# Fonaments dels Sistemes Operatius

Departament de Informàtica de Sistemes i Computadores (DISCA)
*Universitat Politècnica de València*

fSO

# EXERCICIS DEL BT3
# Gestió del Sistema d'Arxius
# UT07, UT08, SUT07 i SUT08
# Versió 1.0

**Contigut**

# 1 Qüestions sobre implementació d'arxius

**1.** Un arxiu requereix un total de N blocs de dades. Assumint que les seues metadades resideixen en memòria principal calculeu el **nombre d'accessos a disc** que són necessaris per a accedir al seu últim bloc de dades (bloc N-1), en un sistema amb:
**a)** Assignació contigua.
**b)** Assignació mitjançant llista enllaçada
**c)** Assignació indexada.
Justifiqueu la vostra resposta.

**2.** Suposem un arxiu el contingut del qual és una pel·lícula. ¿Quin sistema d'assignació de blocs serà més eficient per a el visionat de la mateixa? Justificar la resposta.

**3.** Citeu els principals avantatges i inconvenients de l'**assignació contigua** de blocs en un sistema de fitxers.

# 2 Qüestions de codi sobre canonades i redireccions

**4.** Indiqueu ordenada i seqüencialment els passos necessaris per a establir un **mecanisme de comunicació** entre dos **processos UNIX**, pare i fill, de manera que: tot el que el pare escriga en la seua sortida estàndard, el fill ho llisca des de la seua entrada estàndard. Fiqueu en cada pas les primitives POSIX i instruccions C necessàries.

**5.** Donat el següent codi,

```
//codi proposat
int fd[2], fd1, fde;
   fde = open("error.txt", NEWFILE, MODE644);
   dup2(fde, STDERR_FILENO);
   close (fde);
   pipe(fd);
   if (fork() == 0) {
     /*CODI DEL FILL */
     dup2 (fd[0], STDIN_FILENO);
     close (fd[0]);
     close (fd[1]);
     fd1 = open("salida.txt", NEWFILE, MODE644));
     dup2(fd1, STDOUT_FILENO);
     close (fd1);
     // ---> A) ESTAT TAULA FILL
    ...
  } else
     /* CODI DEL PARE*/
     dup2 (fd[1], STDOUT_FILENO);
     close (fd[0]);
     close (fd[1]);
     // ---> B) ESTAT TAULA PARE
    ...
    }
```

Suposeu que l'estat inicial de la **taula de descriptors** del procés **pare** és el següent

| | |
|---|---|
| 0 | STDIN |
| 1 | STDOUT |
| 2 | STDERR |
| 3 | -- |
| 4 | -- |

Indiqueu com queda la taula de descriptors del procés pare en la línia assenyalada com a B) i en el fill en la línia assenyalada com a A).

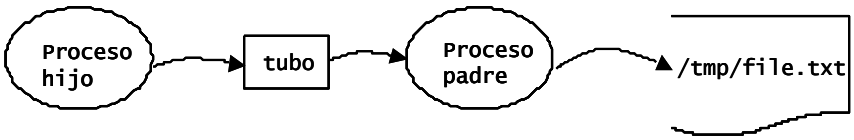
**6.** Donat el següent codi en el qual se generen almenys tres processos: P1,P2, i P3:

```
//codi proposat
...
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]);
  }
}
...
```

Indiqueu per als processos P1, P2 i P3, el contingut, després de l'execució d'aquest codi, de les seues respectives taules de descriptors d'arxius, la relació existent entre ells i quin és l'esquema de comunicació resultant.

**7.** Se pretén implantar un esquema entre dos processos ( procés pare i procés fill) com el que es mostra en la figura següent:




On el procés pare redirigeix la seua sortida al ficfitxer /tmp/file.txt.
Per a això es proposa el següent codi:

```
// Codi proposat
...
  pipe(fd);
  if(fork() == 0){
    dup2(fd[1],STDOUT_FILENO);
    close(fd[0]); close(fd[1]);
    /* resta de codi */
  }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);
     /* resta de codi */
  }
  ...
```

Suposant que la resta de codi que falta no altera la redirecció de l'E/S dels processos, indiqueu si el codi es correcte o no. En caso de no ser-ho escriviu les modificacions que són necessàries realitzar en el codi per a que lo siga.

# 3 Qüestions sobre implementació de directoris i protecció

**8.** Indiqueu si les següents afirmacions sobre sistemes tipus **Unix** són certes. Justifiqueu les vostres respostes:
- **a)** Per a tots els directoris es compleix que el nombre d'enllaços sempre és major que 1
- **b)** Les entrades dels directoris ocupen un àrea dedicada en la capçalera del disc

**9.** Donat el següent llistat d'un directori 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        nou
-rw-rw-r--  1     sterrasa    fso   634310      abr   9     2002        fich1
-rw-rw-r--  1     sterrasa    fso   104157      abr   9     2002        fich3
```

On el programa "nou" còpia el fitxer rebut com a primer argument, donant-li a la còpia el nom rebut com a segon argument. Expliqueu si les següents ordres podran completar-se amb èxit o no suposant que totes elles s'executen en el directori llistat anteriorment
- **a)** "nou fich1 f ich2", amb els atributs eUID=jose i eGID=fso, inicialment.
- **b)** "nou fich1 f ich3", amb els atributs eUID=juan i eGID=grup3 inicialment

**10.** Raoneu si un procés pot modificar els seus UID i GID efectius durant la seua execució. Si es impossible, indiqueu per què. Si fora possible, descriviu cóm ho podria aconseguir.

**11.** Donat el següent llistat del contingut d'un directori UNIX:

