SISTEMAS OPERATIVOS

TRABAJO PRACTICO

MINIX

1er.-C.2007

Objetivos del práctico

Al terminar este trabajo Ud. habrá aprendido a:

1. Instalar MINIX 2.0 en su versión DOSMINIX o sobre un simulador.
2. Utilizar convenientemente los principales comandos de MINIX.
3. Compilar y ejecutar programas escritos en lenguaje C.
4. Aplicar en forma práctica algunos de los conocimientos brindados en la materia.
5. Realizar modificaciones al Sistema Operativo MINIX.

Herramientas necesarias:

Para resolver los ejercicios propuestos necesitará:

1. Una PC con Windows 95/98 con, al menos, 80 Mb de Disco o el entorno necesario para instalar el simulador elegido.

Requisitos de Entrega

Lugar y Fecha de entrega:

1. La fecha y hora de entrega para este práctico es la que figura en el cronograma de la materia. (se alienta y acepta la entrega del trabajo, en su totalidad, en forma anticipada)
2. Los trabajos deben ser entregados PERSONALMENTE a alguno de los docentes de la cátedra en los horarios de clase o de consulta. No se aceptarán trabajos depositados en el Departamento de Computación o en cualquier otro lugar.
3. No se aceptarán trabajos incompletos.

Formato de Entrega.

Se deberá entregar en un medio digital, preferentemente CD, una o varias imágenes del sistema operativo, con los cambios efectuados al mismo y que contenga además todas las demás resoluciones en formato fuente, formato ejecutable y los programas de prueba que se utilizaron para comprobar que los cambios o resoluciones funcionan correctamente, **identificando claramente directorios o subdirectorios en los cuales se encuentran**. Además se deberán entregar todos los archivos necesarios para que esa imagen o imágenes ejecuten perfectamente.

Se deberá, además, entregar un DOCUMENTO IMPRESO. Ese documento debe reunir las siguientes características:

1. Formato de Presentación
   1.1. Impreso en hojas de tamaño A4 encarpetadas.
2. Secciones del documento (Todas obligatorias):
   2.1. **Carátula de presentación:** Debe incluir OBLIGATORIAMENTE:
      2.1.1. Asignatura
      2.1.2. Número y Descripción del trabajo práctico
      2.1.3. Año y Cuatrimestre de Cursado
      2.1.4. Identificación del Grupo
      2.1.5. Nombres, Apellidos y direcciones de correo electrónico de TODOS los Integrantes del grupo
   2.2. Sección Principal: Aquí debe incluirse la resolución de cada uno de los problemas planteados. Para cada respuesta debe indicarse OBLIGATORIAMENTE, el número y título del problema al que corresponde tal como aparece en el enunciado y los comandos y/o programas utilizados para resolverlos. **Se deberá indicar claramente en que directorio y bajo que nombre se encuentran los fuentes, los ejecutables y los programas de prueba**

Cambios al enunciado del práctico, fechas de entrega, etc.

Cualquier cambio en los enunciados, fechas de entrega, etc. será informado utilizando dos métodos:

1. La página Web de la Materia.
2. La lista de correo sisop@dc.uba.ar.

Ud. no puede alegar que no estaba al tanto de los cambios si esos cambios fueron anunciados utilizando alguno de los dos métodos.

SUGERENCIA: Consulte frecuentemente la página de la materia y asegúrese de que ha sido incorporado a la lista de correos.

Los grupos serán de hasta un máximo de tres (3) integrantes.

Ejercicios

Ayudas:

http://www.dc.uba.ar/people/materias/so/html/minix.html

Setear Teclado Español

cp /usr/lib/keymaps/spanish.map /etc/keymap
echo loadkeys /etc/keymap >> /etc/rc
sync; sync; shutdown

Pruebe el ash, el man, el apropos y el mined

1. Instalación del Sistema Operativo

a) Indique gráficamente el layout del sistema operativo MINIX indicando sus componentes y las funciones que brindan. En especial indique los system_calls que contienen.
b) Instalar MINIX en su versión para DOS (DOSMINIX), WINDOWS o LINUX (BOCHS) según se describe en http://www.dc.uba.ar/people/materias/so/html/minix.html

2. Herramientas

Indique que hace el comando make y mknode. Cómo se utilizan estos comandos en la instalación de MINIX y en la creación de un nuevo kernel. Para el caso del make muestre un archivo ejemplo y explique que realiza cada uno de los comandos internos del archivo ejemplo.

3. Comandos Básicos de MINIX/UNIX

3.0. Póngale password root a root.

3.1. ***pwd***

Indique qué directorio pasa a ser su *current directory* si ejecuta:

3.1.1. ***# cd /usr/src***
3.1.2. ***# cd***
3.1.3. ¿Cómo explica el punto 3.1.2?

