insert('end',"Control de $server" );$box1->see('end');
}
##-----------------------------------------------------------------
##   Subrutina ejecucion cliente.pl
##-----------------------------------------------------------------
sub do_sys {
  my ($server,$El_archivo,$top_serv,$tipo,$anio) = @_;
  if($El_archivo=~/(0[1-9]|1[0-2])([0-2][0-9]|3[0-1])/
     || $tipo==2){ $NOM=1;
     if !(-s "$ENV{'PATHSICU'}/temp/$server$El_archivo")) {
      $chk_serv=system "$ENV{'PATHSICU'}/bin/cli_fork.pl\
                        12345 $server $El_archivo $tipo "; }
     if (-e "$ENV{'PATHSICU'}/temp/${server}acct"){ #Informe  acct on /off
      Error_datos('9');
      unlink "$ENV{'PATHSICU'}/temp/${server}acct"; }
     if (-s "$ENV{'PATHSICU'}/temp/$server$El_archivo") {
         $Arc_gif="$server$El_archivo";
      system "$ENV{'PATHSICU'}/bin/awktk.pl $Gra $Opc\
                $server$El_archivo $server $El_archivo $tipo $anio";
            if (-s "$ENV{'PATHSICU'}/temp/$server$El_archivo.gif"){
             $conte1->Photo('imggif', -file =>
              "$ENV{'PATHSICU'}/temp/$server$El_archivo.gif");
             $conte1->create('image','10c','10c',-image => 'imggif');
             $Imp_graph=1; }
            else{ Error_datos('1'); } }
       else { unlink "$ENV{'PATHSICU'}/temp/$server$El_archivo";
```

```
                if($chk_serv==0){ Error_datos('3'); }
                else{ Error_datos('4'); } }
    }
    else{ Error_datos('8'); }
    destroy $top_serv;
}
##-----------------------------------------------------------------
##  ventana top para peticion Historica o Periodica
##-----------------------------------------------------------------
sub TOP_H {
    my ($reporte,$h_p) = @_;
    if($h_p==1){ Salirse(2); }
    my $servidor = ''; my $delmes=''; my $delanio="$DIA[-1]";
    my $top_hist;
    if($h_p==1){ $top_hist = $VP->Toplevel;
       $top_hist->title(Historia); $top_hist->iconname(Historia); }
    if($h_p==2){ $top_hist = $VP->Toplevel;
    $top_hist->title(Periodo); $top_hist->iconname(Periodo); }
    my $frame_toph=$top_hist->Frame(-relief => 'groove',
                  -borderwidth => '5', -height => '10');
    $frame_toph->pack(-fill=>'both',-expand=>'both',-side=>'left');
    $frame_toph->Label(-text =>' ',-relief=>'ridge',-height=>'17',
    -background => 'DarkSlateGray3',
    )->pack(-side => 'left',-fill => 'both', -expand => 'both');
    my $f1=$frame_toph->Frame()->pack(-fill => 'both',
    -expand => 'both',-side =>'top');
    $f1->Label(-text=>'Servidor:')->pack(-side=>'left',-fill=>'x');
    my $Servidor = $f1->LabEntry(-width => 10,
    -labelPack => [qw/-side left -anchor w/],
    -textvariable => \$servidor)->pack(qw/-side right/);
    if ($reporte==1){my$f4=$frame_toph->Frame()->pack(-fill=>'both',
    -expand => 'both',-side =>'top');
    $f4->Label(-text =>' Mes:')
    ->pack(-side => 'left',-fill => 'x');
    my $Delmes = $f4->LabEntry(-width => 10,
    -labelPack => [qw/-side left -anchor w/],
    -textvariable => \$delmes)->pack(qw/-side right/);
    }
    if ($reporte==1 || $reporte==2 ){
```

