Mariona Farré

Entrega Tema 3: Gestió de l'entrada/sortida

1. Escriu un programa que:

(a) Es limiti a fer un echo de l'entrada estandar per la sortida estandar

```c
//1 a) limiti a fer un echo d'entrada estandar a la sortida estandar
int main(){
  char c;
  while(read(0,&c,sizeof(char))) //LLegir un caracter per entrada estandar 0
     write(1,&c,sizeof(char));   //escriure aquest mateix caracter per sortida estandar
}
```

(b) Explica en que consisteix la tecnica del Buffering i posa tres exemples de situacions on pot ajudar. Pots trobar la informació al apartat de Optimització de l'entrada/sortida.

Un buffer ajuda a poder tallar el temps entre l'intercanvi de dades, i la tècnica del buffering al ser una area que guarda les dades temporalment, dona suport a l'execució de processos d'entrada i sortida.
Tres exemples serien: donades les dades per teclat i utilitzar el disc seguentment,
abans d'enviar dades de les cançons en els altaveus sense tenir pauses entre dues cançons i audio o vídeo d'internet perquè hi hagi menys possibilitats de que es talli la reproducció

(c) Modifica el programa del echo, perque utilitzi la tecnica del buffering per a millorar l'eficiencia. Es a dir llegint en un buffer de MAX en MAX en comptes de caracters de un en un.

```c
#define MAX 20
int main(){
  char c[20];
  int n=0;
  while ((n=read(0,c,(MAX*sizeof(char))))>0) //n guardara el valor guardat llegit per pantalla
       //pel canal estandar 0 el char C
   write (1,c,n);     //Escriu per pantalla el echo fet
}
```

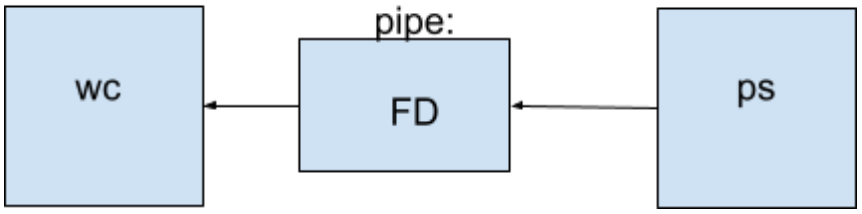
(d) Fa el echo, utilitzant el codi del programa anterior una sola vegada i redireccionant els canals que calgui dins del teu programa, de manera que es permeti el pas de parametres per indicar fitxer(s) de la manera seguent:

- Cap parametre (STDIN ->STDOUT)
- Un parametre (fitxer in ->STDOUT)
- Dos parametres (fitxer in ->fitxer out)

```
#define MAX 20
int main(int argc,char *argv[])
{
   char c[MAX];
   int fprimer,fsegon;
   int x=0;
   if(argc>1){
      close(0);   //redireccio del stdin al fitxer
      fprimer=open (argv[1],O_RDONLY);
      if(fprimer==-1){perror("Error\n");}
   }
   if(argc>2){
      close(1);   //redireccio del stdout al fitxer
      fsegon=open (argv[2],, O_RDWR | O_CREAT,S_IRWXU););
      if(fsegon==-1){perror("Error\n");}
   }
   while(x=read(0,c,(MAX*sizeof(char)))>0) write(1,c,x);
   close(0);
   close(1);
}
```

Almenys en una de les respostes cal que facis el control d'errors de les crides a sistema.
Normes per corregir l'exercici 1: 3 punts en total.
0,75 per cada apartat.
Resta 0,5 si no es fa control d'errors almenys en un d'ells.
→ **NOTA**: 2 punts

2. Dibuixa el diagrama de processos i pipes corresponent a aquest programa i indica justificadament quin sera el resultat d'executar-lo. Pots assumir que cap crida al sistema fallara.



