UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Escuela Nacional de Estudios Profesionales ARAGÓN

TRABAJO DE TESIS

"Análisis y desarrollo del sistema convertidor de elementos de ejecución batch para la plataforma IBM-MVS al sistema operativo SUN-UNIX"

JOSÉ LEONARDO ALBUERNE SÁNCHEZ

ISRAEL VERA MARTÍNEZ

A mis padres José Antonio y Blanca María que me dieron la vida.
A mis abuelos José Leonardo y Lidia quienes me apoyaron en mis estudios.
A mi esposa Hilda y mi hijo José Antonio cuyo amor y cariño me impulsan constantemente.

José Leonardo

A mis padres Margarita y Pedro que con su dedicación y sacrificio me enseñaron siempre a buscar ser alguien mejor.
Al amor y compresión de mi esposa Rocío que hoy en día es el motor de mi vida y la certeza de un futuro grandioso.
Por último dedico este trabajo a mi hijo Israel con la esperanza de que algún día le sirva como ejemplo de logro y esfuerzo.

Israel

Página

INTRODUCCIÓN

El Presente trabajo de Tesis tiene como objetivo generar un sistema de conversión de procesos batch, que permita trasladar los módulos de las aplicaciones utilizadas en la administración de la Base Nacional del Sistema de Ahorro para el Retiro (BNDSAR), desde un sistema MVS OS390 ubicado en una máquina IBM mainframe, hacia un sistema UNIX Solaris en una máquina modelo E10000 de SUN.

En el capítulo uno se define el "Marco Teórico" en el que hacemos mención a los principales conceptos; descripciones técnicas y especificas de las dos plataformas tecnológicas con que estaremos trabajando, además de destacar sus principales virtudes y facilidades de cada uno de estos.

Es también en este primer capítulo donde se establece que uno de los aspectos más importantes en los sistemas computacionales es tener perfectamente definido las reglas del negocio, saber el porque y el cómo se hacen las cosas y en caso de requerir aclarar las dudas que surjan durante cualquiera de las etapas del desarrollo saber a quien podemos recurrir, es por ello que en el capítulo 2 "Situación actual y situación propuesta", abarcamos estos puntos con los cuales es posible responder a todos estos cuestionamientos y además se expone el panorama por el cual actualmente atraviesa la empresa y la situación que proponemos, para lo cual damos razones que sustentan el cambio de plataforma tecnológica que garantizan los resultados operativos del sistema.

Una de las formas que utilizamos para garantizar la operatividad del sistema es mostrar que la política de estandarización para el desarrollo de procesos se cumpla tal y como hasta el momento se encuentra, es por ello que en el capítulo 3 se muestran tanto el actual como el nuevo procedimiento que se empleará para realizar los desarrollos subsecuentes.

Por último en el capítulo 4 se realiza el análisis del sistema y diseño del sistema que estará encargado de convertir los procesos batch a scripts, y para ello se explica detalladamente el algoritmo a seguir para llevar a cabo la conversión de cada uno de sus elementos, dichas explicaciones son apoyadas mediante diagramas y ejemplos ilustrativos que facilitan su comprensión y aplicación.

CAPÍTULO 1. MARCO TEÓRICO

1.1. INTRODUCCIÓN A LOS SISTEMAS ABIERTOS Y CERRADOS

En el actual diseño de sistemas se reconocen principalmente dos arquitecturas, estas son: "Sistemas abiertos" y "Sistemas Propietarios". Hasta 1990 se empleaban sistemas propietarios cerrados, en lugar de sistemas abiertos, en los que la arquitectura es algo propio y protegida mediante patente, de forma que el usuario no puede saber cómo funciona, desconoce la arquitectura del sistema, por lo que se tenía seguridad mediante oscuridad. Sin embargo, a partir de este año, apareció una tendencia hacia sistemas abiertos, provocada por la aparición en 1978 de un modelo de referencia para la interconexión de sistemas abiertos (OSI), que supuso el inicio de la estandarización de las comunicaciones. Este tipo de sistemas abiertos y estandarizados ponen en conocimiento de todos los usuarios su arquitectura, de forma que más gente puede hacer productos e intercambiar módulos. Sin embargo, de esta forma se ponen de manifiesto las debilidades de los sistemas, de forma que es más fácil realizar ataques contra los mismos.

Ambos ofrecen a su vez ciertas ventajas con respecto al otro, de tal forma que la elección del tipo de sistema con el que se desea trabajar, dependerá de los requerimientos del usuario. En la tabla 1.1 se muestra un comparativo de los dos tipos de sistemas.

	Sistemas abiertos	*Sistemas propietarios*
Escalabilidad	**Los sistemas abiertos están diseñados para la implementación a gran escala.**	Varía de un producto a otro pero suele requerir un servidor adicional y soporte administrativo
Herramientas de administración	Empiezan a aparecer sólidas herramientas de administración.	**La competencia y las múltiples revisiones realizadas han generado opciones robustas**
Fiabilidad	No hay suficiente control sobre la ruta que sigue el intercambio de información con otros sistemas ni recursos para la verificación de entrega.	**Proporciona notación de entrega y un tiempo de entrega garantizado**
Costos de utilización	**Requiere menor soporte de servicio y de tipo administrativo**	Requiere mayor soporte administrativo y de servicio, pero varía según el producto
Antecedentes probados	Los estándares y protocolos principales son relativamente nuevos en el mercado	**Años de experiencia práctica**
Comunicaciones externas	**Puede enviar y recibir correo electrónico hacia o desde cualquier máquina que tenga una conexión IP**	Los principales fabricantes tienen parcial o totalmente implementada la capacidad de enviar o recibir mensajes sobre una conexión IP
Integridad de la información	**Los archivos adjuntos y los formatos más utilizados llegan intactos**	Hay que convertir los archivos adjuntos en la pasarela de un servidor, lo que constituye una fuente potencial de error
Rendimiento	Tiempo de entrega variable	**Puede garantizar la entrega o notificar el fallo, en cuestión de segundos.**
Seguridad	Estándares aún inacabados	**Un nivel de seguridad relativamente elevado dentro de la red**
Portabilidad	**El hardware tiene la posibilidad de trabajar con distinto software**	El hardware solo acepta cierto tipo de software
Compatibilidad	**Diferentes versiones de sistemas operativos son aceptadas**	La posibilidad de cambiar de versión de sistema operativo es muy limitada
Interoperatibilidad	**Computadoras de diferentes fabricantes pueden operar entre si**	Es prácticamente nula

Tabla 1-1. Comparativo de ventajas y desventajas de los sistemas abiertos y propietarios.

1.2. SISTEMA IBM MAINFRAME MVS OS/390

1.2.1. HISTORIA 0S/390

A mediados de los cincuentas con la aparición de las primeras computadoras transistorizadas, se adoptó el sistema de procesamiento por lotes, la idea era colectar los trabajos en un cuarto de entrada, leerlas y después de tener cierto número de procesos, se transferían a una cinta magnética para llevarlas a la unidad de proceso, el operador cargaba entonces un programa especial (el antecesor del sistema operativo), el cual leía el primer trabajo de la cinta y lo ejecutaba. La salida se escribía en una segunda cinta y al terminar cada trabajo ejecutaba el siguiente trabajo, al terminar de procesar la cinta de entrada se retiraba junto con la cinta de salida, la cual era procesada para imprimirla mientras se comenzaba a procesar otra cinta en la que ya se habían recopilado nuevos trabajos para proceso, esto se representa en la figura 1-1.

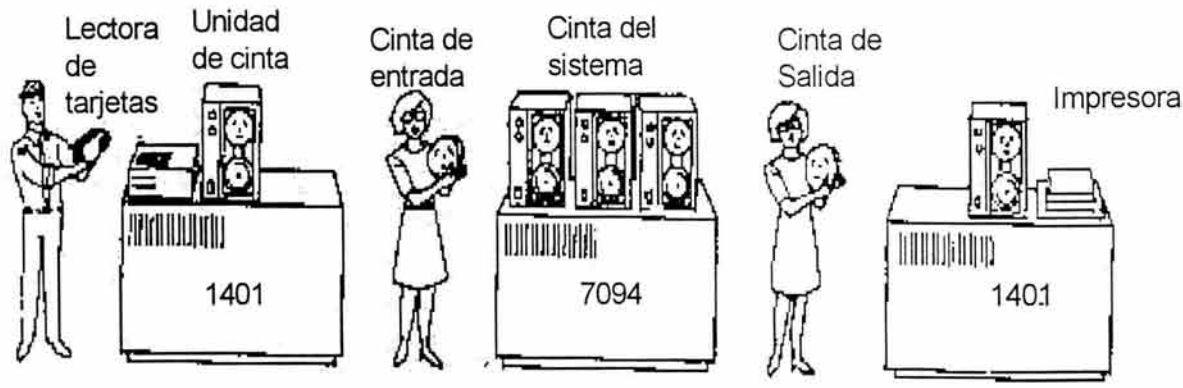


Figura 1-1. Procesamiento por lotes.

A principios de los sesentas IBM (International Bussines Machine) intentó resolver los problemas de diseño de una máquina que ofreciera la capacidad de resolver problemas científicos de cálculos complejos, además del manejo de

caracteres y que existiera compatibilidad en su escalabilidad, es decir que las aplicaciones creadas para una máquina pequeña pudieran trasladarse a una de mayor capacidad y poderlos ejecutar sin problema, así surgió el sistema 360.

La 360 fue la primera línea que utilizó circuitos integrados, lo que proporcionó una gran ventaja de precio y desempeño, y puesto que tenían la misma arquitectura y conjunto de instrucciones, al menos en teoría, los programas escritos para una máquina podían ejecutarse en las otras, la 360 se diseñó para hacer cálculos tanto científicos como comerciales.

La intención era que todo el software y sistema operativo debería funcionar en todos los modelos científicos y comerciales, pero era difícil escribir un pedazo de software que cumpliera con todos estos requisitos en conflicto. El resultado fue un sistema operativo enorme y extraordinariamente complejo, que constaba de millones de líneas de lenguaje ensamblador, escrito por miles de programadores, con miles y miles de errores, que requería de un flujo continuo de nuevas versiones, en un intento por corregirlos. Cada nueva versión resolvía algunos errores pero introducía otros nuevos, porque es probable que el número de errores fuera constante respecto al tiempo.

A pesar de los problemas el OS/360 realmente pudo satisfacer en forma razonable, a la mayoría de sus clientes. También popularizaron técnicas como la multiprogramación, el "spooling" (operación simultánea y en línea de periféricos), con lo que se obtenía una respuesta más rápida desapareciendo el uso de cintas. El deseo de una rápida respuesta, abrió el camino al tiempo compartido o "timesharing", variante de la multiprogramación en donde cada usuario tenía una terminal en línea, en donde la unidad central de proceso (Central Procesing Unit "CPU") se va asignado conforme la van demandando los procesos.

Muchos de los usuarios de la 360, tanto dentro como fuera de IBM, deseaban tener tiempo compartido, de esta forma surgió la versión VM/370, en donde el

sistema es la capa entre el software y el hardware, que simula una o mas máquinas virtuales, es decir genera una copia exacta del hardware simple con su núcleo, E/S, interrupciones y todo lo que posee la máquina real, separando la función de multiprogramación del sistema operativo en sí.

En la figura 1.2 se puede observar de manera gráfica la forma en que el sistema operativo simulaba las máquinas virtuales.

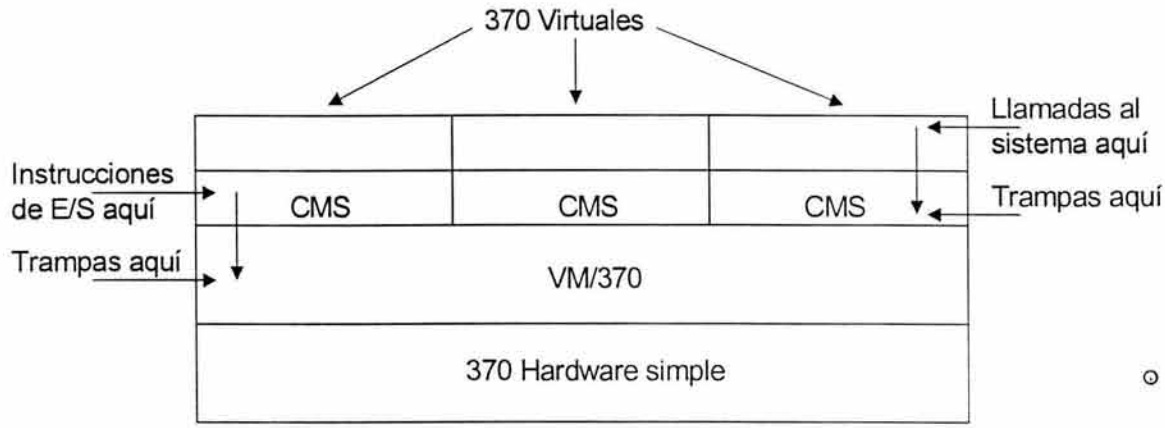


Figura 1-2. Sistemas virtuales.

Puesto que cada máquina virtual es idéntica al hardware real, cada una puede ejecutar cualquier sistema operativo que se ejecute en forma directa sobre el hardware. Así algunas máquinas virtuales pueden ejecutar procesamiento por lotes y otros sistemas interactivos para usuarios de tiempo compartidos. Al hacer una separación total de las funciones de multiprogramación y proporcionar una máquina extendida, cada una de las partes puede ser mas sencilla, flexible y tener un fácil mantenimiento.

1.2.2. ARQUITECTURA DEL SISTEMA IBM 0S/390.

La arquitectura de un sistema se describe como la relación entre las partes físicas de una computadora o hardware y el sistema operativo o software que se ejecuta en dicha máquina[1].

La arquitectura de sistemas incluye una lista de instrucciones que pueden ser ejecutadas, cómo puede un programa leer o escribir datos (entrada/salida), cuantas instrucciones pueden ejecutarse simultáneamente (multiprocesamiento) y cómo se puede acceder a la memoria de la computadora (acceso real o virtual).

En 1964 IBM anunció que todas sus máquinas utilizarían la arquitectura 360, unificando todas las arquitecturas de los modelos previos, con el fin de que sus clientes, pudieran crecer y/o modificar sus equipos sin necesidad de realizar grandes cambios en sus aplicaciones, ya que utilizaban el mismo conjunto de instrucciones, y estaban orientadas al manejo de números binarios, decimales, de punto flotante, ya sea de precisión simple o doble, caracteres y palabras en la misma máquina.

El sistema operativo en el OS/360, evolucionaría a la par de la evolución del hardware, ofreciendo mejoras cada vez más ventajosas y así, una aplicación que residiera en un modelo 30 de la 360, que fue un modelo diseñado para manejar requisitos de procesamiento mínimos, se podría transportar a un modelo 50 con una velocidad 50 veces mayor, sin necesidad de modificar ninguna línea de código de la aplicación.

La ultima versión del antiguo sistema OS/360, es el OS/390, que es el que definiremos en este tema. Como es sabido, todas las computadoras actuales cumplen ciertas características en cuanto a la estructura de su hardware,

1 Gary DeWard Brown, "System 370/390 JCL"

poseen una unidad de proceso central, unidades de entrada y salida, una memoria.

Básicamente el sistema operativo, es quién inicialmente tiene el control del procesador, cuando se le pide ejecutar algún programa lo hace instrucción por instrucción, debido a que los sistemas OS/390 son multiusuarios, generalmente ejecutan mas de un programa a la vez, el sistema operativo le pasa el control del procesador a algún programa para que ejecute sus tareas, almacenando su estado en los registros de control del procesador, una vez que el programa que corre, ejecutó un número de instrucciones, utilizó el procesador un tiempo determinado, o entró en algún proceso de espera de recursos; el sistema toma el control del procesador nuevamente, almacena en los registros de control el estatus del programa que estaba corriendo, y pasa el control a otro programa, ya sea este nuevo, o que previamente estaba ejecutándose y cuyo estatus estaba almacenado en los registros de control del procesador.

Según la configuración de los procesadores en los sistemas OS/390, estos pueden funcionar como:

1.2.2.1. UNIPROCESADORES

Como se mencionó la arquitectura de los sistemas de la IBM OS/390 es el conjunto de productos de hardware y software y la relación de su funcionamiento[2], dentro del hardware tenemos al procesador central, el área de almacenamiento del procesador y el subsistema de canales de comunicación, tal como se muestra en la figura 1-3

[2] Gary DeWard Brown, "System 370/390 JCL"

Figura 1-3. Esquema de un uniprocesador.

El conjunto arriba señalado es conocido como el procesador central complex (Central Procesor Complex "CPC"), el programa primario que se ejecuta sobre dicha estructura de hardware es el sistema operativo, que es el de Almacenamiento Virtual Múltiple (Multiple Virtual Storage MVS), en esta estructura del procesador central, procesa una y solo una instrucción de programa a la vez y solo ejecuta una copia del MVS, esta estructura es llamada uniprocesador.

1.2.2.2. MULTIPROCESADORES NO PARTICIONADOS

El sistema de uniprocesador es como se puede ver el más simple y no soporta recuperaciones en caso de fallas en el procesador, pero si añadimos más procesadores a un CPC, como se ve en la figura 1-4.

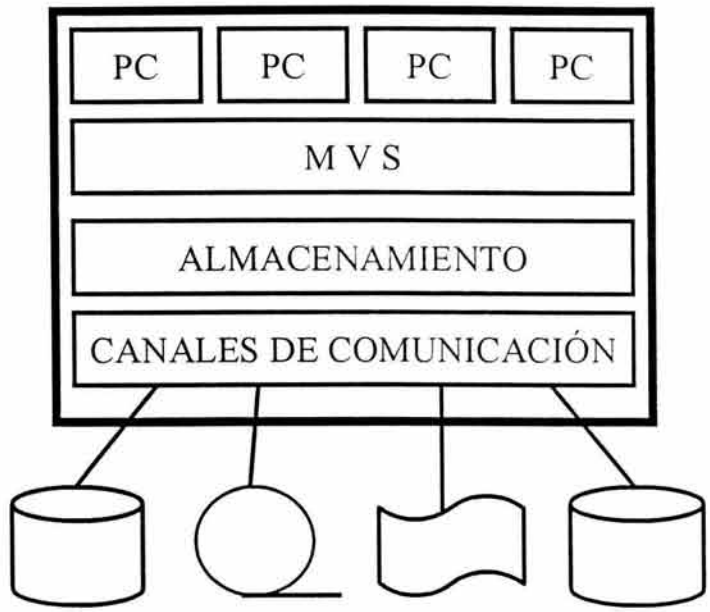


Figura 1-4. Esquema de multiprocesador no particionado.

Esto permite al sistema ejecutar varias líneas de programa simultáneamente, como todos los procesadores comparten la unidad de almacenamiento central y ejecutan una copia del sistema MVS, el trabajo es asignado al primer procesador disponible, en caso de falla de un procesador, el trabajo puede ser asignado a otro procesador, este tipo de configuración es conocida como multiprocesador no particionado.

1.2.2.3. MULTIPROCESADORES PARTICIONADOS.

En contraste a un multiprocesador no particionado, algunos sistemas OS/390 tienen una capacidad de partición física, esto les permite operar en dos formas diferentes en un modo de imagen simple donde se comportan como un multiprocesador no particionado como en la figura 1-5, y ejecutan solo un sistema operativo.

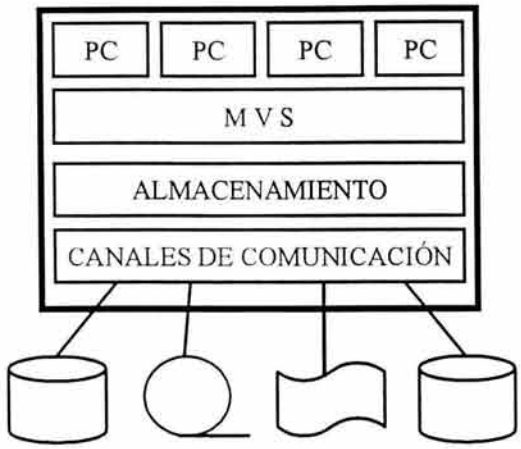


Figura 1-5. Esquema de multiprocesadores particionados en modo de imagen simple.

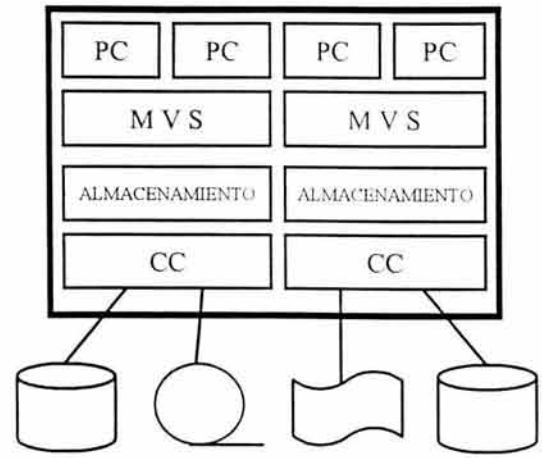


Figura 1-6. Esquema de multiprocesadores particionados en modo de partición física.

El otro modo de operación es un modo de partición física, donde los procesadores y los recursos de memoria y canales de comunicación, están divididos para ejecutar diferentes imágenes del mismo sistema operativo como en la figura 1-6.

1.2.2.4. PARTICIÓN LÓGICA.

Con el desarrollo del sistema manejador de recursos del procesador, los sistemas OS/390 fueron capaces de ser particionados en múltiples particiones, ya que se particionan los procesadores, unidades de almacenamiento temporal y expandida, canales de comunicación, es como tener mas de una máquina en el mismo equipo como se muestra en la figura 1-7.

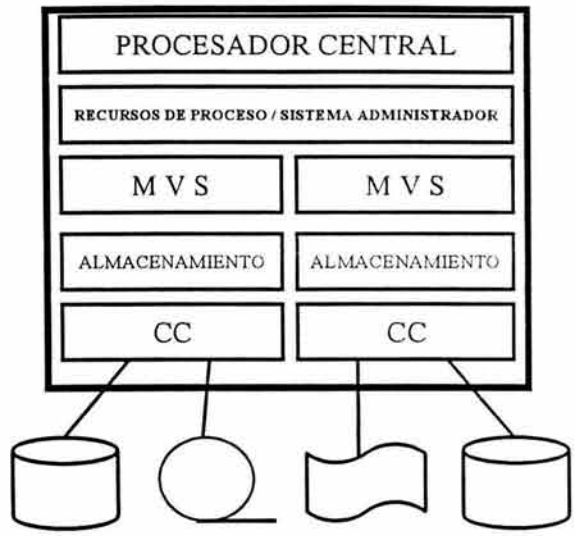


Figura 1-7. Esquema de una partición lógica.

De este modo las diferentes particiones en una sistema OS/390 comparten los dispositivos, mas no los recursos, ya que cada cual tiene su partición de recursos dentro del dispositivo compartido, al cual cada partición se conecta por su propio sistema de comunicación como se observa en la figura 1-8.

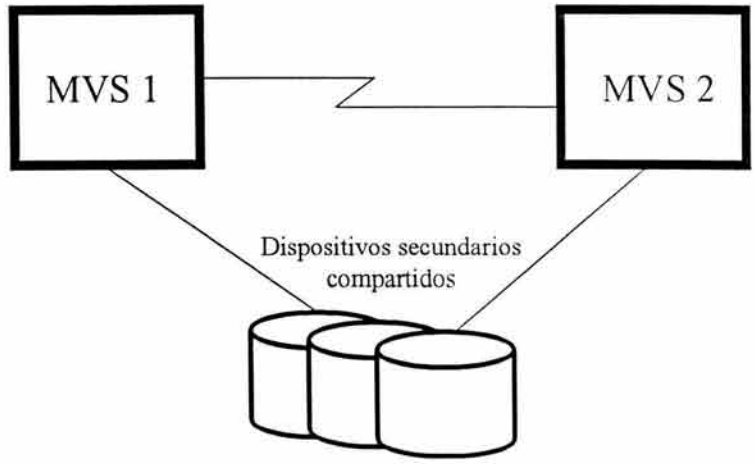


Figura 1-8. Compartición de dispositivos secundarios en un sistema acoplado.

Los sistemas particionados permiten el respaldo cuando un procesador central falla, pero, ¿cuando la falla ocurra en alguna otra parte del sistema?

Un MVS puede coordinar con más de una imagen del mismo sistema MVS, mediante el subsistema de entrada de trabajos y la serialización global de recursos, es decir, múltiples sistemas MVS pueden compartir recursos y trabajo, estos sistemas acoplados tienen más de una unidad central de proceso compartiendo dispositivos de almacenamiento secundario, pero no el espacio.

Un sistema acoplado incrementa la capacidad y disponibilidad, pero requiere gran administración y manejo, además de que cada sistema tiene que ser manejado individualmente, por esto IBM proporciona el sistema cruzado de acoplamiento (Cross-Coupled Facility "XCF"), que permite manejar los sistemas acoplados mas fácilmente, y a las aplicaciones autorizadas de un sistema comunicarse con aplicaciones del mismo u otro sistema.

En un sistema sysplex, que son dos o más sistemas complex acoplados que se ven como uno solo, las unidades centrales de proceso conectadas canal de comunicaciones a canal comunicaciones y con el conjunto de los datos que están compartidos para permitir la comunicación, y cuando están involucrados mas de una central de proceso están en un sistema sysplex, se necesita un temporalizador o reloj que sincroniza estos sistemas llamados temporalizador sysplex, como se muestra en la figura 1-9.

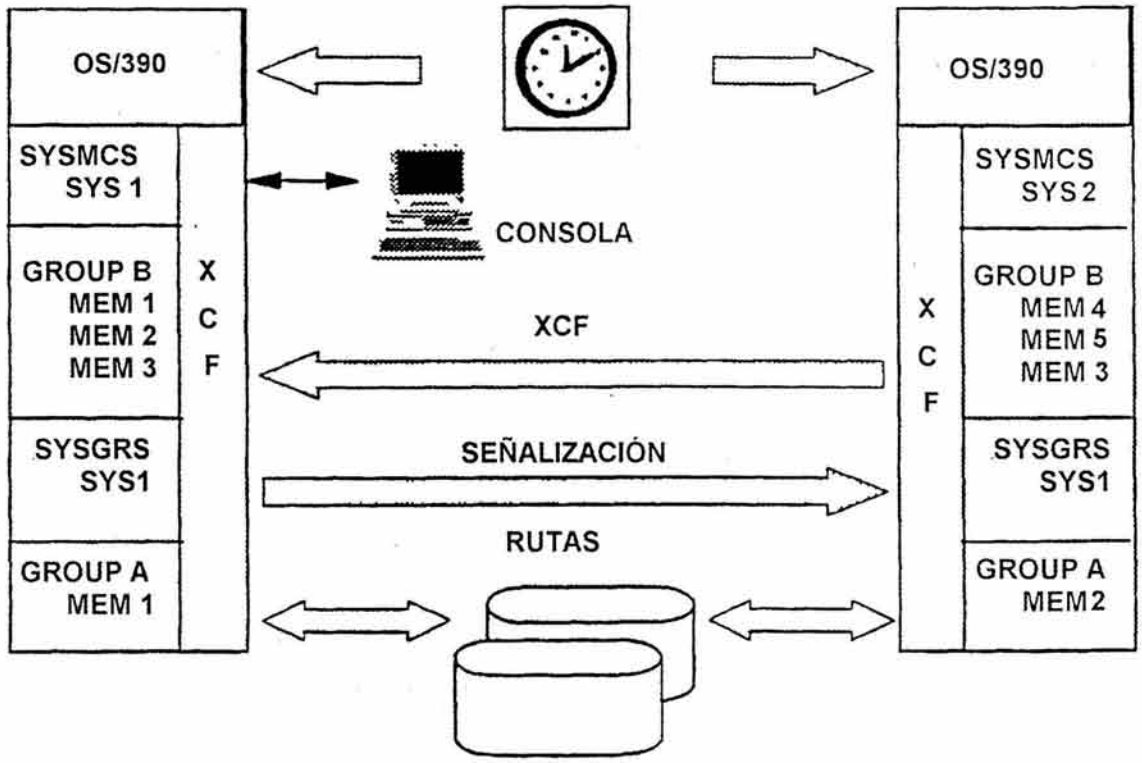


Figura 1-9. Sistema sysplex.

1.2.2.5. MEMORIA.

La memoria real o memoria central es la memoria que utiliza el procesador para almacenar las instrucciones con las que está operando[3], en cambio la memoria virtual es simplemente un rango de direcciones disponibles para todos los programas que se ejecutan en el sistema[4], debido al multiproceso que realiza el sistema, la memoria central por muy grande que sea, no es suficiente para contener toda la información.

El principio de la memoria virtual radica en utilizar espacio de dispositivos secundarios, como lo son los discos para emular memoria central, es decir el

[3] Stallings, William, "Sistemas operativos"

[4] Stallings, William, "Sistemas operativos"

MVS utiliza memoria secundaria para sus procesos, simulando que es parte de su memoria central, pero cuando los datos que necesita se encuentran en los dispositivos secundarios, la carga en memoria central para su proceso, y descargara la información de algún otro proceso que este en espera de algún recurso, o que ya utilizó su tiempo de proceso.

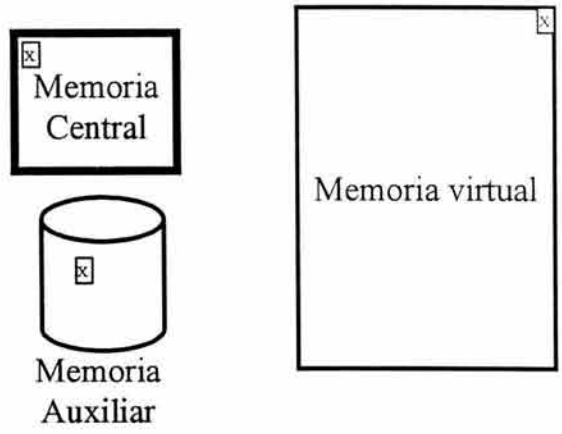


Figura 1-10. Estructura de la memoria virtual.

Para realizar este cambio dinámico de información, el MVS, mantienen un mapa de la memoria virtual en la memoria central, para "saber", dónde se encuentran los datos de cada proceso, ya sean estas instrucciones o datos, este mapa es el conjunto de direcciones de la memoria virtual.

Tanto la memoria central como la virtual están administradas en bloques de 4Kb, llamados páginas, estas páginas se van asignando de acuerdo a como lo demande la aplicación y pueden direccionar 16 megabytes o 2 gigabytes de memoria virtual para cada aplicación, y según el balanceo de recursos predispuesto en el sistema, es como la memoria virtual se carga en la central para su procesamiento, este manejo de memoria en dispositivos secundarios es más lento que la memoria central, pero cada proceso tiene su área de memoria para sí.

Utilizando memoria virtual, a cada usuario se le proporciona un rango de direccionamiento de 16 megabytes.

1.3. CARACTERÍSTICAS DEL SISTEMA MVS OS/390.

Antes del OS/390, el sistema 390 consistía en gran cantidad de productos, eran pedidos, instalados y actualizados separadamente, por lo que su liberación era asíncrona y en diferentes tamaños. Por lo que para algún sistema de información requería evaluar y saber que prerrequisitos u otras actualizaciones se deberían de hacer para que un producto pudiera ser ordenado, además de evaluar su impacto en el funcionamiento de otros productos ya instalados en el sistema.

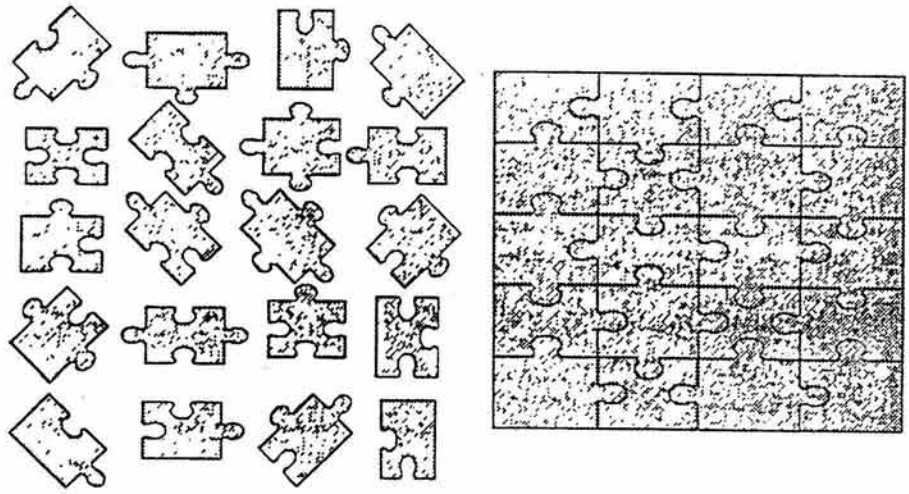


Figura 1-11. Antes del OS/390. Después del OS/390.

Con el OS/390 IBM dio a sus clientes la posibilidad de actualizar sus sistemas en bloques, garantizando que los productos eran compatibles unos con otros, además de tener su propio procedimiento de instalación que reducía el tiempo de este considerablemente. Así el OS/390 se entregaba al cliente en dos componentes principales, las funciones y elementos básicos del OS/390, que se encuentran invariablemente en cada OS/390, y las opcionales, que pueden ser habilitadas o deshabilitadas dinámicamente por el usuario.

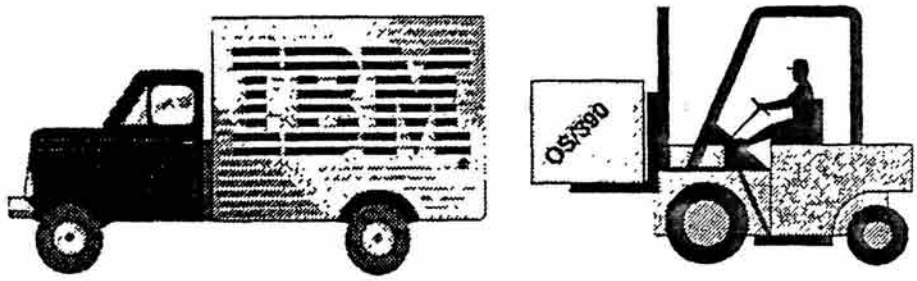


Figura 1-12. Entrega del OS/390.

Así el apoyo a la comunicación, los accesos en línea, interfaces gráficas son consideradas elementos básicos, que siempre estarán en funcionamiento, en cambio las características opcionales, pueden ser las dinámicamente habilitadas, es decir si el cliente las pide se entregan activadas, en caso contrario, si el cliente no las requiere, esta existen pero inactivas, existe una segunda clase de funciones opcionales que deber ser explícitamente ordenadas para su adición al sistema.

El sistema MVS que es la piedra angular de muchos sistemas computacionales actualmente ofrece modernas soluciones de cliente servidor, tecnología de objetos y conectividad en redes en adición a las prestaciones funcionales de los servicios de MVS.

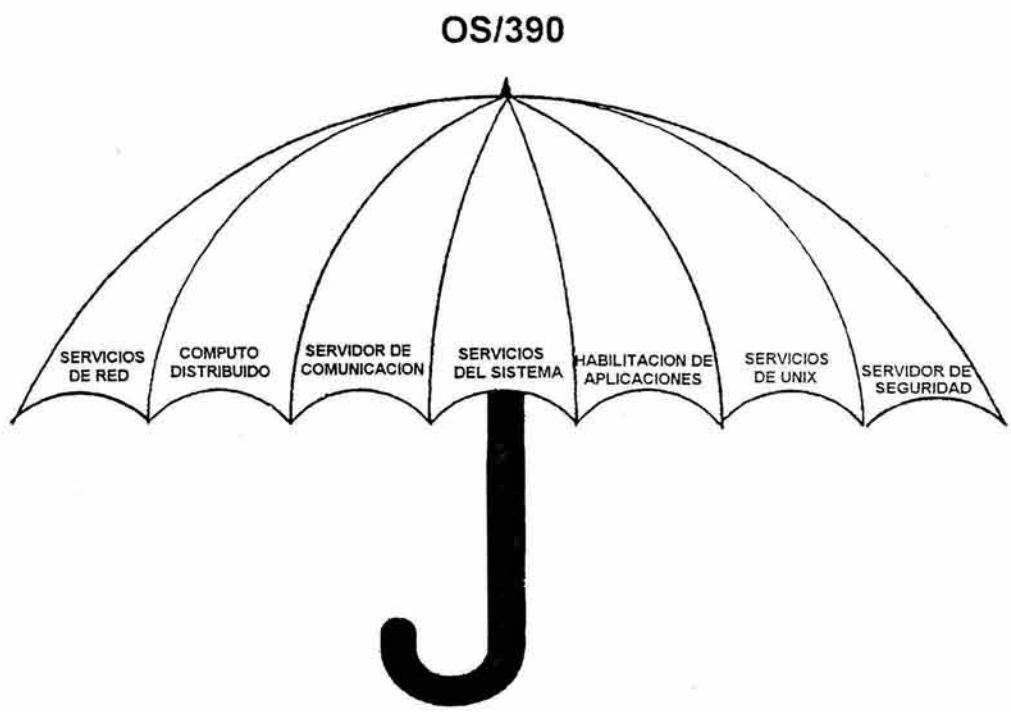


Figura 1-13. Apoyo en comunicación de OS/390.