3.2. ***cat***

Cual es el contenido del archivo ***/usr/src/.profile*** y para que sirve.

3.3. ***find***

En que directorio se encuentra el archivo
proc.c

3.4. ***mkdir***

Genere un directorio ***/usr/<nombregrupo>***

3.5. ***cp***

Copie el archivo ***/etc/passwd*** al directorio ***/usr/<nombregrupo>***

3.6. ***chgrp***

Cambie el grupo del archivo ***/usr/<grupo>/passwd*** para que sea ***other***

3.7. ***chown***

Cambie el propietario del archivo ***/usr/<grupo>/passwd*** para que sea ***ast***

3.8. ***chmod***

Cambie los permisos del archivo ***/usr/<grupo>/passwd*** para que

- el propietario tenga permisos de lectura, escritura y ejecución
- el grupo tenga solo permisos de lectura y ejecución
- el resto tenga solo permisos de ejecución

3.9. ***grep***

Muestre las lineas que tiene el texto **include** en el archivo
/usr/src/kernel/main.c

Muestre las lineas que tiene el texto **POSIX** que se encuentren en todos los archivos
/usr/src/kernel/

3.10. ***su***

3.10.1. Para qué sirve?
3.10.2. Que sucede si ejecuta el comando ***su*** estando logueado como **root**?
3.10.3. Genere una cuenta de <usuario>
3.10.4. Entre a la cuenta <usuario> generada
3.10.5. Repita los comandos de 3.10.2

3.11. ***passwd***

3.11.1. Cambie la password del usuario ***nobody***
3.11.2. presione las teclas ALT-F2 y verá otra sesion MINIX. Logearse como ***nobody***
3.11.3. ejecutar el comando ***su***.
3.11.3.1. ¿Que le solicita ?
3.11.3.2. ¿Sucede lo mismo que en 3.10.2? ¿Por qué?

3.12. ***rm***

Suprima el archivo **/usr/<grupo>/passwd**

3.13. ***ln***

Enlazar el archivo ***/etc/passwd*** a los siguientes archivos ***/tmp/contra1 /tmp/contra2***
Hacer un ls – l para ver cuantos enlaces tiene ***/etc/passwd***

3.14. ***mkfs***

Genere un Filesystem MINIX en un diskette

3.15. ***mount***

Montelo en el directorio
/mnt
Presente los filesystems que tiene montados

3.16. ***df***

Que espacio libre y ocupado tienen todos los filesystems montados? (En KBYTES)

3.17. ***ps***

3.17.1. Cuantos procesos de usuario tiene ejecutando ?
3.17.2. Indique cuantos son del sistema

3.18. ***umount***

3.18.1. Desmonte el Filesystem del directorio
/mnt

3.18.2. Monte el Filesystem del diskette como read-only en el directorio
/mnt

3.18.3. Desmonte el Filesystem del directorio /mnt

3.19. ***fsck***

Chequee la consistencia de Filesystem del diskette

3.20. ***dosdir***

Tome un diskette formateado en DOS con archivos y ejecute
dosdir a
Ejecute los comandos necesarios para que funcione correctamente el comando anterior

3.21. ***dosread***

Copie un archivo de texto desde un diskette DOS
al directorio **/tmp**

3.22. ***doswrite***

Copie el archivo /etc/passwd al diskette DOS

4. ***Uso de STDIN, STDOUT, STDERR y PIPES***

4.1. *STDOUT*

4.1.1. conserve en el archivo ***/usr/<grupo>/fuentes.txt*** la salida del comando ***ls*** que muestra todos los archivos del directorio ***/usr/src*** y de los subdirectorios bajo ***/usr/src***

4.1.2. Presente cuantas líneas, palabras y caracteres tiene ***/usr/<greupo>/tmp/fuentes.txt***

4.2. *STDOUT*

4.2.1. Agregue el contenido, ordenado alfabeticamente, del archivo ***/etc/passwd*** al final del archivo ***/usr/<grupo>/fuentes.txt***

4.2.2. Presente cuantas lineas, palabras y caracteres tiene ***usr/<grupo>/fuentes.txt***

4.3. *STDIN*

4.3.1. Genere un archivo llamado ***/usr/<grupo>/hora.txt*** usando el comando ***echo*** con el siguiente contenido:

2355

4.3.2. cambie la hora del sistema usando el archivo ***/usr/<grupo>/hora.txt*** generado en 4.3.1

4.3.3. Presente la fecha del sistema

4.4. *STDERR*

Guarde el resultado de ejecutar el comando **dosdir k** en el archivo ***/usr/<grupo>/error.txt.*** Muestre el contenido de ***/usr/<grupo>/error.tmp***

4.5. *PIPES*

Posiciónese en el directorio **/** (directorio raíz), una vez que haya hecho eso:

4.5.1. Liste en forma amplia los archivos del directorio ***/usr/bin*** que comiencen con la letra **s**. Del resultado obtenido, seleccione las líneas que contienen el texto ***sync*** e informe la cantidad de caracteres, palabras y líneas.