```
      my $f6=$frame_toph->Frame()->pack(-fill => 'both',
      -expand => 'both', -side =>'top');
      $f6->Label(-text =>'A#o: ')
      ->pack(-side => 'left',-fill => 'x');
      my $Delanio = $f6->LabEntry(-width => 10,
      -labelPack => [qw/-side left -anchor w/],
      -textvariable => \$delanio)->pack(qw/-side right/); }
   $frame_toph->Button(-text => 'Salir',
   -command=>sub{destroy $top_hist})->pack(-side=>'bottom');
   my $ejecutar=$frame_toph->Button(-text =>'Ejecutar')
   ->pack(-side => 'bottom');
   my $commando = sub {Hacer_GH($servidor,$reporte,$delmes,
                        $delanio,$h_p,$top_hist)};
   $ejecutar->configure(-command => $commando);
   $box1->insert('end',"historia" );$box1->see('end');
}
##-------------------------------------------------------------------
##   Subgrafica Grafica Historica
##-------------------------------------------------------------------
sub Hacer_GH {
  my ($servidor,$reporte,$mes,$anio,$tipo,$top_hist) = @_;
  my @arr_ser=("polaris","mira","deneb","berenice8","ursa");
  my $Err_val2=0; $NOM=2;
  case5:{
  if ($reporte==1){
    for ($i=0;$i<@arr_ser;$i++){
     if($servidor eq $arr_ser[$i]){ $Err_val2=0; last;}
     else { $Err_val2=1; } }
    if ($Err_val2==1){Error_datos(10);last case5;}
    else{
      if(!($mes=~/(0[1-9]|1[0-2])/)){$Err_val2=1;Error_datos(11);
        last case5;}
      if(!($anio=~/199[7-9]|200[1-9]/)){$Err_val2=1;Error_datos(12);
        last case5;} }
    $nombre="${servidor}D_${mes}${anio}_$Opc"; last case5;}
  if ($reporte==2){
      for ($i=0;$i<@arr_ser;$i++){
        if($servidor eq $arr_ser[$i]){ $Err_val2=0; last;}
        else { $Err_val2=1; } }
```

```
          if ($Err_val2==1){Error_datos(10);last case5;}
          else{
            if(!($anio=~/199[7-9]|200[1-9]/)){$Err_val2=1;
                Error_datos(12);last case5;} }
       $nombre="${servidor}M_${anio}_$Opc";last case5;}
   if ($reporte==3){
       for ($i=0;$i<@arr_ser;$i++){
         if($servidor eq $arr_ser[$i]){ $Err_val2=0; last;}
          else { $Err_val2=1; } }
      if ($Err_val2==1){Error_datos(10);last case5;}
      $nombre="${servidor}_A$Opc";last case5;}
 }
 if ($Err_val2==0 && $tipo==1){
      system "$ENV{'PATHSICU'}/bin/hawktk.pl $servidor $reporte\
                  $mes $anio $Opc";
       if (-s "$ENV{'PATHSICU'}/temp/$nombre.gif"){
         $conte1->Photo('imggif', -file =>
                  "$ENV{'PATHSICU'}/temp/$nombre.gif");
         $conte1->create('image','10c','10c',-image => 'imggif');
         $Imp_graph=1; }
       else{ Error_datos('1'); }
      destroy $top_hist; }
if($Err_val2==0&&$tipo==2){do_sys($servidor,$mes,$top_hist,2,$anio)
}
##-----------------------------------------------------------------
##   Renovacion de la barra  de estado
##-----------------------------------------------------------------
sub BARRA {
case:{
  if ($Gra==1) {$Barra_gra='Barra'; last case;}
  if ($Gra==2) {$Barra_gra='Pie'; last case;}
}
case2:{
  if ($Opc==1) {$Barra_opc="Usuarios  CPU"; last case2;}
  if ($Opc==2) {$Barra_opc="Usuarios Memoria"; last case2;}
  if ($Opc==3) {$Barra_opc="Usuarios Disco"; last case2;}
  if ($Opc==4) {$Barra_opc="Usuarios Tiempo Conexion";last case2;}
  if ($Opc==5) {$Barra_opc="Comandos  CPU"; last case2;}
  if ($Opc==6) {$Barra_opc="Comandos  Memoria"; last case2;}
```