Debido a que actualmente los sistemas de información requieren reducir costos para ser competitivos, el OS/390 es la solución integral ya que ofrece:

- Servicios del sistema, confiabilidad y flexibilidad del sistema operativo, tiene capacidad de recuperación automática, gracias a las funciones redundantes que posee en disponibilidad, permite tener capacidades de reconfiguración dinámica y robustecimiento con mínima inversión de infraestructura.

- Integridad de los datos a cargo del sistema de administración de almacenamiento, además de respaldos automáticos, opciones de recuperación, seguridad.

- Paralelismo de sistemas complejos llamados Sysplex (por su mnemónico en inglés), donde varios sistemas complejos, complex, dan la apariencia de un solo sistema sysplex.

- Manejo de las cargas de trabajo, mediante políticas establecidas en términos de respuestas a requerimientos.

- Opciones tecnológicas en respuesta a la demanda tecnológica, con fácil mantenimiento.

- Servicio al usuario final, con aplicaciones que resuelven necesidades específicas de negocios, con apoyo en la nueva tecnología de objetos.

- Servicios de red, proporcionando datos impresoras y periféricos, administración de grandes sistemas empresariales, seguridad integrada para redes mediante el control de acceso a recursos (Resource Access Control Facility "RACF").

- Los servicios UNIX ofrecidos soportan estándares abiertos como posix y XPG4.

- Ofrece soporte para aplicaciones distribuidas usando soluciones industriales tales como ambiente de cómputo distribuido (Distributed Computing Enviroment "DCE"), sistemas de archivos distribuidos (Distrubuted File System "DFS"), y sistemas de archivos en red (Network File System "NFS").

- Productividad, hacer más sin incrementar los recursos.

1.3.1. SERVIDOR DE COMUNICACIÓN

El servidor de comunicación es parte de la base del sistema operativo, y provee la infraestructura de red para sistemas computacionales abiertos y distribuidos instalados en empresas extensas. Incluye una gran variedad de interfaces como lo son los sockets, llamadas a procedimientos remotos, y el uso de protocolos de redes como lo son el TCP/IP (Transfer control Protocol / I Protocol) y VTAM.

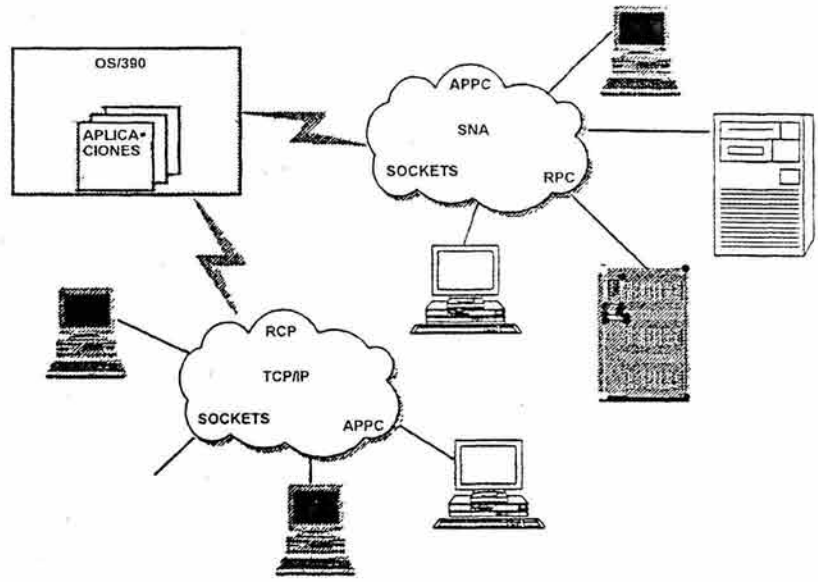


Figura 1-14. Servicios de comunicación.

La capacidad del servidor de comunicaciones de manejar el protocolo multiprotocolo de transporte de redes (Multiprotocol Transport Network "MPTN") proporciona el poder tener aplicaciones UNIX comunicándose sobre una red SNA o tener APPC (SNA) con aplicaciones comunicándose sobre una red con TCP/IP.

En sí el servidor de comunicación permite manejar y compartir información y transacciones a través de diferentes plataformas y sistemas de redes.

1.3.2. SERVICIOS DE RED

El OS/390 incluye los servicios de administración área de red local, tales como el acceso a los usuarios para almacenar y acceder datos, usar servicios de impresión, desde un manejador central, gran capacidad y desempeño, para una solución de proceso distribuido.

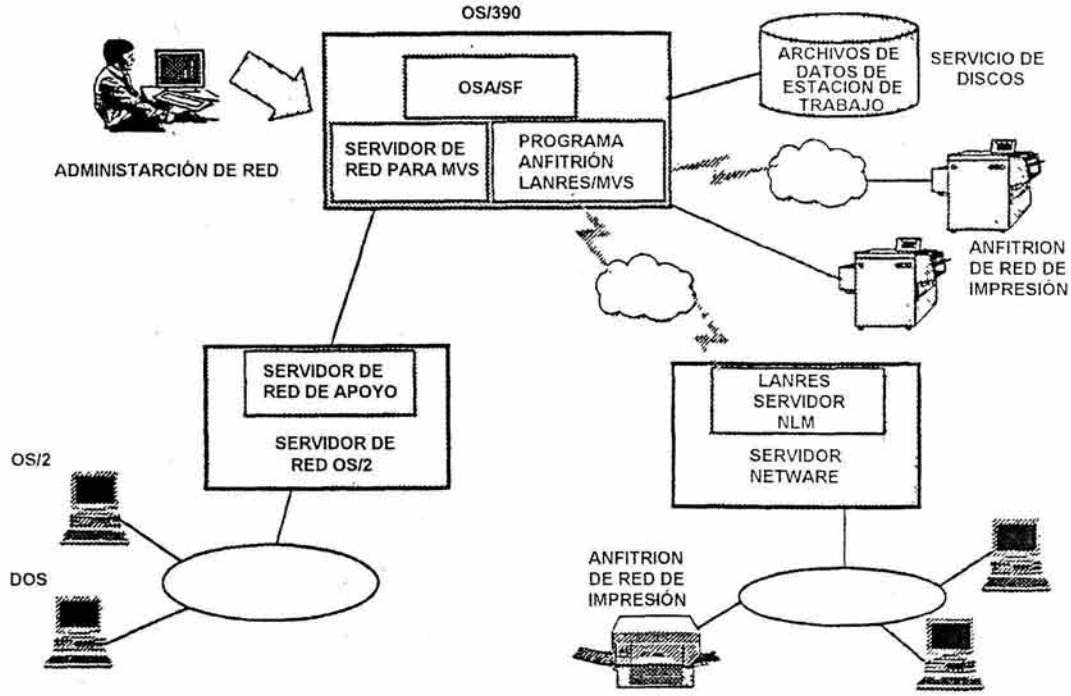


Figura 1-15. Servidor de red.

Estos servicios son proporcionados por: el LANRES/MVS, que proporciona los servicios de impresión, distribución de datos y administración central para redes Novell. El LAN server o servidor de área local, provee los servicios de compartición de los sistemas de archivo, permite múltiple servidores OS/2 LAN y que los clientes NFS puedan compartir datos, repositorios, y tiene soporte para multimedia; y por último tenemos el adaptador de sistemas abiertos (OSA por sus siglas en inglés), que permite que los procesadores OS/390 puedan ser conectados directamente a las redes.

1.3.3. HABILITACIÓN DE APLICACIÓNES

El OS/390 incluye servicios de aplicación, los cuales proporcionan soporte para la tecnología orientada a objetos, un ambiente común de ejecución, y herramientas para crear nuevas interfaces de usuario a las aplicaciones tradicionales del MVS.

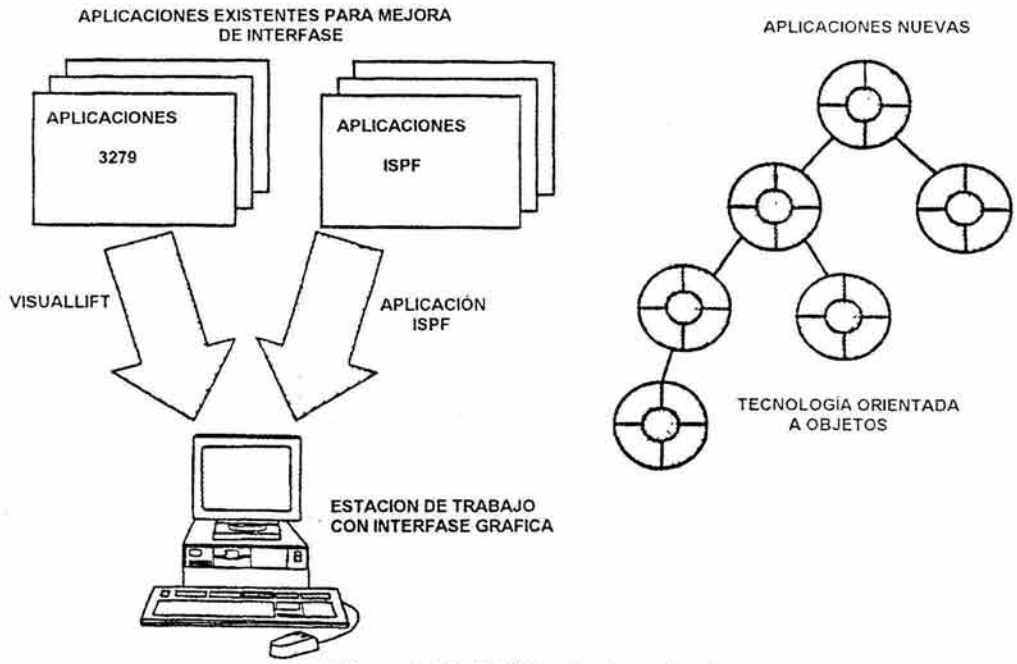


Figura 1-16. Habilitación de aplicaciones.

Las aplicaciones tradicionales del MVS que usan una interfase de usuario 3270, pueden ser mejoradas usando la herramienta VisualLift, para añadir ventanas de interfase gráfica (Graphical User Interface "GUI") a las estaciones de trabajo sin reprogramar la aplicación.

Proporciona también SOMobject, basado en modelo del sistema de objetos (System Model Object "SOM"), el cual define una arquitectura para manejar y definir librerías de clases de objetos, trayendo como consecuencia las ventajas de simplicidad, reutilización, compartición, reducción del esfuerzo de desarrollo, ciclo de vida, calidad del código de la programación con objetos.

Una biblioteca común de ejecución para C/C++, Cobol, Fortran, PL/I para el desarrollo de aplicaciones con tecnología orientada a objetos y distribuidas en cliente servidor, además de ofrecer apoyo hacia los estándares de sistemas abiertos.

1.3.4. SERVICIOS UNIX

Los servicios de aplicación UNIX que son incluidos en el MVS/390 son llamados Edición Abierta (OpenEdition), tales como la interfase shell y la estructura jerárquica de archivos, esto con el fin de aprovechar la fuerza, capacidad y flexibilidad del plataforma 390, como lo son la velocidad y capacidad de los dispositivos secundarios de almacenamiento, la velocidad de impresión, seguridad, manejo de funciones del sistema, capacidad de procesamiento, entre otras.

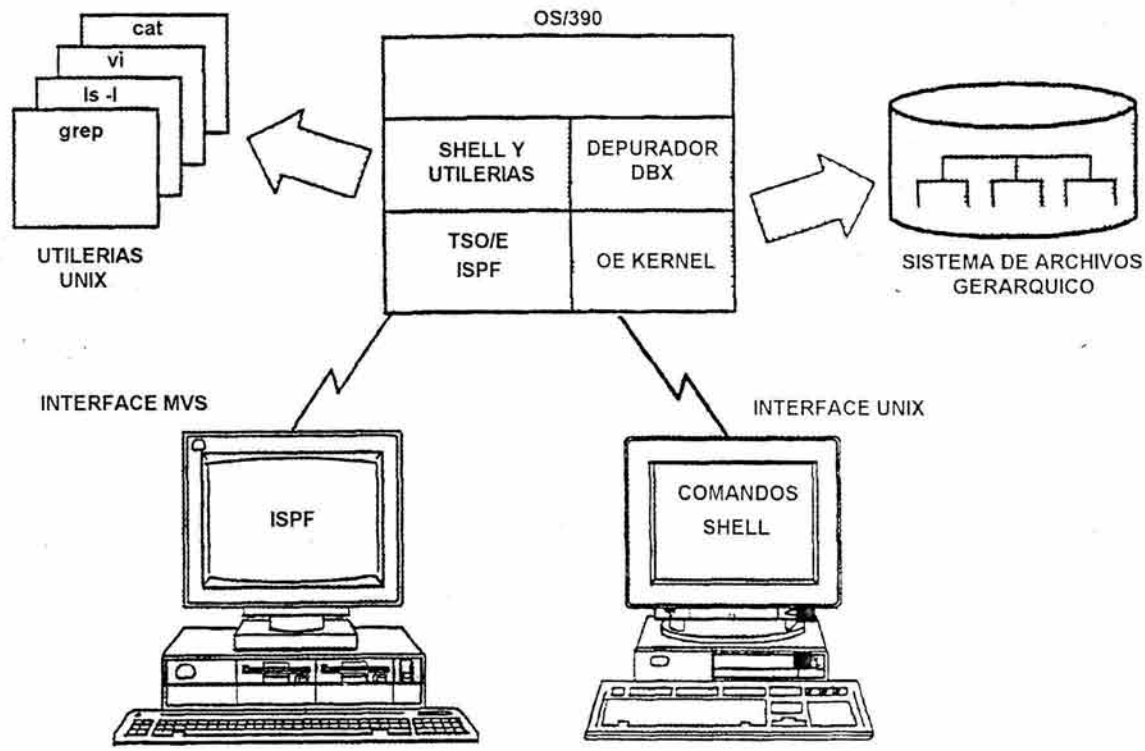


Figura 1-17. Servicios de aplicación UNIX.

1.3.5. CÓMPUTO DISTRIBUIDO

El ambiente de cómputo distribuido (Distributed Computing Enviroment "DCE") ofrece para la tecnología cliente/servidor un conjunto de servicios y

herramientas que apoyan la creación, uso y mantenimiento de aplicaciones distribuidas en un ambiente heterogéneo.

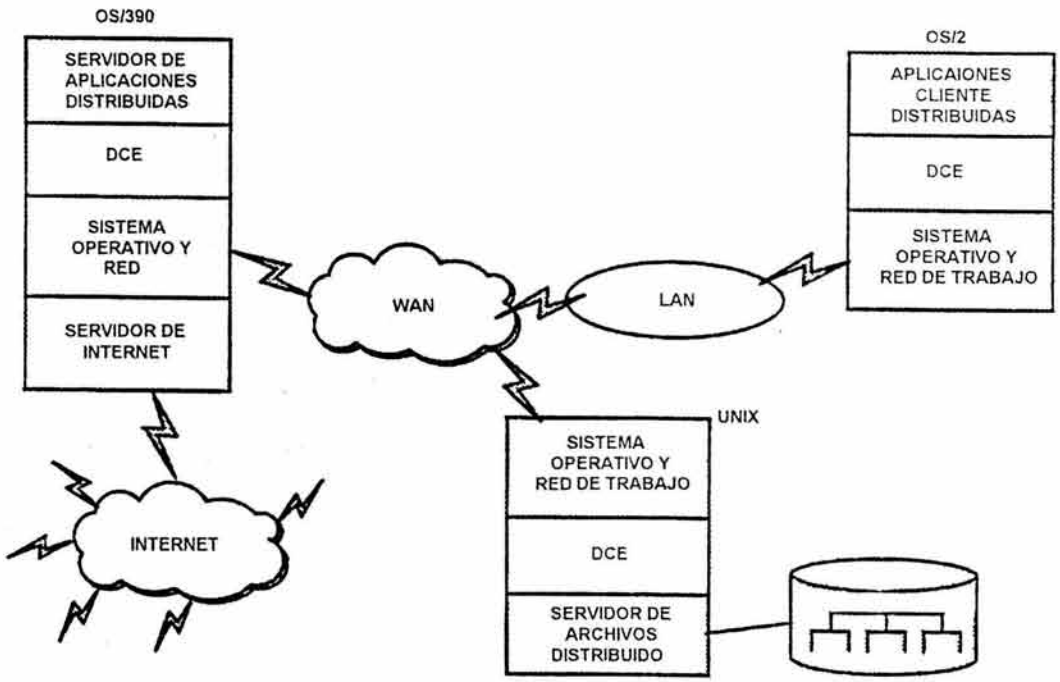


Figura 1-18. Esquema de cómputo distribuido.

El sistema de archivos distribuido (Distributed File System "DFS") es un servicio del DCE que proporciona acceso a archivos remotos de manera transparente, y desde cualquier lugar de la Red DCE, lo que permite a los usuarios fácilmente compartir datos en un ambiente distribuido.

El sistema de archivos de red de trabajo (Network File system "NFS") por sus siglas en inglés es una solución que permite a los usuarios compartir y acceder datos en un ambiente heterogéneo. Es ampliamente utilizado en sistemas UNIX. NFS permite a las estaciones de trabajo acceder a conjuntos de datos en un sistema OS/390.

Asimismo el OS/390 proporciona el servidor de conexión de Internet para MVS, lo que lo convierte en un servidor de Internet, con almacenamiento de texto,

imágenes, sonido y multimedia, almacenados en un sistema de archivos jerárquico.

1.3.6. SERVIDOR DE SEGURIDAD

El control de acceso a recursos (Resources Access Control Facility "RACF") es la solución que IBM proporciona para proveer seguridad en mainframes centralizados utilizando MVS, en cambio para sistemas abiertos, cliente/servidor y archivos distribuidos, las necesidades de seguridad son mas amplias y en constante crecimiento, OS/390 ofrece el servidor de seguridad como una solución, el cual esta basado en la seguridad del DCE y RACF, esto quiere decir que se cubren tanto las necesidades de funciones de seguridad distribuida y local.

Las aplicaciones distribuidas que no estén basadas en DCE, también pueden tomar ventaja del servidor de seguridad del OS/390, mediante el uso de un servicio genérico de seguridad API, el cual permite a las aplicaciones explotar la seguridad del servidor sin necesidad de rediseñar el modelo de distribución de aplicaciones.

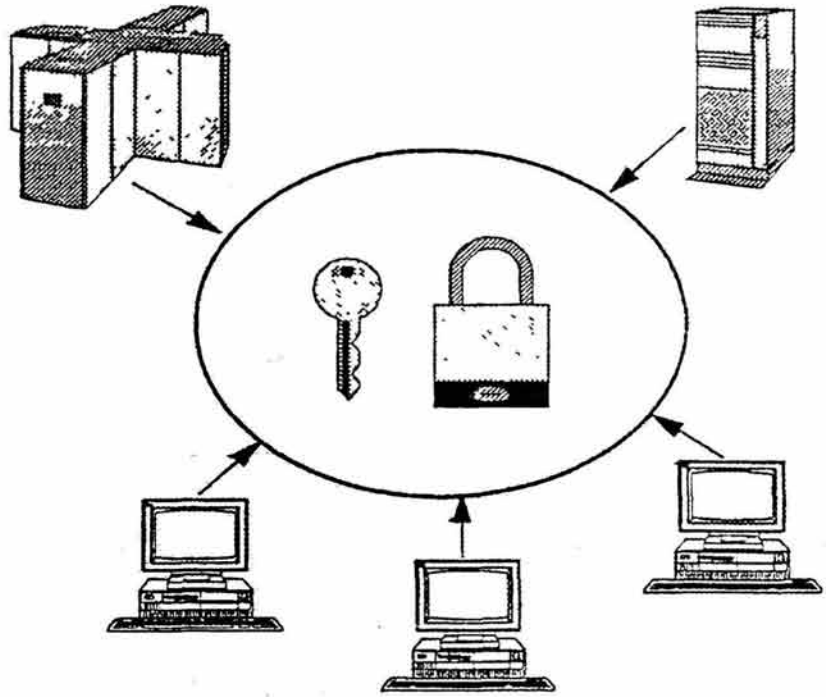


Figura 1-19. Esquema de seguridad mediante RACF en OS/390.

1.4. PLATAFORMA TECNOLÓGICA CON EQUIPO SUN ENTERPRISE 10000.

1.4.1. ARQUITECTURA

El Sun Enterprise 10000 (E-10000), popularmente conocido como Starfire, es el primer "mainframe" UNIX y esta diseñado para cumplir con los desafíos que actualmente presentan los centros de datos. Esta basada en la tecnología de aplicaciones cliente/servidor como transacciones en línea (on-line transaction procesing "OLTP"), Sistema de Soporte de Decisiones (Decision Support Systems "DSS") y servicios de comunicación o multimedia.

El la figura 1-20 se muestra una vista frontal del servidor Enterprise 10000.

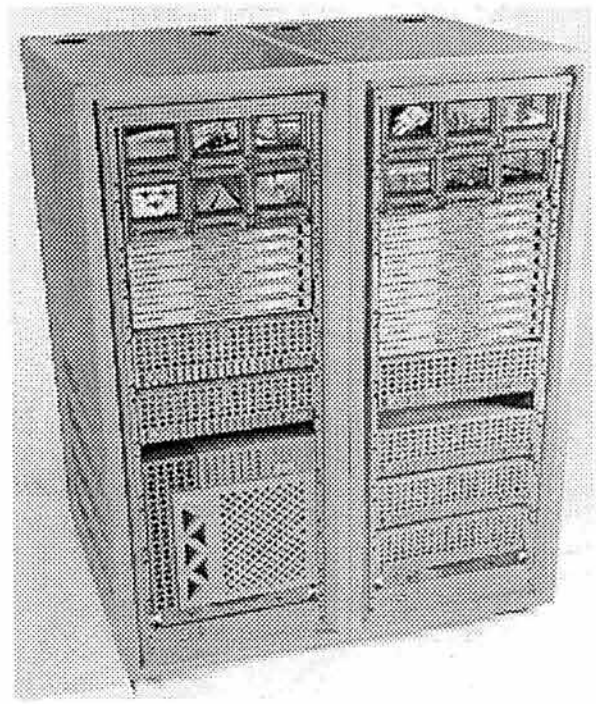


Figura 1-20. Servidor Enterprise 10000.

1.4.2. CARACTERÍSTICAS GENERALES

Las características principales del servidor Enterprise 10000 se muestran en las siguientes tablas:

PROCESADORES EN LA E-10K	
No. de procesadores	4 a 64
Secuencia de reloj	400 Mhz
Arquitectura	Ultra SPARC versión 9
Caché por procesador	Primario: 16-KB Secundario: 4 megabytes (caché externo)
Interfaz de CPU	64-bit Arquitectura Ultra Port

Tabla 1-2. Características de procesadores de la E-10000.

TARJETAS DEL SISTEMA
Máximo de 16 system board's por equipo E-1000, la configuración mínima es de 1 system board, cada una con hasta 4 procesadores, hasta 4 tarjetas de Sbus o 2 tarjetas PCI y un módulo de memoria con 4 bancos de 8 SIMMS cada uno.

Tabla 1-3. Características del sistema de tarjetas.

MEMORIA PRINCIPAL
Capacidad de memoria de 2-gigabytes a 64 gigabytes por sistema.

Tabla 1-4. Características de la memoria principal.

INTERFACES

SBus	Ancho del bus de datos de 64-bit a 25 Mhz
PCI	Ancho del bus de datos de 32-bit o 64-bit a 66 Mhz

Tabla 1-5. Características de interfases.

CUALIDADES
• Intercambio en línea de componentes (On-line hot swap) tales como tarjetas y componentes del sistema de enfriamiento y de poder. • Tolerancia de errores (Fault-tolerant) de los sistemas de enfriamiento y de poder • Corrección de errores de interconexión • Herramientas de monitoreo de procesos • Sistema automático de recuperación • Ancho de banda mayor a 3.2 gigabytes por segundo • Reconfiguración dinámica • Suministros de energía, ventiladores y componentes del sistema pueden ser reemplazados mientras el sistema se encuentra activo. • Manejo de Dominios • Se puede adicionar memoria y slots de entrada-salida mientras el sistema se encuentra funcionando. • Verificación de paridad • Monitoreo de ambiente • Soporte mediante consola remota • Redundancia de elementos • Conexión de inter-dominios

Tabla 1-6. Cualidades.

PROGRAMAS	
Sistema operativo	Solaris 2.6 (equivalente a UNIX system V versión 4)
Lenguajes	C, C++, Pascal, FORTRAN, Java, Cobol Microfocus
Protocolos	ONC, NFS, TCP/IP, SunNet, OSI, MHS, X.25, DCE, JTAG, Netware
Sistema de monitoreo	SSP 3.1
Software de sistema	Sun Enterprise Volume Manager, VERITAS Files System, StorEdge Enterprise HSM, Solstice Site Manager, Solstice Domain Manager and Solstice Enterprise Manager

Tabla 1-7. Programas de aplicación.

1.4.3. CARACTERÍSTICAS ESPECÍFICAS

A continuación se detallan algunas de las características más significativas del servidor E-10000.

1.4.3.1. SISTEMA OPERATIVO

El sistema operativo con el que cuenta es Solaris 2.6 (equivalente a UNIX system V versión 4).

UNIX es un sistema operativo multiusuario propuesto al principio para minicomputadoras pero implementado para un amplio rango de máquinas desde poderosas microcomputadoras hasta supercomputadoras[5]. La figura 1-21 proporciona una descripción general de la arquitectura de UNIX.

[5] **Sun educational service, "Ultra Enterprise 10000. Administration ES-400"**

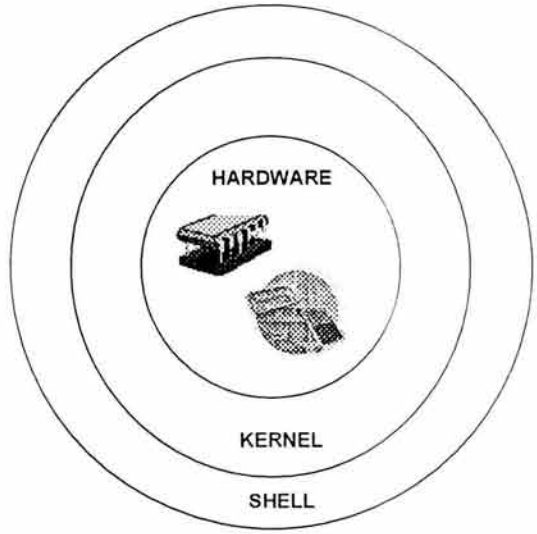


Figura 1-21. Esquema del sistema operativo UNÍX.

El software rodea al hardware básico en el sistema operativo. El sistema operativo se llama con frecuencia el kernel del sistema o solo kernel para enfatizar su aislamiento del usuario y aplicaciones. Esta parte de UNIX es considerada el alma del sistema, no obstante, UNIX viene equipado con varios servicios de usuario e interfaces que se consideran parte del sistema. Estas pueden agruparse en el shell, otro software de interfaz y los componentes del compilador C (compilador, ensamblador, cargador). La capa externa de este consta de aplicaciones de usuario de la interfaz del usuario al compilador C.

1.4.3.2. SISTEMA DE SERVICIO DE PROCESOS

El sistema de servicio de procesos (System Service Procesor "SSP") es usado para monitorear y controlar el servidor E-10000, y ejecutar funciones del sistema de administración. El SSP requiere de una estación SPARC con 64

megabytes de memoria RAM y espacio en disco duro de 2.1 gigabytes con sistema operativo Solaris 2.5.1 con ambiente OpenWindows además del siguiente software:

- Control y manejo de demonios (daemons). El SSP ejecuta un determinado número de demonios que controlan y monitorean al servidor E-10000 y sus dominios.

- Visualizador (Hostview). Este es una interfaz gráfica de usuario (Graphical User Interface "GUI") la cual auxilia al administrador con el mantenimiento del hardware del Enterprise 10000.

- Autocomprobador de poder (Power On Self Test "POST"). Este Software se asegura de que el Hardware conectado al dominio se encuentre listo para ser usado.

- Consola de red (Network console "Netcon"). Esto le permite al administrador crear un sistema de consola remota, usando de interfaz una línea de MODEM.

Por medio del SSP el administrador puede realizar las siguientes tareas:

- Inicializar dominios.

- Desactivación automática de algún dominio en caso de urgencia. Por ejemplo si la temperatura del procesador excede los limites permitidos.

- Crear dominios.

- Reconfiguración dinámica de dominios.

- Monitoreo de temperatura y niveles de voltaje de uno o más sistemas de tarjetas o dominios.

- Control de operación de ventiladores.

- Ejecución de programas de diagnóstico.

Además el ambiente SSP:

- Previene de problemas de poder, tales como altas temperaturas o mal funcionamiento de los suministros de poder.
- Notifica problemas y errores de software cuando esto ocurre.
- Mantiene una bitácora de transacciones entre el SSP y los dominios.

1.4.3.3. SISTEMA DE DOMINIOS

Un dominio consiste en un grupo de una o más tarjetas de sistema o "system board" lógicamente agrupadas por el SSP[6], con lo cual cada dominio configurado aparenta ser un equipo completo e independiente. Otra forma de describir a un dominio es verlo como grupo de system board que pueden ser agrupados para formar de uno hasta 16 diferentes dominios los cuales ejecutan una copia del sistema operativo, cada dominio es totalmente independiente de errores de hardware o software que puedan ocurrir en cualquier otro dominio.

En el figura 1-22 se muestra la distribución lógica de un servidor E-10000 con tres dominios, cada uno tiene un número diferente de system board y cada cual cuenta con un determinado espacio en disco en donde debe residir una copia del sistema operativo como mínimo.

[6] Sun educational service, "Ultra Enterprise 10000. Administration ES-400"

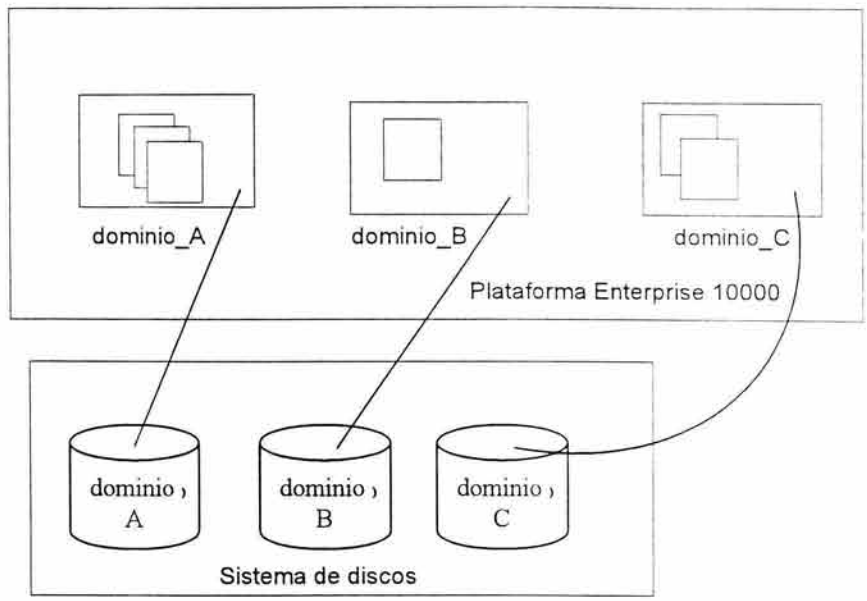


Figura 1-22. Sistema de dominios.

Cada dominio debe ser configurado con sus correspondientes conexiones de red y suficiente memoria y espacio en disco para su adecuado funcionamiento. La capacidad del dominio la limita el número de tarjetas de sistema que este contenga.

Algunas propiedades de los dominios son:

- Cada dominio contiene múltiples tarjetas.
- Cada dominio tiene su propia e independiente copia de Solaris.
- Existe completo aislamiento de fallas entre dominios.
- Una tarjeta de sistema solo puede pertenecer a un dominio a la vez.
- Todos los componentes de la tarjeta del sistema pertenecen al dominio y no pueden ser compartidos con otros dominios.
- Pueden existir de 1 a 16 dominios.

- Varias tarjetas de sistema en un mismo dominio:
    - Tienen una misma dirección de espacio.
    - Ejecutan la misma copia de Solaris.
    - Se puede ver todos los componentes en todas las tarjetas de sistema en el dominio.

1.4.3.4. RUTAS ALTERNATIVAS

Las rutas alternativas (Alternate Pathing "AP") provee al dominio la posibilidad de tener dos distintas rutas de acceso a un mismo dispositivo de E/S, proporcionando redundancia a la etiqueta del controlador de E/S y del cableado.

La mayoría de dispositivos de red (incluyendo las interfases Ethernet y FDDI) son soportados al igual que los dispositivos de almacenamiento StorEdge A5000 y SPARCstorage.

El AP crea una nueva etiqueta a los manejadores de dispositivos (también conocido como meta-disk y meta-networks), a la cual acceden cualquiera de los dos rutas alternativas. De tal forma que los programas aplicativos al hacer referencia a estos nombres de meta-dispositivos no se percatan de la presencia de los AP.

El único requerimiento para poder hacer uso del AP es que el dominio debe de estar equipado con dos interfaces al dispositivo en cuestión, contar con un cable controlador por separado o tener una segunda interfaz a la misma red.

El la figura 1-23 se observa un dominio con dos system-board y que tienen la posibilidad de acceder al arreglo de discos mediante una ruta alternativa. De

esta manera si la ruta activa presenta alguna falla, la comunicación con los dispositivos de almacenamiento no será afectada.

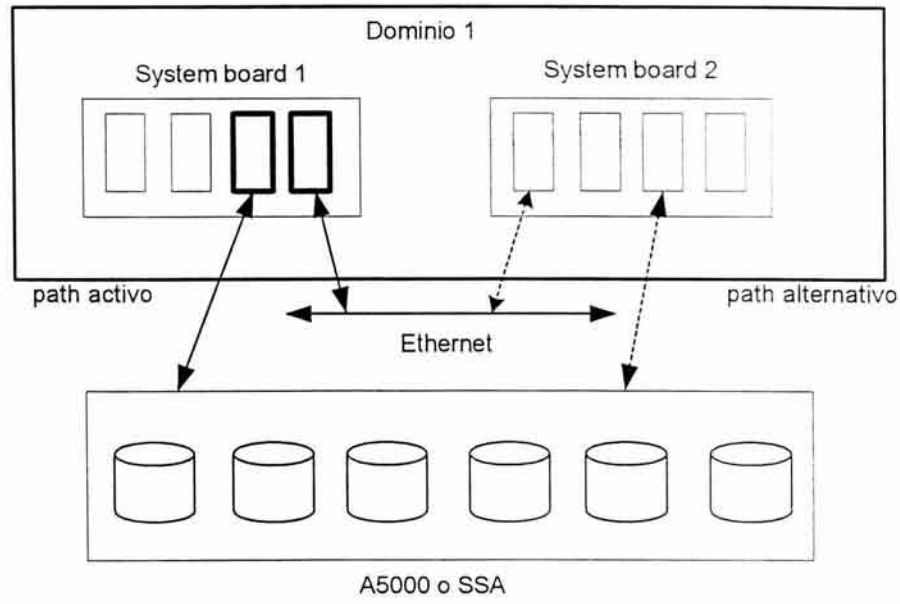


Figura 1-23. Ruta Alternativa (Alternative path)

1.4.3.5. REDUNDANCIA DE COMPONENTES

Por medio de la redundancia de componentes es posible asegurar que el servicio en las áreas productivas y/o críticas no sea interrumpido por causa de falla de alguno de sus elementos.

En el servidor Enterprise 10000 todos sus elementos pueden ser redundantemente configurados si es que así se desea. Los system board pueden ser configurados con redundancia y ser capaz de operar independientemente. Además de los system board también se puede tener redundancia de componentes en:

- Tarjetas de control
- Tarjetas de soporte Centerplane

- Almacenamiento de discos
- Subsistemas de poder
- Suministros de poder
- Controladores periféricos y canales de comunicación
- El SSP y sus interfases

1.4.3.6. RECONFIGURACIÓN DINÁMICA

Mediante la reconfiguración dinámica (Dynamic Reconfiguration "DR") es posible adicionar o mover un system board a un dominio activo, es decir, no importa que dicho dominio se encuentre activo o en ejecución.

