Ejemplo

Enunciado:

Escribe un programa (signals.c) que cree N procesos hijo, dónde N es el primero y único argumento del programa. Entre la creación de un hijo y el siguiente, el padre esperará a la recepción de un SIGUSR1 utilizando una espera bloqueante. Los procesos hijo deben mostrar por pantalla un mensaje con su PID y esperar 1 segundo con una espera activa. Al cabo de 1 segundo, el proceso debe mandar un SIGUSR1 a su padre y acabar. El padre debe controlar el caso en que el SIGUSR1 llegue antes de realizar la espera.

Solución:

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <string.h>

int ppid = 0;

void error_y_exit(char *msg,int exit_status)
{
    perror(msg);
    exit(exit_status);
}

void trata_alarma(int s) {
  if(s == SIGALRM){
    kill(ppid,SIGUSR1);
      exit(0);
  }
  else if(s == SIGUSR1){
  }
```

```
}

int main(int argc, char* argv[]){
    char buff[256];

    ppid = getpid();

    struct sigaction sa;
    sigset_t mask;

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigaddset(&mask, SIGUSR1);
    sigprocmask(SIG_BLOCK,&mask, NULL);

    sa.sa_handler = &trata_alarma;
    sa.sa_flags = SA_RESTART;
    sigfillset(&sa.sa_mask);

    if (sigaction(SIGALRM, &sa, NULL) < 0 ||
  sigaction(SIGUSR1, &sa, NULL) < 0) error_y_exit("sigaction", 1);

    int n = atoi(argv[1]);
    for(int i = 0; i < n; ++i){
        switch(fork()){
            case 0:
              sprintf(buff, "Hola, soy el proceso nº %d\n", getpid());
          write(1, buff, strlen(buff));
                alarm(1);
                sigfillset(&mask);
                sigdelset(&mask, SIGALRM);
                sigsuspend(&mask);
                //while(1);
                break;
            case -1:
                error_y_exit("fork",1);
                break;
            default:
                sigfillset(&mask);
                sigdelset(&mask, SIGUSR1);
                sigsuspend(&mask);
                break;
        }
    }

}
```

Enunciado:

Solución:

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <string.h>

void Usage(){
  char buffer[256];
  sprintf(buffer, "Usage: programName arg1 arg2\nEste programa hace X cosa");
  write(1, buffer, strlen(buffer));
  exit(0);
}

void error_y_exit(char *msg, int exit_status){
  perror(msg);
  exit(exit_status);
}

pid_t process_id;

int main(int argc, char* argv[]){
  char buff[256];

  int n = atoi(argv[1]);

  sigset_t mask;
  sigfillset(&mask);

  for(int i = 0; i < n; ++i){
    int pid = fork();
    switch(pid){
      case 0:
        if(i == 0) {
          sprintf(buff, "El PID de mi padre es %d.\n", getppid());
          write(1,buff,strlen(buff));
        }
        else{
          sprintf(buff, "El PID de mi hermano mayor es %d.\n", process_id);
          write(1,buff,strlen(buff));
        }
        exit(0);
        break;

      case -1:
        error_y_exit("fork", 1);
        break;

      default:
        process_id= pid;
        break;
    }
  }
  sigprocmask(SIG_BLOCK, &mask, NULL);
  while(waitpid(-1,NULL,0) < 0){
  }
  sigprocmask(SIG_UNBLOCK, &mask, NULL);
```

```
  exit(0);
}
```