```
}
  $des = "Tipo de Grafica $Barra_gra, Contabilidad de $Barra_opc";
  $box0->destroy if Exists($box0);
  $box0 = $w_barra->Label(-text => "$des",
                          -relief => 'flat',-borderwidth => 2,
                          width => '100',height => '1'
                          )->pack(-side => 'top',-fill => 'x');
}
##-----------------------------------------------------------------
##   Rutinas Abrir archivos graficos
##-----------------------------------------------------------------
sub Abrir {
    my $top_abrir = $VP->Toplevel;
    $top_abrir->title('Abrir'); $top_abrir->iconname('Abrir');
    $con_izq = $top_abrir->Frame(-relief => 'groove',
               -borderwidth => 5)->pack(-side =>'left',-fill => 'y');
    my $con_dir=$con_izq->Frame->pack(-side =>'top');
    my $dir_Etiq = $con_dir->Label(-text => 'Directorio:');
    my $dir_Img = "$ENV{'PATHSICU'}/Img";
    $dir_nom = $con_dir->Entry(-width => 25, -textvariable => \$dir_Img);
    my $frog0 = $top_abrir->Frame; my $frog = $con_izq->Frame;
    my $file_label = $frog->Label(-text => 'Archivo:');
    my $f = $frog->Frame; my(@pl) = qw/-side top -anchor w/;
    $dir_Etiq->pack(-side =>'left' ); $dir_nom->pack(-side =>'left');
    $con_izq->Button(-text => 'Salir',
            -command => sub {destroy $top_abrir})->pack(-side => 'bottom');
    $con_izq->Button(-text => 'Imprimir',
                -command => sub {Imprime()})->pack(-side => 'bottom');
    $frog0->pack(-side =>'left'); $frog->pack(qw/-side top/);
    my $toad = $frog0->Frame; $toad->pack(qw/-side right -expand yes/);
    $file_label->pack(-side =>'left'); $f->pack(@pl);
    my $f_list = $f->Listbox(-width => 20, -height => 10);
    $dir_nom->bind('<Return>' => [\&Ya_dir_img, $f_list, \$dir_Img]);
    my $f_scroll = $f->Scrollbar(-command => [$f_list => 'yview']);
    $f_list->configure(-yscrollcommand => [$f_scroll => 'set']);
    @pl = qw/-side left -fill y -expand 1/;
    $f_list->pack(@pl); $f_scroll->pack(@pl);
    my $j;
    local *DIR;
```

```
    opendir DIR, "$ENV{'PATHSICU'}/Img";
    foreach $j (sort readdir DIR) {
       $f_list->insert('end', $j);
     }
    closedir DIR;
    my $image2a=$top_abrir->Frame(-relief =>'groove',
                                  -borderwidth =>5)->Photo;
   $f_list->bind('<Double-1>'=>[\&Ya_img_img,$image2a,\$dir_Img]);
   my $image_label = $toad->Label();
   my $image = $toad->Label(-image => $image2a);
   @pl = qw/-side top -anchor w/; $image_label->pack(@pl);
   $image->pack(-expand => 1 );
}
sub Ya_dir_img {
    my($e, $l, $dir_nom) = @_;
    $l->delete(0, 'end'); my $i;
    local *DIR;
    opendir DIR, $$dir_nom;
    foreach $i (sort readdir DIR) {
       $l->insert('end', $i);
    }
    closedir DIR;
}
sub Ya_img_img {
    my($l, $i, $dir_nom) = @_; my $e = $l->XEvent;
    my($x, $y) = ($e->x, $e->y);
    $i->configure(-file => "$$dir_nom/" . $l->get("\@$x,$y"));
    $Arc_gif="$$dir_nom/" . $l->get("\@$x,$y");
    $NOM=3; $Imp_graph=1;
}
##-----------------------------------------------------------------
##   Rutina Salvar archivos
##-----------------------------------------------------------------
sub Salvar {
if($Imp_graph){
  my $top_salvar = $VP->Toplevel;
    $top_salvar->title('Salvar'); $top_salvar->iconname('Salvar');
    $contenedor = $top_salvar->Frame(-relief => 'groove',
                          -borderwidth => 5)->pack(side =>'left',
```

