PROGRAMACION DE SISTEMAS

CAPITULO I: SUBSISTEMAS DE UN SSOO

- **Principios Generales**
- **Sistemas locales y en red**
- **Integración de plataformas (POSIX).**
- **Herramientas de ayuda al desarrollo.**
    - **Make**
    - **DLL / ar**
    - **Compilador**
    - **Control de versiones.**
    - **Depurador simbólico.**

1.PRINCIPIOS GENERALES

El Unix system5 Release4 proviene de cuatro ramas:

− *AT&T.* Con su sistema SVR3

−*Microsoft.* Realmente Microsoft dejó este sistema para pasar al WIN NT.

−*Sun Microsystems.*

−*Universidad de Berkeley.* Con su sistema BSD4.2

Las principales características del UNIX S5R4 son:

- Procesamiento en Tiempo Real
- Incorporación de distintos tipos de planificador de tareas.
- Mejora de la Memoria Virtual
- Mejora en la interfaz del manejo de interrupciones software
- Asignación dinámica de memoria para los procesos del sistema
- Mayor número de archivos procesables simultáneamente por cada proceso. En el SVR3 la cantidad máxima estaba limitada a 20 (3 del sistema, 17 libres). En el SVR4 el manejo de archivos se realiza de modo dinámico, de modo que no hay limitación de archivos procesables simultaneamente.
- Aparece el Sistema Virtual de Archivos VFS (Virtual File System).
- Interfaz estándar de funciones y estructuras para el diseño de los controladores de dispositivo (drivers).
- Incorpora la posibilidad de controlar varias CPUs con la misma prioridad (procesamiento simetrico).

El Subsistema de SVR4 esta formado por tres componentes:

− Gestión de Archivos

− Gestión de Procesos

− Gestión de la Entrada/Salida

SUBSISTEMA DE PROCESOS

Es un conjunto de rutinas del sistema que permite la gestión de varias actividades. Permite varias funcionalidades:

- *Gestión de la CPU.* Se encarga de sacar y meter los procesos en la CPU.
- Debe permitir a los programas de usuario el acceso a las primitivas del Kernel (system calls).
- Realizar las operaciones de *swap−in* y *swap−out*. Esto es lo que permite sacar de la CPU tareas que han consumido su tiempo de CPU o que necesitan realizar operaciones de I/O. Debe poder guardar esta tarea en el área de swapping y cuando llegue el momento de meterla en CPU, se encargará de planificar la entrada.
- La **incorporación** más importante es la *Técnica de Intercomunicación entre Procesos:*

− Tuberías sin nombre.

− Tuberías con nombre.

− Señales 8Interrupciones Software)

− Entornos (Tablas de Variables y Datos de Procesos hijos a padres)

− Técnica de IPCS

· Segmentos de Memoria Compartida (shm)

· Colas de Tareas (msg)

· Semaforos (sem)

− Paso del Testigo ("*Passing de Buck")*.

SUBSISTEMA DE ARCHIVOS

El Subsistema de Archivos es un conjunto de áreas administrativas de disco, de bloque de almacenamiento de disco y de algoritmos que permiten la rápida y comoda gestión de los archivos de un usuario. Las funciones claras son:

- Mantener las estructuras necesarias para el tratamiento y acceso a los archivos de disco.
- Mantener los buffers necesarios para agilizar los accesos a disco(vnodes y vfs).
- Ejecutar las funciones necesarias para el manejo de archivos (open, ...).

SUBSISTEMA DE ENTRADA/SALIDA

Es el responsable final de la gestión de todos los requerimientos de I/O realizados por los usuarios. Funcionalidades básicas:

- Gestión en modo carácter o bloque de los dispositivos físicos.
- Proporcionar la estructura de buffers necesaria para realizar estas operaciones de I/O
- Posibilidad de definir e incorporar nuevos controladores de dispositivo al sistema
- Gestionar las interrupciones necesarias para acceder a estos dispositivos perifericos.

2. SISTEMAS LOCALES EN RED

Hasta el que apareció el SVR4 el UNIX era un SSOO local y solo podía comunicarse con otros sistemas con UUCP (Aplicaciones de comunicación de nivel OSI 7). Esto ya no se usa porque se ha definido, en SVR4, unos protocolos TCP/IP.

Existen un conjunto de funciones TLI (Transport Level Interface) que estan orientados al desarrollo de aplicaciones. Permiten el establecimiento de comunicación entre varias maquinas.

Esto llevó a crear un sistema XDR que permite la transmisión de datos con formato (Reales, Doble precisión, Enteros, ...). Las funciones RPC fueron diseñadas en SUN Microsystems. Son primitivas de nivel OSI 7 que permiten además desarrollar aplicaciones que están en dos maquinas.

