Mariona Farré

Entrega Tema 2: Gestió de Processos

1. Quin serà el resultat d'executar el següent programa?

```
int g1, g2;
void main()
{
        int l1, l2;
        int pid;
        g1 = 0; g2 = 0;
        l1 = 0; l2 = 0;
        printf(\G1 = %d G2 = %d L1 = %d L2 = %d\n", g1, g2, l1, l2);
        pid = fork();
        g1 = 1; l1 = 2;
        printf(\G1 = %d G2 = %d L1 = %d L2 = %d\n", g1, g2, l1, l2);
        if (pid == 0) {
        g2 = 3;
        l2 = 4;
        }
        printf(\G1 = %d G2 = %d L1 = %d L2 = %d\n", g1, g2, l1, l2);
}
```

1r print: G1 = 0 G2 = 0 L1 = 0 L2 = 0
2r print: G1 = 1 G2 = 0 L1 = 2 L2 = 0 (pare)
3r print: G1 = 1 G2 = 0 L1 = 2 L2 = 0 (fil)
4r print: G1 = 1 G2 = 0 L1 = 2 L2 = 0 (pare)
5r print: G1 = 1 G2 = 3 L1 = 2 L2 = 4 (fill)

Normes per corregir l'exercici: 1,5 punts en total si resposta correcta i ben raonada.
→ **NOTA**: 1 punt (raonar)

2. Escriu un programa que emuli el comportament d'un shell per a l'execució de processos en foreground. Per a llegir una comanda de l'entrada estàndar, disposeu de la rutina int llegir comanda (char *com) que emmagatzema en com la comanda llegida i ens retorna la seva longitud Podeu suposar que la comanda no té arguments. Es necessari el canvi d'imatge? I crear algun fill?
Una possible implementació de la rutina llegir comanda perquè pugueu provar el programa:

```
#define MAX 20
int llegir_comanda(char *com){
        int len=0;
        char tmp[80];
        sprintf(tmp,"Escriu una comanda acabada amb Ctrl-D (exit per sortir)\n");
        write(1,tmp,strlen(tmp));
        if((len=read(0,com,MAX))==-1){
        printf("Error al llegir comanda\n");
        }
        return len;
}
```

Nota: en el cas que creeu fills, cal que tingueu cura dels zoombies. També cal fer control d'errors. I imprimir per pantalla el codi de finalització dels fills
Un exemple d'execució del codi seria el següent:

```
% ./procsShellv1
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
lsExecutant ls amb PID=69786...
Makefile funcionsAux.o procsShellv1.o procsShellv2.o procsShellv3.o
funcionsAux.c procsShellv1 procsShellv2 procsShellv3
funcionsAux.h procsShellv1.c procsShellv2.c procsShellv3.c
El proceso 69786 termina con exit code 0
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
exit
%
```

El programa s'executa sense paràmetres i mostra un missatge per pantalla Escriu una comanda acabada amb Ctrl-D (exit per sortir). En l'exemple l'usuari ha introduït¨ıt pel teclat ls i després exit per sortir.

```c
int llegir_comanda(char *com){   //funcio donada
   int len=0;
   char tmp[80];
   sprintf(tmp,"Escriu una comanda acabada amb Ctrl-D (exit per sortir)\n");
   write(1,tmp,strlen(tmp));
   if((len=read(0,com,MAX))==-1){
   printf("Error al llegir comanda\n");
   }
   return len;
}

void sortida(int pid, int code) {
   int statuscode,signcode;
   char buffer[90];
   if (WIFEXITED(code)){ //estat del programa
   statuscode=WEXITSTATUS(code);
   sprintf(buffer,"El proces fill %d  ha acabat amb el exit %d\n",pid,statuscode);
   }else{
   signcode=WTERMSIG(code);
   sprintf(buffer,"El proces fill %d  ha acabat amb el signal %d\n",pid,signcode);
   }
   write(1,buffer,strlen(buffer));
}
int main() {
   char cmd[MAX];
   int comanda;
   int pid;
   int exitcode;
   while{
   comanda=llegir_comanda(cmd);
   if (comanda>0) {
   if (strcmp(cmd,"exit")==0)
       exit(0);
   if(fork()==0){
       printf("Executant %s amb PID=%d...\n",cmd,getpid());
       execlp(cmd,cmd,NULL);
       printf("Error en executar comanda\n");
   }else{
       pid=wait(&exitcode);
       if((pid==-1)
           printf("Error en wait\n");
       else
           sortida(pid,exitcode);
       }
   }
   }
}
```

Normes per corregir l'exercici: 3 punts en total si:

- funciona correctament,
- no queden fills zoombies perquè el pare espera el seu acabament correctament i on toca perquè l'execució sigui foreground,
- es controla el possible error de totes les crides al sistema
- es recull i imprimeix el codi de finalització de tots els fills.

Resta 0,75 per cadascun dels punts anteriors que falti o sigui incorrecte.

