Fundamentos de los Sistemas Operativos

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

fSO

EJERCICIOS DEL BT3
Gestión del Sistema de Archivos
UT07, UT08, SUT07 y SUT08
Versión 1.0

Contenido

1 Cuestiones sobre Implementación de Archivos 2

2 Cuestiones de código sobre Tuberías y Redirección 2

3 Cuestiones sobre Implementación de Directorios y Protección 4

4 Cuestiones y Problemas sobre Minix 5

1 Cuestiones sobre Implementación de Archivos

1. Un archivo requiere un total de N bloques de datos. Asumiendo que sus metadatos residen en memoria principal calcule el **número de accesos a disco** que son necesarios para acceder a su último bloque de datos (bloque N-1), en un sistema con:
a) Asignación contigua.
b) Asignación mediante lista enlazada
c) Asignación indexada.
Justifique su respuesta.

2. Supongamos un archivo cuyo contenido es una película. ¿Qué sistema de asignación de bloques será más eficiente para el visionado de la misma? Justificar la respuesta.

3. Cite las principales ventajas e inconvenientes de la **asignación contigua** de bloques en un sistema de ficheros.

2 Cuestiones de código sobre Tuberías y Redirección

4. Indique ordenada y secuencialmente los pasos necesarios para establecer un **mecanismo de comunicación** entre dos **procesos UNIX**, padre e hijo, de manera que: todo lo que el padre escriba en su salida estándar, el hijo lo lea desde su entrada estándar. Ponga en cada paso las primitivas POSIX e instrucciones C necesarias.

5. Dado el siguiente código,

```
//código propuesto
int fd[2], fd1, fde;
   fde = open("error.txt", NEWFILE, MODE644);
   dup2(fde, STDERR_FILENO);
   close (fde);
   pipe(fd);
   if (fork() == 0) {
     /*CODIGO DEL HIJO */
     dup2 (fd[0], STDIN_FILENO);
     close (fd[0]);
     close (fd[1]);
     fd1 = open("salida.txt", NEWFILE, MODE644));
     dup2(fd1, STDOUT_FILENO);
     close (fd1);
     // ---> A) ESTADO TABLA HIJO
    ...
  } else
    /* CODIGO DEL PADRE*/
    dup2 (fd[1], STDOUT_FILENO);
    close (fd[0]);
    close (fd[1]);
    // ---> B) ESTADO TABLA PADRE
   ...
   }
```

Suponga que el estado inicial de la **tabla de descriptores** del proceso **padre** es el siguiente

0	STDIN
1	STDOUT

2	STDERR
3	--
4	--

Indique como queda la tabla de descriptores del proceso padre en la línea señalada como B) y en el hijo en la línea señalada como A).

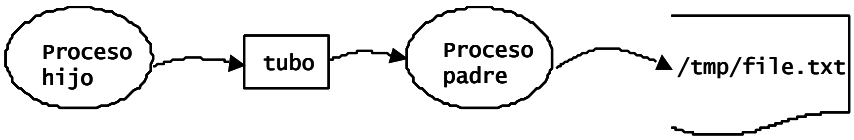
6. Dado el siguiente código en el cual se generan al menos tres proceso P1,P2, y P3:

```
//codigo propuesto
...
pipe(fd);
pipe(fd2);
if(fork() != 0){
  /***Proceso P1 ***/
  dup2(fd[1],STDOUT_FILENO);
  close(fd[0]);
  close(fd[1]);
  dup2(fd2[0],STDIN_FILENO);
  close(fd2[0]);
  close(fd2[1]);

}else{
  /***Proceso P2 ***/
  dup2(fd[0],STDIN_FILENO);
  close(fd[0]);
  close(fd[1]);
  pipe(fd3);
  if(fork() != 0){
    close(fd2[0]);
    close(fd2[1]);
    dup2(fd3[1],STDOUT_FILENO);
    close(fd3[0]);
    close(fd3[1]);
  }else{
     /***Proceso P3 ***/
     dup2(fd3[0],STDIN_FILENO);
     close(fd3[0]);
     close(fd3[1]);
     dup2(fd2[1],STDOUT_FILENO);
     close(fd2[0]);
     close(fd2[1]);
  }
}
...
```

Indique para los procesos P1, P2 y P3, el contenido, después de la ejecución de este código, de sus respectivas tablas de descriptores de archivo, la relación existe entre ellos y cuál es el esquema de comunicación resultante.