El DR puede ser usado para:

- Actualizar el system board
- Modificar la configuración del hardware del dominio
- Reparar fallas del system board
- Reubicar memoria en uso o ponerla en disponibilidad para el dominio
- Remover procesadores en uso o adicionarlos al dominio
- Desconectar o adicionar cualquier dispositivo de E/S al dominio
- Ejecutar test sobre los elementos del dominio

El DR es controlado por el SSP. No requiere que el sistema operativo sea reiniciado después de realizar alguna actualización, lo cual es bastante

conveniente para cuestiones de soporte por no requerir un tiempo exclusivo para estas tareas.

1.5. CONECTIVIDAD LÓGICA DE SERVIDOR ENTERPRISE 10000

El gabinete de la Enterprise 10000 mide 1.8 metros de alto (70"), 1 metro de ancho (39") y 1.3 metros de profundidad (50"). Su peso aproximado (completamente configurado y con el equipo básico) es de 638 kilogramos y requiere de 13.6 kVA de poder.

En la siguiente figura 1-24 se observa el gabinete de la E-10000 así como el System Service Procesor (SSP).

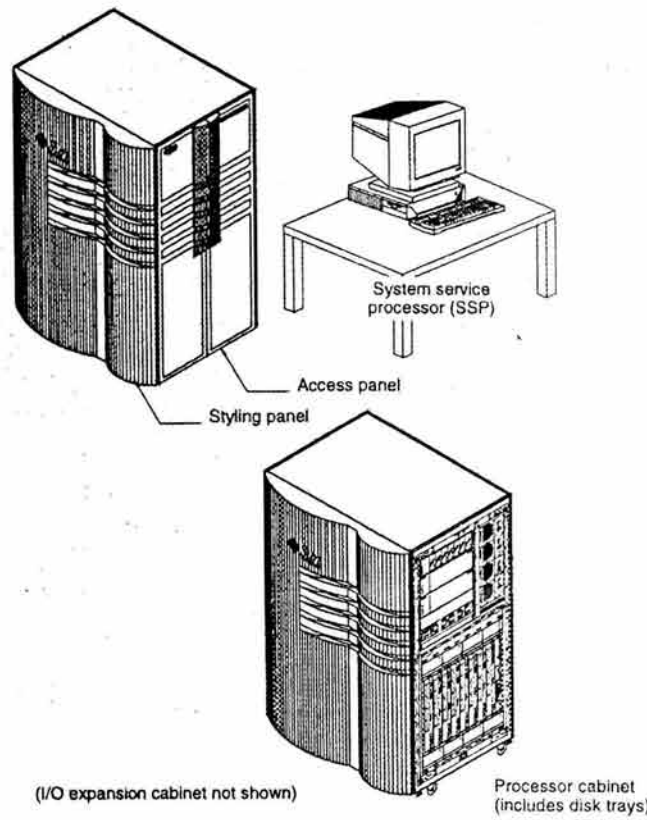


Figura 1-24. Gabinete de la Enterprise 10000.

En la figura 1-25 se muestra el diagrama de la conexión lógica externa del servidor Enterprise 10000 y del Servidor de Servicios de Procesador (SSP). El SSP es conectado vía Ethernet a la tarjeta de control del sistema de la E-10000. La tarjeta de control incluye un procesador que interpreta los datos transmitidos por medio de TCP/IP y lo convierte a JTAG (Joint Test Action Group). Las características especiales de este estándar permite a la tarjeta de control y a otros componentes realizar a si mismos un diagnóstico en vez de requerir probadores especiales.

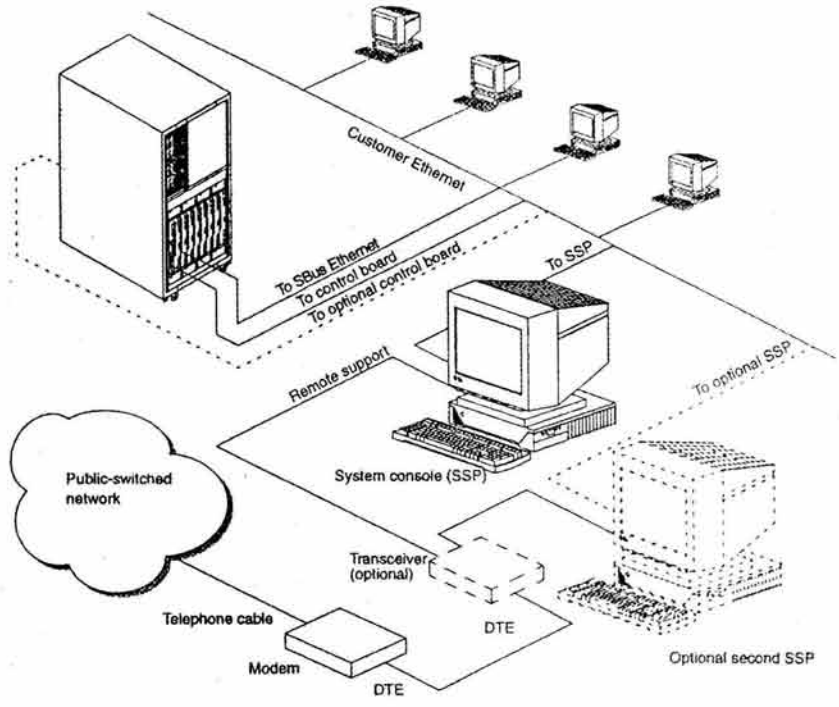


Figura 1-25. Conectividad lógica de la E-10000

1.5.1. LOCALIZACIÓN DE COMPONENTES

En general, los componentes del sistema se enumeran de arriba abajo e izquierda a derecha. En la parte trasera se tienen menor número de componentes en comparación con la parte frontal. Los componentes son:

AC	Módulos de control de corriente alterna de entrada
CB	Tarjeta de control
CSB	Tarjeta de soporte de Centerplane
FT	Bandeja de ventilador
PDU	Unidad de secuencia y distribución de poder de entrada
PS	Suministro de poder de corriente alterna
RCP	Conectores de control remoto de poder
SB#	Tarjetas de sistema (system board)

En la figura 1-26 y 1-27 se muestra una vista frontal y trasera, respectivamente, del gabinete de la E-10000.

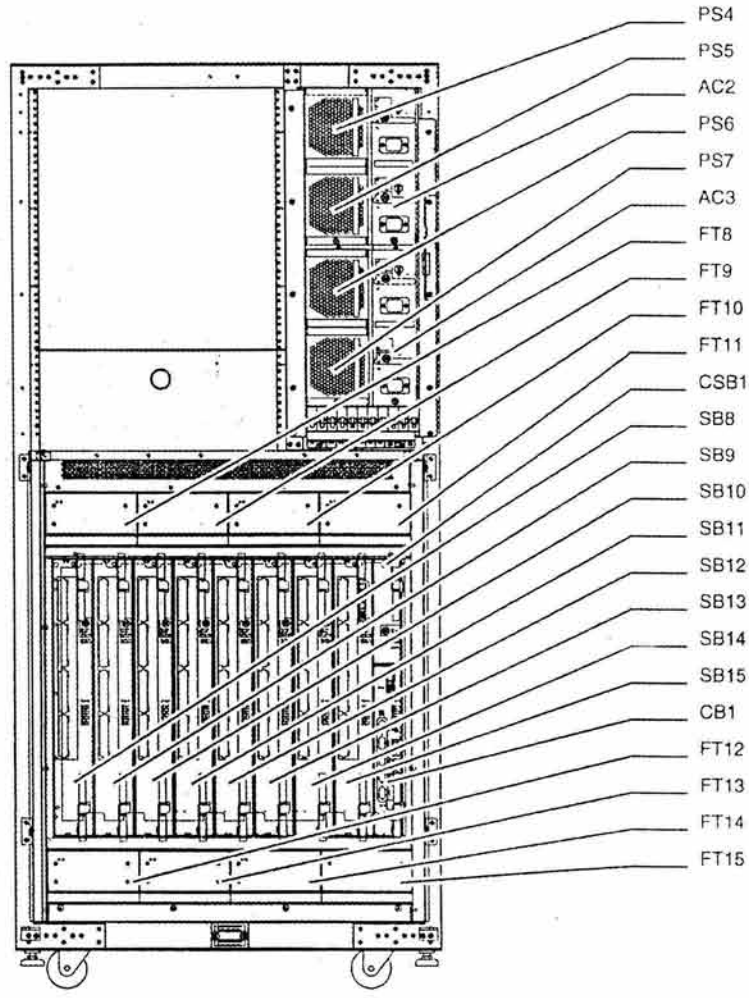


Figura 1-26. Vista frontal del gabinete de la Enterprise 10000.

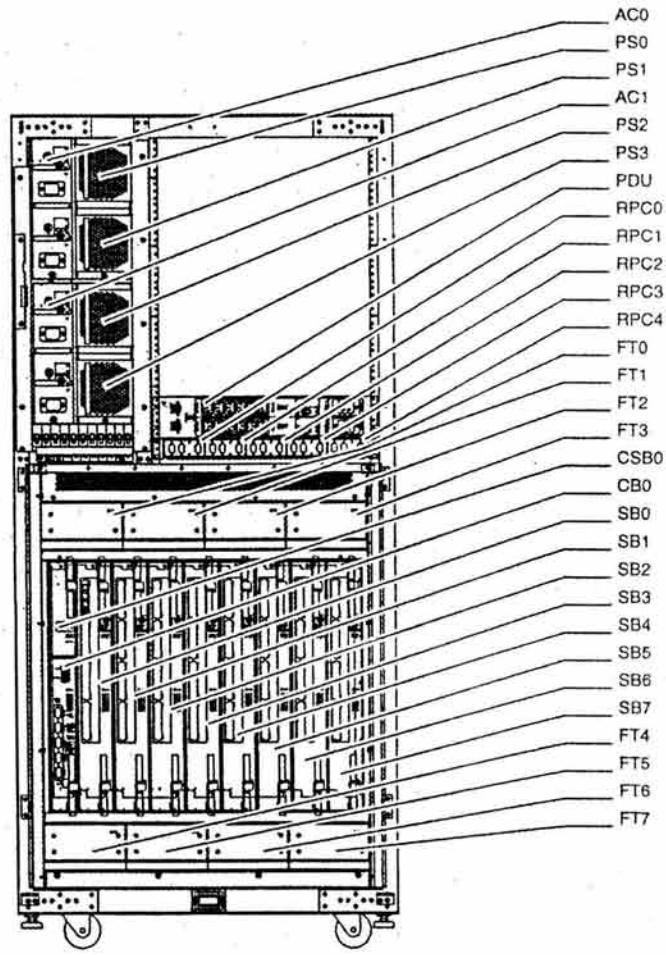


Figura 1-27. Vista trasera del gabinete de la Enterprise 10000.

1.5.2. LA TARJETA DE SISTEMA

La tarjeta de sistema o "System Board" es un circuito impreso multicapas cuya función es la de conectar los procesadores, la memoria principal, los subsistemas de E/S al centerplane. Cada tarjeta de sistema puede soportar:

- Cuatro CPU's de 400 Mhz UltraSPARC-II
- Cuatro bancos de memoria con capacidad de 4 gigabytes por módulo (en total 64 gigabytes para el sistema Enterprise 10000)
- Dos buses de E/S, cada system board con 2 slot Sbus o 1 slot PCI, dando un total de 32 Sbuses con 64 slots Sbus por sistema o 32 slots

PCI por sistema. Los slots PCI y Sbus no pueden ser mezclados en un mismo system board pero si pueden ser mezclados en un dominio.

El diagrama de distribución de los elementos dentro de un system board se presenta en la figura 1-28

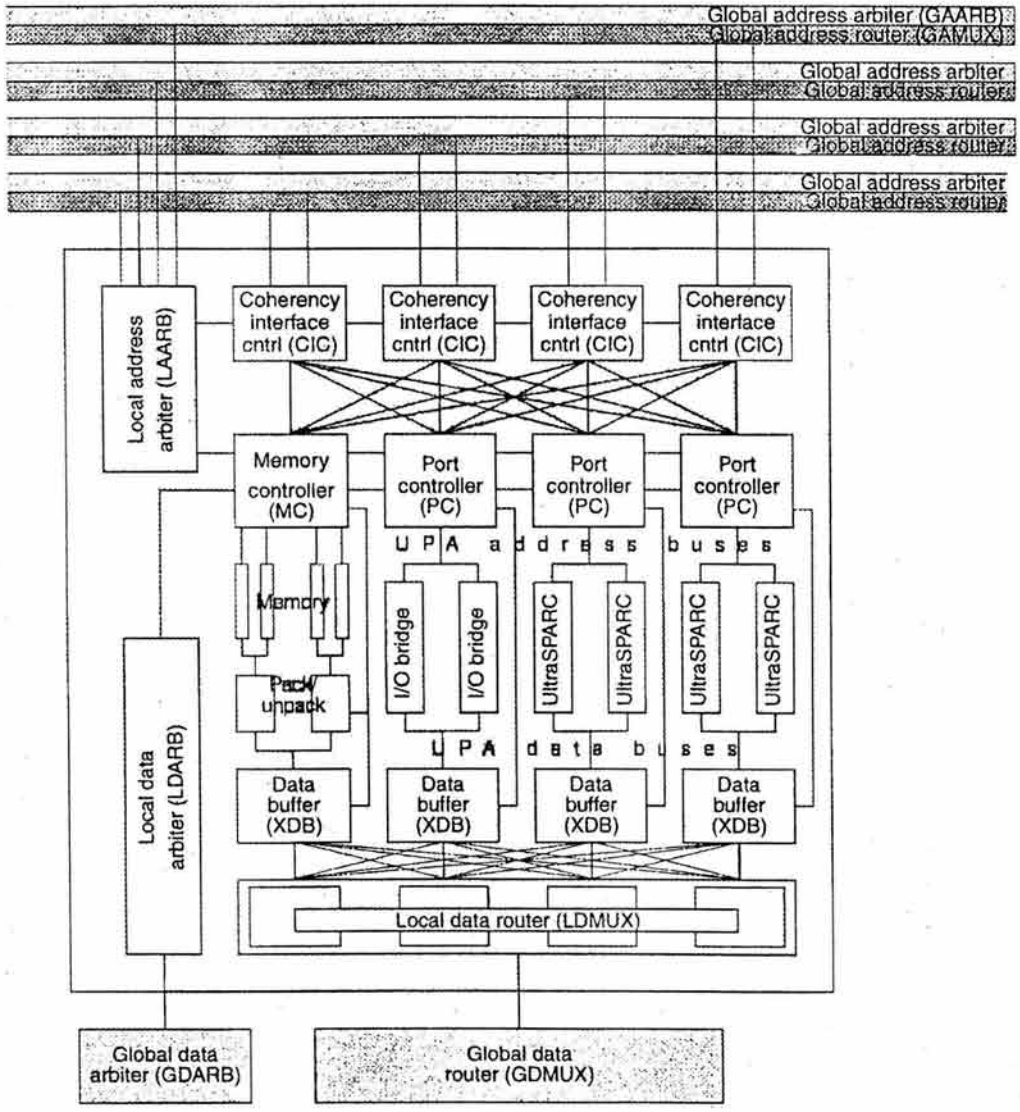


Figura 1-28. Vista lógica del System Board.

CAPÍTULO 2. SITUACIÓN ACTUAL Y SITUACIÓN PROPUESTA

2.1. ¿QUÉ ES ProceSAR?

Con la promulgación de la nueva Ley de IMSS en 1994, se abre la posibilidad de que grupos financieros no gubernamentales administren los fondos de retiros y pensiones de los empleados del sector privado, beneficiando al trabajador, al no tener que generar antigüedad en una sola empresa, que se encargaría de su jubilación.

Así mismo surgen algunas necesidades o áreas de oportunidad de los servicios necesarios para controlar y administrar el cúmulo de información que esta nueva ley implica, es así como el gobierno federal convoca a la Creación de las Administradoras de los Fondos de Ahorro para el Retiro (AFORE), quienes a su vez determinan la necesidad de contar con un elemento central que se encargue de coordinar el flujo de información que este nuevo esquema requiere, conocida como empresa operadoras, así surge ProceSAR.

ProceSAR es la empresa que posee desde 1994 la concesión por parte del gobierno federal para operar la base nacional de datos de SAR (Sistema de Ahorro para el Retiro).

2.1.1. FUNCIONES

Las principales funciones de ProceSAR son:

- Admi4nistrar la Base de Datos Nacional SAR
- Generar y mantener actualizado un listado de los trabajadores que no hayan elegido administradora, que contenga su domicilio y el nombre de su patrón.
- Llevar un sistema contable aprobado por la Comisión Nacional del Sistema de Ahorro para el Retiro (CONSAR).

- Mantener actualizada la Base de Datos Nacional SAR, entre otros datos, con:

  - La información del registro de trabajadores en las administradoras e institutos de seguridad social respectivamente.

  - Los números de seguridad social y claves únicas de registro de población de los trabajadores, proporcionados por el Instituto Mexicano del Seguro Social, y en su caso, el Instituto de Seguridad y Servicios Sociales de los Trabajadores del Estado, y Registro Nacional de Población.

  - La información de los retiros realizados con cargo a las cuentas individuales.

- Recibir del Instituto del Fondo Nacional de la Vivienda para los Trabajadores la información relativa a los trabajadores, a los que les asignó o les cancele créditos, así como informar de lo anterior a las administradoras.

- Informar al Instituto del Fondo Nacional de la Vivienda para los Trabajadores (INFONAVIT), de las aportaciones y descuentos que se reciban y correspondan a trabajadores a los que se les haya asignado un crédito del mencionado Instituto.

- Informar a quien la Comisión les indique, las tasas de rendimiento de la cuenta concentradora que, a su vez, les haya informado el Banco de México.

- Informar a las administradoras sobre las tasas de rendimiento que deberán aplicar a las subcuentas de vivienda de los trabajadores que tengan registrados en términos de lo dispuesto por la Ley del INFONAVIT que, a su vez, les haya informado el Banco de México.

2.1.2. ESTRUCTURA ORGANIZACIONAL

En el organigrama de la figura 2.1 se puede observar la estructura de la dirección general de ProceSAR y las cuatro direcciones que de esta dependen.

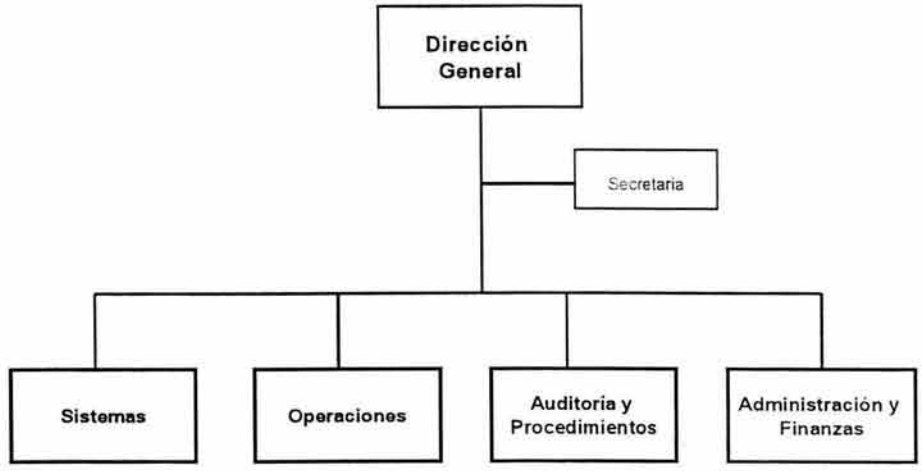


Figura 2-1. Organigrama de la dirección general

La empresa se encuentra conformada por una dirección general y cuatro direcciones, las cuales son:

- Dirección de Administración y Finanzas. Sus principales funciones son:
  - Planear, administrar y controlar los Recursos Humanos y Financieros de la empresa.

- Coordinar, planear y administrar los servicios internos hacia las áreas operativas de la empresa. (Recursos Humanos, Seguros, Compras, Tesorería, Mensajería y Recepción.

- DIRECCIÓN DE AUDITORIA Y PROCEDIMIENTOS. Sus principales funciones son:

  - Verificar y validar, a través de la aplicación de pruebas selectivas, que las distintas operaciones de la empresa se realicen de acuerdo a las Normas, Políticas y Procedimientos establecidos por el Consejo de Administración, la Alta Dirección, CONSAR y/o cualquier otro organismo regulatorio.

  - Coordinar las diferentes auditorias externas.

- DIRECCIÓN DE OPERACIONES. Sus principales funciones son:

  - Coordinar, definir y supervisar la operación de los procesos que se deriven de las funciones de conciliación y dispersión de las cuotas de retiro, cesantía y vejez del Seguro Social, registro de trabajadores a las AFORES, traspaso de cuentas entre los participantes del sistema de pensiones (Bancos, AFORES), retiros por jubilación, recaudación y otorgamiento de créditos de vivienda, cálculo y dispersión de la cuota social y aportación gubernamental, administración de la información de los trabajadores pendientes de registro, conciliación de la cuenta concentradora en Banco de México, inversión (temporal) de los recursos de la recaudación para su posterior dispersión utilizando los mecanismos de pagos electrónicos disponibles en el sistema financiero.

- DIRECCIÓN DE SISTEMAS. Sus principales funciones son:

- Definir, desarrollar e implantar los servicios informáticos necesarios para la operación del nuevo sistema de pensiones, integrando a las soluciones desarrolladas, los elementos tecnológicos necesarios para su óptimo funcionamiento y el beneficio directo de los accionistas de la empresa. Se entiende por servicios informáticos a la integración de procesos operativos, elementos tecnológicos, sistemas aplicativos, políticas y procedimientos necesarios, para establecer los flujos de y medios de procesamiento de información requerida para la operación del nuevo sistema de pensiones[7].

- Definir, desarrollar e implantar los servicios informáticos necesarios para la operación de la empresa, bajo los lineamientos descritos en el párrafo anterior.

- Definir e instrumentar constantemente planes de optimización de los servicios informáticos mencionados, con el objetivo de minimizar costos, maximizar la eficiencia de los mismos y garantizar su correcto funcionamiento bajo los estándares que determina la normatividad asociada.

- Garantizar que el diseño y desarrollo de los servicios informáticos se apegue en todo momento a lo que marca la normatividad asociada a la operación del nuevo sistema de pensiones.

- Determinar y aplicar las políticas generales de análisis, diseño y desarrollo de los servicios informáticos, buscando en todo momento cumplir con los estándares internacionales de calidad y eficiencia.

7 Manuales de políticas y procedimientos de ProceSAR

- Desarrollar y resguardar los elementos de análisis y diseño de los servicios informáticos, tales como diagramas conceptuales, técnicos, arquitectura de datos, entre otros.

- Elaborar planes integrales de Tecnología para la empresa usando sus conocimientos profundos de las soluciones tecnológicas existentes y requeridas.

- Administrar, coordinar y supervisar la ejecución de los planes tecnológicos de acuerdo a las necesidades y proyectos establecidos para la empresa.

- Manejar los departamentos técnicos, coordinando sus esfuerzos para que se alcancen los objetivos de la empresa.

- Coordinar las relaciones con proveedores de servicios outsourcing de software, hardware y telecomunicaciones.

2.1.3. INTERELACIONES

En la figura 2.2 se puede observar las interrelaciones que existen dentro de ProceSar.

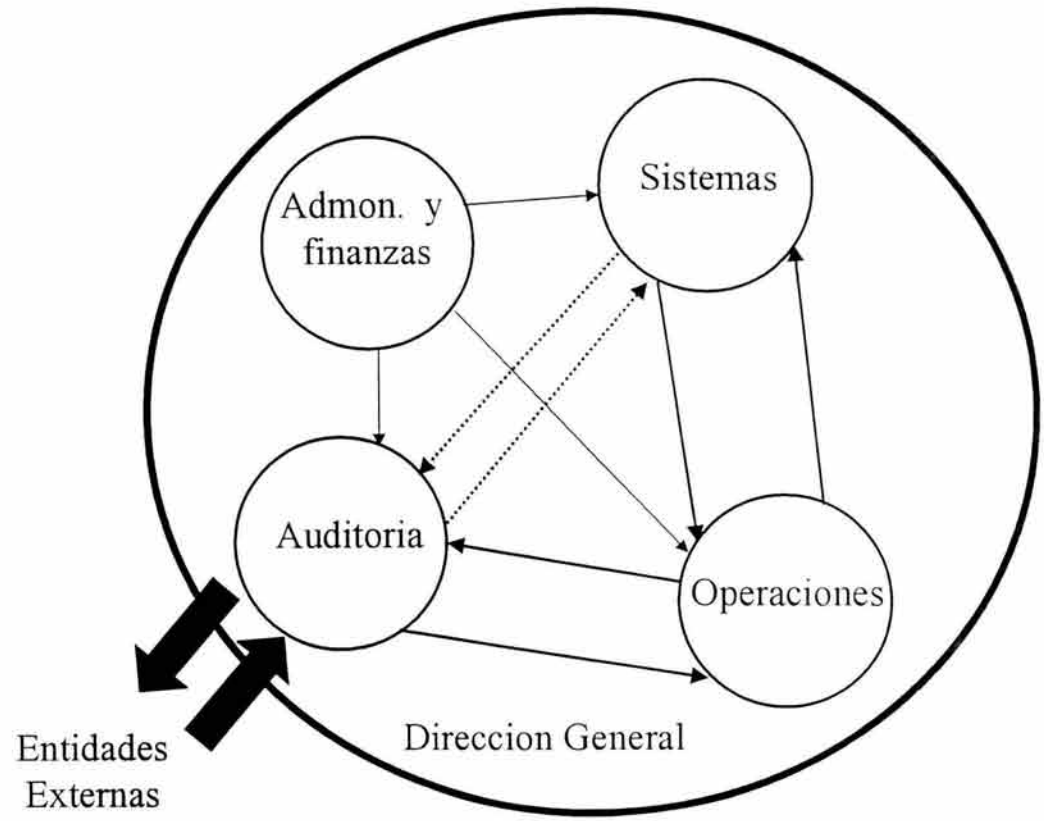


Figura 2-2. Interrelaciones en ProceSAR.

La dirección general gobierna las políticas generales de la empresa, mientras que el área de Administración y finanzas se encarga de que los recursos humanos y financieros se aprovechen adecuadamente. El flujo de la información y procesos inician en la dirección de auditoria la cual es canal oficial de comunicación entre las entidades externas (clientes), con las direcciones de operaciones y de sistemas las cuales se encargan de procesar la información administrada y entregar los resultados solicitados. Ambas direcciones (operaciones y sistemas) se encuentran permanentemente vigiladas por la dirección de auditoria con el fin de comprobar que se cumpla puntualmente con los procedimientos establecidos.

2.1.4. RELACIÓN ENTRE ProceSAR Y SUS CLIENTES

Los clientes de ProceSAR se encuentran agrupados en distintos grupos los cuales se pueden identificar claramente en la tabla 2.1

Tipo	Descripción	Nombre de cliente(s)
Administradoras de fondos de ahorro para el retiro (AFORES)	Instituciones encargadas de administrar los fondos para el retiro de los trabajadores afiliados al sistema de pensiones.	• Zurich, S.A. De C.V. • Tepeyac, S.A. De C.V. • XXI, S.A. De C.V. • Sólida Banorte, S.A. De C.V. • Bancrecer-Dresdner, S.A. De C.V. • Profuturo GNP, S.A. De C.V. • Principal AFORE, S.A. De C.V. • Santander-Mexicano, S.A. De C.V. • Bital-Ing, S.A De C.V. • Garante, S.A. De C.V. • Inbursa, S.A. De C.V. • Banamex, S.A. De C.V. • Bancomer, S.A. De C.V
Prestadoras de servicio	Instituciones encargadas de administrar los fondos para el retiro de los trabajadores que aún no han elegido alguna afore.	• Zurich, S.A. De C.V. • Tepeyac, S.A. De C.V. • XXI, S.A. De C.V. • Sólida Banorte, S.A. De C.V. • Bancrecer-Dresdner, S.A. De C.V. • Profuturo GNP, S.A. De C.V. • Principal AFORE, S.A. De C.V. • Santander-Mexicano, S.A. De C.V. • Bital, S.A. De C.V. • Garante, S.A. De C.V. • Inbursa S.A. De C.V. • Banamex, S.A. De C.V. • Bancomer, S.A. De C.V.
ICEFAS	Instituciones crediticias	• Banco Nacional de México

	especializadas en fondos de ahorro del sistema del 1992 (SAR 92)	S.A. • Serfin S.A. • Banco del Atlantico S.A. • Citibank México S.A. • Bancomer S.A. • Banco internacional, S.A. • Banco inverlat, S.A. • Banco mercantil del norte, S.A.
Organismos	Instituciones que representan al gobierno federal y su fin es el de vigilar el manejo correcto del fondo de ahorro de los trabajadores, así como el regular y auditar todos los procesos que se presentan dentro del nuevos sistema de pensiones.	• CONSAR (Comisión Nacional del Sistema de Ahorro para el Retiro) • TESOFE (Tesorería de la Federación) • CONDUSEF
Institutos	Estos son dos organismos estatales. Uno se encarga de gobernar todas las operaciones realizadas en nuevos sistema de pensiones y el otro es el encargado de otorgar los créditos de vivienda a los trabajadores.	• IMSS (Instituto Mexicano del Seguro Social) • INFONAVIT (Instituto del Fondo Nacional de Vivienda de los Trabajadores)
Entidades receptoras de pagos patronales (bancos)	Son las instituciones bancarias acreditadas para realizar la recepción de los pagos patronales.	• Banco Nacional de México • Banca Serfin • banco del atlántico • Bancomer • Banco Santander-Mexicano • Banco Bilbao Vizcaya • Bital • Inverlat • Promex

		• Banpais • Banco del Norte • Bancrecer

Tabla 2.1 Clientes

En la figura 2.3 se puede observar claramente la relación que existe entre ProceSAR y sus distintos clientes:

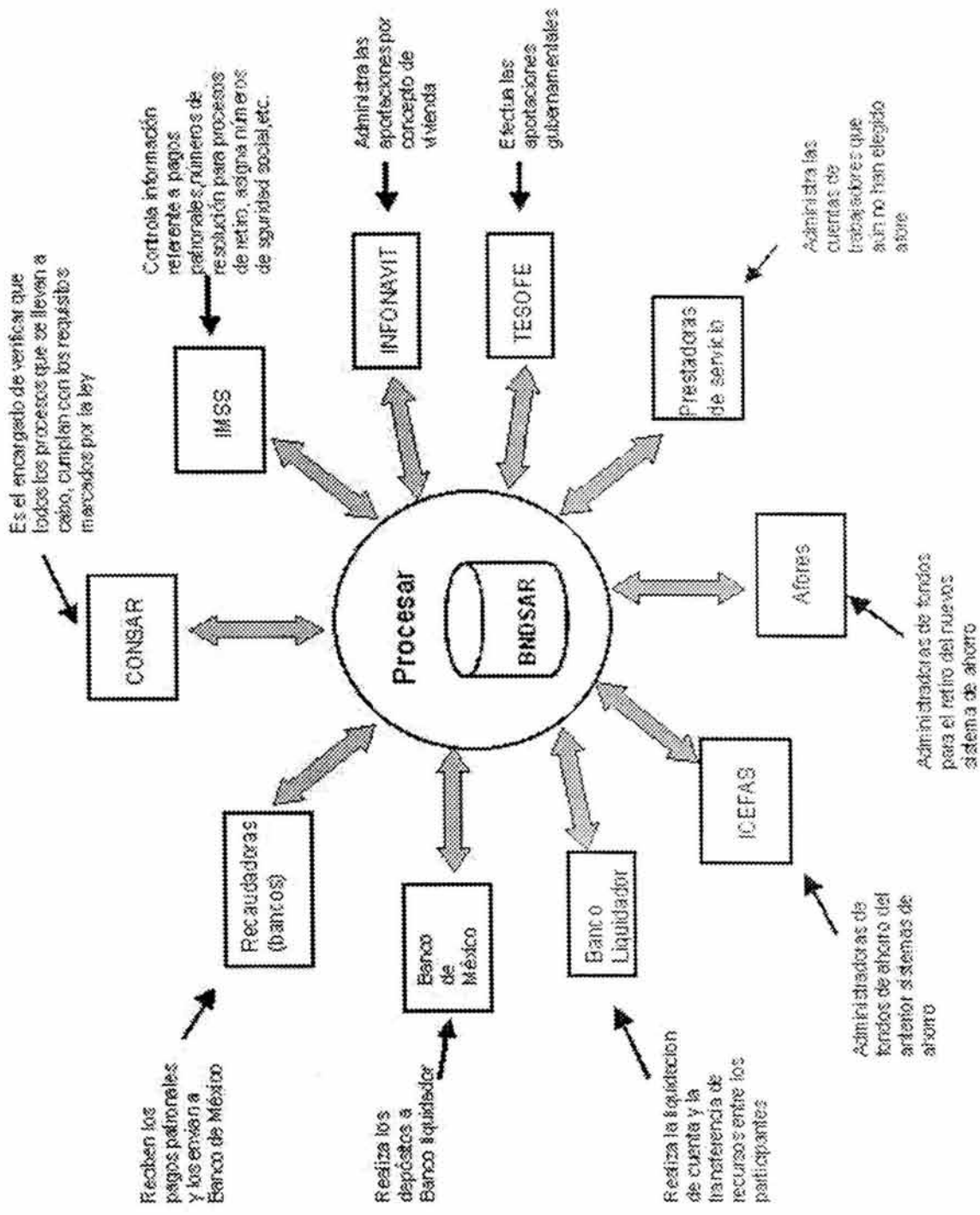


Figura 2- 3. Relaciones entre ProceSar y sus clientes

2.2. IDENTIFICACIÓN DE LOS DISTINTOS AMBIENTES DE DESARROLLO DE SISTEMAS

Con la finalidad de que todos los sistemas elaborados se encuentren debidamente documentados y validados existen distintos ambientes o etapas de desarrollo, en cada uno de estos ambientes se realiza una actividad distinta.

Para que una aplicación, programa o modificación sea instalado al ambiente productivo debe antes pasar por:

- Test. Aquí es donde se crean todos los programas y aplicaciones. Este ambiente es exclusivo para los desarrolladores. La capacidad de este ambiente es limitada en cuestión de almacenamiento de base de datos ya que en este punto los constructores (programadores) solo realizan las pruebas unitarias de programas que son generados basándose en las especificaciones proporcionadas por los analistas.

- Validación. En esta segunda etapa del desarrollo el ambiente se encuentra mas fortalecido que el de "Test" ya que es aquí donde se realizan las primeras pruebas integrales tomando como entrada muestras de información productiva. Es en este punto donde se realiza la validación de cada uno de los elementos construidos por los programadores. En caso de existir inconsistencias, se genera una lista de las deficiencias encontradas para que sean corregidas en el ambiente de Test.

- Aceptación. Esta es la última escala que los programas y aplicaciones deben de hacer antes de su puesta en marcha en el ambiente productivo. El objetivo principal de este ambiente es el de realizar las pruebas finales con el usuario y obtener su visto bueno.

En la figura 2.4 se presenta esquemáticamente el flujo que un programa o aplicación debe de seguir antes de ser instalado en el ambiente productivo.

Figura 2-4. Ambientes de desarrollo.

2.3. IDENTIFICACIÓN DE LAS APLICACIONES PRODUCTIVAS

2.3.3. REGISTRO

Esta aplicación es la encargada de realizar la certificación (Registro) para dar de alta a los empleados en la Base de datos Nacional del SAR.

Su objetivo es formar la BASE DE DATOS NACIONAL DEL SAR a través de la información que envían las Afores vía File Transfer (Conect Direct) o Medio Magnético.

En la figura 2.5 se muestra gráficamente el flujo conceptual que se lleva a cabo para realizar un registro.