→ **NOTA**: 2,50 punts (falten alguns errors i exits)

3. Modifica el programa anterior per a que sigui capa¸c de tractar, a més a més, comandes llan¸cades en background. Pots indicar-li que la comanda s'ha d'executar en background amb el caràcter especial & a final de comanda (tal i com ho fa el Shell de UNIX) o amb qualsevol altre caràcter que tu escullis. Recorda que aquest caràcter no forma part de la comanda i l'hauràs de treure del buffer abans d'executar la comanda. Ho pots fer substituint-lo per caràcter NULL. Per exemple:

```
if(cmd[num-1]=='&') //Mode background
{
cmd[num-1]='\0'; //Eliminem &
}
```

Nota: en el cas que creeu fills, cal que tingueu cura dels zoombies. També cal fer control d'errors. I imprimir per pantalla el codi de finalització dels fills.
Un exemple d'execució de la comanda ls en foreground primer i en background (ls&) després

```
% ./procsShellv2
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
lsExecutant fg ls amb PID=69927...
Makefile funcionsAux.h procsShellv1 procsShellv2 procsShellv2.o
funcionsAux.c funcionsAux.o procsShellv1.c procsShellv2.c procsShellv3.c
El proceso 69927 termina con exit code 0
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
ls&Escriu una comanda acabada amb Ctrl-D (exit per sortir)
Executant bg ls amb PID=69928...
Makefile funcionsAux.h procsShellv1 procsShellv2 procsShellv2.o
funcionsAux.c funcionsAux.o procsShellv1.c procsShellv2.c procsShellv3.c
exit
%
```

Noteu com en en cas de l'execució background el missatge de Escriu una comanda ... em surt immediatament.

```c
int llegir_comanda(char *com){   //funcio donada
   int len=0;
   char tmp[80];
   sprintf(tmp,"Escriu una comanda acabada amb Ctrl-D (exit per
sortir)\n");
   write(1,tmp,strlen(tmp));
   if((len=read(0,com,MAX))==-1){
   printf("Error al llegir comanda\n");
   }
   return len;
}

void sortida(int pid, int code) {
   int statuscode,signcode;
   char buffer[90];
   if (WIFEXITED(code)){ //estat del programa
   statuscode=WEXITSTATUS(code);
   sprintf(buffer,"El proces fill %d  ha acabat amb el exit
```

```
%d\n",pid,statuscode);
   }else{
   signcode=WTERMSIG(code);
   sprintf(buffer,"El proces fill %d  ha acabat amb el signal
%d\n",pid,signcode);
   }
   write(1,buffer,strlen(buffer));
}
int main() {
   char cmd[MAX];
   int comanda;
   int pid;
   int exitcode;
   while{
   comanda=llegir_comanda(cmd);
   if (comanda>0) {
   if (strcmp(cmd,"exit")==0)
       exit(0);
   if(fork()==0){
       printf("Executant %s amb PID=%d...\n",cmd,getpid());
       execlp(cmd,cmd,NULL);
       printf("Error en executar comanda\n");
   }else{
       pid=wait(&exitcode);
       if((pid==-1)
           printf("Error en wait\n");
       else
           sortida(pid,exitcode);
       }
   }
   }
}
```

Autocorrecció: 3 punts en total si:

- funciona correctament,
- l'espera dels fills es fa on toca per l'execució foreground i background
- no queden fills zoombies
- es controla el possible error de totes les crides al sistema i es recull i imprimeix el codi de finalització de tots els fills.

Resta 0,75 per cadascun dels punts anteriors que falti o sigui incorrecte.

→ **NOTA**: 2 punts (queden fills zoombies sense els exits)

4. Modifica el programa anterior perquè demani de llegir una comanda nova cada 2 segons. Exemple d'Execució:

```
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
ls Makefile funcionsAux.h procsShellv1 procsShellv2.c procsShellv3.c
funcionsAux.c funcionsAux.o procsShellv1.c procsShellv3 procsShellv3.o
El proceso 69973 termina con exit code 0
exit^D
Escriu una comanda acabada amb Ctrl-D (exit per sortir)
```

En aquest cas com que llegeix comandes cada 2 segons ens pot passar que l'usuari entri la comanda com en el cas del exit i el programa la llegeixi després, per això veiem el missatges Escriu una comanda ... després d'haver introduït la comanda.

```
int main() {
   char buff[MAX];
   int num=0,pid;
   int alarma2seg;
   signal (SIGALRM,alarma2seg);
   while {
   num=llegir_comanda(buff);
   if (num>0) {
   if (strcmp(buff,"exit")==0)
   exit(0);
   buff[num-1]='\0';
   }
   else {
   int pid=waitpid(pid,&exitcode,0);
   while(pid>0){
        sortida(pid,exitcode);
   }
   alarm(10);
   while(alarma2seg==0) pause();
   alarma2seg=0;
   }
   }
}
```

Normes per corregir l'exercici: 3 punts en total si:

- funciona correctament
- el signal es programa correctament
- el signal s'espera correctament
- no queden fills zoombies, es controla el possible error de totes les crides al sistema i es recull i imprimeix el codi de finalització de tots els fills.

Resta 0,75 per cadascun dels punts anteriors que falti o sigui incorrecte.
→ **NOTA**: 2,75 punt (les alarmes funcionen, pero falta esperar els fills quan acabin)

5. Explica en el context de planificació de processos què és el Round Robin i posa un exemple del seu funcionament diferent al dels apunts. Per exemple. si tenim 3 processos que triguen 10,20 i 5 segons i un Quantum de 3. Com seria el seu diagrama de Gantt (en quin ordre s'executarien)?

Normes per corregir l'exercici: 1,5 punts si resposta correcte i raonada

→ **NOTA**: 1 punt (raonar més)

p1= 10

p2=20 Q=3

p3=5