```
                        -fill => 'both', -expand => 'both');
    my $con_dir=$contenedor->Frame->pack(side =>'top');
    my $dir_Etiq = $con_dir->Label(-text => 'Salvar:');
    my  $arc_salvar;
    if($NOM==1){
     $arc_salvar = "$ENV{'PATHSICU'}/Img/$Arc_gif$Gra$Opc.gif";
    }
    else{ $arc_salvar = "$ENV{'PATHSICU'}/Img/$nombre.gif"; }
    $dir_nom=$con_dir->Entry(-width=>25,-textvariable=>\$arc_salvar);
    my $frog0 = $top_salvar->Frame; my $frog = $contenedor->Frame;
    my $file_label = $frog->Label(-text => "Archivos:");
    my $f = $frog->Frame; my(@pl) = qw/-side top -anchor w/;
    $dir_Etiq->pack(-side =>'left' ); $dir_nom->pack(-side =>'left');
    $contenedor->Button(-text => "Salir",
                -command =>sub{destroy $top_salvar})
               ->pack(-side => 'bottom');
    $contenedor->Button(-text => 'salvar',
               -command => sub {salvar2()})->pack(-side =>'bottom');
    $frog0->pack(-side =>'left'); $frog->pack(qw/-side top/);
    my $toad = $frog0->Frame;
    $toad->pack(qw/-side right -expand yes/);
    $file_label->pack(-side =>'left'); $f->pack(@pl);
    my $f_list = $f->Listbox(-width => 20, -height => 10);
    $dir_nom->bind('<Return>'=>[\&Ya_dir_img,$f_list,\$arc_salvar]);
    my $f_scroll = $f->Scrollbar(-command => [$f_list => 'yview']);
    $f_list->configure(-yscrollcommand => [$f_scroll => 'set']);
    @pl = qw/-side left -fill y -expand 1/;
    $f_list->pack(@pl); $f_scroll->pack(@pl); my $j;
    local *DIR;
    opendir DIR, "$ENV{'PATHSICU'}/Img";
    foreach $j (sort readdir DIR) {
       $f_list->insert('end', $j);
     }
    closedir DIR;
}
else{ Error_datos('5'); }
}
sub salvar2 {
if ($NOM==1){
```

```
'cp "$ENV{'PATHSICU'}/temp/$Arc_gif.gif"
          "$ENV{'PATHSICU'}/Img/$Arc_gif$Gra$Opc.gif"'; }
else{
'cp"$ENV{'PATHSICU'}/temp/$nombre.gif""$ENV{'PATHSICU'}
    /Img/$nombre.gif"';}
}
##-----------------------------------------------------------------
##   Subrutina Imprimir
##-----------------------------------------------------------------
sub Imprime (){
if($Imp_graph){
 my $La_impresora='';
 my   $top_imp = $VP->Toplevel;
   $top_imp->title("Imprimir"); $top_imp->iconname('Imprimir');
   $top_imp->Label(-text => "Introducir nombre Impresora")
           ->pack(-side =>'top');
  my $frame_top=$top_imp->Frame(-relief => 'flat',
                                -borderwidth => 5);
     $frame_top->pack(-fill=>'both',-expand=>'both',-side=>'left');
  my $frame_top2=$top_imp->Frame(-relief=>'flat',-borderwidth=>5);
     $frame_top2->pack(-fill=>'both',-expand=>'both',-side=>'right');
     $frame_top->Label(-text=>'Impresora:')
               ->pack(-side => 'top',-fill => 'x');
  my $Entrada=$frame_top2->LabEntry(-width => 40,
       -labelPack => [qw/-side left -anchor w/],
       -textvariable => \$La_impresora)->pack(qw/-side top/);
  my $ejecutar=$frame_top2->Button(-text =>'Imprimir')
      ->pack(-side=>'left');
  my $commando = sub {do_imp($La_impresora)};
    $ejecutar->configure(-command => $commando);
    $Entrada->bind('<Return>' => $commando);
  $frame_top2->Button(-text=>'Salir', -command=>sub{destroy top_imp})
            ->pack(-side => 'right');
      $box1->insert('end',"Imprir en $La_impresora" )
      ;$box1->see('end');
 }
 else{ Error_datos('6'); }
}
sub do_imp {
```