3.INTEGRACION EN PLATAFORMAS (POSIX)

Existe un comité que es el POSIX, que establece el conjunto de funcionalidades definidas como estándar para que en cualquier plataforma sea posible el desarrollo de aplicaciones software con independencia de las plataformas en que se han desarrollado.

4.HERRAMIENTAS DE AYUDA AL DESARROLLO

CC es el compilador que vamos a utilizar en UNIX. Los archivos *.c* van a ser los archivos fuente, y los archivos *.o* son los objeto. Por defecto al compilar con CC se crea el ejecutable *a.out.*

CC a1.c a2.c a3.c ... ak.c ==> a.out

Se pueden mezclar al compilar archivos objeto y archivos fuente. Esto puede ser útil cuando un fuente falla y los demás no; variamos el que falla y luego lo compilamos con los objeto que no han fallado sus fuentes y así ganamos tiempo de compilación.

CC a1.c a2.o a3.o ... ak.o ==> a.out

Para variar el nombre del ejecutable que vamos a producir debemos utilizar la opción **−o**

CC −o ejec1 a1.c ... ak.c

Para generar archivos objeto (extensión .o) y no generar el ejecutable deberemos utilizar la opción **−c.** CC −c a1.c a2.c ... ak.c . En este caso no debemos poner el archivo que contiene el main para que se pueda utilizar estos objeto en otros programas.

PREPROCESADOR ENSAMBLADOR

Fuente CC −P fuente.c fuente.i CC −S fuente.{c,i}

Fuente.s

EDITOR DE ENLACES COMPILADOR

CC fuente.{c,i,s,o} Fuente.o CC −C fuente.{c,i,s}

a.out

PROCESO DE COMPILACION AUTOMATICA (MAKE)

Si tenemos cientos de módulos utilizar CC puede llegar a ser muy engorroso ya que si falla alguno, deberemos volver a escribir la línea entera que puede ser muy extensa. Para evitar esto disponemos de la técnica de los make (makefiles).Vamos a verlo con un ejemplo. Suponemos:

x.c −− # include "def.h" y.c .. # include "def.h" z.c

$vi makefile

Makefile

ejec: x.o y.o z.o

CC x.o y.o z.o −o ejec1

x.o: x.c def.h

CC −c x.c

y.o: y.c def.h

CC −c y.c

z.o: z.c

CC −c z.c

^D

Si escribo $make y en el directorio existe makefile(o Makefile) ejecuta lo que encuentre dentro. También se puede llamar de otra manera y luego ejecutarlo con $make *nombre*.

DEPURADOR SIMBOLICO. SDB

Es un depurador muy primitivo que no necesita entorno gráfico. Si existe entorno gráfico se puede usar DBX.

$lint *fuente.c* Depurador léxico, sintáctico y semántico de programas C fuente. Simula la compilación y te va informando de lo que has hecho.

$cflow *fuente.c* Permite obtener un diagrama de flujo simbólico del programa que le hemos dado como argumento.

$cxref *fuente.c* Permite obtener la lista de referencias externas que utiliza nuestro programa.

$..... *fuente.c* Mete sangrías en el fuente.

BIBLIOTECAS

Las bibliotecas pueden ser de enlace estático o enlace dinámico.

ENLACE ESTATICO

Una biblioteca de enlace estático es un contenedor de varios archivos.

$ar Crea un archivo contenedor de otros archivos.

$ar r Crea una biblioteca de enlace estático.

$ar r $HOME/lib/**lib**complejo**.a** mult.o add.o ==> Crea una biblioteca de enlace estático, llamada libcomplejo.a, que contiene los archivos mult.o y add.o. El nombre de la librería tiene dos restricciones importantes:

- Siempre debe comenzar por lib.
- Su extensión ha de ser .a

El sistema contesta con el siguiente mensaje:

ar: creating /home/usr/lib/libcomplejo.a

Si no existe la librería la crea, y si existe añade los nuevos archivos a la librería.

$ar t lista el contenido de la libreria

$ar d *biblioteca módulo* Borra el módulo de la librería.

$ar x *biblioteca módulo* Extrae el módulo de la librería al directorio actual como un archivo.

El comando **ar no utiliza el signo menos** para las sus modificadores.

Para generar un ejecutable cuyas referencias externas son a una librería estática se utiliza: $CC **−L**/home/usr/lib complejo.c **−l**complejo −o ejec1

La opción **−L** da el directorio donde se encuentra la libreria

La opción **−l** da el nombre de la libreria(**−l**complejo indica: *lib*complejo.*a*)

ENLACE DINÁMICO (DLL) (MODULOS COMPARTIDOS)

No guarda en el ejecutable que creamos con CC el código del módulo de la librería, sino una referencia a la DLL.