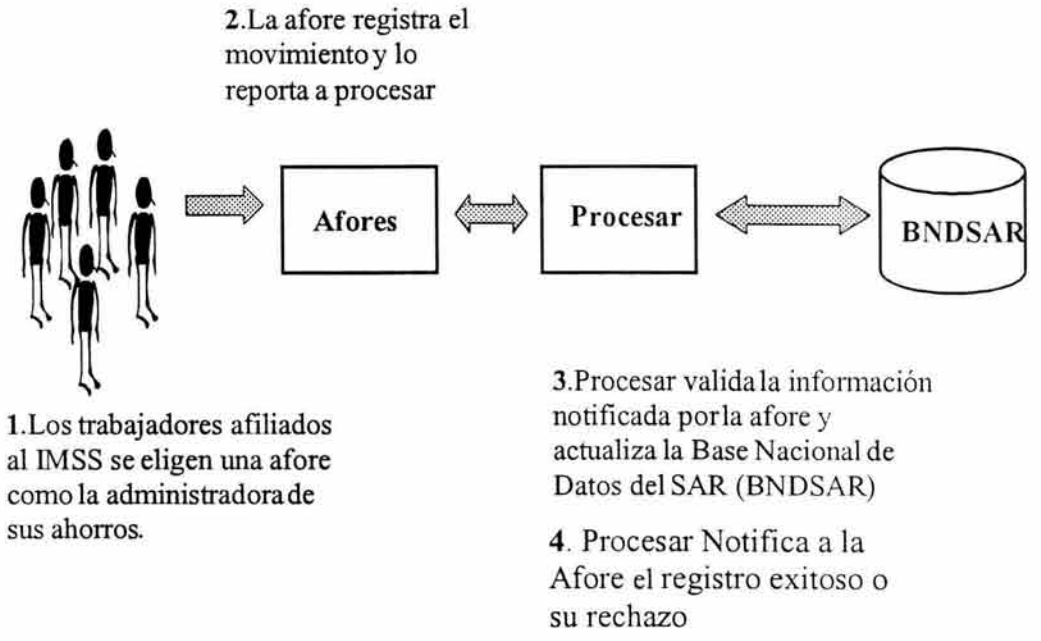


Figura 2-5. Flujo conceptual de una registro.

2.3.2. TRASPASOS

El nuevo Sistema de Pensiones establecido por la CONSAR especifica que cada trabajador tiene el derecho de elegir la AFORE de su preferencia[8], por lo tanto, éste podrá hacer el traspaso de sus recursos a la AFORE deseada.

[8] Ley del sistema de ahorro para el retiro, "articulo 74".

Los requisitos para que el trabajador pueda realizar el traspaso de los recursos a la AFORE están regulados por la Ley del Seguro Social y la Ley del Sistema del Ahorro para el Retiro. ProceSAR fungirá como intermediario en el proceso de recepción, procesamiento y envío de solicitudes de traspasos y liquidación a las entidades cedentes (actuales administradoras del SAR) y las receptoras.

En la figura 2.6 se muestra claramente el proceso que debe seguir un trabajador cuando desea realizar un cambio de afore.

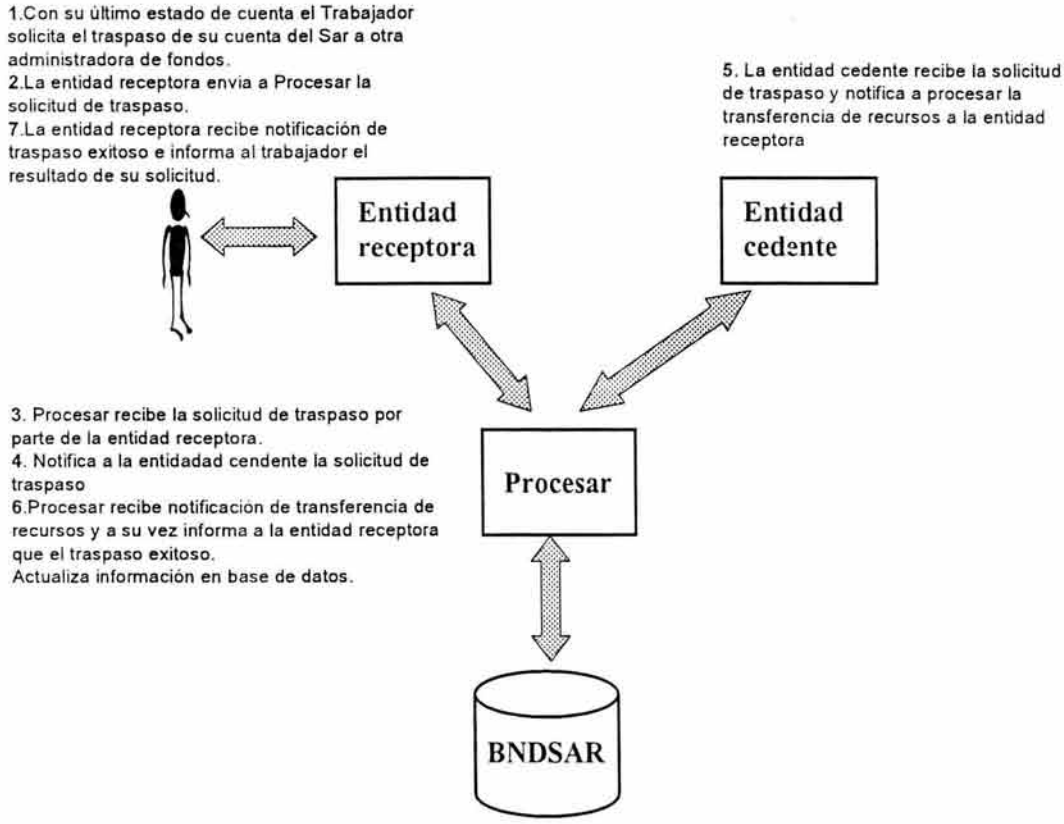


Figura 2-6. Flujo conceptual para realizar traspaso de afore.

2.3.3. RECAUDACIÓN

En este módulo se realiza la conciliación e individualización de pagos patronales y aportaciones tanto de los trabajadores como del gobierno federal.

El principal objetivo principal es el de hacer llegar el dinero recaudado a sus respectivas cuentas de ahorro. En la figura 2-7 se puede observar como se lleva a cabo este proceso.

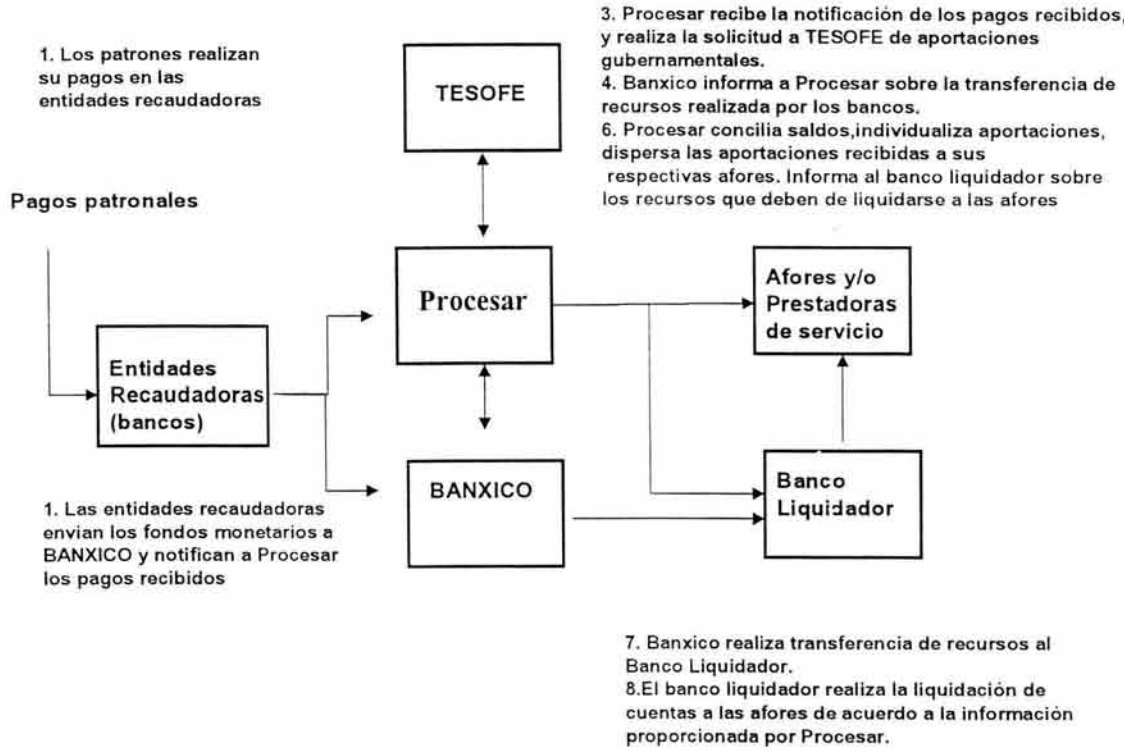


Figura 2-7. Flujo conceptual del proceso de recaudación

2.3.4. RETIROS

El Nuevo Sistema de Pensiones establecido en la Ley del Seguro Social especifica que un trabajador puede ejercer su derecho a retirar los fondos acumulados en su cuenta individual, siempre y cuando este mismo cumpla con ciertos requisitos especificados en la Ley antes mencionada.

En el proceso de Retiros, la función de ProceSAR será la de fungir como intermediaria para la recepción y el envío de la información entre el Gobierno Federal, los Institutos y las AFORES.

En la figura 2.8 se muestra el modelo conceptual de un proceso de retiro.

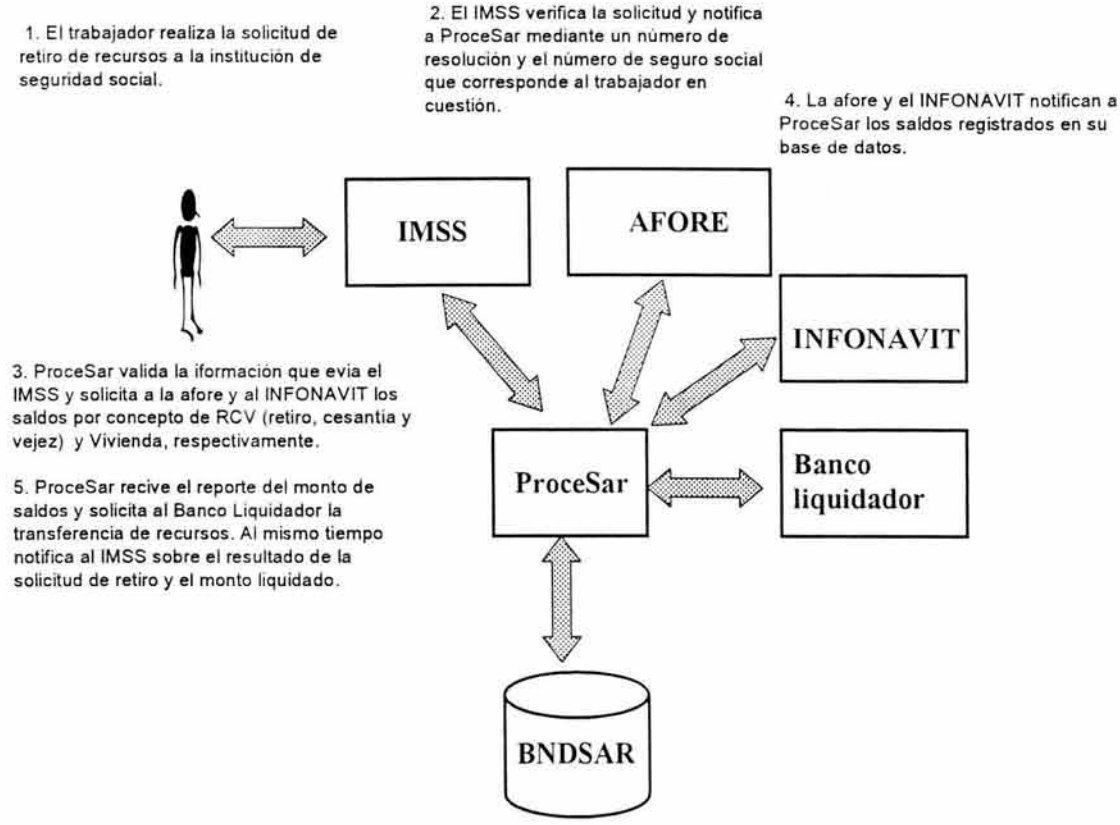


Figura 2-8. Flujo conceptual del proceso de retiro.

2.4. SITUACIÓN ACTUAL

ProceSAR como una compañía mexicana líder en el ramo de procesamiento automático de información, y única por el tipo de información que procesa, tiene que estar en constante mejoramiento de su infraestructura tecnológica, tanto en hardware como software, por esto continuamente se esta investigando para

encontrar las herramientas o métodos de solución a problemas que día a día se presentan, siempre tratando de cubrir las necesidades de sus clientes, las Afores, y los organismos gubernamentales como lo son el IMSS e INFONAVIT.

2.4.1. ESTRUCTURA TECNOLÓGICA DE ProceSAR

Debido a la naturaleza de los servicios tecnológicos que presta ProceSAR, esta empresa cuenta con una completa infraestructura de cómputo, tanto de hardware como de software, para el procesamiento, telecomunicación, y explotación de la información relacionada con la administración de la BNDSAR.

2.4.1.1. EQUIPO MAINFRAME

Sin duda alguna una parte muy importante de ProceSAR en su estructura tecnológica es que está basada en IBM, cuenta como unidad principal un MAINFRAME IBM 9672-RB6 con sistema operativo OS-390 en el cual se realizan los principales procesos productivos, que corresponden a la operación y explotación de la BNDSAR, este equipo cuenta con dos procesadores que realizan 165 Mips (millones de instrucciones por segundo), tiene una memoria de 1Gb donde 728 Mb corresponden a memoria real y 256 Mb a expandida. Así mismo cuenta con un dispositivo central de almacenamiento (DASD) con 2.3 TB en unidades de disco, y utiliza 12 cartucheras para el almacenamiento secundario, administradas por un sistema automático Storage Tek capaz de manejar 5000 cartuchos en su base de datos, la capacidad de los cartuchos varia de 2.5, 10 y 30 Gb según su tipo. Y cuenta con 12 canales paralelos de 4.2 Mb/seg y 48 canales escon de 17 Mb/seg, los cuales se utilizan para controlar las unidades de discos, cartucheras, comunicación con las estaciones de trabajo y conectividad para la recepción de archivos de lotes de proceso. En

la figura 2.9 se muestra una vista frontal del equipo mainframe con el que cuenta ProceSar.

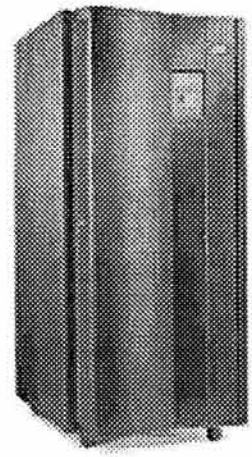


Figura 2-9. Vista frontal del Mainframe tipo utilizado en ProceSAR

El mainframe cuenta con una terminal gráfica y dos terminales tontas desde donde es administrado y configurado, el SITE donde se encuentra dicho equipo está arrendado a la misma empresa IBM, quien se encarga de proporcionar todas las características de instalación para el buen funcionamiento del equipo, así como de tener que recurrir a una red WAN para su conectividad con otros equipos.

2.4.1.2. EQUIPO HP 9000

Como un equipo auxiliar de los procesos estadísticos de obtención de resúmenes de información productiva y capa intermedia de seguridad en la transferencia de archivos, se cuenta con la unidad HP9000 con sistema operativo UNIX.

El servidor HP9000 es un series 800/K640, el cual posee dos procesadores PA 8000/180 Mhz con 2Mb de cache en memoria, y memoria principal de 1 Gb, cuenta con dos discos internos SCSI de 4 y 6 Gb respectivamente, y un arreglo de 10 discos con 4.6 Gb por disco y un segundo arreglo de 12 discos con 36 Gb en cada disco, como dispositivo de almacenamiento masivo cuenta con una unidad CD-ROM y dos unidades de cinta, una HP C1533A de 4mm y una DLT4000 de ½ pulgada, posee una consola de operación, dos tarjetas de red 10/100 base T, un multiplexor integrado de 16 puertos TS232, 6 tarjetas FWD SCSI.

2.4.1.3. INTERRELACIÓN TECNOLÓGICA

Podemos definir que la pieza central de la estructura tecnológica de ProceSAR es el mainframe IBM, y que alrededor de éste se conectan y superponen las capas de la red conteniendo los sistemas de consulta, servidores de seguridad, y por ultimo las terminales finales.

Como usuarios finales podemos mencionar los diseñadores y desarrolladores de los sistemas, los operadores del sistema, los especialistas en sistemas IBM, los auditores y personal de control de operaciones, y las entidades externas, quienes proporcionan la información para la operación del sistema y realizan consultas sobre esta misma. En la figura 2.10 se muestra un esquema de la interrelación tecnológica que existe en ProceSAR.

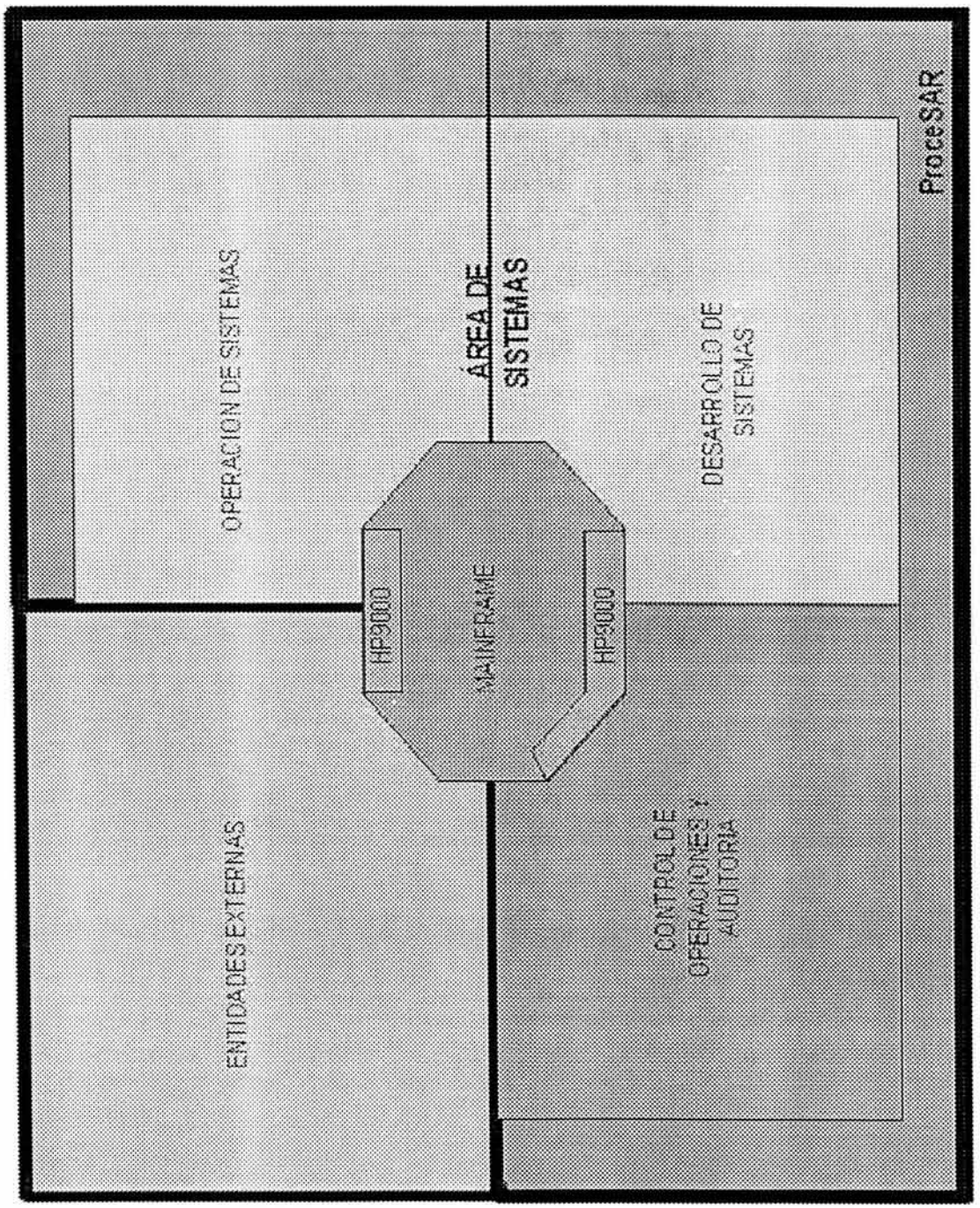


Figura 2-10. Topología tecnológica de ProceSar

Del esquema anterior podemos señalar las diferentes áreas de ProceSAR que actúan con el sistema, tenemos así a desarrollo, quien hace uso del mainframe y a la HP9000 para desarrollar y probar los nuevos sistemas creados, operación de sistemas, quien realiza en si la ejecución de los procesos. También tenemos al área de control de operaciones y auditoria que mayormente acceden vía el sistema de consultas a través de la HP9000, y por ultimo a los participantes externos quienes transmiten y reciben información al mainframe y realizan consultas también mediante los sistemas implementados en la HP9000.

2.5. PROBLEMÁTICA ACTUAL

Debido a que el mercado en el que ProceSAR se encuentra es un mercado de clientes cautivos y que además esos mismos clientes son los dueños de la empresa, resulta ser que esta empresa debe trabajar única y exclusivamente con los recursos indispensables para su operación.

Por lo anterior resulta primordial que la empresa este en condiciones de prestar todos estos servicios y que además estos sean de muy buena calidad y adicionalmente tener el costo más bajo para los socios de la empresa.

Con este afán, la directiva de ProceSAR, evaluando los actuales costos de operación y los servicios que actualmente se prestan, y los que se tienen proyectados prestar a futuro, ha decidido tomar una serie de medidas para reducir costos de operación. En este sentido se han venido aplicando una serie de reestructuraciones de las áreas que conforman la empresa.

En el área de sistemas, específicamente en desarrollo de sistema, se ha procurado recurrir cada vez en menor medida al outsorcing y consultoría

externa, conformando un área autosuficiente para desarrollar nuevos productos, mantener los ya existentes, y responder rápida y oportunamente a los problemas que surjan con los procesos ya productivos.

Actualmente el mainframe IBM demanda una gran cantidad de personal del área de sistemas de soporte técnico, para mantener en optimas condiciones de funcionamiento este equipo, se cuenta con cuatro especialistas en el sistema operativo OS/390, que se encargan de monitorear el funcionamiento del mainframe, a fin de detectar posible fallas o rendimiento inadecuado, por lo que continuamente lo están poniendo en punto (tunning), y cada día obtienen nuevas actualizaciones al sistema, y en casos muy drásticos, se recurre a especialistas de IBM para la corrección de problemas.

Así mismo se alquila el SITE con un espacio de más de 150 m^2 para que el equipo IBM y todos sus periféricos cuenten con las condiciones de operación adecuadas, así mismo, periódicamente personal de IBM, da mantenimiento preventivo a todos los equipos que aquí funcionan, sea unidad principal, unidades de discos, robot de cartuchos, sistema de corriente regulada e ininterrumpida.

Todos estos servicios prestados por IBM tienen tarifas fijadas por la misma compañía IBM, y debido a que no existe en el mercado otra compañía no relacionada con IBM, que pueda competir con esta, se puede decir que se está cautivo en sus servicios, mientras el equipo sea IBM, se tendrán que pagar las tarifas fijadas por esta.

Esto genera el inconveniente de tener una plataforma cerrada, que aun cuando es muy segura, no permite la expansión hacía nuevas tecnologías que IBM aun no considera, tales como una seguridad integrada a accesos de Internet, donde

se pueden dar servicios utilizando tecnologías como lo es el JAVA o servlets, tecnología que ProceSAR ya ha venido aplicando y utilizando con mucho éxito en servicios vía web, ya sea por intranet, la red WAN o Internet

Y por último, el inconveniente más importante, es el ser cliente cautivo de IBM. Es así como se llegó a la decisión de cambiar la plataforma principal de procesos, es decir el mainframe IBM 390 por el equipo Enterprise 10000 el cual opera con sistema operativo "UNIX".

2.6 RAZONES PARA CAMBIAR HACIA UNA PLATAFORMA DE SISTEMAS ABIERTOS

Los "Sistemas Abiertos (S.A.) son aquellos sistemas y componentes que pueden ser especificados y adquiridos de fuentes distintas en un mercado competitivo" [9]

Cuando se habla de sistemas abiertos, existen dos puntos en que todo el mundo concuerda: que éstos constituyen una gran idea y que conlleva a grandes beneficios al final, pero al mismo tiempo esto puede ser confuso por la gama de posibilidades que permite.

Los sistemas abiertos están basados en estándares de la industria y éstos envuelven de muchas maneras un objetivo que no está claramente definido. Una apertura, por su naturaleza crea opciones y selecciones que tiende a la confusión. De ahí surge la pregunta: ¿Cómo se debe proceder?

Años atrás, los sistemas existentes (de varios fabricantes) no podían compartir información entre ellos. Las comunicaciones se realizaban a través de

9 http://www.monografias.com/trabajo5/sistab/sistab2.stml

teléfonos, faxes y cartas. Esto tenía que cambiar para poder optimizar el flujo y manejo de la información dentro de la organización.

Hoy en día son muchas las Empresas que están cambiando a sistemas abiertos. La necesidad de mover información libremente a través y entre organizaciones es lo que conlleva a este tipo de sistemas, y mientras más rápido se conviertan en una parte de este cambio, mejor.

2.6.1. ¿POR QUÉ UN SISTEMA ABIERTO?

Los Sistemas Abiertos ofrecen soluciones viables y desde su origen hasta la actualidad han evolucionado rápidamente dando lugar a un nuevo ambiente competitivo, donde tanto usuarios como proveedores tienen su participación.

Los Sistemas Abiertos se presentan como una alternativa rentable y confiable por lo que se deben tomar en cuenta en la selección de un sistema de procesamiento electrónico de datos, aumentando cada vez más el número de empresas que adquieren esta tecnología para el manejo y procesamiento de sus informaciones.

El movimiento hacia los sistemas abiertos se ha convertido en una disciplina complementaria de la Informática. También ha dado lugar a la conformación de Organismos para fomentar la estandarización en el uso de éstos sistemas, por lo cual consideramos de gran importancia valorar los beneficios que se obtendría al realizar la migración de un sistema cerrado a uno abierto. A continuación se exponen algunos de los puntos importantes que justifican la migración hacia un sistema abierto.

2.6.1.1. LIBERTAD DE ELECCIÓN.

Tanto en lo que toca al equipo físico, como a los programas o servicios. Esta es justamente la situación opuesta a lo se conoce como el síndrome de "cliente cautivo"[10], y que con frecuencia conduce a que ampliaciones o sustituciones de equipos en entornos "propietarios" se hagan en unas condiciones de negociación muy desfavorables para la Administración. Por lo que se refiere a los programas hay otro importante elemento a destacar, y este es la amplia variedad de programas disponibles en el mercado para sistemas abiertos, que realizan funciones similares o funciones para cuya realización no existen programas en entornos propietarios.

2.6.1.2. PROTECCIÓN DE LA INVERSIÓN.

En este apartado hay que destacar la cuestión de la continuidad de la oferta. En la situación de crisis en que se desenvuelve hoy la industria informática mundial, los usuarios corren el riesgo de que su suministrador desaparezca del mercado. Si se ha adoptado una solución basada en S.A. ello no es grave, puesto que la ampliación del sistema y los servicios asociados a su uso pueden ser contratados con un tercero.

2.6.1.3. MEJOR RELACIÓN PRECIO/RENDIMIENTO.

En el caso de los Sistemas Abiertos, los precios no se determinan en régimen de monopolio de oferta, existe la posibilidad de integrar dispositivos de varias

---

75[10] http://www.monografias.com/trabajo5/sistab/sistab2.stml

procedencias en un sistema, y se produce una rápida adopción de la nueva tecnología, gracias a la competencia existente.

CAPÍTULO 3. ESTÁNDARES DE DESARROLLO

3.1. ACTUALES ESTÁNDARES DE DESARROLLO

Toda empresa que cuente con sistemas automatizados, requiere de la definición y utilización de estándares; los estándares son las reglas para el manejo de utilerías o de sintáxis, las cuales se deben tomar para la elaboración de trabajos (JCL`s), nombres de archivos, nombres de bibliotecas y definiciones de base de datos, con el objeto de ordenar y controlar los elementos relacionados de cada sistema.

La definición de estándares se reliaza en base a la estructura y necesidades propias de cada empresa.

3.1.1. ESTÁNDARES DE NOMENCLATURA

3.1.1.1. ESTRUCTURA LÓGICA DE BIBLIOTECAS

Como ya se mencionó en el anterior capítulo, los módulos del sistema son: registro, recaudación , traspasos y retiros, y se cuenta con distintos ambientes: test, validación, aceptación y produccion, cada aplicación cuenta con sus propia estructura de bibliotecas, la cual se repite en cada uno de los ambientes.

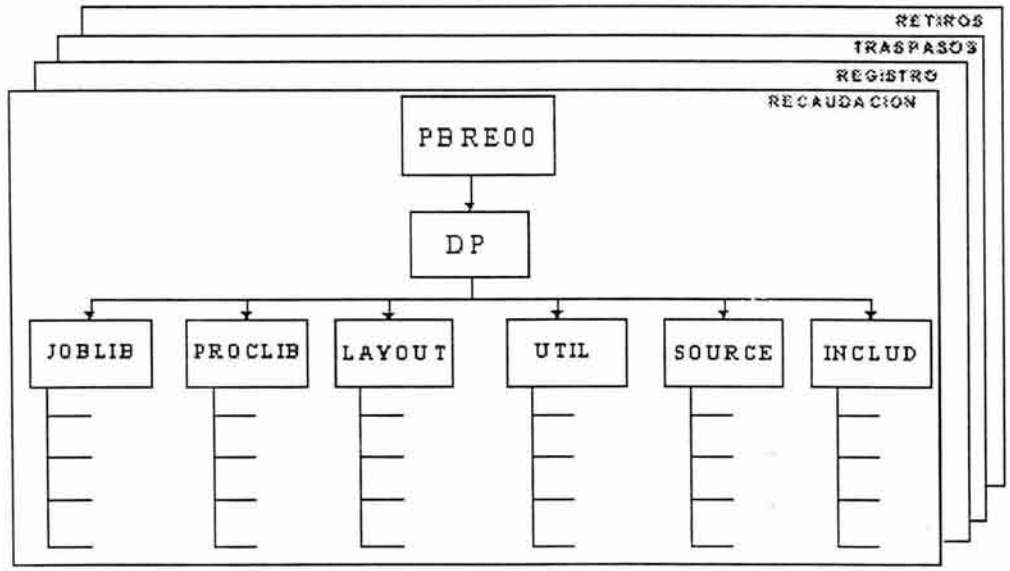


Figura 3-1. Organización de bibliotecas productivas.

En la figura 3-1 se puede observar que la estructura lógica de las bibliotecas productivas es la misma en todas las aplicaciones. Cada biblioteca almacena distintos tipo de elementos. A continuación se proporciona una breve descripción del tipo de datos que almacena cada una de ellas:

- PBRE00.DP.JOBLIB. Contiene los disparadores de procesos (también llamados JCL's por su nombre en inglès Job Control Languaje, o lenguaje de control de trabajos)

- PBRE00:DP.PROCLIB. Contiene los procesos (también llamados PROC's), que son invocados por los JCL`s.

- PBRE00.DP.UTIL. Contiene la tarjetas de ejecución (también llamadas tarjetas de RUN).

- PBRE00.DP.LAYOUT. Contiene los archivos requeridos para la definición de tablas, parámetros y criterios del sistema.

- PBRE00.DP.SOURCE. En esta biblioteca se almacenan todos lo programas fuente.

- PBRE00.DP.LOAD. Aquí se almacenan los programas objeto o ejecutables.

- PBRE00.DP.INCLUD. Esta biblioteca contiene los elementos que definen las variables del sistema.

En la figura 3-2 se muestra el caso particular de ProceSar en donde existen 4 diferentes aplicaciones productivas: Registro, Retiros, Traspasos y Recaudación.

Se puede observar que todos los elementos del sistema se encuentran diferenciados entre si de acuerdo a la aplicación a la que pertencen, asi como al ambiente en que se encuentran y al tipo que pertenezcan; esto es gracias a la nomeclatura que se utiliza para ubicarlos dentro del sistema.

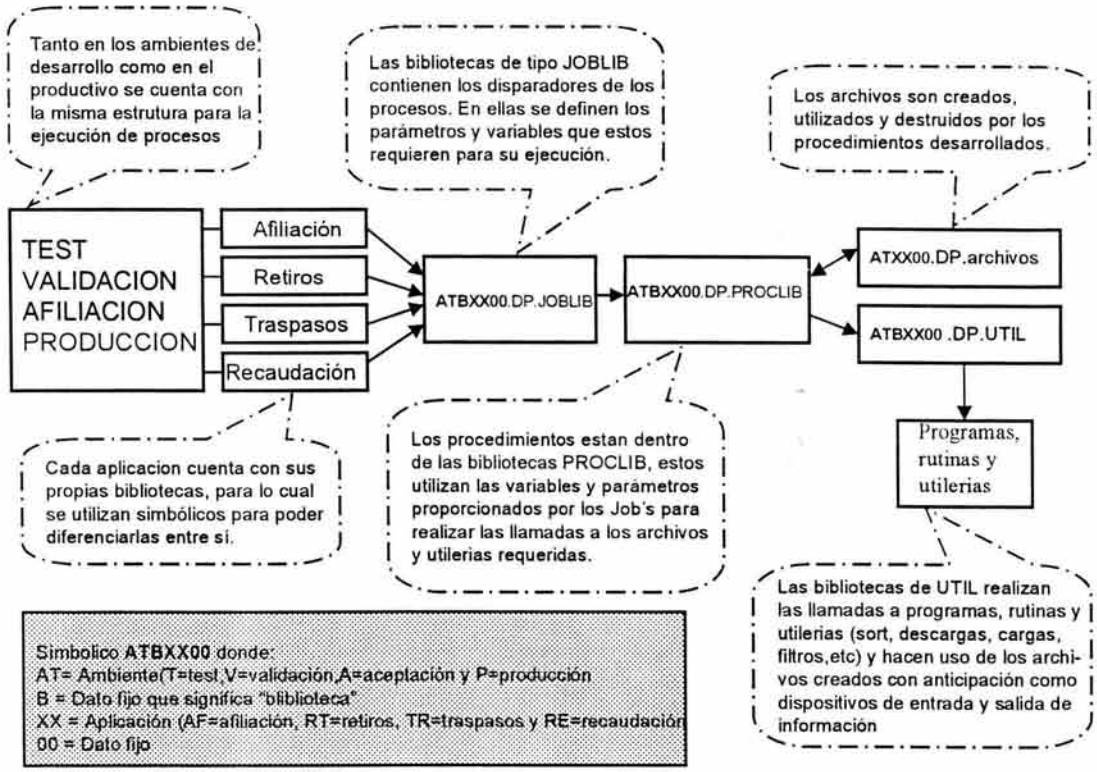


Figura 3-2. Estructura de ejecución de procesos.

Los directorios o carpetas tienen la siguiente nomenclatura:

ABMMNN.DP.TTTTTT

En donde: **A=** Ambiente (T=test, V=validación, A=aceptación, P=producción)

B= Indica que se trata de una biblioteca

MM= Módulo Aplicativo (AF=registro, RT=retiros, TC=traspasos, RE=recaudación)

NN= Subsistema (00)

.

D= Dispositivo de residencia

D= Disco

T= Cinta

P= Permanencia

P= Permanente

T= Temporal

TTTTTT= Nombre del directorio o carpeta (JOBLIB, PROCLIB, INCLUD, UTIL, SOURCE y LOAS)

3.1.1.2. NOMBRES DE TRABAJOS (JOB'S) Y PROCEDIMIENTOS(PROC'S).

Los nombres de los trabajos(job's) y procedimientos (proc's) deben de ser el mismo, con longitud de 8 caracteres y se asignan tomando en cuenta al ambiente de trabajo y aplicación a la que pertenecen además de indicar su periodicidad, es decir, si serán ejecutados, diariamente; semanalmente, mensualmente o eventualmente. Su nomeclatura es:

AMMTPSEC Donde:

A = Ambiente (T=test, V=validación, A=aceptación, P=producción)

MM = Módulo o Aplicación (AF=registro, RT=retiros, TC=traspasos, RE=recaudación)

TP = Tipo de proceso (PD=diario, PS=semanal, PM=Mensual, PE=eventual PB= bimestral)

SEC = *Secuencia (000 – ZZZ)*

El ejemplo de la figura 3-3 muestra el nombre de un elemento:

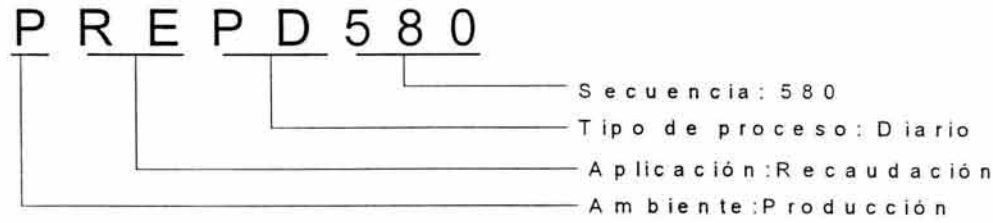


Figura 3-3. Ejemplo del nombre de un elemento.