```
my ($impre) =@_;
case:{
if($NOM==1){ 'giftoppm $ENV{'PATHSICU'}/temp/$Arc_gif.gif
 > $ENV{'PATHSICU'}/temp/IMP.ppm'; last case; }
if($NOM==2){ 'giftoppm $ENV{'PATHSICU'}/temp/$nombre.gif
 > $ENV{'PATHSICU'}/temp/IMP.ppm'; last case; }
if($NOM==3){ 'giftoppm $Arc_gif
 > $ENV{'PATHSICU'}/temp/IMP.ppm'; last case; print "entro al3";}
}
'pnmtops $ENV{'PATHSICU'}/temp/IMP.ppm > $ENV{'PATHSICU'}/temp/IMP.ps';

if((system ("lpr -P$impre $ENV{'PATHSICU'}/temp/IMP.ps")) > 0){
Error_datos('7'); }
}
##-----------------------------------------------------------------
##   Subrutina  control de errores
##-----------------------------------------------------------------
sub Error_datos{
my ($num_err) = @_;
case3:{
 if($num_err eq '1')
 {$error_tex="No hay datos para esta solicitud barra";last case3;}
 if($num_err eq '2')
 {$error_tex="No hay datos para esta solicitud pie"; last case3;}
 if($num_err eq '3'){$error_tex="Archivo no existe";last case3;}
 if($num_err eq '4'){$error_tex="Peticion al servidor fallida";last case3;}
 if($num_err eq '5'){$error_tex="No hay grafica para salvar";last case3;}
 if($num_err eq '6'){$error_tex="No hay grafica para imprimir";last case3;}
 if($num_err eq '7'){$error_tex="Impresora no valida";last case3;}
 if($num_err eq '8'){$error_tex="Nombre de Archivo no valido";last case3;}
 if($num_err eq '9'){$error_tex="Servidor con contabilidad off";last case3;}
 if($num_err eq '10'){$error_tex="Servidor no existe";last case3;}
 if($num_err eq '11'){$error_tex="Mes no existe ";last case3;}
 if($num_err eq '12'){$error_tex="A#0 no existe";last case3;}
}
 my $DialogoError=$VP->Dialog(
              -title          => 'Error',
              -text           => "$error_tex",
              -bitmap         => 'error',
```

