Mariona Farré

Entrega Tema 4: Gestió de fitxers

1. Escriu un programa que mostri per la seva sortida estàndard les línies parells d'un fitxer d'entrada. El nom del fitxer d'entrada es rebrà per la línia de comandes.
Recorda que el codi que escriguis ha de tenir control d'errors. Exemple:

```
prompt$ more file
111111
22
33333
444
5555
prompt$ problema file
22
444
prompt$
```

```c
int main (int argc, char *argv[]){
   int fitxer;
   int bytesread;
   char c;
   int par=0;
   if(argc != 2){
            fprintf(stderr,"Usage: %s file_name\n",argv[0]);
        }
   fitxer=open(argv[1],O_RDONLY);

   if(fitxer==-1) { //error
        fprintf(stderr," S'ha produit un error al obrir el fitxer%s\n",argv[1]);
        }

   while((bytesread=read(fitxer,&c,1))>0){
        if(par==1){
            if(write(1,&c,1)<0){ //error
            fprintf(stderr,"Hi ha un error al escriure \n");
            }
        }
        if (c=='\n'){
        if(par==0) par=1;
        else par=0;
        }
   }
   if (bytesread == -1){ //error
        fprintf(stderr,"S'ha produit un error al llegir el fitxer \n");
   }
exit(1);
close(fitxer);
}
```

Normes per corregir l'exercici 1:
1,5 punts en total si el codi es correcte te control d'errors.
→ **NOTA**: 1,5 punts

2. A partir del següent codi, contesta de manera raonada les preguntes que es fan a continuació:

```
void procesar()
{
        char c;
        while(read(0,&c,sizeof(char))> 0)
        write(1,&c,sizeof(char));
        exit(0);
}
int main(int argc, char *argv[])
{
        int fd_ent,fd_sort;
        int pids[N];
        int i=0;
        if((fd_ent=open("in.txt", O_RDONLY))<0)     //obre el fitxer de només lectura
                perror("Error al primer open ");
        if((fd_sort=open("out.txt",O_WRONLY|O_CREAT|O_TRUNC,777))<0)
                perror("Error al segon open ");    //fitxer només escriptura+fitxer no existeix CREA
                                                   //borra contingut fixer anteriri i sobreescriu
        for(i=0;i<N;i++){
                pids[i]=fork();
                if(pids[i]==0){
                        /* open("in.txt" .... */
                        /* open("out.txt" .... */
                        dup2(fd_ent, 0);        //canal entrada fd_ent sortida TDVO =0  ENTRADA
                        dup2(fd_sort,1);        //canal entrada fd_sort sortida TDVO =1  SORTIDA
                        procesar();
                }
        }
        close(fd_ent);                          //tanca fd_ent→ canal entrada
        while(waitpid(-1,NULL,0)>0);
}
```

(a) Indica totes les estructures de dades relatives a la gestió de fitxers, en memòria i en disc, que s'accedeixen i/o actualitzen durant l'execució de totes dues crides open. I en la crida dup2?

En el open de → in.txt

Accedeix al bloc de dades del fitxer → afegir al Dentry

Accedeix al inode del fitxer int,txt → afegir inode a la taula d'inodes

crea nova entrada a la TFO i ocupa la primera entrada lliure a la TC del procés

En el open de → out.txt

Accedeix al bloc de dades del fitxer → afegir al Dentry

Accedeix al inode del fitxer int,txt → afegir inode a la taula d'inodes

crea nova entrada a la TFO i ocupa la primera entrada lliure a la TC del procés

En el cas de dup2→ s'actualitza la TFO tancar el canal 0 si no hi ha entrada de la TC que hi apunta→ modifica la TC del procés per fd_end i actualitza fd_ent

(b) Explica que fa el codi. A l'acabament del programa els fitxers in.txt i out.txt tenen el mateix contingut?

El programa obre el fitxer int.txt com a canal de lectura i el fitxer out.txt com a canal de escritura.

Crea N fills que tindran el mateix punter cam els dos fitxers oberts (pids[i])

Això fa que cada sortida de cada procés sobreescriu el seu resultat al fitxer out.txt, pero segurament no tindran el mateix contingut els dos fitxers ja que alguns caracters es poden moure i modificar al llegir caràcter per carácter o per un desfas.

(c) Si en comptes de fer el primer open (open("in.txt",...)al principi el féssim abans de la crida dup2 que passaria?
Es necessitaria el fitxer destí, en les crides dup2 son les entrades en la taula 0/1, i si es canvies l'open amb la crida dup2 el fitxer s'obriria i donaria un fitxer destí, pero no es podria entrar a través de la taula TDVO i els seus fills no tindran el punter que apunta al fitxer d'entrada, fent que els processos fills no llegiran correctament

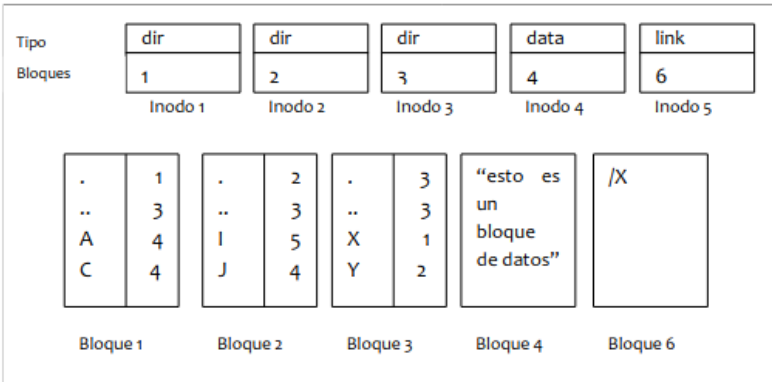
(d) I si fem el mateix amb el segon open (open("out.txt",...)?
Si es canvies el segon open amb un dup2 cada fill obriria un fitxer de sortida i crearia la seva propia entrada en la TFO. Al no tenir el punter del fitxer de lectura o sortida els processos s'aniran sobreescribint i reexectuant N vegades.