Nota 1: Está prohibido, en este ítem, usar archivos temporales de trabajo

Nota 2: si le da error, es por falta de memoria, cierre el proceso de la otra sesion, haga un kill sobre los procesos update y getty.

5. Diseñar y programar un utilitario para el administrador del sistema, que permita crear, borrar o modificar en forma automática cuentas de usuarios.
De forma interactiva pedirá todos los parámetros necesarios del nuevo usuario.
Automáticamente comprobará y validará el identificador del nuevo usuario. Creará en caso de que no exista, su directorio asociado o directorio de trabajo, haciéndolo su propietario. Asignará al nuevo usuario un shell particular. Modificará los archivos group y shadow con los nuevos valores. En caso que el grupo no exista deberá agregarlo. Deberá agregar el usuario al grupo
El borrado de un usuario llevará consigo la desaparición de todos sus archivos y directorios asociados y la modificación de los archivos group y shadow. En todos los casos deberá controlar la consistencia de la información.

6. ***Ejecución de procesos en Background***

Crear el siguiente programa

```
/usr/src/loop.c
#include <stdio.h>
int main()
{
int i, c;

while(1)
        {
        c = 48 + i;
        printf("%d",c);
        i++;
        i = i % idgrupo;
        }
}
```

Compilarlo. El programa compilado debe llamarse **loop,** Indicando a la macro idgrupo el valor de su grupo.

a) Correrlo en foreground. ¿Que sucede ? Mate el proceso con el comando kill
b) Ahora ejecútelo en background

/usr/src/loop > /dev/null &

Que se muestra en la pantalla ?

Que sucede si presiona la tecla F1? Que significan esos datos ?
Que sucede si presiona la tecla F2? Que significan esos datos ?

7. Generación de kernel minix

a) Generar un nuevo minix manteniendo la versión original.
b) Compilar los fuentes del kernel y construir un nuevo diskette boot.

Bootee alternativamente del original y los originados por Ud.
(ver sugerencias al fondo de los enunciados)

8. Explique que tipo de mecanismo utiliza MINIX y LINUX para:

a) Administrar el recurso procesador (scheduler).
b) Administrar el recurso memoria.
c) Administrar las operaciones de Entrada/Salida
d) Administrar la información en disco (File System)

Realice una comparación entre MINIX y LINUX para cada punto anterior.

9. Explique como se genera un driver de dispositivo tipo bloque y como se realiza la vinculación con el sistema de archivos en MINIX. Además describa la estructura del procedimiento de una tarea de E/S (I/O task).
Sugerencias: Como ayuda busque las respuestas a las siguientes preguntas:

Qué es el "major number" de un device. (com.h)
Dónde se declara la entrada a un driver. (table.c)
Cómo se llega desde un proceso usuario al driver. (table.c)

10. Modificación de códigos

a) Modifique el "scheduler" original del MINIX para el nivel de usuarios.
b) Modifique la administración de memoria original del MINIX

En ambos casos deberá describir en el informe cuales fueron las decisiones tomadas, cuáles fueron las expectativas y cuáles fueron los resultados obtenidos e informar el juego de programas utilizados con los cuales se llegó a alguna conclusión. (**tests o pruebas mencionados en Forma de entrega)**

11. Implementar un nuevo ***system call*** llamado ***newcall*** que devuelva según requerimiento, el pid del proceso que lo invocó, el ppid, los punteros al segmento text, al segmento data y al segmento stack, y los muestre por pantalla. (Ver sugerencias al fondo de los enunciados). No olvide los test de prueba. **(Tests o pruebas mencionados en Forma de entrega).**

12. Implementar las primitivas para manejo de semáfororos y las primitivas P y V (deben ser implementadas mediante system-call) (ver sugerencias al fondo de los enunciados)
Pruébelo con el siguiente caso:
Modelo productor/consumidor. No olvide los test de prueba. **(Tests o pruebas mencionados en Forma de entrega).**

13. (Opcional) Resuelva el problema de un productor y dos consumidores que deberán consumir la información generada por el productor en forma alternada, no pudiendo el productor producir información si la misma no fue consumida, ni los consumidores retirar la información que ya retiraron.