Como ya se mencionó, el nombre del los procesos (PROC`s) deben de coincidir con el trabajo (JOB) que lo ejecuta.

3.1.1.3. NOMBRE DE PROGRAMAS

Este es el elemento que permite identificar un programa. Dicho nombre debe de ser de 6 posiciones como a continuación se muestra:

PATNNN

En donde:

P= La letra P fija de ProceSar

A= Aplicación a la que pertenece (E=recaudación, T=Retiros, C= Traspasos, R=Registro).

T= So corresponde a programa será P, si es rutina R

NNN= Consecutivo (001 – ZZZ)

3.1.1.4. NOMBRE DE ARCHIVOS Y CARPETAS

Para asignar nombre a los archivos de datos secuenciales se utiliza la siguiente nomeclatura:

AMMNN.DP.TTTTTT.IIIIII.GGGGGG

En Donde: **A=** Ambiente (T=test, V=validación, A=aceptación, P=producción)

MM = Aplicación (AF=registro, RT=retiros, TC=traspasos, RE=recaudación)

NN = Origen o destino

00 Generado en ProceSar

FT Recibido o para ser eviado po File Transfer.

D = Dispositivo de residenecia

D= Disco

T= Cinta

P = Permanencia

P= Permanente

T= Temporal

.

TTTTTT= Identificador 1 de formato libre

IIIIII= Identificador 2 de formato libre

.

G00V9999= Opcional (archivos GDG's)

Cuando el archivo se use como grupo de generación.

En la figura 3-4 se muestra un ejemplo de nombre de archivo.

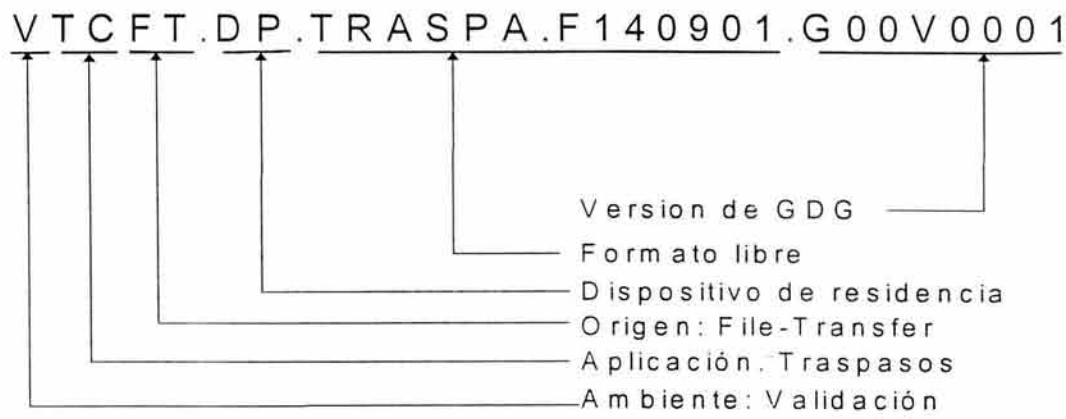


Figura 3-4. Ejemplo del nombre de un archivo

3.1.2. ESTÁNDARES PARA LA CREACIÓN DE JOB'S Y PROC'S

3.1.2.1. CREACION DE JOB'S (DISPARADORES)

En este punto vale la pena aclarar que el objetivo de este trabajo de tesis no es el de explicar el funcionamiento de los comandos e intrucciones propios del sistema operativo o procesos batch, por lo cual estos sólo se mencionarán. La figura 3-5 muestra la estructura de lo que los diparadores deben contener, y a su vez se destacan los pricipales componentes que son:

1. Area de postulados iniciales. En ella se raliza la declaración de las siguientes variables de sistema:

CLASS. La clase de ejecución, la cual tendrá los siguientes valores posibles dependiendo el ambiente en que se ejecuten: "P" (producción), "Q" (aceptación) y V (validación).

MSGCLASS. Los mensajes del proceso deberán estar en clase X.

MSGLEVEL. El nivel de mensajes debe ser completo, es decir, las tarjetas y mensajes de JCL así como los de JES2 y del operador deben quedar en el LOG o en la ejecución del proceso, para obtener esta información deberá codificarse de la siguiente manera: MSGLEVEL=(1,1).

RESTART. En algunas ocasiones se pueden presentar cancelaciones de los procesos por distintas causas, dejando inconcluso dicho proceso, por lo tanto, cuando el error es corregido el proceso reinicia desde el punto en que se haya quedado pendiente.

NOTIFY. La notificación del inicio o terminación del job deberá ser hacia el usuario del operador.

REGION. La memoria que se le asignará al job para su ejecución será de 0M para producción y de 8M para los demas ambientes de desarrollo.

TIME. Se le asignará tiempo máximo de CPU (TIME=1440)

USER. Se asignará usuario permitido para ejecución según el ambiente, (S000PROD para producción)

JCLLIB ORDER=(PBRE00.DP.PROCLIB). Solo deberá invocar a la biblioteca oficial de producción de donde tomará el procedimiento del proceso

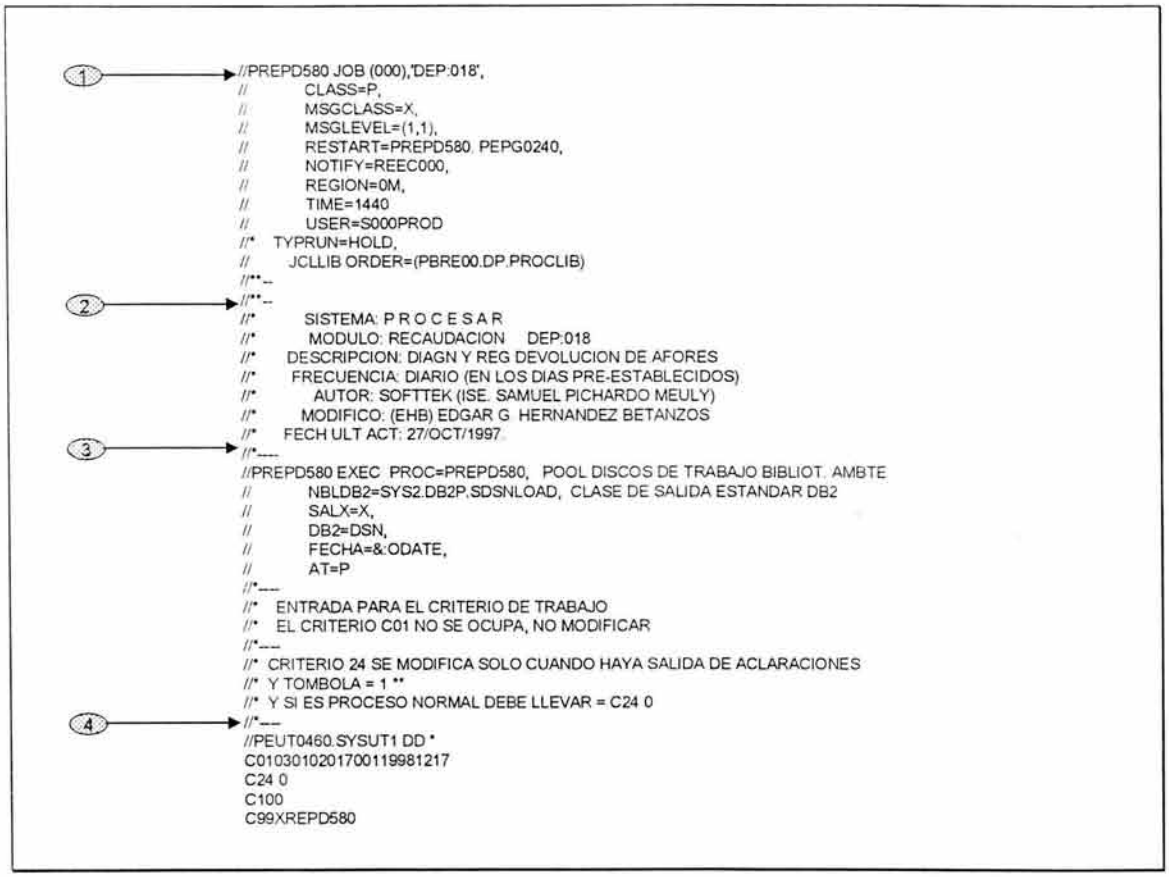


Figura 3-5. Ejemplo de un disparador.

2. Área de comentarios. Aquí se proporciona información general del proceso en cuestión.

3. Área de parámetros simbólicos. En esta parte se definen todas las variables que el procedimiento a ejecutar, llamados comunmente sólo simbólicos, requiere, también puden ir acompañadas de un breve comentario de su función.

Los parámetros simbólicos oficiales son los siguientes:

NBLDB2= Biblioteca de módulos carga de DB2

SYS2.DB2P.SDSNLOAD (Producción)
SYS2.DB2A.SDSNLOAD (Aceptación)
SYS2.DB2T.SDSNLOAD (Validación)

AT= Ambiente de trabajo

P	(Producción)
A	(Aceptación)
V	(Validación)

DB2= Ambiente de trabajo de DB2

DSN	(Producción)
DSNA	(Aceptación)
DSNT	(Desarrollo)

Estos parámetros son los que se usan para definir el ambiente, pueden existir otros parámetros propios de cada aplicación para su operación, estos parámetros serán definidos por el área de desarrollo de sistemas.

4. Área tarjetas de override y de criterios. En esta área se escriben criterios o condiciones especificas de ejecuciòn, que se sustituiran en el procedimiento, siempre y cuando este proceso los requiera, de no ser asi, esta área se omitirá. En ambos casos es necesario que se proporcione una breve explicación del propósito que estos tienen.

3.1.2.2. CREACIÓN DE PROC'S (PROCEDIMIENTOS)

A cada JOB o disparador le corresponde un PROC o procedimiento, el cual contine las instrucciones batch requeridas para llevar a cabo una tarea en particular. Dichos procedimientos deben cumplir con las especificaciones establecidas para su creación.

En la figura 3-6 se muestra un ejemplo de un procedimiento.

Figura 3- 6. Ejemplo de un procedimiento.

En el ejemplo anterior se puden identificar los siguientes elementos:

1. Identificador del procedimiento. En la primer linea se coloca el identificador del procedimiento seguido de un área de comentarios en la que se realiza la identificación del proceso.

2. Comentarios. Los procesos batch se encuentran conformados por una serie de pasos, los cuales deben estar precedidos por un breve explicación de su función.

3. Nombre del paso dentro del PROC. Cada uno de los pasos deben estar plenamente identificados por una etiqueta o nombre del paso. La nomeclatura requerida para proporcionar nombre de etiqueta a los pasos de un procedimiento es:

PATTNNNN Donde:

P = La letra P de Paso

A =Aplicacion a la que pertenece el procedimiento (E=recaudación, T=Retiros, C= Traspasos, R=Registro).

TT =Tipo de programa ejecutado en el paso

PG= Programa

SO= Ordenmiento (SORT)

UT= Utileria

LO= Carga

UL= Descarga

NNNN =Número consecutivo del paso en orden Descendente (0000 – 9999)

4. Condición de ejecución. Este condiciona la ejecución del paso en cuestión siempre y cuando el anterior haya terminado con un código de retorno satisfactorio.

5. Sustitución de variables. Se debe hacer uso de las variables de ambiente para que el procedimiento pueda ser ejecutado desde cualquier ambiente con solo indicarle en el disparador el valor de dichas variables.

6. Direccionamiento de mensajes de salida. La salida de mensajes predeterminada será la estandar.

7. Etiquetas de archivos de entrada y/o salida. Todos los archivos de entrada y/o salida contarán con una etiqueta de identificación y esta será de acuerdo a como la solicite el programa o utilería en cuestión.

8. Parámetros de definición de archivos. Estos definen el tratamiento que se le va a dar al archivo. En la figura 3-7 se muestran los componentes que debe contener la definición de los archivos.

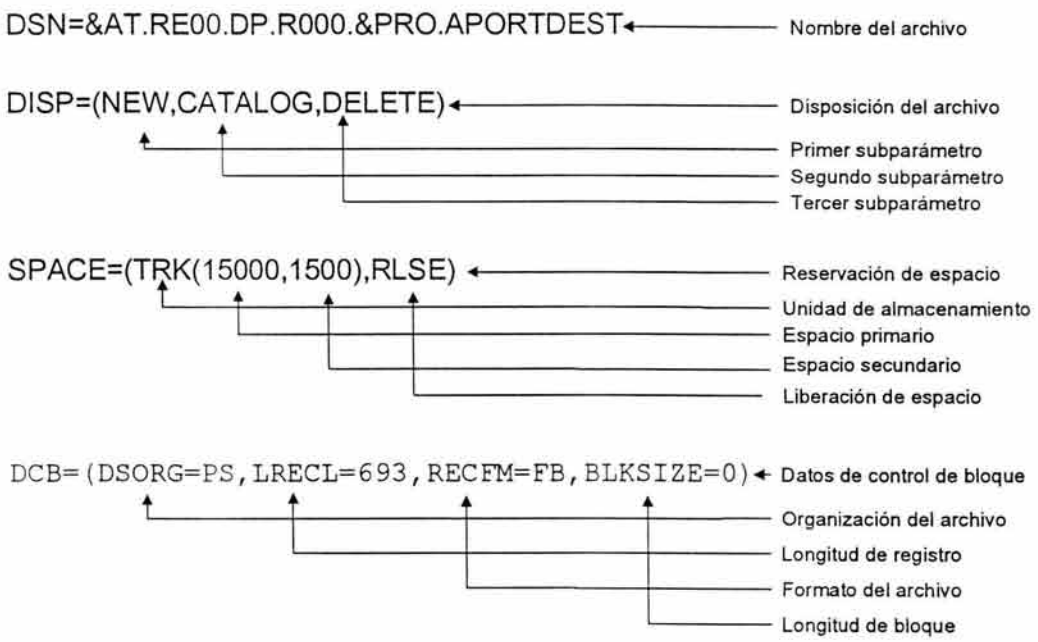


Figura 3-7. Elementos requeridos para la definición de un archivo.

A continuación se proporciona una breve explicación de cada uno de ellos.

Disposición (Disp). Indica al sistema si el archivo va ser creado o ya existía antes del proceso (primer subparámetro) y especifica lo que el desarrollador

quiere que ocurra con el archivo al terminar el proceso (segundo y tercer subparámetros).

La forma en la que se debe usar el parámetro DISP es en base a la lógica de utilización del archivo dentro del JCL; a continuación se muestran las diferentes combinaciones:

		,DELETE		
	NEW	,KEEP	,DELETE	
DISP=(	OLD	,PASS	,KEEP	)
	SHR	,CATLG	,CATLG	
	MOD	,UNCATLG	,UNCATLG	

Primer subparámetro:

NEW. Cuando el archivo va a ser creado.

OLD. Cuando el archivo ya existe, no se desea compartir ni se le van a agregar registros después del último grabado.

SHR. Cuando el archivo ya existe, se desea compartirlo con otro u otros programas que lo requieran (sólo de lectura).

MOD. Cuando el archivo ya existe y se le van a agregar registros a continuación del último grabado.

Segundo subparámetro

DELETE. Para que el archivo sea borrado al terminar el paso de trabajo.

KEEP. Para que el archivo sea conservado al terminar el paso de trabajo.

PASS. Para pasar el archivo a un paso de trabajo posterior.

CATLG. Para que el archivo sea catalogado.

UNCATLG. Para que el archivo sea quitado del catálogo al terminar el paso de trabajo.

Tercer subparámetro

Informa al sistema lo que se desea hacer con el archivo si el paso de trabajo termina anormalmente, con excepción de la opción PASS que no esta permitida como tercer subparámetro, las demás son las mismas y con el mismo significado que en el segundo subparámetro.

El parámetro DISP puede ser omitido cuando el archivo va a ser creado y dado de baja en el mismo paso de JOB, esta omisión equivale a la siguiente codificación:

DISP=(NEW,DELETE,DELETE)

Espacio (Space). Se utiliza para definir el espacio en disco de archivo que será creado.Para calcular el espacio correcto de un archivo, se deberá usar la siguiente formula:

Número de Registros * Longitud del Registro = Total de Bytes a Ocupar

Total de Bytes a Ocupar / 55000 (número de bytes por track) = ESPACIO PRIMARIO RECOMENDADO

Para calcular el espacio secundario se usará la siguiente formula:

ESPACIO SECUNDARIO = 10% de ESPACIO PRIMARIO RECOMENDADO

El resultado de la fórmula se usará para definir en el procedimiento el espacio real para el archivo.

SPACE=(TRK,(espacio primario,espacio secundario)

La optimización del espacio utilizado es muy importate, para ello contamos con un subparámetro que sirve para liberar el espacio no utilizado en la generación de un archivo, el cual se utiliza con el parámetro de SPACE de la siguiente forma:

SPACE=(TRK,(100,10),RLSE)

De esta forma, aseguramos que todo el espacio que no fué utilizado en la generación de un archivo será liberado.

DCB (Data Control Block). Este parámetro se usa para definir el formato, y las longitudes del registro y de los bloques del archivo. La información se proporciona atravez de los siguientes subparámetros:

RECFM. Especifica el formato y características de los registros de un archivo nuevo

LRECL. Especifica la longitud de los registros del archivo

BLKSIZE. Especifica la longitud máxima en bytes de un bloque, si no se especifica, el sistema calcula un óptimo basado en el tamaño del registro.

DSORG. Especifica la organización del archivo, comúnmente PS Physical Sequential.

Como estandar para la utilización del subparámetro BLKSIZE, no se usará ningún valor, es decir, se codificará "BLKSIZE=0" para que se calcule automáticamente basado en el tamaño de registro.

3.1.3. ESTANDARES EN BASE DE DATOS

Al igual que cualquier otro elemento dentro del sistema, las bases de datos se identifican de acuerdo a la aplicación y ambiente al que pertenezca. En la figura

3-8 se puede observar que tanto en los ambientes de desarrollo como en el productivo se cuenta con la misma configuración de base de datos, la diferencia está en la estrutura que define e identifica cada una de estas configuraciones. Cada estructura tiene asignado un alias. Los alias existentes son:

1. TDBSAR01.ISTSAR. Ambiente de test.

2. VDBSAR01.ISVSAR . Ambiente de validación.

3. ADBSAR01.ISASAR. Ambiente de aceptación.

4. PDBSAR01.ISPSAR. Ambiente de producción.

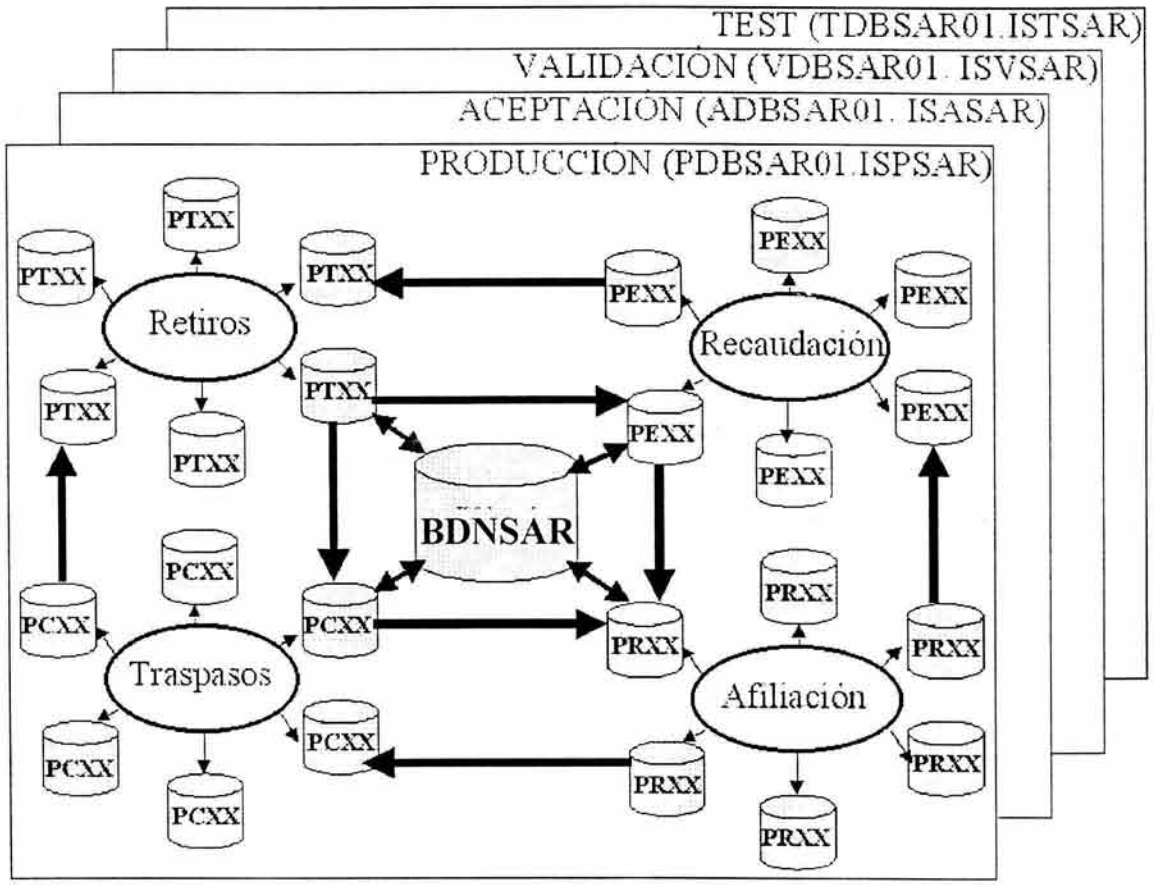


Figura 3-8. Estándares en base de datos.

Adicionalmente al prefijo de ambiente se tiene una nomeclatura predefinida para nombrar a las tablas o entidades propias de cada aplicación, esta nomeclatura es la siguiente:

XXYY Donde:

XX = Aplicación (PT=retiros, PE=recaudación, PR=registro, PC=traspasos)

YY = Caracteres de identificación de la entidad. Estos deben de ser únicos y se pueden hacer combinaciones entre números y letras.

En la figura 3-9 se muestra un ejemplo del nombre de una entidad en el ambiente productivo, para la aplicación de retiros y cuyo objetivo es el de llevar el control de lotes recibidos.

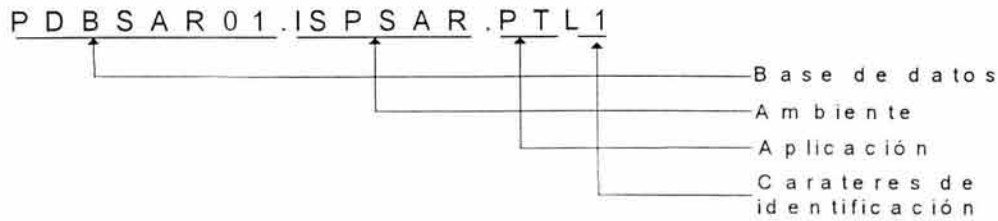


Figura 3-9. Ejemplo del nombre de una entidad.

3.2. NUEVOS ESTÁNDARES DE DESARROLLO

Debido al cambio de plataforma tecnológica es necesario desarollar nuevos estandares de calidad para la creación de shell's los cuales sustituirán en su totalidad a los antiguos procesos batch, ya sean JCL`s o PROC`s. Es importante mencionar que para este proceso de conversión de job's a shell's es de primordial importancia buscar que el impacto operativo y productivo sea el menor posible, es por ello que en la medida de lo posible se procuran conservar las mismas politicas de desrrollo.

3.2.1. ESTÁNDARES DE NOMENCLATURA

3.2.1.1. ESTRUCTURA LÓGICA DE BIBLIOTECAS

Como ya se mencionó en el capítulo 1, el equipo SUN E-10000 tiene la característica de poder proporcionar distintos dominios en un mismo equipo, es por ello que el área de soporte aplicativo con apoyo del área de desarrollo planteó utilizar un dominio exclusivo para el área productiva y utilizar un

segundo dominio para el área de desarrollo. Esta distribución es muy conveniente desde el punto de vista de mantenimiento y versatilidad en los procesos, ya que con esto se asegura que estos serán ambientes totalmente independientes. La estructura de las bibliotecas de trabajo y de producción con la que trabajaremos es la siguiente:

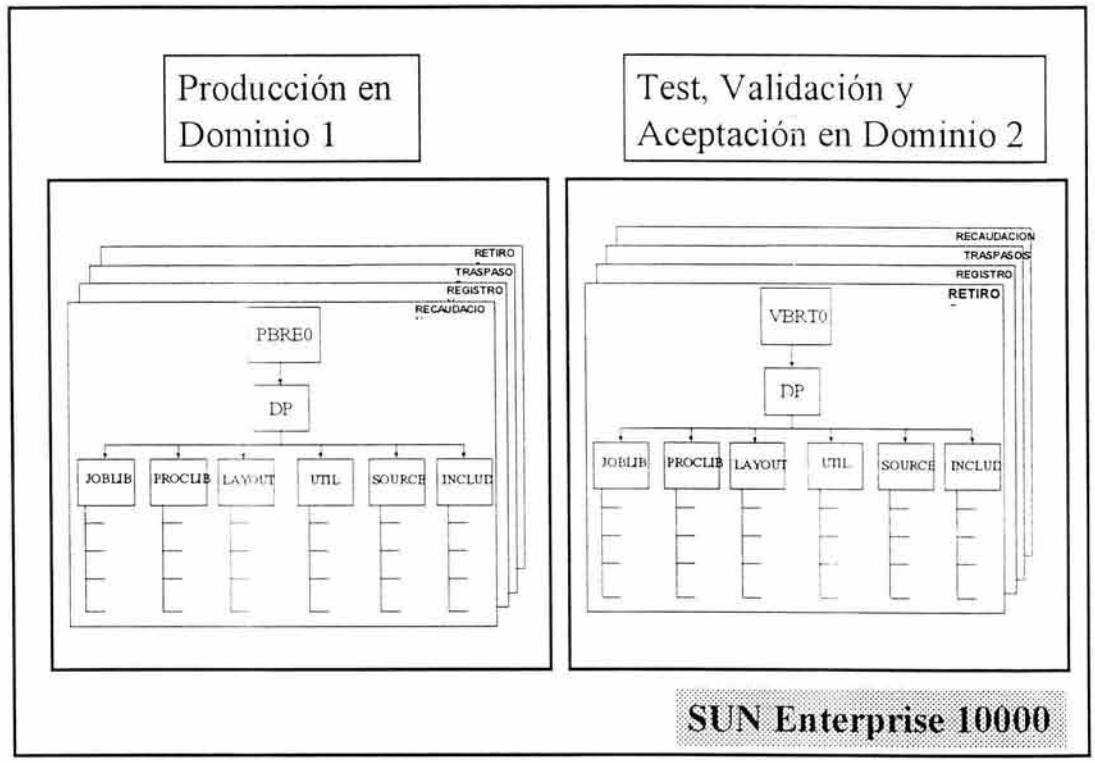


Figura 3- 10.Distribución de ambientes de trabajo en la E-10000.

3.2.1.2. NOMBRES DE TRABAJOS (SHELL'S), PROGRAMAS Y CARPETAS

En la figura 3.10 se puede observar claramente que la estructura lógica de las biblitecas productivas y de desarrollo sigue siendo la misma solo que ahora se encuentran alojadas en diferentes dominios. La metodología para asignar nombres de bibliotecas y elementos tales como programas, tarjetas de

ejecución y procedimientos se seguirá aplicando, solo que en vez de utilizar el punto (.) para realizar la jerarquización del árbol de módulos, se utilizará la diagonal (/). A continuación se proporcionan algunos ejemplos de los actuales nombres contra los nuevos:

NOMBRE ACTUAL		NUEVO NOMBRE
PBRE00.DP.PROCLIB	Cambia por:	/PBRE00/DP/JOBLIB
PBRE00.DP.LAYOUT	Cambia por:	/PBRE00/DP/LAYOUT
VBRT00.DP.SOURCE	Cambia por:	/VBRT00/DP/SOURCE
VBAF00.DP.JOBLIB.VAFPD070	Cambia por:	/VBAF00/DP/JOBLIB/VAFPD070
VTC00.DT.DA45.SOACEPDI	Cambia por:	/VTC00/DT/DA45/SOACEPDI

Tabla 3-1. Ejemplos de los cambios de nombre de archivos

Es importante aclarar que aun cuando en UNIX es posible asignar nombre de bibliotecas y/o elementos mayor a 8 caracteres, por convenecia se seguira respetando esta restricción, ya que el objetivo principal de todo este proceso de cambio de plataforma técnológica es el de procurar obtener el menor impacto posible.

3.2.1.3. ARCHIVOS SECUENCIALES.

El estándar para asignar nombre a los archivos secuenciales es el siguiente:

AMMNN/DP/TTTTTT.IIIIII.G0009999

Fddmmaa.Hhhmmss

En Donde: **A=** Ambiente (T=test, V=validación, A=aceptación, P=producción)

MM = Aplicación (E=recaudación, T=Retiros, C= Traspasos, R=Registro).

NN = Origen o destino

01 Generado en ProceSar

FT Recibido o para ser eviado po File Transfer

/

D = Dispositivo de residencia

D= Disco

T= Cinta

P = Permanencia

P= Permanente

T= Temporal

/

TTTTTT= Formato Libre

.

IIIIII= Formato Libre

G000999= Opcional (GDG's) cuando se use como grupo de generación.

Fddmmaa Fecha en formsto ddmmaa

Hhhmmss Hora en formato hhmmss

Cuando el archivo se use como archivo de recepción o sea necesaria la hora de arribo

3.2.2. ESTÁNDARES PARA LA CODIFICACIÓN DE SCRIPTS

3.2.2.1. CREACIÓN DE UN JOB (SCRIPT) DISPARADOR

De los diferentes tipos de shells para el sistema UNIX, utilizaremos el Korn Shell (ksh). La primer línea de un shell script, determina con qué tipo de shell se ejecutarán los programas, y en este caso, siempre será la misma:

```
#!/bin/ksh
```

Tabla 3-2. Primera línea de un JOB.

Cualquier otra línea, en donde el primer caracter sea # , será un comentario para la documentación interna del script

En la siguiente tabla se exporta la variable JID la cual debe de ser el nombre y número del JOB asignado por el sistema :

```
export JID=PREPD100.${$}
```

Tabla 3-3. Declaración de la variable JID.

Las siguientes líneas realizan la identificación de JOB :

```
# IDENTIFICACION DEL JOB
echo "** JOB=PREPD100"
echo "** NUMERO=$$"

echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
```

Tabla 3-4. Identificación de un JOB.

En las siguientes líneas se indicará la ruta del PROC a ejecutar (exportando la variable PROCLIB):

```
# PATH DE LOS PROCS A EJECUTAR

  export PROCLIB=/PBRE00/DP/PROCLIB
```

Tabla 3-5. Declaración de la variable PROCLIB.

En las siguientes líneas se hace la descripción del JOB :

```
#//*----
#//* SISTEMA: P R O C E S A R
#//* MODULO: RECAUDACION     DEP:004
#//* DESCRIPCION: RECHAZO DE F-TRANSFER
#//* DEVOLUCION AFORES
#//* FRECUENCIA: DIARIO (EN LOS DIAS #//* ESTABLECIDOS)
#//* AUTOR:  OCTAVIO GUTIERREZ
#//* MODIFICO:(GUSTAVO ORDAZ)
#//* FECH ULT ACT: 25/ENE/1998.
#//*----
```

Tabla 3-6. Descripción del JOB.

A continuación se escribe el código en el que se define el proceso a ejecutar. La variable EJECUTA le es asignado el nombre del PROC a ejecutar.

```
# IDENTIFICACION DEL PASO
echo "** PASO=PREPD100"
echo "** EJECUTA=PREPD100"
echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
                                        #
export PASO=PREPD100; export EJECUTA=PREPD100
#
```

Tabla 3-7. Declaración de la variable EJECUTA.

3.2.2.1.1. PARÁMETROS

En caso de que el JOB envíe parámetros al PROC se deberá utilizar el siguiente código:

```
# EXPORTAR LOS PARÁMETROS RECIBIDOS PARA EL PASO
     param=1
     while [ $param -le $# ]
     do
       i=`eval echo $"{$param}"`
       export "$i"
       param=`expr $param + 1`
     done
#
```

Tabla 3-8. Código para la lectura y asignación de parámetros.

3.2.2.1.2. CRITERIOS

En caso de requerir de criterios de ejecución se hará lo siguiente:

```
echo80 "$PEUT0100_SYSUT1" >> ${TMP}/${JID}.PEUT0100_SYSUT1
     export PEUT0100_SYSUT1="${TMP}/${JID}.PEUT0100_SYSUT1"

#
```

Tabla 3-9. Código para la captura de criterios de ejecución.

Donde PEUT0100 es el paso donde se copia el criterio a un archivo de trabajo, y el número después de la orden echo, será el total de caracteres del criterio.

3.2.2.1.3. EJECUCIÓN

Por último se pone el comando "time" para que mediante la sustitución de variable se realice la ejecución del el PROC, en la terminación del PROC se actualiza la variable RC y regrese el código de retorno :

```
Param
time ${PROCLIB}/${EJECUTA}
    rc=$?
    if[$rc –gt $RC]
    then
      RC=$rc
    `fi
rm ${TMP}/${JID}*
echo "** FIN=\"`date'+%d/%m/%G %T'`"\"
echo "** RETURN-CODE=$rc"

return $RC #
```

Tabla 3-10. Código final de un JOB.

El JOB se ejecutará desde la línea de comando de UNIX de la siguiente manera:

```
PREPD100 AT=/P RC=0 REINICIO=0 CENT=000
PEUT0100_SYSUT1="C99PREPD100 C0612"
```

Tabla 3-11. Ejemplo de ejecución de un proceso desde la línea de comandos.

Donde AT, RC, REINICIO y CENT son los parámetros que se le están enviando al PROC.

Las variables RC y REINICIO deben de ir forzosamente, siempre y cuando el PROC tenga más de un paso.

Para ejecutar un JOB con algún reinicio se hace como sigue (en una sola línea):

```
PREPD100 AT=/P RC=99 REINICIO=PREPD100_PEUT0100 CENT=000
PEUT0100_SYSUT1="C99PREPD100 C0612"
```

Tabla 3-12. Ejemplo de reinicio de un proceso.

Donde RC debe de ser diferente de cero (por lo que se utilizará el 99 para que no se confunda con otro código de retorno) y REINICIO debe de llevar el nombre del PROC y la etiquta del paso en donde reiniciará.

3.2.2.2. CREACIÓN DE UN PROC

Para elaborar un PROC lo primero que se hace es la identificación del PROC y la declaración de las variables y/o constantes a utilizar en los siguientes pasos:

```
# IDENTIFICACION DEL PROCEDIMIENTO
  echo "** PROC=PREPD100"
#
# VARIABLE QUE SERVIRA PARA LOS PASOS SUBSECUENTES
  export PROC=PREPD100
#
# PARAMETROS DEL PROCEDIMIENTO
  echo "export SECC2=${AT}RE00.DT.R${CENT}"
  export SECC2='${AT}RE00.DT.R${CENT}'
  echo "export PRO=D100"
  export PRO='D100'
```

Tabla 3-13. Identificación de procedimiento.

Posteriormente se escribe los datos generales del PROC:

```
#//*   SISTEMA: P R O C E S A R
#//*   MODULO: RECAUDACION      DEP:004
#//*   DESCRIPCION: RECHAZO DE FILE-TRANSFER
#//*   DEVOLUCION Y CONFIRMACION AFORES
#//*   FRECUENCIA: DIARIO
#//*   AUTOR: (LAE. RAFAEL ARROYO C.)
#//*   MODIF: (MVS1 ARACELI VEGA S.)
#//*   FECH ULT ACT: 14/ENERO/1999.
```

Tabla 3-13. Datos generales del PROC.

3.2.2.2.1. CODIFICACIÓN DE PASOS

Para todos los pasos se debe iniciar con el siguiente código:

```
#//*----
#//*  DESCRIPCION DEL PASO
#//*----
# IDENTIFICACION DEL PASO
echo "** PASO=PEUT0130"                          <-- Numero de paso
echo "** EJECUTA=PROGRAMA"                       <-- Programa o
                                                  comando a ejecutar
echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
#
# VARIABLES QUE SERVIRAN PARA LOS PROGRAMAS IMPLICADOS
SUBSECUENTES

export PASO=PEUT0130; export EJECUTA=PROGRAMA
```

Tabla 3-14. Código inicial para la programación de un paso.