(e) Dibuixa l'estat de la TC, TFO i T-inodes abans que tots els processos finalitzin l'execució, es a dir just abans del waitpid (pots suposar que finalitzen alhora tots els processos)

(f) Per finalitzar, volem modificar el programa perquè tots els processos fills coneguin el PID dels seus germans. Raona com ho faries fent servir: (a) signals; (b) pipes.
Amb les signals no es podrien enviar les dades de pid entre germans ja que només es poden enviar senyals i no dades.
Amb les pipes si que es podria fer que els fills coneguessin el PID dels seus germans, com el procés para podria fer una pipe entre pare i tots els fills creats i el procés para enviar el PID de tots els germans en un procés fill.

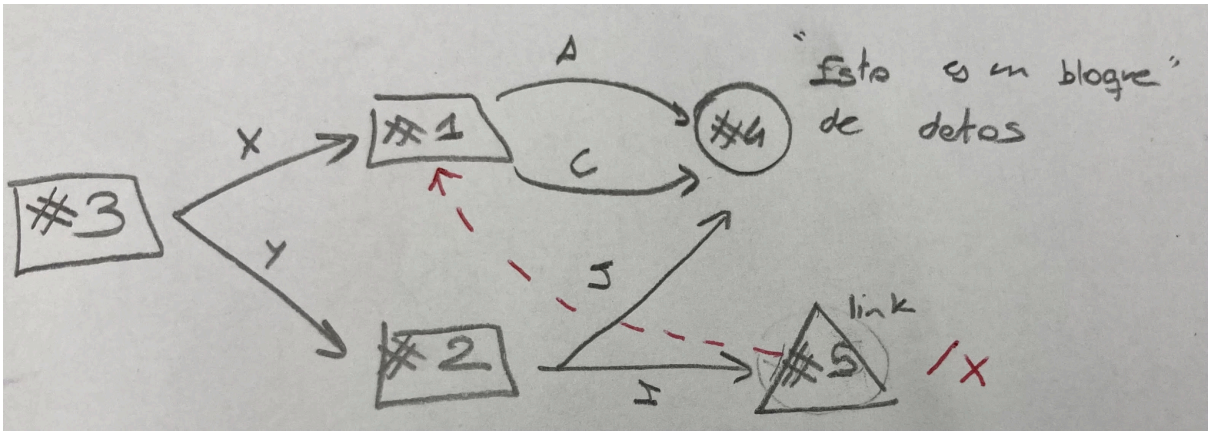
Normes per corregir l'exercici 2: 3 punts en total. 0,5 per apartat correcte.
→ **NOTA**: 2,5 punts (e) incorrecte)

3. Suposeu un sistema de fitxers (basat en inodes) que te els seguents inodes i blocs de dades que es mostren a la figura ??



(a) Dibuixa el graf de directoris que es pot deduir d'aquest esquema de inodes i blocs de dades. Indica quin es l'inode arrel. Indica els fitxers que son soft-link i quins fitxers son hard-link.

Inode arrel es: inode #3
Les entrades soft-link: inode #5
Les entrades hard-link: A,C,J hard-links de inode #4



(b) Indica el numero d'accessos a disc que es generaria si executem la crida a sistema open("/Y/I/A",O_RDONLY) Pots assumir que el superbloc esta a memoria. Indica en el cas que el sistema tingui caches (buffer cache i dentry) i en el cas que no en tingui. Assumeix que els inodes ocupen 1 bloc de disc.
inode 3+ dades bloc 3+ inode 2 +dades bloc 2(Y)+ inode 5+ dades bloc 5=bloc 6(I) + inode 3+ dades bloc 3+ inode 1 + dades bloc 1(X) + inode 4 (A) =11 accesos

Amb buffer cache:
inode 3+ dades bloc 3+inode 2=Y+ dades bloc 2+inode 5=I + dades bloc 6 + inode 1 + dades bloc 1 + inode 4 =A = 9 accessos

(c) Indica també, en el cas anterior, quin inode es guardaria a la taula d'inodes en memòria (vnodes).
El node que es va fent servir es el 4, així que aquest es guardar a memoria

Normes per corregir l'exercici 3: 2,5 punts en total. Puntuació per apartat: 1, 0,75, 0,75.
→ **NOTA**: 2,5 punts

4. Explica que es l'assignació indexada multinivell i quin sistema de fitxers coneixes que la faci servir.
Es un bloc a dins del disc que conté una taula amb l'entrada per cada fiter amb els punters corresponents als seus blocs, aquests permet que es guardin fitxers més grans ja que crea una estructura jeràrquica d'aquests índexs.

Normes per corregir l'exercici 4: 1 punts si resposta correcte
→ **NOTA**: 1 punt

5. Explica quines estratègies coneixes per gestionar l'espai lliure del disc.
Es bastant comú que es facin servir un bitmap dels blocs lliures. Per cada bloc del disc conté un bit que es igual a zero si es un bloc lliure o un bit igual a 1 si es un bloc ocupat. Es una estratègia que ocupa molt poc espai pero que millora l'eficiència a l'entrada o l'obertura de fitxers a dins del disc.

Normes per corregir l'exercici 5: 1 punts si resposta correcte
→ **NOTA**: 1 punt