14. (Opcional) Resuelva el problema de Lectores/Escritores con prioridad Lectores. **(Tests o pruebas mencionados en Forma de entrega).**

15. (Opcional) Implemente "Lista de Control de Accesos" para archivos. Se debe generar la lista para un archivo, adjudicar permisos, revocar permisos, verificar que el usuario que ejecuta una acción sobre el archivo posee el permiso correspondiente, leer la LCA, guardar modificaciones, agregar y modificar permisos. No olvide los test de prueba. **(Tests o pruebas mencionados en Forma de entrega).**

16. (Opcional) Explique y muestre como se implementa Lectores/Escritores con prioridad Escritores en LINUX.

Sugerencias sobre:

Implementación de Semáforos a nivel de usuario
Deberá implementar la gestión de semáforos a utilizar sólo por procesos de usuario.
El administrador de semáforos será el Memory Manager o el File System Manager.
Se sugieren las siguientes llamadas al sistema:
semaf = crear_sem(argumento)
Asigna un semáforo si ya está creado, en el caso de que no haya disponibles debe devolver un error.
val_sem(semaf , argumento)
Retorna el valor del semáforo.
nproc_sem(semaf, argumento)
Retorna la cantidad de procesos bloqueados en ese semáforo
is_sem(semaf, argumento)
Retorna si ***semaf*** es un semáforo asignado al proceso
p_sem(semaf, argumento)
Realiza la operación del operador P
v_sem(semaf, argumento)
Realiza la operación del operador V
liberar_sem(semaf, argumento)
Libera al semáforo y a todos los procesos que aún están bloqueados .
En todos los casos se debe devolver un error cuando corresponda (como por ejemplo hacer P sobre un semáforo no asignado).

Sugerencia de estructura
Como se necesitará tener por un lado datos sobre el semáforo: el valor, su nombre, nro de procesos que lo están utilizando, lista de procesos utilizándolo, lista de procesos bloqueados.
Dado que un proceso puede utilizar N semáforos, por cada semáforo debe haber una lista de hasta M procesos que utilizan el semáforo y otra lista de hasta M procesos bloqueados en el semáforo
Es decir cada lista la podemos representar por una Matriz de NxM

Sugerencias de cómo implementar un system call

src/lib/syscall: **Crear el archivo *newcall.s***
src/lib/syscall: **Modificar el Makefile para que incluya su compilación.**
src/lib/posix: **Crear el archivo *_newcall.c***
src/lib/posix: **Modificar el Makefile para que incluya su compilación.**
include/unistd.h: **Prototipo del System Call *newcall***
include/callnr.h: **Asignarle el número de System Call e incrementar la Cantidad de System Calls *NCALLS* si fuese necesario. ATENCION: esto puede afectar a src/mm/table.c y src/fs/table.c que utilizan NCALLS para sus vectores.**

src/mm/proto.h o src/fs/proto.h : **Agregar el prototipo de la función que invocará el MM/FS con nombre *do_newcall***
src/mm/table.c o src/fs/table.c: **insertar un elemento en la Tabla de funciones de atención de System Calls del MM/FS con nombre *do_newcall*.**
src/mm/newcall.c: **Crear este archivo con el código de la nueva System Call.**
src/mm/Makefile: **Editar este archivo para que incluya la compilación de *newcall.c*.**

Sugerencia para identificar versiones de kernel

Todos los cambios que se hagan al kernel, no deberán alterar la compilación standard. Para eso se utilizará en el archivo /usr/include/minix/config.h:
#define UBA_FCEN 1
para forzar la compilación sobre el código nuevo y si se elimina esta línea se compilará con la versión standard.
Cada parte del código que agregue o cambie deberá estar realizada con compilación condicional utilizando:
#ifdef UBA_FCEN
----------Nuevo código----------
#else
----------Código original----------
#endif

A los efectos de verificar que se está ejecutando un kernel compilado por uno mismo cambiar en el archivo /usr/include/minix/config.h:
/* Minix release and version numbers. */
#define OS_RELEASE "2.0"
#define UBA_FCEN 1
ifdef UBA_FCEN
#define OS_VERSION "2 SISTEMAS OPERATIVOS - UBA-FCEN"
#else
#define OS_VERSION "2"
#endif
Este mensaje se verá al arrancar minix

Antes de realizar cualquier cambio en el kernel, este seguro de tener un conjunto de diskettes de emergencia: disco de booteo, backups del kernel minix original y de los archivos fuentes que fueron modificados. De manera de poder restaurar el sistema en caso de que algo salga mal con la compilación del nuevo kernel.

Como resguardo ante pruebas con distintos kernels es conveniente generar un directorio /minix2 en el /dev/hd2a, copiar allí alguna imagen confiable del kernel desde el /minix. Cuando el monitor de booteo pregunta con que SO arrancar presiones ESC . Allí tipear

image= minix /* Si se quiere bootear con la nueva imagen */
image= minix2 /*Si se quiere bootear con la `imagen vieja */`