7. Se pretende implantar un esquema entre dos procesos (Proceso padre y Proceso hijo) como el que se muestra en la figura siguiente:



Donde el proceso padre redirige su salida al fichero /tmp/file.txt.
Para ello se propone el siguiente código:

```
// Codigo propuesto
...
  pipe(fd);
  if(fork() == 0){
    dup2(fd[1],STDOUT_FILENO);
    close(fd[0]); close(fd[1]);
    /* resto de codigo */
  }else{
     dup2(fd[0],STDIN_FILENO);
     close(fd[0]); close(fd[1]);
     fd_open=open("/tmp/file.txt",O_WRONLY|O_TRUNC|O_CREAT,0600);
     /* resto de codigo */
  }
  ...
```

Suponiendo que el resto de código que falta no altera la redirección de la E/S de los procesos, indique si el código es correcto o no. En caso de no serlo escriba las modificaciones que son necesarias realizar en el código para que lo sea.

3 Cuestiones sobre Implementación de Directorios y Protección

8. Indique si las siguientes afirmaciones sobre sistemas tipo **Unix** son ciertas. Justifique las respuestas:
- **a)** Para todos los directorios se cumple que el número de enlaces siempre es mayor que 1
- **b)** Las entradas de los directorios ocupan un área dedicada en la cabecera del disco

9. Dado el siguiente listado de un directorio en un sistema POSIX:

```
drwxr-xr-x  2     sterrasa    fso   4096        may   8     2002        .
drwxr-xr-x  11    sterrasa    fso   4096        mar   21    14:39       ..
-rwsrw-r-x  1     root        fso   1139706     abr   9     2002        nuevo
-rw-rw-r--  1     sterrasa    fso   634310      abr   9     2002        fich1
-rw-rw-r--  1     sterrasa    fso   104157      abr   9     2002        fich3
```

Donde el programa "nuevo" copia el fichero recibido como primer argumento, dándole a la copia el nombre recibido como segundo argumento. Explique si las siguientes órdenes podrán completarse con éxito o no suponiendo que todas ellas se ejecutan en el directorio listado anteriormente
- **a)** "nuevo fich1 f ich2", con los atributos eUID=jose y eGID=fso, inicialmente.
- **b)** "nuevo fich1 f ich3", con los atributos eUID=juan y eGID=grupo3 inicialmente

10. Razone si un proceso puede modificar sus UID y GID efectivos durante su ejecución. Si es imposible, indique por qué. Si fuera posible, describa cómo podría conseguir hacerlo.

11. Dado el siguiente listado del contenido de un directorio UNIX:

```
-rwsr--r-x 1 calif grupo1 1014  May  17 11:10 miprog
-r---w---- 1 calif grupo1 14487 Jun  25 09:11 datos
-rw----r-- 1 calif grupo1 2099  Jun  25 08:49 punt2
```

Y los usuarios “ramon” (perteneciente al grupo “grupo1”), “marta” (perteneciente al grupo “grupo3”) y “juan” (perteneciente al grupo “grupo2”). Indique, de entre los tres usuarios citados, quienes podrán utilizar el programa “miprog”, y con él, procesar la información existente en los ficheros “datos” y “punt2” (para ello,únicamente se necesita leer el contenido de ambos ficheros).

12. Sea el siguiente código denominado “miprog” encargado de realizar la apertura de un fichero cuyo nombre se le pasa como argumento.

```
int main( int argc, char *argv[] ) {
   int fd;
   char path_command[80], *fichero;
   sprintf(path_command, "%s", argv[1]);
   fd = open( path_command, O_RDONLY );
   if (fd == -1) {
      fprintf(stdout, "error apertura lectura %s", path_command);
   } else {
      fprintf(stdout,"funciona apertura lectura %s\n", path_command);
   }
   fd = open( path_command, O_WRONLY );
   if (fd == -1) {
```

```
      fprintf(stdout, "error apertura escritura %s", path_command);
   } else {
      fprintf(stdout,"funciona apertura escritura %s\n",
path_command);
   }
}
```

Considerando el siguiente listado de los ficheros:

```
-rwsr-xr-x 1 xedu gedu 1014   May 17 11:10 miprog
-rw-r--r-- 1 xedu gedu 14487  Jun 25 09:11 datos
-r--rw-rw- 1 xedu gedu 2099   Jun 25 08:49 punt2
```

Y dado los usuarios “xedu” perteneciente al grupo “gedu” y “xedu2 ” perteneciente al grupo “otros”. Indique los mensajes que aparecen en pantalla tras la ejecución de las siguientes órdenes por xedu y xedu2:

```
$ ./miprog datos
$./miprog punt2
```

4 Cuestiones y Problemas sobre Minix

13. Se ha creado un **sistema de archivos minix** en el dispositivo /dev/imagen. La capacidad del dispositivo es 15360 KBytes y ha sido formateado para albergar hasta 17984 nodos-i. Determine la estructura del dispositivo, indicando las partes que la componen, así como el número de bloques de cada una de las áreas.