```
                -buttons        => ['ok'],);
               my $button = $DialogoError->Show('-global'); }
##-----------------------------------------------------------------
##  Subrutina Ayuda
##-----------------------------------------------------------------
sub Ayuda {system("$ENV{'PATHSICU'}/bin/ayuda.pl &");}
##-----------------------------------------------------------------
##  Subrutina Salir
##-----------------------------------------------------------------
sub Salirse {
 my ($sal)=@_;
 if (-s "$ENV{'PATHSICU'}/temp/arch_aux") {
 `egrep -vf $ENV{'PATHSICU'}/rpt/hist_rpt
 $ENV{'PATHSICU'}/temp/arch_aux > $ENV{'PATHSICU'}/temp/nuev_aux`; }
 if (-s "$ENV{'PATHSICU'}/temp/nuev_aux") {
   `cat  $ENV{'PATHSICU'}/temp/nuev_aux >>
         $ENV{'PATHSICU'}/rpt/hist_rpt`;
    `sort -t: +0 +1n +2n +3n +4n -o $ENV{'PATHSICU'}/rpt/hist_rpt
     $ENV{'PATHSICU'}/rpt/hist_rp t`; }
 if ($sal==1){ system ("rm -r $ENV{'PATHSICU'}/temp"); exit; }
}
##////////////////////////////////////////////////////////////////
##              FIN DE PROGRAMA sicu:sicu.pl
##////////////////////////////////////////////////////////////////
```

5.5.3 Ayuda

El SICU ofrece facilidades de ayuda interactiva que permite al usuario obtener una respuesta sin abandonar la interfaz. El tipo de ayuda es llamada *añadida*, es decir, es incorporada al sistema después de haber sido terminado.

La incorporación de la ayuda es de gran utilidad, es una ventana separada fácil de usuar, basta manipular un puntero de mouse para señalar el título referente al concepto que se requiera. (figura 5.9).

Los mensajes de error del SICU interactuan con el usuario y suministran los títulos de la ayuda, donde se encuentra información referente.

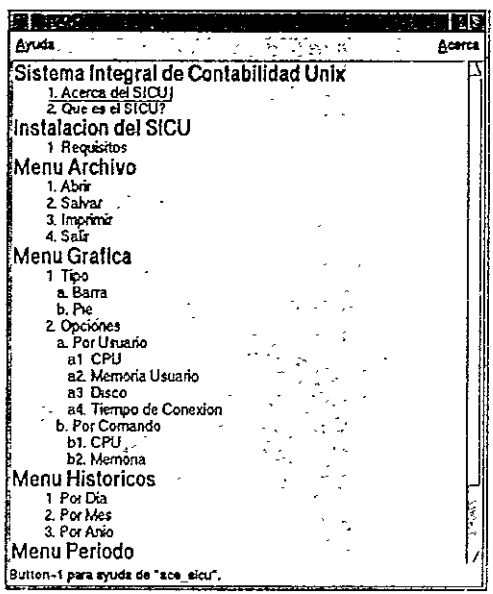

Figura 5.9: Ventana de Ayuda

5.6 Instrumentación del SICU

Se describe la operación del SICU en el Laboratorio de Visualización de la Dirección General de Servicios de Cómputo Académico en la Universidad Nacional Autónoma de México.

5.6.1 Máquinas Involucradas

Las máquinas que SICU analizará en operación normal serán todas las del Departamento de Supercómputo, Área de seguridad en Cómputo y del Laboratorio de Visualización, para efectos prácticos solo se han incluido 6 máquinas.

- **Polaris**, polaris.labvis.unam.mx, Silicon Graphics del laboratorio de Visualización.
- **Mira**, mira.labvis.unam.mx, Silicon Graphics del laboratorio de Visualización.
- **Deneb**, deneb.labvis.unam.mx, Silicon Graphics del laboratorio de Visualización.
- **Ursa**, ursa.labvis.unam.mx, Silicon Graphics del laboratorio de Visualización.
- **Berenice8**, berenice8.cray-origin.unam.mx, Silicon Graphics Origin 2000.
- **Charanda**, charanda.super.unam.mx, Sun Sparc IV del Departamento de Supercómputo.

5.6.2 Sistemas Operativos

Los sistemas operativos son un factor muy importante para la operación del SICU, de ellos depende su correcto funcionamiento. Se mencionan los instalados en las máquinas involucradas:

- **IRIX 5.3, System V**, polaris.labvis.unam.mx
- **IRIX 6.2, System V**, mira.labvis.unam.mx
- **IRIX 6.2, System V**, ursa.labvis.unam.mx
- **IRIX 6.2, System V**, deneb.labvis.unam.mx

- IRIX 6.4, System V, berenice8.cray-origin.unam.mx
- Solaris 2.5.1 System V, charanda.super unam.mx

Estos sistemas se apegan a System V, pero no son 100% System V, contienen funcionalidades de System V y BSD. Lo importante para SICU es que el sistema operativo genere archivos de contabilidad tipo System V.

5.6.3 Configuración del Servidor

La tarea de configurar el servidor es sencilla, los pasos involucrados son:

- Instalación del Servidor del SICU.
- Definir el puerto por el cual se comunicará con el cliente
- Ejecución y pruebas con el servidor.

Instalación del Servidor de SICU

La máquina servidor debe tener disponible el compilador de Perl version 5.0 o posterior.

Se define el lugar donde quedará el programa **serv.pl**, es recomendable que se instale en el directorio.

```
/usr/local/bin/
```

Comúnmente es el directorio para aplicaciones que son agregadas al sistema

Se definen los permisos y dueño de este programa, se recomienda que el propietario sea *adm* y que los permisos únicamente le correspondan a él.

Puerto de Comunicación

Se define el puerto por donde se comunicará con el cliente, se puede utilizar cualquier puerto mayor a 1000 ya que los anteriores estan reservados para el sistema. Con cualquier editor de textos se indica el puerto elegido en la siguiente línea del programa **serv.pl**

```
my $port = shift || 12345;
```

Para fines de ejemplo se eligió el puerto 12345.

Ejecución y Pruebas con el Servidor

Se ejecuta lo siguiente:

```
% serv.pl
```

Con lo cual el servidor quedará como un programa residente en el sistema listo para contestar a sus clientes.

Las pruebas recomendadas que se deben realizar son ver que el programa quede como un proceso dentro del sistema.

```
% ps -fea | grep serv.pl
```

La instrucción anterior debe encontrar y desplegar la informacón asociada al proceso de **serv.pl**.