Normes per corregir l'exercici 2: 1 punts en total si resposta correcta
→ **NOTA**: 1 punt

4. Donat el seguent programa:

```
main() {
int fd1, fd2, fd3;
int pid1, pid2, stat, i;
fd1 = open("s1.txt", O_WRONLY|O_CREAT, 640);
write(fd1, "p", sizeof(char));
if((pid1=fork())==-1) error();
if(pid1==0) /* proces fill */
{
fd2 = open("s2.txt", O_WRONLY|O_CREAT, 640);
for(i=0; i<5; i++)
{
if((pid2=fork())==0) /* proces net */
{
        fd3 = open("s1.txt", O_WRONLY);
        write(fd1, "n", sizeof(char));
        write(fd2, "n", sizeof(char));
        write(fd3, "n", sizeof(char));
        close(fd1);
        close(fd2);
        close(fd3);
        exit(0);
        }
}
for(i=0; i<5; i++) wait(&stat);
        write(fd1, "h", sizeof(char));
        write(fd2, "h", sizeof(char));
        close(fd1);
        close(fd2);
        exit(0);
        }
wait(&stat);
write(fd1, "p", sizeof(char));
close(fd1);
}
```

(a) Calcula el maxim numero d'entrades que poden haver en la TFA en un moment donat (i el fitxer relacionat amb cada entrada).
Hi han em màxim d'entrades de a la TFO es 9,
En fd3 hi ha l'open de s1.txt on entra 6 vegades a la TFO i En fd2 hi ha l'open de s2.txt que nomes entra 1 vegada a la TFO.

(b) Calcula el maxim numero de canals de la TDVA que apuntaran a entrades de la TFA relacionades amb s1.txt i s2.txt.
El st1.txt tindrà 12 canals apuntant cap a ell i s2.txt tindrà 6 canals apuntant a ell.

(c) Suposant que les crides a sistema 'write' son atomiques, determina quines el contingut dels fitxers s1.txt i s2.txt al finalitzar el programa.

Normes per corregir l'exercici 4: 3 punts en total. 1 per cada apartat.
→ **NOTA**: 2 punts

5. Implementa un programa que busqui el caracter ‘z’ en un fitxer durant maxim 7 segons. El nom del fitxer el passem com a parametre per la lınia de comandes.
El programa acaba de seguida que trobi una ’z’ amb el missatge ”He trobat una z aquı: ¨i imprimeix la ultima frase que ha llegit on ha trobat la lletra ’z’. Si no troba la lletra ’z’ Treu un missatge dient que ha exhaurit el temporitzador i no ha trobat la lletra ’z’ o be que ha llegit tot el fitxer i no ha trobat la lletra ’z’.

```c
#define MAX 20
int fi=0;
void alarma(int signo)
{
   fi = 1;
   char tmp[MAXMAX];
   sprintf(tmp,"Alarma exhaurida\n");
   if(write(1,tmp,strlen(tmp))<0) perror("Error al write\n");
}

int main(int argc,char *argv[]){
   int fd,i,x=0;
   if(argc!=2) exit(-1);
   if((fd=open(argc[1],O_RDONLY))<0) perror("Error al obrir\n");

   signal (SIGALRM,&alarma);
   alarm(3);
   char x[MAX];
   int buscarz=0;
   while((n=read(fd,x,MAX))>0) && fi==0 && buscarz==0){
      if(n<0) perror("Error al llegir\n");
      for(i=0,i<n;++i){
         if((x[i]=='z') && not fi){
            alarm(0);
            buscarz=1;      //trobada la z
            printf("S'ha trobat una z\n");
         }
      }
   }
   if(buscarz==0 && n==0){
      printf("NO s'ha trobat z\n");
   }
   close(fd);
}
```

Normes per corregir l’exercici 5: 3 punts en total si:

- •el programa funciona correctament
- •la busqueda es correcta
- •el temporitzador esta ben reprogramat
- •es controla l’error de les crides al sistema

→ **NOTA**: 2 punts

6. Explica que son els sockets i quina avantatge tenen respecte els altres mètodes de comunicació entre processos que coneixes.
Amb les pipes només permet la comunicació amb els processos que estan emparentats entre si, entre pare i fill i entre fills si aquests comparteixen una pipe, pero si es vol comunicar dos procesos que no tenen relació de emparentats, es necessiten els sockets per poder fer la connexió.
ELs sockets es un dispositiu que te la implementació igual a la pipe, pero sobre operacions específiques i té un sol canal per poder fer la lectura i escriptura.

Normes per corregir l'exercici 6: 1 punt si resposta correcta i raonada

→ **NOTA**: 1 punt