Una gran ventaja de las DLL es que puedo variar el código del programa sin cambiar el contenido de este, sino cambiando el contenido de la DLL. Para cambiar el contenido o añadir módulos en la DLL tengo que crear otra.

Las DLL se cargan en memoria y pueden ser usadas por varios programas que las referencian. Esta es la gran ventaja de las DLL.

En el directorio /usr/lib se encuentra el código de las funciones UNIX, salvo las matemáticas, y están en su versión estática (**libc.a**), y en su versión dinámica (**libc.so**).

$ldb *argumento* Si se le da como argumento a este programa un ejecutable, te devuelve como tiene solucionada este programa su edición de enlaces.Devuelve:

dynamic linker:ejec:file loaded:/usr/lib/libc.so

$CC −G ==> Crea una DLL (Dinamic Linked Library)

$CC −G −o /home/usr/lib/libcomplejo.so mult.o add.o ==> Crea una DLL que contiene los archivos mult.o y add.o

$file *archivo_biblioteca* Se utiliza para saber el contenido de la DLL.

Variables de Entorno que intervienen en el Proceso de Creación y Uso que se Linkeditan con DLLs: LD_RUN_PATH, LD_LIBRARY_PATH

- *LD_RUN_PATH:* variable de entorno que permite incluir una lista de directorios que se incrustarán dentro del ejecutable que hace uso de bibliotecas DLL incluidas en su ruta.
- *LD_LIBRARY_PATH:* contiene una lista de directorios que se van a utilizar para obtener referencias externas estáticas.

$CC −L/home/lib complejo.c −lcomplejo −o ejecutable_dinamico

Para que busque una librería estática (*.a*) tengo que incluir uno de los siguientes modificadores:

- ***−dn*** *=>Significa Dinamyc NO*
- ***−Bstatic***

Si decido modificar la DLL:

$CC −G −o libcomplejo.so mult.o ==> Ahora la librería solo contendrá el archivo mult.o

Si ahora intento compilar el fuente que referencia a las dos módulos dará un error, porque la librería ahora solo contiene un modulo (mult.o).

$CC complejo.c −lcomplejo −o ejec2 ==>Dará el siguiente error

undefined first referenced

symbol in file

add complejo.o

ld:ejec2:fatal error:symbol referencig errors. No output written to ejer2.

$ldd Permite examinar las bibliotecas externas de los programas con el fin de adecuarlas al enlace dinámico.

$ldd −r complejo

dynamic linker:complejo:file loaded:/home/usr/lib/libcomplejo.so

dynamic linker:complejo:file loaded:/home/usr/lib/libc.so

dynamic linker:complejo:relocation error:symbol not found:add

Para compilar con referencias estáticas:

$CC −dn a1.c ... ak.c −lm −o ejec3

Para referenciar desde un módulo de una DLL a otro módulo de otra DLL:

$CC −G −o libmia.so −Ldir f1.o f2.o f3.o −lcomp

- **Averiguar que realiza lo siguiente:**

$CC −L/home/milib main.c f1.c f2.c −Bstatic −lmia f3.c −Bdynamic −lcomp

CAPITULO II: SUBSISTEMA DE ARCHIVOS

- **Principios Generales**
- **Tipos de Subsistemas de Archivos**
  - **S5**
  - **Ufs**
  - **Archivos y nodos de índices**
  - **Estructuras de Datos**
  - **Algoritmos de asignación/desasignación de espacio**
  - **Funciones dependientes del sistema de archivos**
- **Sistema Virtual de Archivos VFS**
  - **Estructuras de Datos**
  - **Algoritmos**
  - **Funciones Independientes del Sistema de Archivos.**
- **Llamadas al sistema. Interfaz UNIX S5 R4**

1.PRINCIPIOS GENERALES

Un Subsistema de Archivos es una organización de datos dentro de un disco (también llamada partición). Características de un Subsistema de Archivos:

- Se establece Jerárquicamente (árbol invertido).
- Los archivos de datos no tienen un tipo predefinido de estructurar los datos.
- Incorpora técnicas de control de accesos: Cada archivo tiene unos permisos diferenciados (propietario, grupo, otros).
- Asignación dinámica de espacio en disco. Por defecto el UNIX admite como máximo 2147 millones de bytes.
- Posee la característica de manipular los dispositivos físicos como archivos.
- Se permite la cohabitación de distintos tipos de sistemas de archivos.