3.2.2.2.2.EJECUCIÓN DE PASOS

El siguiente código es para que se ejecute, o no, un paso, en caso de que el anterior haya terminado en error; y para llevar el control de reinicio del proceso.

```
#
if [ "$RC" -eq 0 ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
        Sentencias (Programas, Borrado de archivos, copia de archivos,
        Cargas y descargas de y a base de datos, etc...)
        rc=$?
        if [ $rc –gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
        then
             RC=$rc
        fi
        unset (variables que se definieron al inicio del paso)
fi
echo "** FIN=\"`date '+%d/%m/%G %T'`"\"

echo "** RETURN-CODE=$rc"
```

Tabla 3-15. Código para el control del flujo de ejecución.

3.2.2.2.3.PASOS DE BORRADO DE ARCHIVOS

Al inicio y al final de cada proceso se generará una tarjeta de borrado de archivos de trabajo, mediante un programa especial que se encuentra en cada una de los módulos productivos. Esto es llevado a cabo en tres pasos tal como se describe en el siguiente ejemplo:

1) Se genera un archivo temporal que contiene los parámetros que el sistema necesita para identificar en la base de datos los archivos que serán borrados de acuerdo al proceso que se ejecutará.

```
# PARAMETROS DEL PASO
echo "      export PARM='PREPD580,1,${AT}RE00,${PRO}'"
export PARM="PREPD580,1,${AT}RE00,${PRO}"
#
echo "FILEOUT=\"${TMP}/${JID}.TMP\""
export FILEOUT="${TMP}/${JID}.TMP"
echo "${EJECUTA}"
time ${EJECUTA}
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
    RC=$rc
fi
unset PARM

unset FILEOUT
```

Tabla 3-16. Código del primer paso para el borrado de archivos.

2) Con los parámetros descritos en el paso anterior, lo que continúa es la ejecución del programa que genera la tarjeta que contendrá la lista de archivos que serán borrados.

```
cp /dev/null ${AT}BRE00/DP/UTIL/PEBD5801
echo "DD_ZGR901A1=\"${AT}BRE00/DP/UTIL/PEBD5801\""
export DD_ZGR901A1="${AT}BRE00/DP/UTIL/PEBD5801"
echo "DD_SYSIN=\"${TMP}/${JID}.TMP\""
export DD_SYSIN="${TMP}/${JID}.TMP"
echo "${EJECUTA} `cat $DD_SYSIN`"
time ${EJECUTA} `cat $DD_SYSIN`
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
     RC=$rc
fi
If [ $rc -eq 0 ]
then
  echo "DIVIDE_080 $TMP/${JID} ${AT}BRE00/DP/UTIL/PEBD5801"
  DIVIDE_080 $TMP/$JID ${AT}BRE00/DP/UTIL/PEBD5801
  rc=$?
  if [ $rc –gt $RC ]
  then
     RC=$rc
  fi

fi
```

Tabla 3-17. Código del segundo paso para el borrado de archivos.

3) Por último se efectua el borrado de los archivos.

```
echo "SYSIN=\"${AT}BRE00/DP/UTIL/PEBD5801\""
export SYSIN="${AT}BRE00/DP/UTIL/PEBD5801"

time $SYSIN
```

Tabla 3-18. Código del tercer paso para el borrado de archivos.

3.2.2.2.4. CRITERIOS PARA ARCHIVOS DE TRABAJO

El siguiente código introduce a un archivo de trabajo los criterios de ejecución proporcionados desde el JOB.

```
echo "SYSUT1=\"${PEUT0100_SYSUT1}\""
export SYSUT1="${PEUT0100_SYSUT1}"
echo
"SYSUT2=\"${AT}RE00/DT/R${CENT}.T${TEN}${CONS}.F${FECH}.AHR008
A1\""
export
SYSUT2="${AT}RE00/DT/R${CENT}.T${TEN}${CONS}.F${FECH}.AHR008A1
"
cat $SYSUT1 > $SYSUT2
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
   RC=$rc
fi
unset SYSUT1
unset SYSUT2
```

Tabla 3-19. Código para generar archivo de criterios.

3.2.2.2.5. EJECUCIÓN DE UN PROGRAMA

Para la ejecución de un programa se debe realizar lo siguiente:

En la variable EJECUTA se debe indicar el programa a ejecutar.

```
export PASO=PEPG0120;  export EJECUTA=PEP89D
```

Tabla 3-20. Asignación de la variable EJECUTA para la ejecución de un programa

Las líneas que van despues de la condición que controla el flujo de ejecución quedan como sigue:

```
echo "DD_SYSOUT=\"${AT}RE00/DP/P?PXXX.SYSOUT\""
export DD_SYSOUT="${AT}RE00/DP/P?PXXX.SYSOUT"

echo "DD_PEP89DA1=\"< cat
T${TEN}${CONS}/F${FECH}.TOTRERCC.${NBLOT}\""
export DD_PEP89DA1="< cat
T${TEN}${CONS}/F${FECH}.TOTRERCC.${NBLOT}"
echo "DD_PEP89DA2=\"> cat
>${AT}REFT/DP/P${TEN}${CENT}/FRESULT.C001\""
export DD_PEP89DA2="> cat
>${AT}REFT/DP/P${TEN}${CENT}/FRESULT.C001"
echo "DD_SYSIN=\"${TMP}/${JID}.TMP\""
export DD_SYSIN="${TMP}/${JID}.TMP"
echo "${EJECUTA} ${DD_SYSIN}"

time ${EJECUTA}
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
        RC=$rc
fi
unset DD_PEP89DA1
unset DD_PEP89DA2

unset DD_SYSIN
```

Tabla 3-21. Ejemplo de codificación de un paso

3.2.2.2.6. EJECUCIÓN DE UN SORT

Para hacer uso de la utileria de sort se deberá codificar de la siguiente forma :

En la variable EJECUTA se le asigna el valor SORT (escrito con mayúsculas)

```
export PASO=PEPG0120;  export EJECUTA=SORT
```

Tabla 3-22. Asignación de la variable EJECUTA para un SORT.

En la siguiente tabla se muestra un ejemplo del código que debe escribirse posterior a las condiciones que controla el flujo de ejecución:

```
# PARAMETROS DEL PASO
#
export SORTIN_LREC=514
echo "SORTIN=\"${AT}RE00/DT/R000.${PRO}.APOPEND\""
export SORTIN="${AT}RE00/DT/R000.${PRO}.APOPEND"
echo "VAR_TEMP=\"${AT}RE00/DP/R000.${FROM}.APONUIND\""
VAR_TEMP="${AT}RE00/DP/R000.${FROM}.APONUIND"
echo "SORTIN=\"${SORTIN} ${VAR_TEMP}\""
export SORTIN="${SORTIN} ${VAR_TEMP}"
echo "SORTOUT=\"${AT}RE00/DT/R000.${PRO}.APOPIND\""
export SORTOUT="${AT}RE00/DT/R000.${PRO}.APOPIND"
echo "SYSIN=\"${AT}BRE00/DP/UTIL/PETS134B\""
export SYSIN="${AT}BRE00/DP/UTIL/PETS134B"
echo "${EJECUTA}"
time ${EJECUTA}
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
      RC=$rc
fi
unset SORTIN_LREC
unset SORTIN
unset SORTOUT
unset SYSIN
```

Tabla 3-23. Ejemplo de codificación de un paso de SORT.

SORTIN es el archivo o archivos de entrada.

SORTOUT es el archivo de salida.

SYSIN es la tarjeta en la que se hacen las indicaciones para ejecutar el SORT.

3.2.2.2.7. CARGA Y DESCARGA DE BASE DE DATOS

Para la carga en la variable EJECUTA deberá tener el programa PTLDRIVM

```
export PASO=LIUT0050;  export EJECUTA=PTLDRIVM
```

Tabla 3-24. Asignación de la variable EJECUTA para realizar cargas con utilería PLTDRIVM

El resto del paso queda como a continuación se muestra:

```
# PARAMETROS DEL PASO
echo "      export PARM='EP=UTLGLCTL/${DB2}${REINIT}'"
export PARM="EP='UTLGLCTL/${DB2}${REINIT}"
#
echo "PTILIB=\"SYS3/PLATINUM/T97C1.LOADLIB\""
export PTILIB="SYS3/PLATINUM/T97C1.LOADLIB"
echo "VAR_TEMP=\"SYS2/DB2P/SDSNLOAD\""
VAR_TEMP="SYS2/DB2P/SDSNLOAD"
echo "PTILIB=\"${PTILIB} ${VAR_TEMP}\""
export PTILIB="${PTILIB} ${VAR_TEMP}"
echo "VAR_TEMP=\"SYS2/DB2P/SDSNEXIT\""
VAR_TEMP="SYS2/DB2P/SDSNEXIT"
echo "PTILIB=\"${PTILIB} ${VAR_TEMP}\""
export PTILIB="${PTILIB} ${VAR_TEMP}"
echo "PTIPARM=\"SYS3/PLATINUM/T97C1.PARMLIB\""
export PTIPARM="SYS3/PLATINUM/T97C1.PARMLIB"
echo "PTIXMSG=\"SYS3/PLATINUM/T97C1.XMESSAGE\""
export PTIXMSG="SYS3/PLATINUM/T97C1.XMESSAGE"
echo "SYSUT1=\"${AT}RE00/DT/R000.${PRO}.IT.SYSUT1\""
export SYSUT1="${AT}RE00/DT/R000.${PRO}.IT.SYSUT1"
echo "SYSUT101=\"${AT}RE00/DT/R000.${PRO}.IT.SYSUT101\""
export SYSUT101="${AT}RE00/DT/R000.${PRO}.IT.SYSUT101"
echo "SYSMAP=\"${AT}RE00/DT/R000.${PRO}.IT.SYSMAP\""
export SYSMAP="${AT}RE00/DT/R000.${PRO}.IT.SYSMAP"
echo "SYSERR=\"${AT}RE00/DT/R000.${PRO}.IT.SYSERR\""
export SYSERR="${AT}RE00/DT/R000.${PRO}.IT.SYSERR"
echo "SYSDISC=\"${AT}RE00/DT/R000.${PRO}.IT.SYSDIS\""
export SYSDISC="${AT}RE00/DT/R000.${PRO}.IT.SYSDIS"
echo "SYSREC=\"${AT}RE00/DT/R000.${PRO}.IT.SYSREC\""
export SYSREC="${AT}RE00/DT/R000.${PRO}.IT.SYSREC"
echo "SYSSORT=\"${AT}RE00/DT/R000.${PRO}.IT.SYSSORT\""
export SYSSORT="${AT}RE00/DT/R000.${PRO}.IT.SYSSORT"
echo "SYSULD=\"${AT}RE00/DP/R000.${FROM}.APONOIND\""
export SYSULD="${AT}RE00/DP/R000.${FROM}.APONOIND"
echo "SYSIN=\"${AT}BRE00/DP/UTIL/PEU2LOIT\""
export SYSIN="${AT}BRE00/DP/UTIL/PEU2LOIT"
echo "${EJECUTA}"
time ${EJECUTA}
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then
    RC=$rc
fi
unset PARM
unset PTILIB
unset PTIPARM
unset PTIXMSG
unset SYSUT1
unset SYSUT101
unset SYSMAP
unset SYSERR
unset SYSDIS
unset SYSREC
```

```
unset SYSSORT
unset SYSULD
unset SYSIN
```

Tabla 3-25 Ejemplo de codificación de un paso de carga con utileria PLTDRIVM

Donde :

SYSREC es el archivo de donde se toman los datos a cargar

SYSIN es la tarjeta de descarga en la que se le indica al manejador de la base de datos la infrormación a extraer y el formato correspondiente.

En la descarga la variable EJECUTA al igual que en la CARGA deberá contener el programa PTLDRIVM.

```
export PASO= PCUN0050; export EJECUTA=PTLDRIVM
```

Tabla 3-26. Asignación de la variable EJECUTA para realizar descargas con utileria PLTDRIVM

El resto del paso se escribe como a continuación se indica:

```
# PARAMETROS DEL PASO
echo "      export PARM='EP=UTLGLCTL/${DB2}'"
export PARM="EP='UTLGLCTL/${DB2}"
#
echo "PTILIB=\"SYS3/PLATINUM/T97C1.LOADLIB\""
export PTILIB="SYS3/PLATINUM/T97C1.LOADLIB"
echo "VAR_TEMP=\"SYS2/DB2P/SDSNLOAD\""
VAR_TEMP="SYS2/DB2P/SDSNLOAD"
echo "PTILIB=\"${PTILIB} ${VAR_TEMP}\""
export PTILIB="${PTILIB} ${VAR_TEMP}"
echo "VAR_TEMP=\"SYS2/DB2P/SDSNEXIT\""
VAR_TEMP="SYS2/DB2P/SDSNEXIT"
echo "PTILIB=\"${PTILIB} ${VAR_TEMP}\""
export PTILIB="${PTILIB} ${VAR_TEMP}"
echo "PTIPARM=\"SYS3/PLATINUM/T97C1.PARMLIB\""
export PTIPARM="SYS3/PLATINUM/T97C1.PARMLIB"
echo "PTIXMSG=\"SYS3/PLATINUM/T97C1.XMESSAGE\""
export PTIXMSG="SYS3/PLATINUM/T97C1.XMESSAGE"
echo "SYSCTL01=\"${AT}RE00/DT/R000.${PRO}.IT.SYSCTL\""
export SYSCTL01="${AT}RE00/DT/R000.${PRO}.IT.SYSCTL"
```

```
        echo "SYSREC01=\"${AT}RE00/DT/R000.${PRO}.APOPEND\""
        export SYSREC01="${AT}RE00/DT/R000.${PRO}.APOPEND"
        echo "SYSIN=\"${AT}BRE00/DP/UTIL/PEU0UN80\""
        export SYSIN="${AT}BRE00/DP/UTIL/PEU0UN80"
        echo "${EJECUTA}"
        time ${EJECUTA}
        rc=$?
        if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
        then
            RC=$rc
        fi
        unset PARM
        unset PTILIB
        unset PTIPARM
        unset PTIXMSG
        unset SYSCTL01
        unset SYSREC01
        unset SYSIN
        unset PTIPARM
        unset PTIXMSG
        unset SYSCTL01
        unset SYSREC01
        unset SYSIN
    fi
    echo "** FIN=\"`date '+%d/%m/%G %T'`"\"
    echo "** RETURN-CODE=$rc"
```

Donde :

SYSIN.- Es la tarjeta con los comandos requeridos por el manejador de base de datos para realizar la descarga.

SYSREC01.- Es el archivo donde van a quedar los datos descargados.

Hasta este momento se puede observar que para la codificación de cada uno de los pasos de los procedimientos tienen la misma estructura, lo que varía son los programas que se desean ejecutar y dependiendo de éste son la cantidad de etiquetas de archivos, bibliotecas y variables que deben ser utilizadas. Una breve descripción de cada uno de estos programas llamados utilerías así como la forma de emplearlas se puede encontrar en el anexo 3.

CAPÍTULO 4. ANÁLISIS Y DESARROLLO DEL SISTEMA DE CONVERSIÓN DE APLICACIONES

4.1. CONSIDERACIONES INICIALES

Se pudó observar en el capítulo anterior que los nuevos estándares de desarrollo y generación de procesos batch están diseñados de tal manera que el impacto del cambio sea el menor posible, pero a pesar de este esfuerzo es necesario realizar diversas adecuaciones que nos permitirán utilizar un esquema similar en cuanto a la distribución de bibliotecas, ejecución de programas aplicativos, programas de sistema, herramientas y utilerías. A continuación se mencionan las consideraciones que el sistema convertidor de procesos debe tomar en cuenta para su desarrollo.

4.1.1. ELECCIÓN DEL SHELL DE TRABAJO

Tal como se mencionó con anterioridad el shell con el que trabajará es el Kornshell por lo cual es obligatorio que al inicio de cada proceso disparador se agregue la siguiente línea:

#!/bin/ksh	→	Establece a Kornshell como traductor de instrucciones.

Tabla 4-1 Línea para la elección del shell.

Con esto se asegura que la ejecución de procesos se realizará bajo el estándar de programación de Kornshell.

4.1.2. USO DE VARIABLE PARA LA CREACIÓN DE ARCHIVOS TEMPORALES

Una de las grandes diferencias con las que nos enfrentamos es el manejo de las salidas o mejor conocido como "spool", ya que en MVS esto es controlado por el sistema operativo, pero en UNIX se tendrá que implementar un

mecanismo que nos permita identificar cada uno de los procesos ejecutados en cualquier ambiente ya se productivo, de pruebas o de desarrollo, es por ello que también es obligatorio que los procesos disparadores contengan las siguientes líneas:

 # VARIABLE QUE SERVIRA PARA CREACION DE ARCHIVOS TEMPORALES
export JID=PREPD580.${$} → Declaración de variable para identificar al JOB

Tabla 4-2 Variable para la identificación del JOB.

Con esto será posible contar con la variable "JID" la cual será única en el sistema, ya que estará distinguida por el proceso que la genera más el número de sesión del shell, el cual es único e irrepetible debido a que es asignado por el sistema operativo.

4.1.3. USO DE COMENTARIOS

El manejo de comentarios en los script's se hace por medio del metacaracter **"#"** con el cual se le indica a Kornshell ignorar la línea en cuestión. La forma de detectar los comentarios en un procedimiento de MVS es comprobando si los primeros tres caracteres son **"//*"** entonces se cambian por "#//*". Ejemplo:

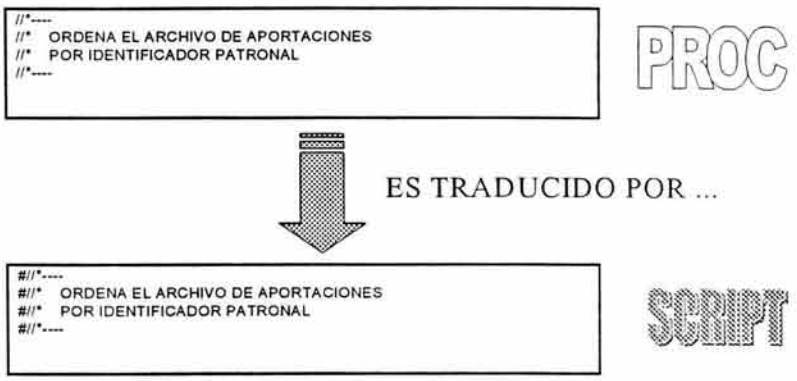


Figura 4-1 Traducción de comentarios.

4.1.4. VARIABLES DE PERFIL DE USUARIO

Es importante hacer la consideración de que para la correcta ejecución de los script's que resultarán de la conversión es necesario que previamente se hayan definido una serie de variables en el perfil del usuario, ya que sin esto será imposible su correcta ejecución. La siguiente tabla muestra una lista de dichas variables.

Variable	Descripción
LOG	Es la ruta que indica el directorio o carpeta en donde se depositarán los resultados de las ejecuciones de los procesos
SCHEDULER	Es la ruta que indica el directorio o carpeta en donde se depositarán los procesos que son generados en automático por algún otro proceso predecesor.
TMP	Es la ruta que indica el directorio o carpeta en donde se depositarán los archivos temporales
CC	Indica la ruta en donde se encuentran las utilerías para el manejo de archivos, descarga de bases de datos, sorts, etc.

LOAD	Aquí se encuentran los programas objeto del sistema.

Tabla 4-3 Variables de perfil de usuario.

4.1.5. REDIRECCIONAMIENTO DE LAS SALIDAS ESTÁNDAR.

Los JCL (Job Control Languaje) son elementos nativos del sistema MVS por lo cual basta activar o desactivar una variable en el JCL para indicarle a que dispositivo (impresora, pantalla, archivo, etc.) debe direccionar los resultados de su ejecución. Por tal motivo esto tendrá que ser emulado en UNIX mediante el direccionamiento de las salidas como a continuación se indican:

 # exec >> $LOG/$JID→ La salida es enviada al archivo $LOG/$JID.out
 # exec 2>> $LOG/$JID → Envía los mensajes de error al archivo $LOG/$JID.out2

Tabla 4-4 Líneas para redireccionar las salidas estándar.

Nótese que aquí se esta haciendo referencia a las variables "JID" y "LOG" las cuales fueron explicadas previamente, además estas líneas se encontrarán inicialmente deshabilitadas por el metacaracter "#", esto es con la finalidad de que el usuario tenga la posibilidad de elegir este método de direccionamiento de la salida o que utilice algún otro que este a su alcance.

También debemos tomar en cuenta que es necesario que la evidencia de la ejecución de los procesos deben ser lo mas claro posible, por tal motivo, es necesario desplegar mensajes que identifiquen que parte del proceso se encuentra en ejecución, que es lo que se está haciendo y los tiempos de inicio y final de cada etapa. Esto se ejemplifica en la siguiente tabla:

MVS		UNIX
Despliega Mensajes	//PREPD580 JOB (000),'DEP:018',	# IDENTIFICACION DEL JOB echo "** JOB=PREPD580" echo "** IDJOB=(000)" echo "** COMENTARIO='DEP:018'"

Contabiliza Tiempo de ejecución	N/A	echo "** INICIO=\"`date '+%d/%m/%G %T'`\"

Tabla 4-5 Redireccionamiento de salida estándar.

4.1.6. DECLARACIÓN DE VARIABLES, ETIQUETAS DE ARCHIVOS Y NOMBRE DE PASOS

Para la sustitución del nombre del paso y etiquetas es necesario detectar cuando la línea inicia con: "//NOMBPASO" ó "//ETIQUETA", en estos casos se debe hacer la declaración de una variable mediante el comando *"export"*. Adicionalmente en el script se tendrá que añadir una línea con el comando "*echo*" con el cual se imprimirá en la salida (puede ser la consola o un archivo en donde se guarden los resultados de la ejecución) la acción que se está realizando. En la siguiente tabla se ejemplifica la manera en que se realizará dicho cambio.

MVS		UNIX
Variable	PRO=D580,	Echo "PROC=PREPD580" Export PROC=PREPD580
Paso	//PARMF520	# IDENTIFICACION DEL PASO Echo "PASO=PARMF520" Export PASO=PARMF520
Etiqueta de archivo	SORTIN DD DSN= &AT.RE00.DT.R000.&PRO..APOPEN,	Echo "SORTIN=\"${AT}RE00/DT/R000.${PRO}.APOPEND\"" Export SORTIN="${AT}RE00/DT/R000.${PRO}.APOPEND"

Tabla 4-6 Traducción de variables y nombres de paso.

En la tabla anterior se puede notar que las variables que se definen dentro de un "JOB" o un "PROC" tienen la siguiente nomenclatura:

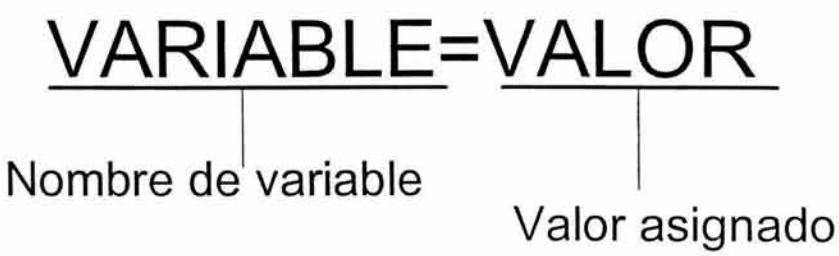


Figura 4-2 Asignación de una variable.

Además en una misma línea pueden declararse una o más variables separadas por coma en cuyo caso se realizará la traducción de cada una de ellas por separado, es decir, que por cada variable encontrada se tendrá que escribir sus correspondientes comandos "echo" y "export". Una vez que ya ha sido declarada e inicializada una variable esta puede ser invocada en cualquier parte del procedimiento haciendo uso del metacaracter "&", de hecho en la tabla 4-6 se observa que al hacer la traducción del la etiqueta de archivo se encuentran las variables "&AT" y "&PRO" las cuales son traducidas como: ="${AT}" y "${PRO}" sucesivamente.

4.2. DISPARADORES (JOB'S)

Ya se ha descrito cada una de sus partes en el capítulo anterior, tanto para OS/390, así como para UNIX. A continuación se explica la manera en que se realizará la traducción de cada sección.

4.2.1.ÁREA DE POSTULADOS INICIALES.

Para la traducción de esta sección, la primera línea que aparece en el disparador es de gran importancia para nuestro sistema, ya que encontraremos

información que es indispensable para interpretar esta área. La siguiente figura muestra la manera en que dicha línea debe ser interpretada.

Figura 4-3 Primer línea de JOB.

En esta misma sección encontraremos la variable JCLLIB la cual indica al MVS la ruta en donde se encuentra el procedimiento a ejecutar, para este caso lo que se debe hacer en UNIX es declarar una variable global llamada PROCLIB y su valor será la ruta donde se localizará su correspondiente procedimiento.

Para realizar la conversión de esta área primero se agregarán los comandos definidos en la sección de consideraciones iniciales (selección del shell, variables para la creación de archivos temporales y redireccionamiento de salida estándar) y que son de tipo obligatorio, posteriormente se agregan las líneas para la identificación del JOB. En la siguiente tabla se muestra la manera en que esta primera parte quedaría convertida.

```
#!/bin/ksh
#
# VARIABLE QUE SERVIRA PARA CREACION DE ARCHIVOS TEMPORALES
  export JID=PREPD580.${$}
#
# REDIRECCIONAMIENTO DE STDOUT Y STDERR
#  exec >> $LOG/$JID
#  exec 2>> $LOG/$JID
## IDENTIFICACION DEL JOB
echo "** JOB=PREPD580"
echo "** IDJOB=(000)"
echo "** COMENTARIO='DEP:018'"
echo "** NUMERO=$$"
echo "** INICIO=\"`date '+%d/%m/%G %T`"\"
#
# PATH DE LOS PROCS A EJECUTAR
  export PROCLIB=/PBRE00/DP/PROCLIB
```

Tabla 4-7 Traducción del área de postulados iniciales.

4.2.2. ÁREA DE COMENTARIOS

Esta área por ser de comentarios solo se procederá a traducirlos como anteriormente se expuso.

4.2.3. ÁREA DE PARÁMETROS SIMBÓLICOS

La línea que inicia esta parte tiene la singularidad de contar con el comando "EXEC" que indica el nombre del proceso a ejecutar, por lo cual la traducción a realizar es la siguiente:

MVS	UNIX
EXEC PROC=PREPD580	# VARIABLES QUE SERVIRAN PARA LOS PROGRAMAS IMPLICADOS SUBSECUENTES export EJECUTA=PREPD580

Tabla 4-8 Primer línea del área de parámetros simbólicos.

Nótese que la nueva variable "EJECUTA" contendrá el nombre del proceso que el JOB invocará para ser ejecutado.

El resto de esta sección son variables y parámetros a los que se les asignará su correspondiente valor desde la línea de comandos de UNIX. En el siguiente diagrama de flujo se muestra el método que se diseñó para hacer dicha asignación de valores.

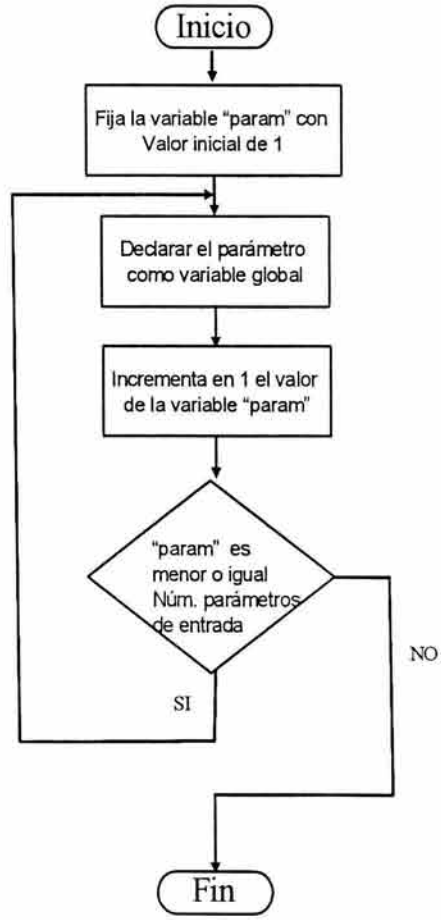


Figura 4-4 Diagrama del método para la lectura de variables.

Este cambio es con el fin de evitar que el JOB deba de ser editado cada vez que tenga que ser ejecutado para actualizar sus parámetros simbólicos. Otro cambio significativo será el de fijar valores a dos nuevas variables que son "RC" y "REINICIO" las cuales serán utilizadas para poder manipular el reinicio de los procesos (este concepto será explicado a detalle mas adelante).

A continuación se muestra un ejemplo de cómo se tendrá que ejecutar el proceso llamado PREPD580, el cual tiene una variable con nombre "AT" y un criterio para una tarjeta de override.

```
PREPD580 AT=/V RC=99 REINICIO=PREPD580_PESO0190
PEUT0460_SYSUT1="C24_0 C100 C0103010200100120010101
C99XREPD580"
```

Tabla 4-9 Ejemplo de ejecución del proceso PREPD580 con reinicio.

Nótese que en este ejemplo que las variables "RC" y "REINICIO" no son igual a cero, esto indica que el proceso PREPD580 será reiniciado a partir del paso PESO0190. En caso de que el objetivo fuese ejecutar el proceso completo entonces la instrucción tendría que ser como a continuación se muestra.

```
PREPD580 AT=/V RC=0 REINICIO=0 PEUT0460_SYSUT1="C24_0 C100
C0103010200100120010101 C99XREPD580"
```

Tabla 4-10 Ejemplo de ejecución del proceso PREPD580 completo.

4.2.4. ÁREA DE TARJETAS DE OVERRIDE Y CRITERIOS

En caso de existir esta sección lo que se procederá a hacer será generar un archivo temporal con la ayuda de la utilería "echo801" (la cual se explicará mas adelante) en donde se depositará el valor del criterio proporcionado. El siguiente diagrama muestra el algoritmo que implementará dicho cambio.

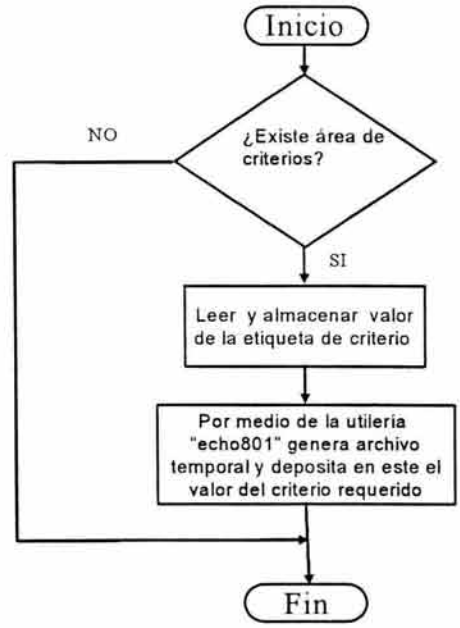


Figura 4-6 Diagrama del método para traducir el área de criterios

Al final lo que nos resta por hacer es invocar el proceso ejecutar, tomar el tiempo de inicio y final del proceso y verificar el estado del retorno del control para comunicárselo al operador y así este pueda conocer si el proceso fue exitoso. Todo esto se realiza agregando el siguiente código al JOB:

```
time ${PROCLIB}/${EJECUTA}
     rc=$?
     if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
     then
          RC=$rc
     fi
     rm ${TMP}/${JID}*
echo "** FIN=\"`date '+%d/%m/%G %T'`"\"
echo "** RETURN-CODE=$rc"
return $RC
```

Tabla 4-11 Traducción del área de criterios

4.3. PROCEDIMIENTOS (PROC'S).

4.3.1. IDENTIFICADOR DEL PROCEDIMIENTO.

Tal como se vió en el anterior capítulo la primera línea dentro de un "PROC" es el identificador del procedimiento. La figura muestra la manera en que dicha línea es interpretada:

Figura 4-7 Primer línea del PROC

La traducción de esta línea quedará de la siguiente manera:

```
# IDENTIFICACION DEL PROCEDIMIENTO
  echo "** PROC=PRTPD400"
#
# VARIABLE QUE SERVIRA PARA LOS PASOS SUBSECUENTES
  export PROC=PRTPD400
```

Tabla 4-12 Traducción de la primer línea del PROC

4.3.2. ESTRUCTURA GENERAL EJECUCIÓN DE PASOS

El resto de los procedimientos se refiere a la ejecución de cada uno de los pasos los cuales invocan y ejecutan los programas y utilerías del sistema para la obtención de los resultados esperados, dichos pasos cumplen con una estructura general gracias a los estándares de construcción con los que se

cuenta. La siguiente figura muestra la estructura general con la que actualmente se codifican dichos pasos.

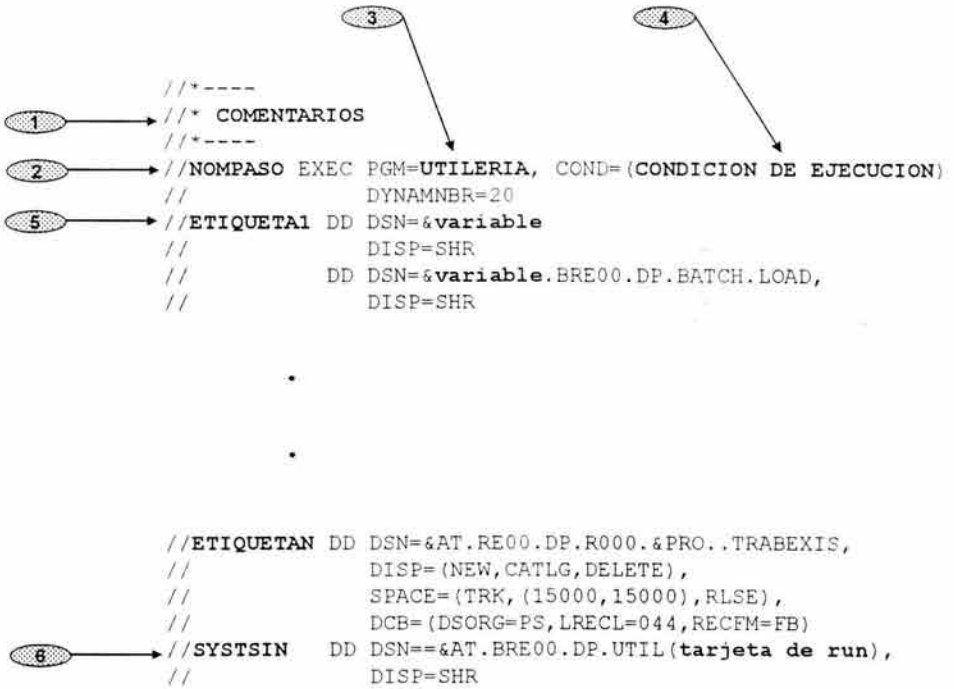


Figura 4-8 Ejemplo de la estructura general de un paso dentro del PROC

En la primer parte se encuentra un área de comentarios donde se proporciona una breve descripción del objetivo del paso. La traducción de esta se hará de la misma forma en que expusimos anteriormente.

En la sección 2 se encuentra la línea que identifica al paso mediante una etiqueta a la cual se le aplicará la misma regla de traducción para etiquetas (véase "Declaración de variables y etiquetas en este mismo capítulo).

Además podemos observar que en esta también se encuentran los comandos que nos dicen que programa ejecutaremos (sección 3) y cual es la condición de ejecución (sección 4) para las cuales se hará la siguiente traducción

```
export EJECUTA= UTILERIA
if [ "$RC" -le 0 ] || [ "${REINICIO}" = "${PROC}_${PASO}" ]
then

{AQUÍ SE ENCONTRARAN EL RESTO DE LOS COMANDOS QUE
INTEGRAN EL PASO}
```

Tabla 4-13 Traducción las secciones 2,3 y 4 de la estructura general de los pasos de un PROC

Lo que en la tabla anterior se esta haciendo es, en primer lugar, cambiar la cadena EXEC PGM=PROGRAMA por el comando export EJECUTA=PROGRAMA, con lo cual definimos la variable EJECUTA y almacenamos en ella el nombre de la utilería que será empleada en dicho paso. Cabe hacer mención que la razón por la que en este punto solo se invocan utilerías en vez de también hacer llamadas a programas, es debido a que los estándares de construcción vigentes así lo indican, por ello mas adelante se explicara más ampliamente la manera en que finalmente se ejecutan los programas.

El resto de la tabla muestra el código encargado de controlar el flujo de la ejecución. En otras palabras, esto significa que es aquí donde se tomará la decisión de ejecutar o no ejecutar el siguiente paso del proceso dependiendo del código de retorno del paso anterior, para lo cual dicho código de retorno debe de ser verificado al principio y al final de cada paso. Al igual que la sección anterior a esta, también se le dedicará un apartado mas adelante para exponer claramente el método de "Control de ejecución del flujo".

En la sección indicada como número 5 de la figura 4.8 se aplicarán las siguientes reglas:

1. Regla para la "Declaración de variables, etiquetas de archivos y nombre de pasos" la cual vimos al inicio de este capítulo.