```
-rwsr--r-x 1 calif grup1 1014   May  17 11:10 miprog
-r---w---- 1 calif grup1 14487 Jun   25 09:11 dades
-rw----r-- 1 calif grup1 2099   Jun  25 08:49 punt2
```

Y els usuaris "ramon" (pertanyent al grup "grup1"), "marta" (pertanyent al grup "grup3") i "juan" (pertanyent al grup "grup2"). Indiqueu, d'entre els tres usuaris esmentats, qui podrà utilitzar el programa "miprog", i amb ell, processar la informació existent en els fitxers "dades" i "punt2" (per a això, únicament es necessita llegir el contingut d'ambos fitxers).

**12.** Siga el següent codi denominat "miprog" encarregat de realitzar l'obertura d'un fitxer el nom del qual se li passa com a argument.

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

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

Considerant el següent llistat dels fitxers:

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

I donats els usuaris "xedu" pertanyent al grup "gedu" i "xedu2 " pertanyent al grup "altres". Indiqueu els missatges que apareixeran en pantalla rere l'execució de les següents ordres per xedu i xedu2:

```
$./miprog dades
$./miprog punt2
```

# 4 Qüestions i problemes sobre Minix

**13.** S'ha creat un **sistema d'arxius minix** en el dispositiu /dev/imagen. La capacitat del dispositiu és 15360 KBytes i ha sigut formatejat per a albergar fins a 17984 nodes-i. Determineu l'estructura del dispositiu, indicant les parts que la composen, així com el nombre de blocs de cadascuna de les àrees.

**NOTA:** Les grandàries estàndard de Minix són les següents: nodes-i de 32 bytes (7 punters directes, 1 indirecte,1 doble indirecte), entrades de directori de 16 bytes, 1 zona=1 bloc=1024 bytes, punter a zona de 16 bits.

**14.** En una **partició Minix**, el sistema d'arxius del qual conté el següent arbre de directoris sent b,d,e,h,i,j fitxers regulares, i la resta directoris:

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

Representeu el contingut de tots els arxius de directori i calculeu la seua grandària.
*NOTA: asumiu l'assignació de nodes-i per ordre alfabètic*

**15.** En un **sistema d'arxius MINIX** estàndard (aquell explicat en classe amb zones de 1 KB, nodes-i de 32 bytes amb 7 punters directes, 1 indirecte simple i 1 indirecte doble, entrades de directori de 16 bytes, punter a zona de 16 bits) es desitja copiar un fitxer de 230 KB amb l'ordre "cp fitx1.txt fitx2.txt". Se demana:

**a)** Determineu el nombre de zones noves que Minix haurà marcat com a ocupades com a conseqüència de l'execució de l'ordre cp i el tipus d'informació que contindran aquestes zones noves.

**b)** b) Llisteu la seqüència de crides al sistema necessàries per a completar aquesta còpia, proporcionant un algorisme senzill que descriga el que fa realment aquesta ordre.

**16.** S'ha creat un **sistema d'arxius minix** amb 8100 blocs en total i amb capacitat per a 10016 nodes-i. Determineu l'estructura del dispositiu, indicant les parts que la composen, així com el nombre de blocs de cadascuna de les àrees.

**NOTA**: Les grandàries estàndard de Minix són les següents: nodes-i de 32 bytes (7 punters directes, 1 indirecte,1 doble indirecte), entrades de directori de 16 bytes, 1 zona=1 bloc=1024 bytes, punter a zona de 16 bits.

**17.** En un dispositiu de 128Mbytes de capacitat, se crea un **sistema d'arxius MINIX** per a 3500 nodes-i. Indiqueu de forma justificada l'espai que queda lliure per a dades després de donar format al dispositiu. Nota: les grandàries estàndard de MINIX són:

- Grandària de bloc 1024 bytes amb 1bloc=1zona
- Referència a bloc 2 bytes
- Entrada de directori 16 bytes
- Node-i de 32 bytes

**18.** Donat un **sistema d'arxius MINIX**, inicialment buit, i muntat en /minix, s'executa la següent seqüència d'ordres:

```
$ 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
```

Tenint en xompte les grandàries estàndard de Minix (16 bits por referència a bloc, 16 Bytes per entrada de directori, 32 Bytes por node-i, bloc=zona=1K Byte), indiqueu:

a) Nombre total de nodes-i ocupats , i el seu desglosse per fitxers
b) Grandària en Bytes del directori /minix , i valor del seu atribut "nombre d'enllaços"

**19.** Indique detalladament quants nodes-i estarien ocupats, i quin seria el valor del campo "enllaços" de cadascun d'ells, en un **sistema de fitxers UNIX** que conté únicament els següents arxius:

- /f1 (és un fitxer regular de 2Mbytes)
- /f2 (és un enllaç simbòlic a f1)
- /f3 (és un enllaç físic a /f1)
- /dir1 (és un directori buit)