Vfs ==> Virtual File System. Se utiliza para poder acceder a cualquier tipo de partición. Debe encontrarse en los sistemas que tengan varios tipos de subsistemas de archivos (ufs, veritas file system, s5).

2.TIPOS DE SUBSISTEMAS DE ARCHIVOS

En un mismo sistema UNIX podemos disponer de varios tipos de subsistemas de archivos situados en varias particiones. Para controlar esto se utiliza el subsistema de archivos vfs, que es un sistema virtual. Para crear una partición se utiliza el siguiente comando:

$mkfs −F <tipo particion> nº grupos_cilindros Bloques_lógicos nºnodos−i Tamaño

partición(Kbytes) nombre_físico

Además las particiones se van a montar todas sobre un mismo árbol de directorios. Esto quiere decir que la organización física de las particiones va a ser invisible para el usuario. En un directorio se montará la partición y esa será su ubicación lógica. Para montar una partición:

$mount /dev/<nombre partición> /<directorio al que se asocia la partición>

SUBSISTEMA DE FICHEROS SYSTEM 5

Es el sistema de archivos procedente de AT&T. Este sistema de archivos se podía exclusivamente manejar información almacenada en Discos Duros y Floppys. La información se almacena en bloques.

Las técnicas de acceso permiten la recuperación de n*bloques (hasta n=64). Esto significa que tengo que tener los buffers suficientes que permitan esto. Con mkfs formateo una partición y estructura los bloques. La partición se divide en:

- 1 bloque para el área de ***BOOT***. (512 bytes). Es el bloque número 0.Deberá almacenarse el 2nd Level Boot. Esto debe ser capaz de arrancar el sistema. Si no es un disco de arranque este bloque estará vacío.
- 1 bloque que es el ***Superbloque***. (512 bytes). Es un área administrativa. Es el bloque número 1, y guarda una lista enlazada con las ubicaciones de los i−nodo libres.
- Lista en cuadricula que son los ***i−nodes***.(Nodos de índice). Cada nodo−i ocupa 64 bytes. Se indican los atributos de un archivo y su ubicación física en disco. La cantidad de nodos−i que tengamos en la partición va a ser el número máximo de archivos que vamos a poder tener en esa partición.
- ***Bloques de Almacenamiento***:

− Bloque de datos.

− Bloque de punteros a bloques de datos.

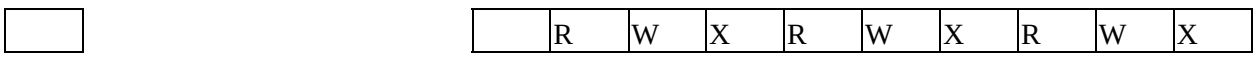
− Punteros a bloques libres.

− Bloque no asignado por ningún archivo.

− Bloque de la lista de bloques libres.

NODO−I

También llamado *i−nodo* o *i−node*. Cada i−nodo ocupa 64 bytes de los cuales 2 bytes pertenecen al **modo de un archivo.** Dentro del modo de un archivo cada bit tiene su significado.



Propietario Grupo Otros

Tipo Set−UserID

Set−GroupID

Sticky−Bit **MODO**

El Tipo puede ser: Ordinario, Directorio, Archivo orientado a bloques, Archivo orientado a caracter, FIFO, Enlace simbólico.

Tipo	**Valor**	**Se crea con**
Ordinario	1000	open(), creat()
Directorio	0100	mkdir()
Especial−Carácter	0010	mknod()
Especial−Bloque	0110	mknod()
Pipe	0001	mknod()
Enlace simbólico	1010	link()
Inodo libre	0000	

El **nodo−i** consta principalmente de los siguientes campos:

- **Modo**.
- **Nº Enlaces**. Indica cuantas veces un archivo esta siendo referido en el árbol de directorios (hard links).
- **User−id**
- **Group−id**
- **Tamaño**
- **Direcciones de los bloques**
- **Fecha acceso**. Fecha y hora que un archivo ha sido accudido.
- **Fecha modIficación**
- **Fecha creación**

TAMAÑOS de ARCHIVOS en una PARTICION SYSTEM 5

Tamaño Bloque	**Directo**	**Simple**	**Doble**	**Triple**
512	5.120	70.656	859.264 (8 MB)	1.091.200.000 (1 GB)
1024	10.240	272.384	67 MB	17 GB
2048	20.480	1 MB	537 MB	275 GB

Esto es en teoría, ya que el campo tamaño en un nodo−i tiene 4 bytes, y por lo tanto el tamaño máximo creable en una partición system 5 es de 4 GB.

ESTRUCTURA DE ACCESO A DATOS

La estructura de punteros de bloques para acceso a datos (acceso a bloques de datos), se jerarquiza de la siguiente forma.