2. Regla de "Excepción de comandos y palabras reservadas para MVS" la cual veremos más adelante.

Por último, la sección 6 se identifica fácilmente porque tiene la etiqueta SYSTIN y en ella vamos a encontrar la ruta completa de la "tarjeta de run", a estas tarjetas se les conoce así porque contienen la definición de variables y elementos indispensables para la correcta ejecución de los programas y demás acciones del sistema. La siguiente tabla muestra un ejemplo de una tarjeta de run utilizada para ejecutar un programa.

```
DSN SYSTEM(DSNT)
  RUN PROG(PTP612) PLAN(VRTBATPL)
END
```

Tabla 4-14 Traducción de la etiqueta SYSTIN

La siguiente tabla muestra el resultado de la traducción de la sección 6:

```
echo "SYSIN=\"${AT}BRE00/DP/UTIL/PETS134B\""
export SYSIN="${AT}BRE00/DP/UTIL/PETS134B"
echo "${EJECUTA}"
time ${EJECUTA}
rc=$?
if [ $rc -gt $RC ] || [ "${REINICIO}" = "${PROC}_${PASO}"
]
then
   RC=$rc
fi
unset SYSIN
```

Tabla 4-15 Traducción la sección 6 de la estructura general de los pasos de un PROC

Como se puede observar la primer línea de la sección 6 es traducida con la regla de "Declaración de variables, etiquetas de archivos y nombre de pasos", las dos siguientes líneas son para la ejecución de la utilería definida al principio del paso, posteriormente se captura el código de retorno en la variable "rc" y se verifica con el paso anterior. Finalmente se destruyen las variables que son únicas para el paso. Para saber cuales son las variables que deben ser destruidas en caso de ser utilizadas, véase el anexo 3.

La figura 4-9 muestra un ejemplo de un paso de un proceso del módulo de recaudación llamado PREPD200 y en la figura 4-10 se puede observar el resultado que este sufriría después de aplicar las reglas de traducción anteriormente descritas.

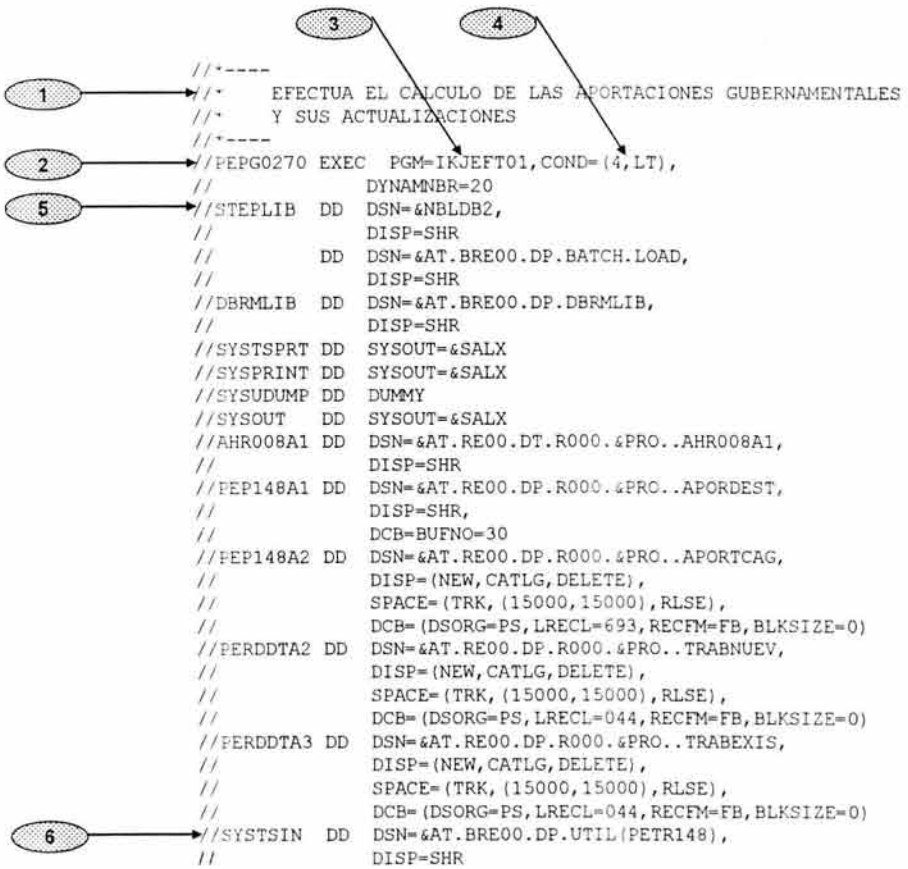
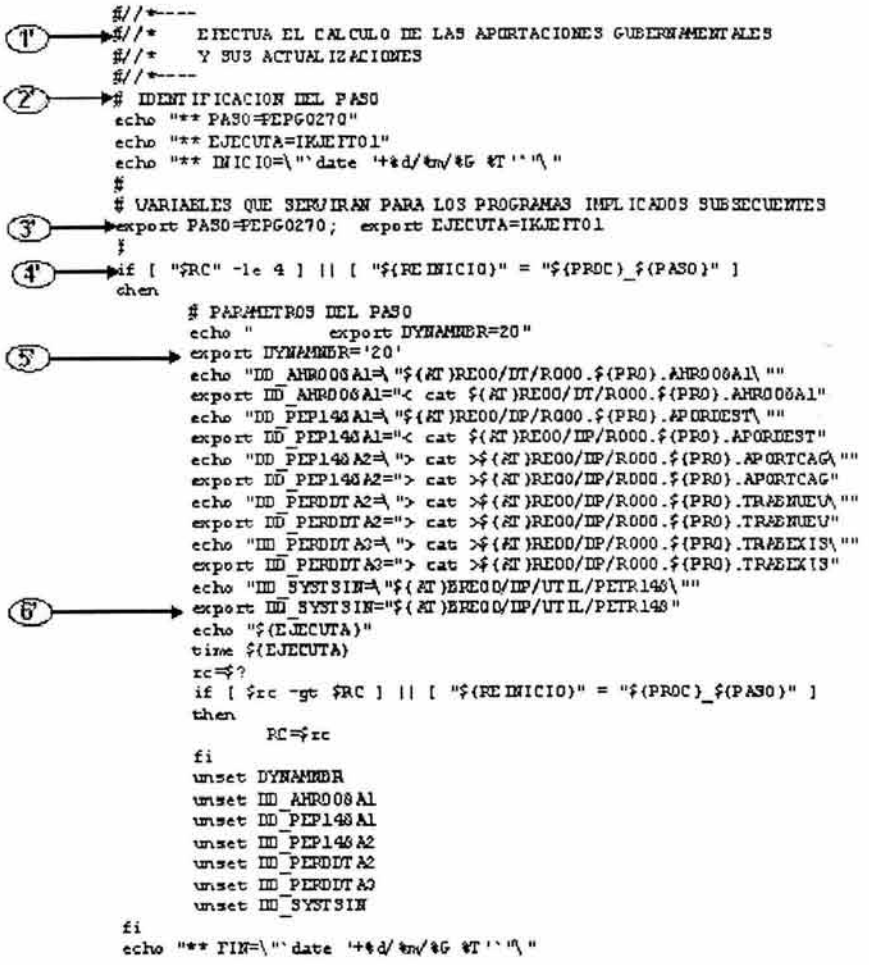


Figura 4-9 Paso del PREPD200 antes de ser traducido




```
#//*----
#//*    EJECTUA EL CALCULO DE LAS APORTACIONES GUBERNAMENTALES
#//*    Y SUS ACTUALIZACIONES
#//*----
# IDENTIFICACION DEL PASO
echo "** PASO=PEPG0270"
echo "** EJECUTA=IKJEFT01"
echo "** INICIO=\"`date '+%d/%m/%G %T'`\"
#
# VARIABLES QUE SERVIRAN PARA LOS PROGRAMAS IMPLICADOS SUBSECUENTES
export PASO=PEPG0270;  export EJECUTA=IKJEFT01
#
if [ "$RC" -le 4 ] || [ "${RE_INICIO}" = "${PROC}_${PASO}" ]
then
      # PARAMETROS DEL PASO
      echo "       export DYNAMNBR=20"
      export DYNAMNBR='20'
      echo "DD_AHR008A1=\"${AT}RE00/DT/R000.${PRO}.AHR008A1\""
      export DD_AHR008A1="< cat ${AT}RE00/DT/R000.${PRO}.AHR008A1"
      echo "DD_PEP148A1=\"${AT}RE00/DP/R000.${PRO}.APORDEST\""
      export DD_PEP148A1="< cat ${AT}RE00/DP/R000.${PRO}.APORDEST"
      echo "DD_PEP148A2=\"> cat >${AT}RE00/DP/R000.${PRO}.APORTCAG\""
      export DD_PEP148A2="> cat >${AT}RE00/DP/R000.${PRO}.APORTCAG"
      echo "DD_PERDITA2=\"> cat >${AT}RE00/DP/R000.${PRO}.TRABNUEV\""
      export DD_PERDITA2="> cat >${AT}RE00/DP/R000.${PRO}.TRABNUEV"
      echo "DD_PERDITA3=\"> cat >${AT}RE00/DP/R000.${PRO}.TRABEXIS\""
      export DD_PERDITA3="> cat >${AT}RE00/DP/R000.${PRO}.TRABEXIS"
      echo "DD_SYSTSIN=\"${AT}BRE00/DP/UTIL/PETR148\""
      export DD_SYSTSIN="${AT}BRE00/DP/UTIL/PETR148"
      echo "${EJECUTA}"
      time ${EJECUTA}
      rc=$?
      if [ $rc -gt $RC ] || [ "${RE_INICIO}" = "${PROC}_${PASO}" ]
      then
            RC=$rc
      fi
      unset DYNAMNBR
      unset DD_AHR008A1
      unset DD_PEP148A1
      unset DD_PEP148A2
      unset DD_PERDITA2
      unset DD_PERDITA3
      unset DD_SYSTSIN
fi
echo "** FIN=\"`date '+%d/%m/%G %T'`\""
```

Figura 4-10 Paso del PREPD200 después de ser traducido

4.3.3. CONTROL DE EJECUCIÓN DEL FLUJO

En el sistema MVS, el control de flujo de ejecución se realiza al inicio de cada paso en el que se pregunta por código de ejecución de paso anterior, si es satisfactorio, procede la ejecución, de lo contrario se omite y continua con el siguiente paso. Otra variante es la del reinicio de un proceso en un punto

determinado, esto se hace indicando en el disparador o JOB el nombre del procedimiento y el paso donde tiene que iniciar su ejecución. En la siguiente figura se muestra claramente la forma en que dichos controles operan.

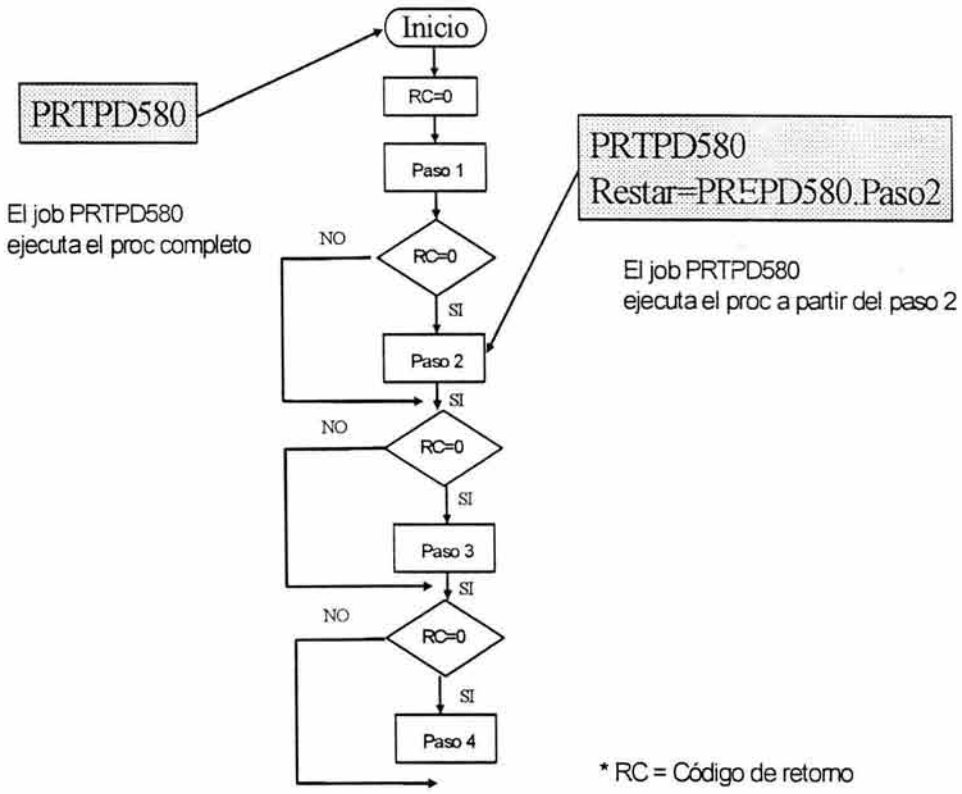


Figura 4-11 Diagrama del método para el control de reinicios

La manera de sustituir el control de ejecución en UNIX, es capturando el código de retorno en una variable y antes de iniciar la ejecución del siguiente paso, evaluar si cumple con el código esperado, si no es así entonces omite su ejecución y continúa con el siguiente paso. Para el control de reinicio se realizará por medio de la variable "REINICIO" la cual contendrá el *nombre* del "*PROC*" y el paso a reiniciar.

En la tabla anterior se observa que la condición de ejecución es sustituida por una validación, la cual solo permitirá procesar dicho paso si el código de retorno es valido o si la variable *"REINICIO"* coincide con el paso actual.

4.3.4. EJECUCIÓN DE PROGRAMAS, UTILERÍAS Y MANEJO DE ARCHIVOS

Para la ejecución de programas en el sistema MVS se hace uso de una utilería de sistema llamada IKJEFT01 por medio del comando "EXEC PGM= IKJEFT01", para lo cual es necesario definir por medio de etiquetas tanto los archivos de entrada como de salida del programa así como el elemento UTIL, que indica cual es el programa aplicativo a ejecutar. La principal función del programa IKJEFT01, es propiciar al programa aplicativo el ambiente de ejecución al realizar la conexión con la base de datos. El siguiente dibujo representa en forma gráfica la manera en que esta operación es realizada:

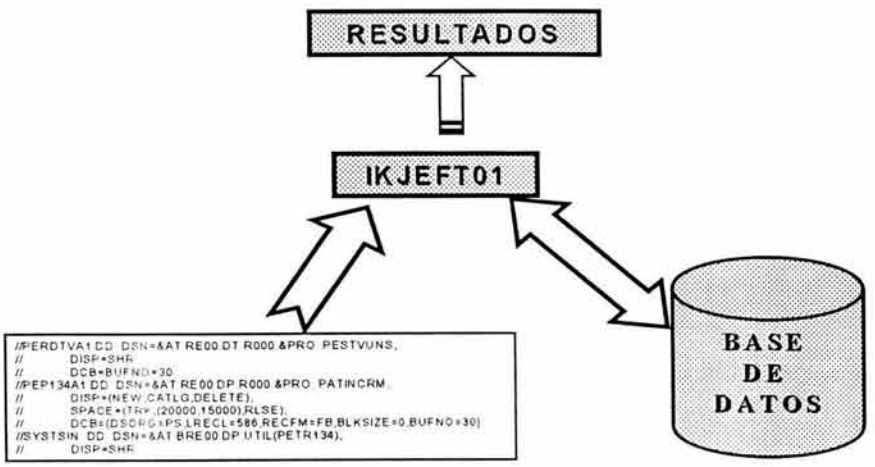


Figura 4-12 Funcionamiento general del programa IKJEFT01

En UNIX no es necesario un programa de sistema que realice la conexión a la base de datos ya que esta se encuentra definida desde el perfil de cada

usuario, pero como se ha venido mencionando, uno de nuestros objetivos es el conservar en la medida de lo posible la actual forma de ejecución de procesos, por lo tanto, se diseñó un script que emula esta funcionalidad. El siguiente diagrama muestra la lógica interna del emulador del IKJEFT01.

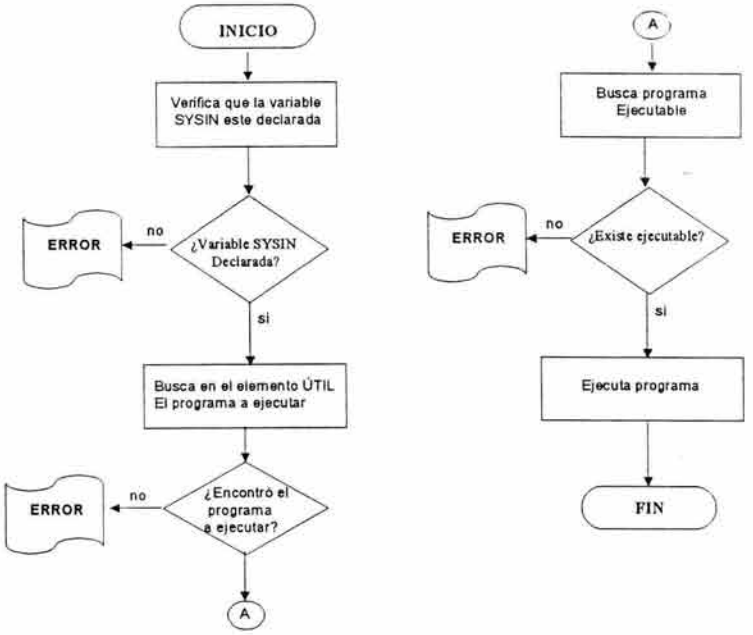


Figura 4-13 Diagrama del método para la sustitución del programa IKJEFT01

Por lo anterior la sustitución quedará de la siguiente manera:

	MVS	UNIX
Comando de ejecución	EXEC PGM=IKJEFT01,	echo "** EJECUTA=IKJEFT01" export EJECUTA=IKJEFT01

Tabla 4-16 Sustitución del I alinea del programa IKJEFT01

Otra de las utilerías que sufre un cambio considerable es la llamada IEFBR14, cuyo objetivo principal es el borrado de archivos para liberar el espacio que estos utilizan y el de reservar el espacio requerido para los archivos que se van a utilizar durante el proceso. Esto para UNIX ya no es necesario, por lo tanto, en el caso especial de esta utilería lo que se debe considerar básicamente es

que esta puede ser usada tanto para borrar archivos como para alojarlos. En la tabla 4-17 se muestra la traducción que se realizará en ambos casos.

	MVS	UNIX

Borra archivos	//DEASLORH DD
	DSN=&SECC..F&FECH..CONFAF,
	DISP=(MOD,**DELETE**,DELETE),
	UNIT=&NBVOLTRB,
	SPACE=(TRK,0),
	DCB=(DSORG=PS,LRECL=080,RECFM=FB)
		rm ${SECC}/F${FECH}/CONFAF

Aloja archivos	//DEASLORH DD
	DSN=&SECC..F&FECH..CONFMQ,
	DISP=(MOD,**CATALG**,DELETE),
	UNIT=&NBVOLTRB,
	SPACE=(TRK,0),
	DCB=(DSORG=PS,LRECL=080,RECFM=FB)
		cp /dev/null
		${SECC}/F${FECH}/CONFMQ

Tabla 4-17. Empleo de la utilería IEFBR14.

Como se puede observar en la tabla anterior la diferencia para saber si se trata de borrar un archivo o alojarlo es mediante la palabra "DELETE" o "CATLG" en la línea de disposición ("DISP"). En ambos casos lo que se procede a hacer es generar un comando de borrado o copiado de archivo según corresponda.

Al igual que el IKJEFT01 o el IEFBR14, se cuenta con otros programas que tienen una aplicación específica y que por consiguiente se requiere su correspondiente emulador. La siguiente tabla muestra la lista de estos elementos y una breve explicación de su función.

ELEMENTO	FUNCIÓN
SORT	Realiza el ordenamiento e indexamiento de archivos de acuerdo a un orden determinado
DMBATCH	Realiza la conexión al proceso de transferencia de archivo (FTP "file transfer proces")

IECEGENER IEBGENER	Son utilizados para realizar copias de un archivo a otro.
PARMFILE	Verifica si existen las variables de ambiente PARM y FILEOUT , si es el caso entonces escribe el PARM en el archivo FILEOUT.
IEAMAXC0	Se utiliza para la ejecución de tarjetas de borrado de archivos de trabajo
IEFBR14	Borrado y alojado de archivos de trabajo
IDCAMS	Realiza la copia de archivo VSAM
PTLDRIVM	Es la utilería de PLATINUM para realizar cargas y descargas de la base de datos
Echo801	En un archivo temporal aloja los criterios de ejecución para la ejecución de procesos

Tabla 4-18 Lista del utilerías y programas a sustituir

En la tabla anterior se observa que se cuenta con las utilerías IEFBR14, IEBGENER, IDCAMS, IEAMAXC0, DMBATCH, IECEGENER para la creación borrado, copiado y transferencia de archivos. En UNIX estas operaciones se pueden realizan de una manera mucho más sencilla, sin embargo se generaron shell's que emularán las acciones de estos programas y que por supuesto tendrán los mismos nombres. A continuación se ejemplifican estas sustituciones:

	MVS	UNIX
IEFBR14	//*---- //* ALOJA ARCHIVOS QUE SE //* UTILIZARAN EN EL //* PROCEDIMIENTO //*---- //PEUT0150 EXEC PGM=IEFBR14,	#//*---- #//* ALOJA ARCHIVOS QUE SE UTILIZARAN EN EL #//* PROCEDIMIENTO #//*---- # IDENTIFICACION DEL PASO echo "** PASO=PEUT0150" echo "** EJECUTA=IEFBR14" export PASO=PEUT0150; export EJECUTA=IEFBR14
IEAMAXC0	//*---- //* BORRA ARCHIVOS //*---- //*---- //PEUT0080 EXEC PGM=IEAMAXC0	#// #// BORRA ARCHIVOS #// echo "** PASO=PEUT0080" echo "** EJECUTA=IEAMAXC0" export PASO=PEUT0080; export EJECUTA=IEAMAXC0 #

IEBGENER	//*---- //* COPIA EL ARCHIVO PEP309 //*---- //PEUT0200 EXEC PGM=IEBGENER	#//*---- #//* COPIA EL ARCHIVO PEP309. #//*---- # IDENTIFICACION DEL PASO echo "** PASO=PEUT0210" echo "** EJECUTA=IEBGENER" export PASO=PEUT0210; export EJECUTA=IEBGENER

Tabla 4-19 Ejemplos de traducción de las líneas de utilerías.

4.3.5. MANEJO DE ARCHIVOS TEMPORALES

Para la generación de archivos temporales se hará uso de un directorio especial para cada una de las aplicaciones, por ejemplo: para la aplicación de Retiros, el directorio temporal será /PRT00/TMP además, por cada aplicación y ambiente de desarrollo se tendrá la variable "TMP" en la cual se guardará la ruta de dicho directorio. La siguiente tabla muestra el valor de la variable TMP de acuerdo a la aplicación y el ambiente.

APLICACIÓN / AMBIENTE	TEST	VALIDACIÓN	ACEPTACIÓN	PRODUCCIÓN
RECAUDACIÓN	TMP=/TRE00/TMP	TMP=/VRE00/TMP	TMP=/ARE00/TMP	TMP=/atp/re
REGISTRO	TMP=/TAF00/TMP	TMP=/VAF00/TMP	TMP=/AAF00/TMP	TMP=/atp/af
TRASPASOS	TMP=/TTC00/TMP	TMP=/VTC00/TMP	TMP=/ATC00/TMP	TMP=/atp/tc
RETIROS	TMP=/TRT00/TMP	TMP=/VRT00/TMP	TMP=/ART00/TMP	TMP=/atp/rt

Tabla 4-20 Valores del la variable TMP

Actualmente, al realizar la declaración de un archivo temporal tanto su nombre como su ubicación son determinados por el sistema operativo. Para la

traducción de este proceso nos tendremos que auxiliar de la variable temporal "JID" que será declarada en el "JOB" mediante la instrucción:

export JID=NombreJob.${$}

Donde: NombreJob= nombre del "JOB" que lo ejecuta

Ejemplo: export JID=PREPD580.${$}

Resultado: JID=PREPD580.1018

De esta manera el sistema asociara el número de sesión de Kornshell (el cual es único) al "JOB" que lo ejecuta obteniendo como resultado una variable irrepetible en el sistema. La combinación de las variables "TMP" y "JID" nos permitirán generar archivos temporales.

Con lo anteriormente expuesto, podemos formular la siguiente regla de traducción para generar archivos temporales:

	MVS	UNIX

Genera Archivo Temporal	//FILEOUT DD DSN=&&TMP1,
	// DISP=(,PASS),
	// SPACE=(TRK,1),
	// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
		echo "VAR_TEMP=\"${TMP}/${JID}.TMP1\""
		
		VAR_TEMP="${TMP}/${JID}.TMP1"

Tabla 4-21 Traducción de archivos temporales

4.3.6. CONCATENACIÓN DE ARCHIVOS

En MVS enlista los archivos una tras otro, son leídos como uno solo mediante la concatenación. Esto resulta muy útil cuando se requiere realizar consolidados partiendo de 2 o más archivos. La manera de efectuar esta concatenación de archivos en UNIX será mediante el uso de la variable temporal "VAR_TEMP" la

cual será la liga entre los archivos a unir. El siguiente diagrama muestra la forma en que este proceso trabajará.

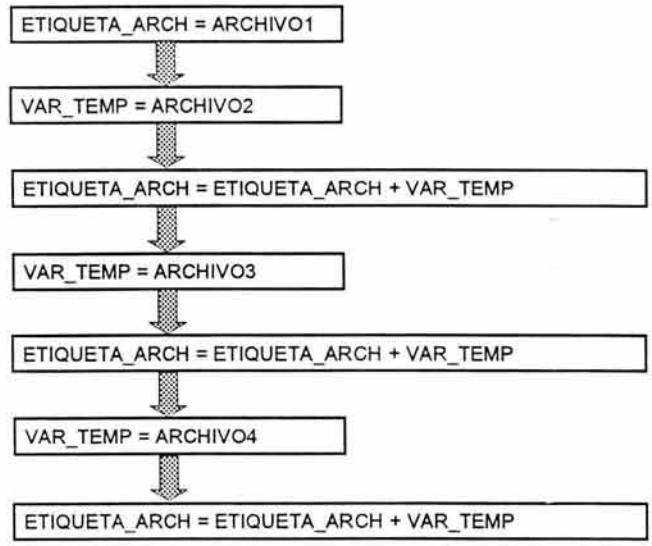


Figura 4-14 Concatenación de archivos.

En la figura anterior se muestra la manera en que se unen 4 archivos diferentes en una misma variable de etiqueta, con lo cual el proceso que haga uso de esta etiqueta podrá utilizar los cuatro archivos como si fuera uno solo.

El siguiente ejemplo muestra la manera en que sería traducido un proceso en el que se realiza la concatenación de varios archivos.

	MVS	UNIX

Une Archivos	//AHR008A1 DD DSN=&AT.RE00.DT.AHR008A1,
	// DISP=SHR
	//PEP313A1 DD DSN=&AT.RE00.DP.MOVSALDO,
	// DISP=SHR,
	// DCB=BUFNO=30
	// DD DSN=&AT.RE00.DP.MOVINDIV,
	// DISP=SHR,
	// DCB=BUFNO=30
	// DD DSN=&AT.RE00.DP.MOVSEXCE,
	// DISP=SHR,
	// DCB=BUFNO=30
	// DD DSN=&AT.RE00.DT.MOVSALTE,
	// DISP=SHR,
	// DCB=BUFNO=30
	// DD DSN=&AT.RE00.DT.MOVINTTE,
	// DISP=SHR,
	// DCB=BUFNO=30
	// DD DSN=&AT.RE00.DP.MOVINTBX,
	// DISP=SHR,
	// DCB=BUFNO=30
	//SYSTSIN DD
	DSN=&AT.BRE00.DP.UTIL(PETR313),
	// DISP=SHR
		echo "DD_AHR008A1=\"${AT}RE00/DT/AHR008A1\""
		export DD_AHR008A1="${AT}RE00/DT/AHR008A1"
		echo "DD_PEP313A1=\"${AT}RE00/DP/MOVSALDO\""
		export DD_PEP313A1="${AT}RE00/DP/MOVSALDO"
		echo "VAR_TEMP=\"${AT}RE00/DP/MOVINDIV\""
		VAR_TEMP="${AT}RE00/DP/MOVINDIV"
		echo "DD_PEP313A1=\"< cat ${DD_PEP313A1} ${VAR_TEMP}\""
		export DD_PEP313A1="< cat ${DD_PEP313A1} ${VAR_TEMP}"
		echo "VAR_TEMP=\"${AT}RE00/DP/R000.${PRO}.MOVSEXCE\""
		VAR_TEMP="${AT}RE00/DP/R000.${PRO}.MOVSEXCE"
		echo "DD_PEP313A1=\"${DD_PEP313A1} ${VAR_TEMP}\""
		export DD_PEP313A1="${DD_PEP313A1} ${VAR_TEMP}"
		echo "VAR_TEMP=\"${AT}RE00/DT/R000.${PRO}.MOVSALTE\""
		VAR_TEMP="${AT}RE00/DT/R000.${PRO}.MOVSALTE"
		echo "DD_PEP313A1=\"${DD_PEP313A1} ${VAR_TEMP}\""
		export DD_PEP313A1="${DD_PEP313A1} ${VAR_TEMP}"
		echo "VAR_TEMP=\"${AT}RE00/DT/R000.${PRO}.MOVINTTE\""
		VAR_TEMP="${AT}RE00/DT/R000.${PRO}.MOVINTTE"
		echo "DD_PEP313A1=\"${DD_PEP313A1} ${VAR_TEMP}\""
		export DD_PEP313A1="${DD_PEP313A1} ${VAR_TEMP}"
		echo "VAR_TEMP=\"${AT}RE00/DP/R000.${PRO}.MOVINTBX\""
		VAR_TEMP="${AT}RE00/DP/R000.${PRO}.MOVINTBX"
		echo "DD_PEP313A1=\"${DD_PEP313A1} ${VAR_TEMP}\""
		export DD_PEP313A1="${DD_PEP313A1} ${VAR_TEMP}"
		echo "DD_SYSTSIN=\"${AT}BRE00/DP/UTIL/PETR313\""
		export DD_SYSTSIN="${AT}BRE00/DP/UTIL/PETR313"
		echo "${EJECUTA}"
		time ${EJECUTA}

Tabla 4-22 Traducción de la concatenación de archivos.

4.3.7. LAYOUT'S DISPARADORES DE PROCESOS

Una tarea común en el procesamiento de información de los procesos batch, es la de generar o disparar nuevos procesos dependiendo del flujo ejecutado. Un ejemplo de ello se muestra en la siguiente figura:

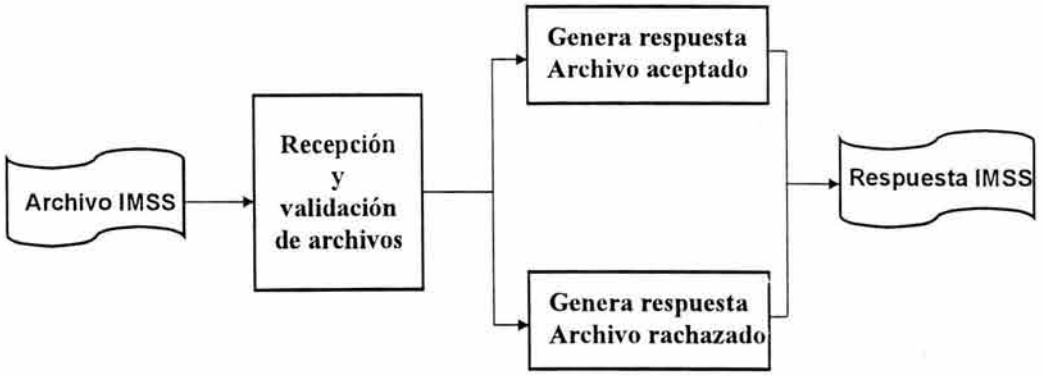


Figura 4-15 Modelo conceptual del disparó automático de procesos

En la figura anterior se puede observar que el módulo "Recepción y validación de archivos" después de realizar una serie de validaciones, puede aceptar o rechazar el archivo en cuestión, por lo que los procesos "Genera respuesta de archivo aceptado" o "Genera respuesta de archivo rechazado" pueden ser generados y ejecutados automáticamente.

La manera de realizar esta automatización es mediante programas disparadores que buscan la información necesaria para generar el job en una base de datos en donde se almacena por cada uno de los disparadores su respectivo "layout" o "esqueleto" del "JOB", por medio del cual éste puede ser escrito en su biblioteca correspondiente para su ejecución. La siguiente figura esquematiza dicho procedimiento.

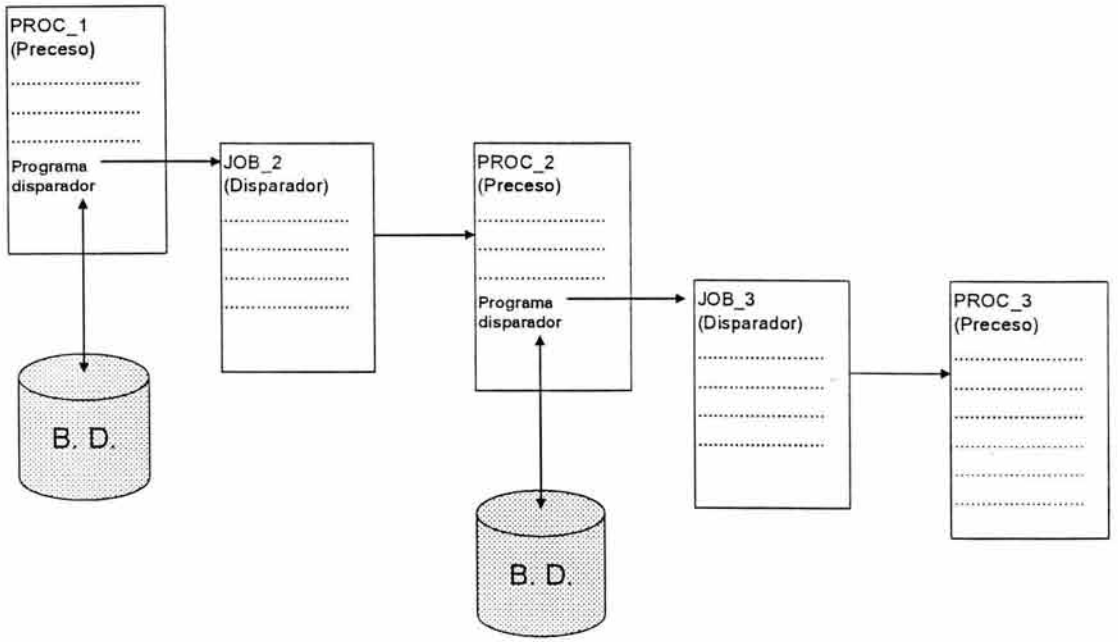


Figura 4-16 Funcionamiento conceptual del los layout's de disparo.

En la figura anterior se observa que un programa disparador del PROC_1 genera al JOB_2 y este a su vez ejecuta su correspondiente PROC_2 el cual también contiene un programa disparador que genera al JOB_3.

La principal diferencia entre un "JOB" generado por este método contra el original, es que en este no existe el algoritmo del área de parámetros simbólicos ya que en estos casos el programa encargado de armar el "JOB" transfiere los valores correspondientes a cada una de las variables requeridas en el proceso que le continua.