```
adm  6541  1  0  Jul 20 ?   0:00 /usr/bin/perl serv.pl
```

Otra prueba que se debe realizar es referente al puerto que esta ocupando el servidor y ningún otro servicio puede ocuparlo, por ejemplo:

```
% serv.pl
```

Lo anterior trata de apartar nuevamente el puerto que está utilizando el primer **serv.pl** que se ejecutó, por lo que debemos obtener lo siguiente.

```
% No puedo apartar  puerto 12345!. at serv.pl line 18.
```

Una vez realizado lo anterior se asegura el buen funcionamiento del servidor.

5.6.4 Configuración del Cliente

La tarea de configurar el cliente consiste en lo siguiente:

- Instalación del cliente de SICU.
- Definir el puerto donde el servidor espera peticiones.
- Definir los servidores a solicitar archivos de contabilidad.
- Ejecución y pruebas con el cliente.

Instalación del Cliente de SICU

La máquina cliente debe tener disponible lo siguiente:

- *Perl 5.0 o posterior.*
- Tk 4.0 o posterior.
- GIFgraph 1.01 o posterior.
- GD 1.16 o posterior.
- pmbplus

Se define el lugar donde quedarán el conjunto de directorios necesarios para el funcionamiento del cliente SICU, se recomienda crear el siguiente directorio:

```
/usr/local/sicu
```

Dentro de este directorio y con el archivo sicu.tar se hace los siguiente.

```
% tar xvf sicu.tar
```

Esto creará los siguientes subdirectorios:

/bin	Contiene los programas en Perl.
/img	Almacena archivos gráficos seleccionados.
/rpt	Guarda archivo histórico de peticiones.
/ayuda	Contiene archivos solicitados por ayuda.
/temp	Archivos temporales.

Se definen los permisos y dueño de este directorio y programas, se recomienda que el propietario sea *adm* y que los permisos únicamente le correspondan a él.

Puerto de Comunicación

Se define el puerto por dónde lo atenderá el servidor. Con cualquier editor de textos se indica el puerto elegido en la siguiente línea del programa **sicu.pl**

```
$chkserv=system"$ENV{'PATHSICU'}/bin/cli_fork.pl\
12345 $server $El_archivo $tipo";
```

El puerto es un argumento para el programa **cli.pl**, llamado por **sicu.pl** para fines de ejemplo se eligió el puerto 12345.

Definición de Servidores

El el programa **sicu.pl**, se eligen los servidores que atenderán peticiones de solicitud de sus archivos de contabilidad, se edita por cada servidor lo siguiente:

```
$conte->Button(-text=>'Polaris',-relief=>'ridge',-height=>'7',
              -background => 'DarkSlateGray3',
              -command => sub {TOP($top_ser,'polaris')})
                 ->pack(-fill => 'both',-expand  => 'both');
```

Se pasa como argumento el nombre del servidor a la llamada de la rutina **TOP**, si el servidor deseado no se encuentra en la misma red será necesario pasar en el argumento la dirección completa del servidor (polaris.labvis.unam.mx).

Ejecución y Pruebas con el Cliente

Se crea una *variable de ambiente*[26] de la siguiente manera:

```
% setenv PATHSICU /usr/local/sicu
```

La variable de ambiente PATHSICU le indica al cliente de SICU dónde localizar sus directorios y archivos necesarios para su funcionamiento.

Se ejecuta lo siguiente:

```
% /usr/local/sicu/bin/sicu.pl
```

Desplegará la interfaz gráfica de usuario, lista para ser manipulada.

[26] Variable presente en el sistema durante la sesión

Conclusiones

En este apartado se mencionan los resultados obtenidos, así como las conclusiones de este trabajo.

Dentro de las tareas del administrador de sistemas, una que es de suma importancia para el buen mantenimiento del sistema es la revisión de las bitácoras y archivos de contabilidad que genera. Esta acción debe ser llevada a cabo ya sea de manera manual o automática. El administrador requiere de soluciones o herramientas que le permitan conocer la información del área a su cargo de manera puntual, rápida y real.

En la parte de contabilidad de los recursos utilizados, los reportes gráficos que brinda el SICU ayudarán a los administradores del Departamento de Supercómputo y del Área de Seguridad en Cómputo a mantener un buen conocimiento de la utilización de los sistemas, pues ahora analizarán reportes gráficos y de una manera más fácil podrán ver los datos que revelen algún posible mal uso del sistema, entonces se procederá cuando sea necesario a optimizar código de usuarios o inclusive a realizar pruebas de seguridad.

La capacitación y actualización debe ser constante en los administradores, deben tener conciencia de la responsabilidad de administrar un sistema, de cuidar sus recursos y todo aquello que mejore el rendimiento del mismo.

Durante el desarrollo del SICU se pudo conocer la filosofía de UNIX, practicarla y desarrollar un sistema de contabilidad gráfico de acuerdo a las características generales de UNIX.

Los conceptos que fueron necesarios aprender y manejar para el desarrollo de SICU, fueron:

- Las actividades de un administrador de sistemas.
- La arquitectura general de UNIX, subsistema de procesos y subsistema de archivos.

- Archivos de configuración del sistema.

- La secuencia de inicialización del sistema de contabilidad en UNIX.

- Los formatos y tipos de archivos generados por el sistema de contabilidad de UNIX.

- La secuencia de terminación del sistema de contabilidad en UNIX.

- Las llamadas al sistema UNIX.

- Servicios de red en el sistema, herramientas de desarrollo de aplicaciones cliente/servidor.

- Procesos de servicio.

Este sistema me permitió conocer la herramienta de desarrollo para interfaces de usuario Perl/Tk y su extensión, Giphgrap.

Todo lo anterior fue aplicado en el desarrollo del sistema SICU. Por otro lado, me permitió comparar las capacidades y facilidades del lenguaje Perl.

El sistema SICU hoy cumple con su principal objetivo, que es el de realizar reportes de contabilidad gráficos en el Laboratorio de Visualización y el Departamento de Supercómputo, de la Dirección General de Servicios de Cómputo Académico en la UNAM.

Difusión

El SICU se presentó a través de videoconferencia a varias universidades del país dentro del ciclo de seminarios GASU (Grupo de Administarción de Sistemas Unix), que organiza el Área de Seguridad en Cómputo de la Dirección General de Servicios de Cómputo Académico en la UNAM.

Referencias Bibliográficas

[Bac86] Maurice J. Bach. *The Desing of the Unix Operating System.* Prentice Hall PTR, Englewood Cliffs, New Jersey, 1986.

[Cra90] Cray Research,inc. *TCP/IP and OSI Network User's Guide.* 1990.

[Fri95] Æleen Frisch. *Essential System Administration.* O'Reilly & Associates, Inc , 103 Morris Street, Suite A,Sebastopol,CA 95472, 1995.

[Fri97] Jeffrey E.F. Friedl. *Mastering Regular Expressions.* O'Reilly & Associates, Inc . 103 Morris Street, Suite A,Sebastopol,CA 95472, 1997.

[JPL97] Jerry Peek,Tim O'Reilly and Mike Loukides. *Unix Power Tools* O'Reilly & Associates, Inc., 103 Morris Street, Suite A,Sebastopol,CA 95472, 1997.

[Lou90] Mike Loukides. *System Performance Tuning.* O'Reilly & Associates, Inc . 103 Morris Street, Suite A,Sebastopol,CA 95472. 1990.

[Nem95] Evi Nemeth. *Unix System Administration Handbook.* Prentice Hall PTR, Upper Saddle River New Jersey 07458, 1995

[Ort96] Alejandro López Ortiz. Perl: Un Lenguaje Para Procesar Cadenas. *Soluciones Avanzadas*, (39).60, 1996.

[Pat96] Smith Patrick. *Client/Server Computing.* SAMS, Indiana USA, 1992.

[Pre93] Roger S Pressman. *Ingeniería del Software.* McGraw-Hill. Edificio Valrealty, 1ra. planta Basaur, 17 28023 Avalaca (Madrid), 1993.

[Sch94] Randal L. Schwartz. *Learnig Perl.* O'Reilly & Associates, Inc., 103 Morris Street, Suite A,Sebastopol,CA 95472, 1994.

[Sri97] Sriram Srinivasan. *Advanced Perl Programming.* O'Reilly & Associates, Inc., 103 Morris Street, Suite A,Sebastopol,CA 95472, 1997.