NOTA: Los tamaños estándar de Minix son los siguientes: nodos-i de 32 bytes (7 punteros directos, 1 indirecto,1 doble indirecto), entradas de directorio de 16 bytes, 1 zona=1 bloque=1024 bytes, puntero a zona de 16 bits.

14. En una **partición Minix**, cuyo sistema de archivos contiene el siguiente árbol de directorios siendo b,d,e,h,i,j ficheros regulares, y el resto directorios:

- /a/b
- /a/c/d /a/c/e /a/f/g/h
- /a/i /a/j /a/k

Represente el contenido de todos los archivos de directorio y calcule su tamaño.
NOTA: asuma la asignación de nodos-i por orden alfabético

15. En un **sistema de archivos MINIX** estándar (el explicado en clase con zonas de 1 KB, nodos-i de 32 bytes con 7 punteros directos, 1 indirecto simple y 1 indirecto doble, entradas de directorio de 16 bytes, puntero a zona de 16 bits) se desea copiar un fichero de 230 KB con la orden “cp fich1.txt fichero2.txt”. Se pide:

a) Determine el número de zonas nuevas que Minix habrá marcado como ocupadas como consecuencia de la ejecución de la orden cp y el tipo de información que contendrán estas zonas nuevas.

b) b) Liste la secuencia de llamadas al sistema necesarias para completar dicha copia, proporcionando un algoritmo sencillo que describa lo que hace realmente esta orden.

16. Se ha creado un **sistema de archivos minix** con 8100 bloques en total y con capacidad para 10016 nodos-i. Determine la estructura del dispositivo, indicando las partes que la componen, así como el número de bloques de cada una de las áreas.
NOTA: Los tamaños estándar de Minix son los siguientes: nodos-i de 32 bytes (7 punteros directos, 1 indirecto,1 doble indirecto), entradas de directorio de 16 bytes, 1 zona=1 bloque=1024 bytes, puntero a zona de 16 bits.

17. En un dispositivo de 128Mbytes de capacidad, se crea un **sistema de archivos MINIX** para 3500 nodos-i. Indique de forma justificada el espacio que queda libre para datos después de dar formato al dispositivo. Nota: Los tamaños estándar de MINIX son:

- Tamaño de bloque 1024 bytes con 1bloque=1zona
- Referencia a bloque 2 bytes
- Entrada de directorio 16 bytes
- Nodo-i de 32 bytes

18. Dado un **sistema de archivos MINIX**, inicialmente vacío, y montado en /minix, se ejecuta la siguiente secuenciade órdenes:

```
$ echo "hola SO1" > /minix/fich1
$ cat /minix/fich1 | grep h | wc –l > /minix/lineas
$ mkdir /minix/maria
$ ls /proc
$ ps -la
$ mkdir /minix/bin
$ ln /minix/fich1 /minix/maria/fich2
$ ln -s /minix/fich1 /minix/maria/fich3
$ ln -s /minix/lineas /minix/maria/fich4
```

Teniendo en cuenta los tamaños estándar de Minix (16 bits por referencia a bloque, 16 Bytes por entrada de directorio, 32 Bytes por nodo-i, bloque=zona=1K Byte), indique:

a) Número total de nodos-i ocupados , t su desglose por ficheros
b) Tamaño en Bytes del directorio /minix , y valor de su atributo “número de enlaces”

19. Indique detalladamente cuántos nodos-i estarían ocupados, y cuál sería el valor del campo “enlaces” de cada uno de ellos, en un **sistema de ficheros UNIX** que contiene únicamente los siguientes archivos:

- /f1 (fichero regular de 2Mbytes)
- /f2 (es un enlace simbólico a f1)
- /f3 (es un enlace físico a /f1)
- /dir1 (es un directorio vacío)