La siguiente figura muestra un ejemplo del layout del proceso identificado como PRTPD395.

```
TZ5    PRTPD395001        #!/bin/ksh
TZ5    PRTPD395002        #
TZ5    PRTPD395003        # VARIABLE QUE SERVIRA PARA CREACION DE ARCHIVOS TEMPORALES
TZ5    PRTPD395004          export JID=PRTPD395.${$}
TZ5    PRTPD395005        #
TZ5    PRTPD395006        # VARIABLE CON EL CODIGO DE RETORNO DE LOS PASOS
TZ5    PRTPD395007          export RC=0
TZ5    PRTPD395008        #
TZ5    PRTPD395009        # VARIABLE CON EL PASO DE REINICIO
TZ5    PRTPD395010          export REINICIO=0
TZ5    PRTPD395011        #
TZ5    PRTPD395012        # REDIRECCIONAMIENTO DE STDOUT Y STDERR
TZ5    PRTPD395013        # exec ^^ $LOG/$JID
TZ5    PRTPD395014        # exec 2^^ $LOG/$JID
TZ5    PRTPD395015        #
TZ5    PRTPD395016        # IDENTIFICACION DEL JOB
TZ5    PRTPD395017        echo "** JOB=PRTPD395"
TZ5    PRTPD395018        echo "** IDJOB=(009)"
TZ5    PRTPD395019        echo "** COMENTARIO='D.R.AUT R-P'"
TZ5    PRTPD395020        echo "** NUMERO=$$"
TZ5    PRTPD395021        echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
TZ5    PRTPD395022        #
TZ5    PRTPD395023        # PATH DE LOS PROCS A EJECUTAR
TZ5    PRTPD395024          export PROCLIB=/>1/>2/PROCLIB
TZ5    PRTPD395025        #
TZ5    PRTPD395026        #/*JOBPARM L=99999
TZ5    PRTPD395027        #//*----------
TZ5    PRTPD395028        #//*           SISTEMA: P R O C E S A R .
TZ5    PRTPD395029        #//*            MODULO: R E T I R O S .
TZ5    PRTPD395030        #//*        DISPARADOR: PRTPD395.
TZ5    PRTPD395031        #//*       DESCRIPCION: D.R. SOL.DE VERIF. RESOL. IMSS SAR
TZ5    PRTPD395032        #//*                DEP: PTDT050.
TZ5    PRTPD395033        #//*        FRECUENCIA: DIARIO (EN LOS DIAS PRE-ESTABLECIDO
TZ5    PRTPD395034        #//*AUTOR: SOFTTEK(ING.JULIO CESAR GHENO FLORES)
TZ5    PRTPD395035        #//*        FECH ULT ACT: 14/MAYO/1997.
TZ5    PRTPD395036        #//*     OBSERVACIONES:
TZ5    PRTPD395037        #//*----
TZ5    PRTPD395038        # IDENTIFICACION DEL PASO
TZ5    PRTPD395039        echo "** PASO=PRTPD395"
TZ5    PRTPD395040        echo "** EJECUTA=PRTPD395"
TZ5    PRTPD395041        echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
TZ5    PRTPD395042        #
TZ5    PRTPD395043        # VARIABLES QUE SERVIRAN PARA LOS PROGRAMAS IMPLICADOS SUBSECUENTES
TZ5    PRTPD395044        export PASO=PRTPD395;  export EJECUTA=PRTPD395
TZ5    PRTPD395045        #
TZ5    PRTPD395046            # PARAMETROS DEL PASO
TZ5    PRTPD395047            export CENT='>6'
TZ5    PRTPD395048            export TEN='>7'
TZ5    PRTPD395049            export FECH='>8'
TZ5    PRTPD395050            export IOPER='>14'
TZ5    PRTPD395051            export AT='/V'
TZ5    PRTPD395052            export NBVOLTRB='3390'
TZ5    PRTPD395053            export NBLDB2='SYS2.DB2P.SDSNLOAD'
TZ5    PRTPD395054            export SALX='X'
TZ5    PRTPD395055            #
TZ5    PRTPD395056            export PTUT0100_SYSUT1="${TMP}/${JID}.PTUT0100_SYSUT1"
TZ5    PRTPD395057        echo80 "
TZ5    PRTPD395058        >9
TZ5    PRTPD395059        " ^ ${TMP}/${JID}.PTUT0100_SYSUT1
TZ5    PRTPD395059        time ${PROCLIB}/${EJECUTA}
```

Figura 4-17 Ejemplo de Layout de disparo

Tal como se aprecia, un layout es un cuasi shell, donde en la primer columna se identifica el tipo de tabla con la que estamos tratando, la segunda columna identifica a cada uno de los registros numerados secuencialmente como único dentro del layout ya que sus primeras 8 posiciones hacen referencia al "JOB" al que pertenecen y las siguientes 3 posiciones son un número consecutivo con el cual evitamos duplicados dentro de la base de datos y de este modo el programa disparador de procesos conoce el orden en que tiene que leerlo y escribirlo. Esto nos obliga a que por cada línea que se traduzca esta numeración sea alterada ya que el número de líneas inevitablemente variará con respecto al original.

En la figura 4-17 se han encerrado en una cuadrícula punteada las secciones del layout en donde se encuentran los simbólicos que el programa disparador se encargará de resolver de modo que el resultado final será un archivo del JOB escrito en la ruta de la biblioteca "SCHEDULER" del usuario en espera de ser ejecutado.

El "JOB" que resultaría del layout anterior quedaría de la manera que a continuación se muestra:

```
#!/bin/ksh
#
# VARIABLE QUE SERVIRA PARA CREACION DE ARCHIVOS TEMPORALES
export JID=PRTPD395.$($)
#
# VARIABLE CON EL CODIGO DE RETORNO DE LOS PASOS
export RC=0
#
# VARIABLE CON EL PASO DE REINICIO
export REINICIO=0
#
# REDIRECCIONAMIENTO DE STDOUT Y STDERR
# exec >> $LOG/$JID
# exec 2 >> $LOG/$JID
#
# IDENTIFICACION DEL JOB
echo "** JOB=PRTPD395"
echo "** IDJOB=(009)"
echo "** COMENTARIO='D.R.AUT R-P'"
echo "** NUMERO=$$"
echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
#
# PATH DE LOS PROCS A EJECUTAR
export PROCLIB=/PRRT00/DP/PROCLIB
#
#/*JOBPARM L=99999
#//*----------
#//*            SISTEMA: P R O C E S A R .
#//*             MODULO: R E T I R O S .
#//*         DISPARADOR: PRTPD395.
#//*        DESCRIPCION: D.R. SOL.DE VERIF. RESOL. IMSS SAR
#//*                DEP: PTDT050.
#//*         FRECUENCIA: DIARIO (EN LOS DIAS PRE-ESTABLECIDO
#//*AUTOR: SOFTTEK(ING.JULIO CESAR GHENO FLORES)
#//*         FECH ULT ACT: 14/MAYO/1997.
#//*      OBSERVACIONES:
#//*-----
# IDENTIFICACION DEL PASO
echo "** PASO=PRTPD395"
echo "** EJECUTA=PRTPD395"
echo "** INICIO=\"`date '+%d/%m/%G %T'`"\"
#
# VARIABLES QUE SERVIRAN PARA LOS PROGRAMAS IMPLICADOS SUBSECUENTES
export PASO=PRTPD395;  export EJECUTA=PRTPD395
#
# PARAMETROS DEL PASO
export CENT='516'
export TEN='01'
export FECH='030317'
export IOPER='24PE'
export AT='/V'
export NBVOLTRB='3390'
export NBLDB2='SYS2.DB2P.SDSNLOAD'
export SALX='X'
#
export PTUT0100_SYSUT1="${TMP}/${JID}.PTUT0100_SYSUT1"
echo80 "
C0104015162420030317RE
" ${TMP}/${JID}.PTUT0100_SYSUT1
time ${PROCLIB}/${EJECUTA}
```

Figura 4-18 JOB resultante de un layout de disparo

Con base a lo anterior podemos decir que la forma en que los layout's de disparo deben de ser traducidos, será la misma que se emplea para un "JOB", excepto que en el área de parámetros simbólicos no se aplicará el algoritmo descrito anteriormente ya que estos serán tratados como si fuera de una sencilla declaración de variables.

4.4. Programas COBOL y BASES DE DATOS

4.4.1. COBOL

Debido a la utilización de un COBOL neutro, no se aprovecharon las funciones y rutinas que mejoran la versión de COBOL incorporada en el MVS, lo único que se tuvo que realizar a estos, fue su copiado y compilación en el ambiente desarrollado en el sistema SUN.

4.4.2. BASES DE DATOS

El personal encargado de administrar la base de datos determinó que era más ventajoso para el downsizing establecer la versión de DB2 UDB de base de datos (De los productos con los que IBM incursiona en el mercado de sistemas abiertos), por lo que prácticamente se clonó el sistema de base de datos de MVS a UNIX, siendo esto muy beneficioso para la migración de los programas COBOL.

CONCLUSIONES.

CONCLUSIONES.

Es indudable que las corrientes de globalización están afectando la tendencia tecnológica así como a los sistema abiertos, es decir, los que son compatibles en las formas de procesar la información, son los que comienzan a predominar en el mercado, y es precisamente la necesidad de los consumidores lo que impulsa una competencia comercial de infinidad de estos productos y sistemas.

Es ciertamente, ésta competencia en el mercado de productos y servicios la que obliga a plantear un esquema de muy alta eficiencia con costos cada vez más bajos, es por ello que las empresas se ven obligadas a buscar alternativas y soluciones que garanticen el acceso veloz, eficiente y con capacidad de crecimiento y evolución, para proporcionar la información que apoye en la toma de decisiones que contribuyan al éxito de la empresa.

Impulsados por esta necesidad de mejora y actualización existente en el mundo de la información, ProceSAR, la empresa concesionada por el Gobierno Federal para operar la base de datos nacional del sistema de ahorro para el retiro, emprendió un ambicioso proyecto de Downsizing. Gracias a la adecuada coordinación y planeación de los diferentes grupos de trabajo, fue posible alcanzar los objetivos planteados y logar que los sistemas que se desarrollaron para una plataforma tecnológica cerrada operen en una plataforma tecnológica abierta.

En concreto el Downsizing consiste en transportar todo el sistema desarrollado originalmente en una plataforma IBM-MVS, que es un ambiente propietario, a un sistema SUN con UNIX Solaris, el cual es un sistema abierto e incompatible en cuanto a trasportabilidad. Se aplicó el esfuerzo humano para lograr esto de una manera relativamente rápida y trasparente para la operación, donde un equipo de especialistas en diferentes partes del sistema contribuyó para el éxito del proyecto.

Dentro del proyecto de Downzising, la tarea que nos fue encomendada representó para el proyecto el punto más importante y delicado, ya que al adecuar correctamente los procesos operativos, se reflejaría el esfuerzo y el trabajo de toda el área de sistemas de la empresa, por lo cual de alguna manera se tenía que asegurar la transparencia durante la traducción de los procesos productivos para garantizar que el producto final cumpliera con los requisitos establecidos previamente con nuestros clientes. Por esto a pesar de que los procesos batch para el sistema operativo UNIX pueden ser programados de una manera mucho más sencilla que lo que se mostró en esta tesis, preferimos adecuarnos (lo más posible) a los actuales esquemas de desarrollo, pruebas y ejecuciones productivas, con lo cual se logró reducir el impacto del cambio de manera muy significativa, ya que para nuestros usuarios internos de las áreas de operaciones, producción y auditoria, el cambio es imperceptible con tiempo y costos reducidos. Para los desarrolladores lo único que es obligatorio es capacitarlos en el uso y manejo del sistema operativo UNIX lo cual será indispensable para los subsecuentes desarrollos.

Uno de los aspectos más importante a destacar y que básicamente permitió automatizar la migración de componentes, es que desde un principio se establecieron estándares de desarrollo estructurados, con lo cual no solo se obtienen las ventajas inherentes de desarrollo y mantenimiento de los sistemas, sino que, como ya se mencionó, se establece una base sólida para que mediante un proceso automático, se puedan traducir los procesos desarrollados para un ambiente propietario cerrado, a un ambiente abierto, sin haber tenido que recurrir a un proyecto maratónico y/o a largo plazo, donde se hubieran visto involucrados la mayor parte de los recursos de la empresa.

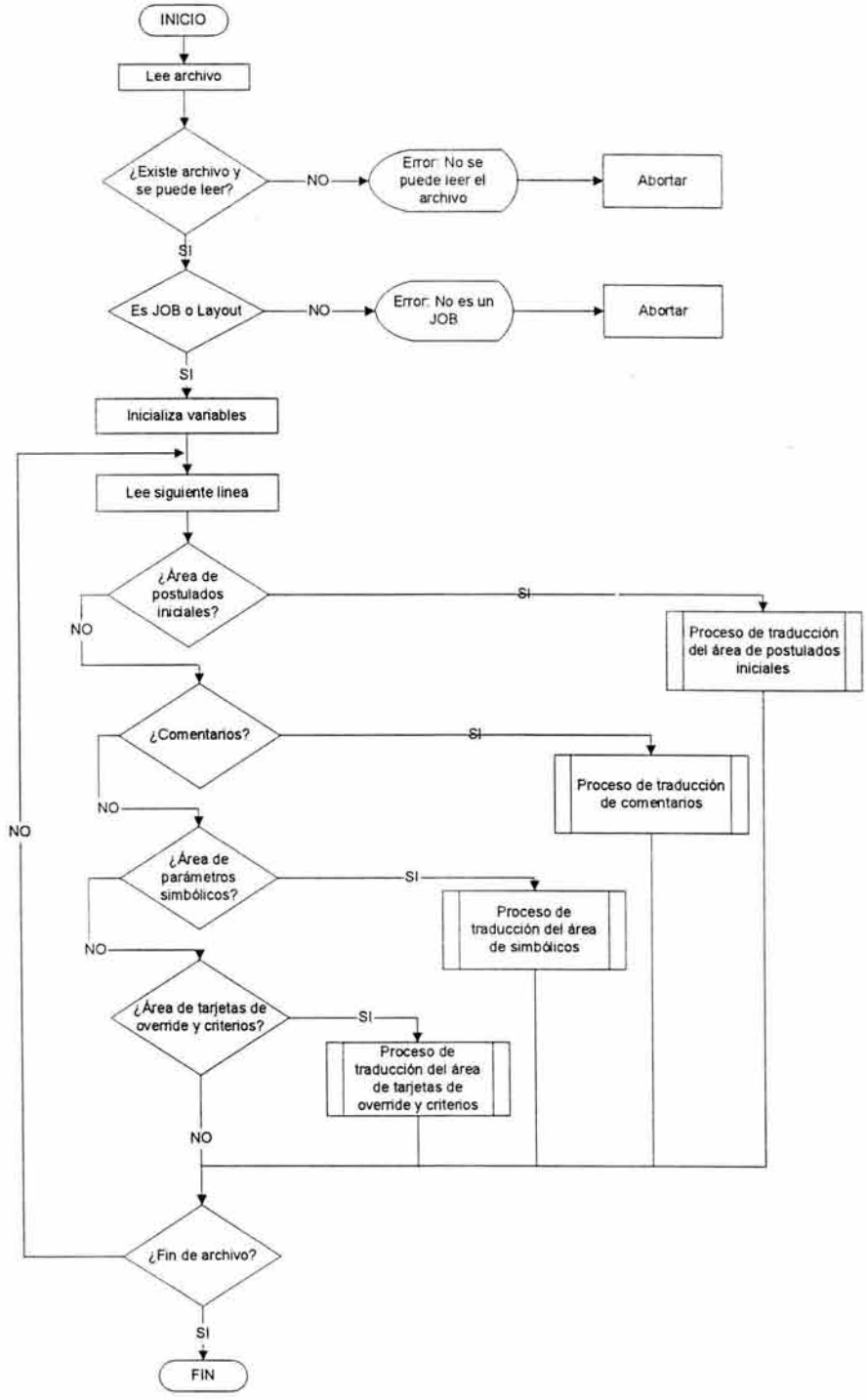
Con el presente trabajos comprobamos que si se siguen estándares de desarrollo fijos y estructurados en el desarrollo de sistemas, independientemente de la naturaleza operativa de este, es posible transportar las aplicaciones de una

manera automatizada de manera automatizada, independientemente de las herramientas de desarrollo utilizadas.

Ya sea siguiendo nuestros ejemplos, o adaptándolos según sea el caso, el presente trabajo será de gran utilidad para que un líder de proyectos que se encuentre con la necesidad de migrar procesos desarrollados para una plataforma tecnológica propietaria, o sistema cerrado, a una plataforma de sistema abierto, logre se generen sistemas inteligentes que cumplan con mejores estándares de desarrollo sin que este tipo de proyecto represente un obstáculo mayor, y demanden tantos recursos humanos y económicos.

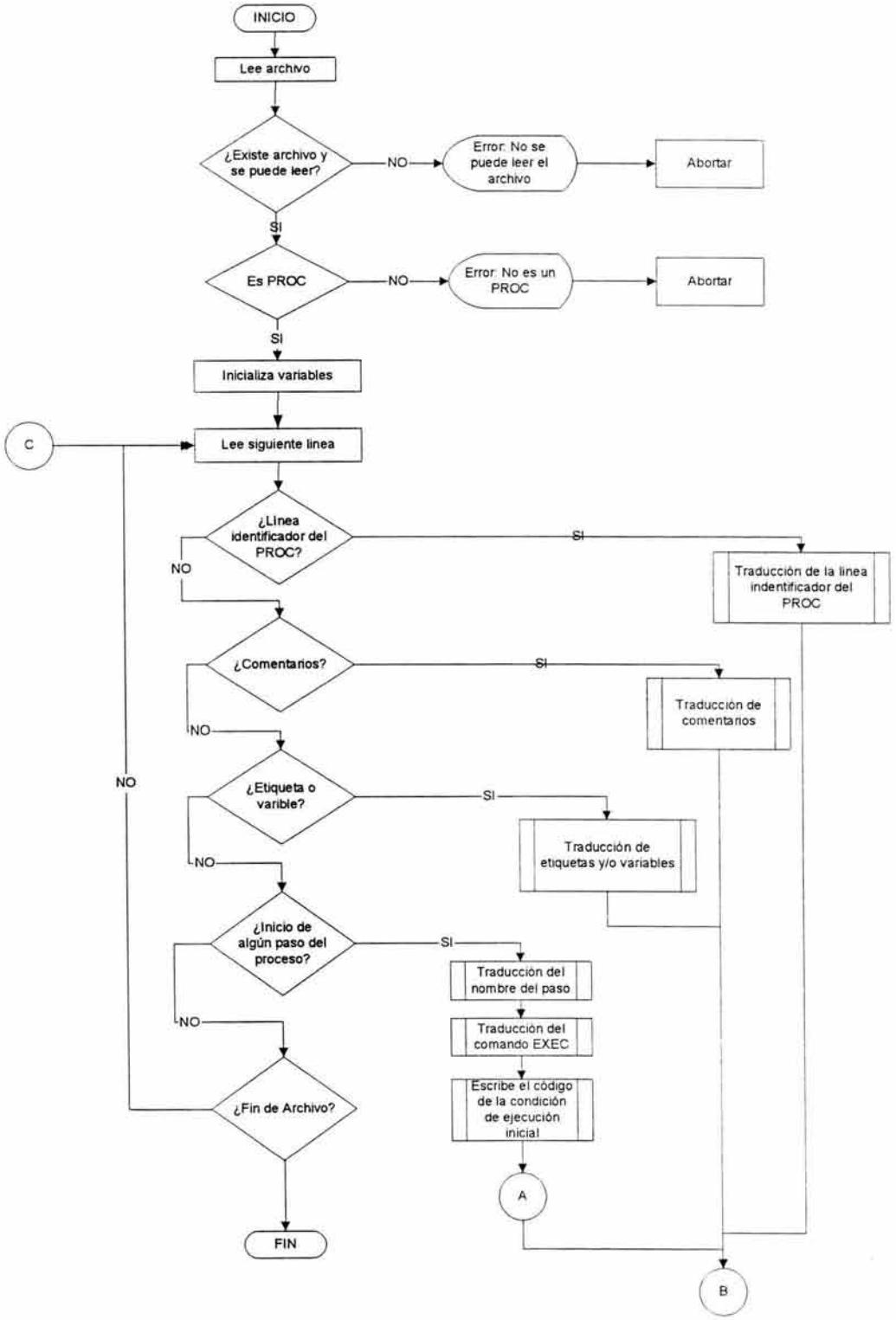
Anexo 1.

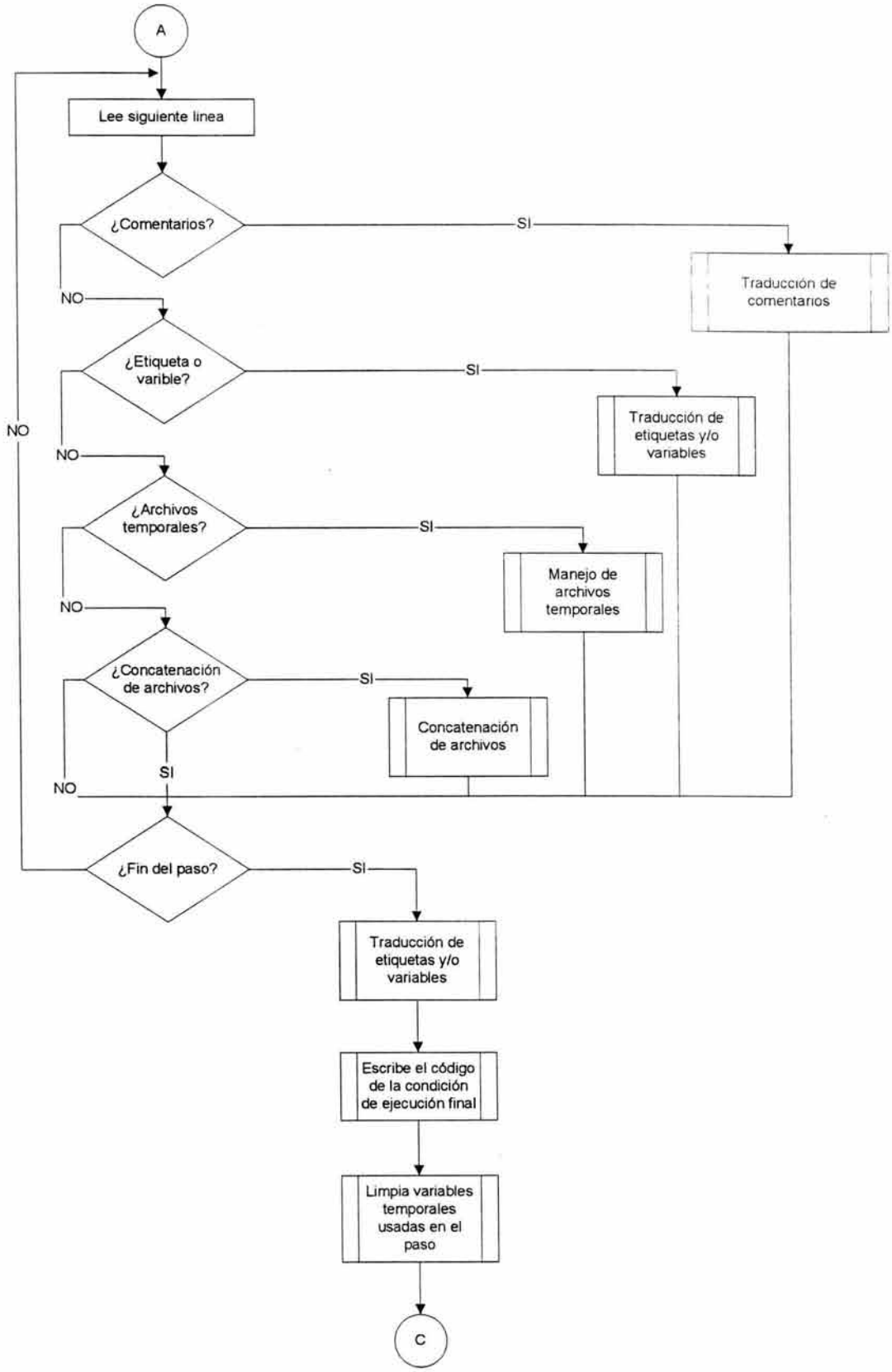
Anexo 1. Diagrama de flujo para la traducción de JOB's y Layout's de disparo



ANEXO 2.

Anexo 2. Diagrama de flujo para la traducción de PROC's




A
Lee siguiente linea
¿Comentarios?
SI
Traducción de comentarios
NO
¿Etiqueta o varible?
SI
Traducción de etiquetas y/o variables
NO
¿Archivos temporales?
SI
Manejo de archivos temporales
NO
¿Concatenación de archivos?
SI
Concatenación de archivos
SI
NO
NO
¿Fin del paso?
SI
Traducción de etiquetas y/o variables
Escribe el código de la condición de ejecución final
Limpia variables temporales usadas en el paso
C

ANEXO 3.

Anexo 3. Manual de usuario

Para realizar la conversión del proceso batch es necesario seguir las siguientes instrucciones:

1. Realizar la transferencia de archivos contenidos en las actuales bibliotecas de los procedimientos y de los job's hacia la nueva plataforma para lo cual previamente ya se debieron haber cubierto todos los requisitos de conectividad y seguridad que ambas plataformas exigen. Se recomienda que dichos archivos sean alojados en un sistema de directorios que por si solo identifique el tipo de elemento del que se trate. Por ejemplo, para transportar los elementos Job's del módulo de retiros se tendrá que crear en el sistema UNIX la biblioteca /PBRT00/MVS/JOBLIB y en esta alojar o copiar todos los elementos de la biblioteca PBRT00.DP.JOBLIB del sistema MVS.

2. Copiar el programa que realiza la conversión de procedimientos en la biblioteca donde se encuentran los archivos a convertir. Por ejemplo, para los elementos de la biblioteca /PBRT00/DP/PROCLIB se deberá contar con el programa convertidor que para este caso su ruta absoluta será /PBRT00/DP/PROCLIB/convierte.

3. En este punto ya estaremos listos para iniciar con la conversión de procesos, para lo cual se deberá invocar el programa "convierte" y pasarle una serie de parámetros que a continuación se muestran en la figura A3-1.

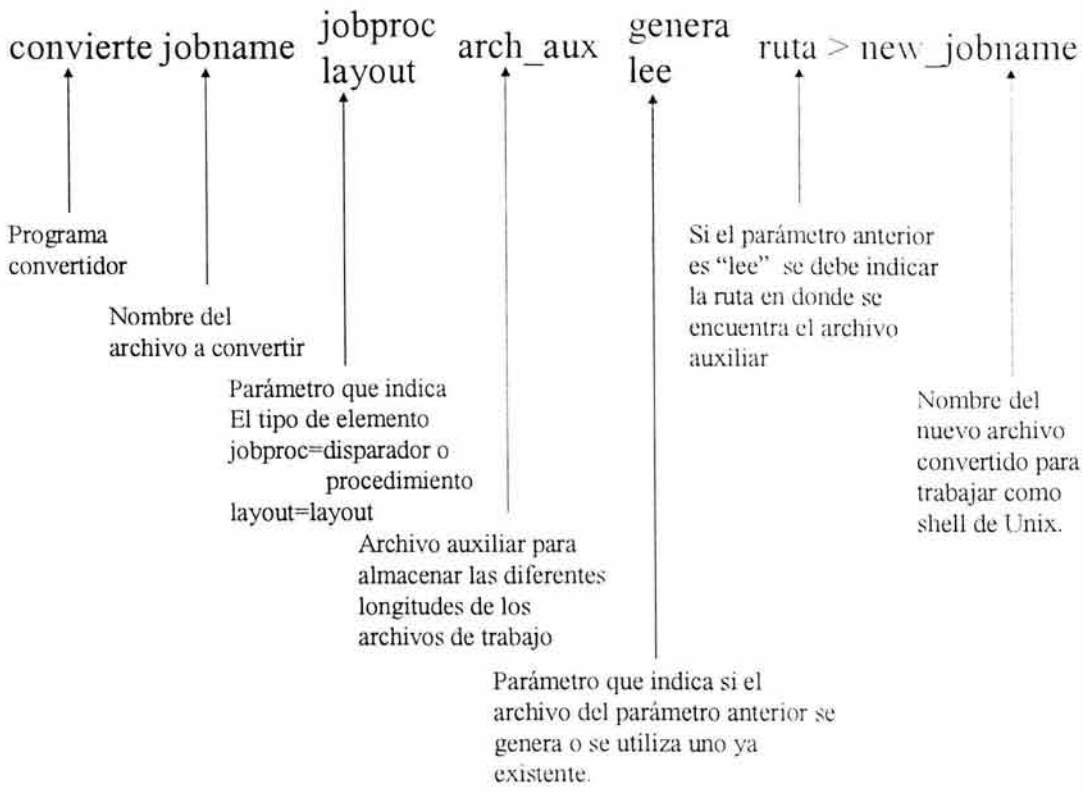


Figura A3- 1 Ejecución del convertidor de aplicaciones

Como se puede observar en la figura anterior, el programa convertidor va acompañado de una serie de parámetros de los cuales únicamente el 5° parámetro depende de que si en el anterior se indicó que el archivo auxiliar será utilizado como de lectura, en caso contrario, este no es necesario ya que si el archivo no existe se genera en la actual ruta o si ya existe entonces se agregan los datos que se requieran.

Al final de la instrucción de ejecución se utiliza el redireccionamiento de la salida estándar para guardar el resultado en el archivo que se convertirá en el Shell que sustituirá los anteriores procesos.

A continuación se proporcionan algunos ejemplos de para la conversión de elementos:

Para convertir el procedimiento PREPD580 se podrá hacer con la siguiente instrucción:

```
convierte PREPD580 jobproc longitudes genera /PBRE00/DP/UTIL
>/PBRE00/DP/PROCLIB/PREPD580
```

Ejemplo E-1 Ejemplo de conversión del proceso PREPD580

En el ejemplo anterior se está asumiendo que ya cumplimos con los 3 puntos mencionados al principio y que el directorio /PBRT00/DP/UTIL ya ha sido generado con anticipación para que sea depositado el archivo "longitudes". El resultado de este comando nos dará como resultado el archivo /PBRE00/DP/PROCLIB/PREPD580 listo para ser ejecutado con el sistema operativo UNIX.

La siguiente instrucción se utiliza para la conversión de un layout:

```
convierte TZ588D01 jobproc longitudes genera /PBRE00/DP/UTIL
>/PBRE00/DP/PROCLIB/TZ588D01
```

Ejemplo E-1 Ejemplo de conversión del proceso PREPD580

Los ejemplos anteriores muestran claramente la manera en que podemos realizar la conversión de cada uno de los elementos, pero resulta poco practico ejecutar una instrucción por cada elemento que debe ser convertido. Para evitar tener que hacer esto se generó un sencillo shell que ejecutará el sistema convertidor tantas veces como archivos encuentre en el directorio en cuestión. La figura A3-2 muestra este proceso.

```
for i in /PBRE00/MVS/PROCLIB/*
do
   convierte $i jobproc archivo genera /PBRE00/DP/UTIL  > temp
   rc=$?
   if [ $rc -ne 0 ]
   then
      echo "$i $rc" >> /PBRE00/DP/UTIL/errores
      cp $i /PBRE00/DP/PROCLIB /$i
   else
      cp temp /PBRE00/DP/PROCLIB /$i
   fi
```

Figura A3- 2 Shell para la conversión de procesos

El algoritmo que se propone para la generación de este shell consta de un ciclo que va a leer archivo por archivo en el directorio indicado y por cada uno va a ejecutar el programa convertidor, al final de cada ejecución se captura el código retorno y si este indica que sucedió algún error, entonces, tanto el nombre del elemento como el código de error son escritos en el archivo errores, en caso contrario el resultado es escrito en su correspondiente archivo convertido.

A continuación se muestra una lista de los diferentes tipos de error con los que nos podemos encontrar y su correspondiente descripción.

Código de error	Descripción
1	Falta encontrar el valor de la variable LREC para una tarjeta de SORT
2	Falta el cierre de algún paréntesis

3	Falta el cierre de alguna variable
4	Continuación esperada no recibida
5	No se puede abrir el archivo o no se tienen permisos de lectura
6	Existen varias posibilidades para la longitud de registro
7	Falta encontrar una LREC
8	No se puede abrir el archivo del argumento 5
9	N/A
10	No se que hacer con esta DISP
11	No se que hacer con esta instrucción
12	No se puede abrir el archivo del argumento 3
13	No hay ningun criterio abierto
14	No esta definido si es JOBPROC o LAYOUT en el argumento 2
15	No esta definido si inserta o lee el archivo de longitudes

Tabla 4-21 Traducción de archivos temporales

ANEXO 4.

Anexo 4. PROGRAMAS Y UTILERÍAS.

Utilería: SORT

Objetivo: Realizar el ordenamiento de archivos planos de acuerdo a uno o varios campos.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
SORTIN	Contiene el nombre del archivo o los archivos a ordenar.
SORTOUT	Indica el archivo en el que se depositarán los resultados del proceso de ordenamiento
SORTIN_LREC	Es la longitud de registro del archivo o archivos del que se encuentran en la variable "SORTIN" Esta longitud deberá ser tomada del procedimiento original de la variable "LREC"
SYSIN	Contiene el nombre de la tarjeta de sort en donde se indican las condiciones y características en la que se ejecutara el proceso de ordenamiento.

Utilería: DMBATCH

Objetivo: Realizar la conexión al proceso de transferencia de archivos.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
DMNETMAP	Contiene la ruta completa del archivo NETMAP.
DMPUBLIB	Indica la biblioteca donde se encuentran las utilerías del sistema.
DMMSGFIL	Contiene la ruta completa del archivo V3R2M0.MSG.
SYSIN	Contiene el nombre de la tarjeta de transferencia de archivos en donde se indican las condiciones y características en la que se ejecutara dicho proceso.

Utilería: ICEGENER e IBGENER

Objetivo: Efectúa la copia de un archivo a otro y en caso de que el archivo destino sea de tipo "File-transfer" realiza la transferencia de información a un dominio diferente en el cual los clientes (afores e institutos) pueden acceder para tomar su información de respuesta a sus procesos.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
SYSUT1	Archivo origen
SYSUT2	Archivo destino.
SYSIN	Archivo temporal que se utiliza como enlace en caso de requerir del proceso de transferencia de información.

Utilería: PARAMFILE

Objetivo: Escribe en un archivo de trabajo lo valores proporcionados por la variable PARM.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
PARM	Contiene los parámetros que serán almacenados en el archivo de trabajo.
FILEOUT	Archivo de trabajo que contendrá los parámetros indicados en la variable PARM..

Utilería: IEAMAXCO.

Objetivo: Ejecución de las tarjetas de borrado de archivos.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
AMSDUMP	Archivo temporal de trabajo.
SYSIN	Contiene el nombre de la tarjeta de borrado de archivos a ser ejecutada.

Utilería: IEFBR14.

Objetivo: Para el actual sistema realiza el borrado y alojamiento de archivos. Para el nuevo sistema no tendrá ninguna función ya que tales tareas se realizarán mediante comandos de UNIX.

Tabla de variables requeridas: No requiere de variables.

Utilería: IDCAMS.

Objetivo: El manejo de archivos secuenciales y archivos VSAM (Virtual Storage Access Method).

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
SYSUT1 **DD01** **AMSDUMP**	Archivos de trabajo.
SYSUT2 **DD02**	Archivos de trabajo
SYSUT1_LREC	Indica la longitud del archivo con la etiqueta SYSUT1.
SYSUT2_LREC	Indica la longitud del archivo con la etiqueta SYSUT2.
SYSIN	Contiene el nombre de la tarjeta con los comandos a ejecutar.

Utilería: PTLDRIVM.

Objetivo: Carga y descarga de información de la base de datos.

Tabla de variables requeridas:

VARIABLE	DESCRIPCIÓN
PTILIB **PTIPARM** **PTIXMSG**	Bibliotecas del manejador de base de datos (DB2) requeridas para realizar la carga o descarga de información
SYSCTL01	Archivo de trabajo para el manejador de base de datos.
SYSREC01	En caso de que se trate de una operación de carga de información indica el archivo que contiene la información a cargar y en caso de ser una descarga de información en este archivo se depositarán los resultados de dicha descarga.
SYSIN	Contiene el nombre de la tarjeta con los comandos que le indican al manejador de base de datos la manera en realizará las operaciones solicitadas.

BIBLIOGRAFÍA

1. *Stallings, William*, "Sistemas operativos". México: Limusa, 1995.

2. *Cearra, Luis José*, "Sistemas abiertos". Madrid: Fundación General de la Universidad Politénica de Madrid, 1998.

3. *Tare, Ramkrishna S.* "Procesamiento de datos en unix: con informixsqsl, esql/ C, c-isam y turbo". México: McGraw-Hill, 1990.

4. *Toporek, Jerome D*, "Ibm mvs/ Tso clists for running bmdp interactively". Los Ángeles, California: Bmdp Statistical Software, 1985.

5. *Sun educational service*, "Ultra Enterprise 10000. Administration ES-400". Sun Microsystems, Inc. 1998.

6. *Gary DeWard Brown*, "System 370/390 JCL". Canada: Jonh Wiley & Sons,Inc. 1991.

7. *Deitel Harvey M.*, "Introducción a los sistemas operativos". Wilmington: Addison Wesley Iberoamericana ,1993.

8. http://www.sunes/tecnologia/sistemas/servidoresent.shtml

9. http://det.bi.ehu.es/~isa/asignaturas/tema7/pagina1.html

10. http://www.map.es/sci/slice/tlwcpu.html

11. http://www.monografias.com/trabajo5/sistab/sistab2.stml

12. http://www.sun.com/products-n-solutions/hardware/docs

13. *Robert H. Johnson*, "MVS".

14. *Tim Martyn*, "DB2".

15. *Jay Ranade, Mukesh Sehgal*, "DB2 concepts programming and design".