Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 3. Llamadas al sistema para el Control de Procesos

Nombre y apellidos:

Jesús Manuel García Palma

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de …… minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: si (si/no). En caso de haber contestado “no”, indica los motivos por los que no las has resuelto:

2. Tengo que trabajar algo más los conceptos sobre:

Proceso padre-hijo

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

Mi solución al **ejercicio 1** ha sido:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
using namespace std;


int main(int argc,char *argv[])
{
	if(argc!=2)
	{
		perror("\nError, parametro = un numero a comprobar\n");
		exit(-1);
	}
	pid_t pid;
	pid=fork();

	if(pid<0)
	{
		perror("Error en el fork\n");
		exit(-1);
	}

	if(pid==0)
	{
		//Entramos en el proceso hijo
		printf("Proceso hijo, pid: %d ; pid del padre: %d. Comprueba si el numero introducido es par o impar\n",getpid(),getppid());
		int n=atoi(argv[1]); //atoi convierte un string en integer
```

```
		if(n%2==1)
			printf("EL numero %d es impar\n",n);
		else printf("El numero %d es par\n",n);
	}else if(pid)
	  {
		printf("\n Proceso padre, pid: %d , es igual al del proceso hijo. Comprobando si el numero es divisible por 4\n",getpid());
		int n=atoi(argv[1]);
		if(n%4==0)
			printf("El numero %d introducido es divisible por 4\n",n);
		else printf("El numero %d introducido es indivisible por 4\n",n);


	  }
return 0;
}
```

Mi solución a la **ejercicio 3** ha sido:

En el primer bucle, se recorre un número de nprocesos donde childpid es igual a un proceso hijo. Si el proceso hijo es igual a -1, ha sido imposible crear el proceso hijo, y el bucle continúa. Si el proceso hijo se crea correctamente, el bucle finaliza.

El segundo bucle realiza el mismo proceso que en el anterior pero la segunda condición es distinta ya que si no existe el proceso hijo (!childpid) el bucle finaliza.

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
```

```
#include <errno.h>
#include <stdlib.h>
#include <string.h>

int main(int argc,char *argv[])
{
        if(argc!=1)
                perror("\n uso inválido\n");
                exit(-1);
        int nprocs=20;
        pid_t pid;
        pid=fork();
        for(int i=1;i< nprocs;i++){
                if(pid==-1)
                        fprintf(stderr,"Could not create child % d: %s\n",i,strerror(errno));
                        exit(-1);

                if(pid)
                        printf("Proceso hijo, pid: %d ; pid del padre: %d\n",getpid(),getppid());

        }
exit(0);
}
```

en el segundo caso utilizar el mismo código pero cambiar la segunda condición poniendo !pid.

Mi solución a la **ejercicio 4** ha sido:

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main()
{
	int j;
	pid_t PID;

	for(int i=0;i<5;i++){
		if((PID=fork())<0){
			perror("Error en fork\n");
			exit(-1);
		}
		if(PID==0){
			printf("Soy el hijo PID = %i\n",getpid());
			exit(0);
		}
	}

	for(int i=4;i>=0;i--){
		PID=wait(&j);
		printf("Ha finalizado mi hijo con PID = %i\n",PID);
		printf("Solo me quedan %i hijos vivos\n",i);
	}
}
```

Mi solución a la **ejercicio 6** ha sido:

Es un programa que indica el proceso hijo que ha finalizado y que imprime también el estado en el que finaliza.

Existe también una condición inicial que imprime un error de fork si el proceso hijo es menor que 0.