```
/*
          hijo.c
           Tarea del hijo
            no hacer nada y dormir intermitentemente.
       linea de compilacion     cc hijo.c -o hijo
*/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define Romer 1

int main(void)
    {
      printf(" \n ******* Fui creado, ya estoy aquí, me conocen como %d ******* \n
",getpid());
      while(Romer) sleep(1);
      exit(0);
    }
```

```
/*
       padre.c

       cc padre.c –o padre

*/

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

#define MAXPROCESS 8

void InicializarVector(pid_t []);
void MostrarVector(pid_t []);
void CrearProceso(pid_t []);
void InformeProcesos(void);
void TerminarTodosLosHijos(pid_t []) ;
void TerminarUnHijo(pid_t []) ;
void EliminarZombies(pid_t []);

int Menu(void);

int main(void)
```

```
{
 pid_t vecpids[MAXPROCESS] ;
 InicializarVector(vecpids) ;
 int op ;
 while ((op = Menu()) != 0)
 {
   switch(op)
   {
     case 1: MostrarVector(vecpids); break ;
     case 2: CrearProceso(vecpids) ; break ;
     case 3: InformeProcesos() ; break ;
     case 4: TerminarTodosLosHijos(vecpids) ; break ;
     case 5: TerminarUnHijo(vecpids) ; break ;
     case 6: EliminarZombies(vecpids);break;
     default: break;
   }
 }
 system("killall hijo");
 exit(0) ;
}


void  InicializarVector(pid_t vecpids[])
{
 int i ;
 for (i = 0 ; i < MAXPROCESS ; i++)
   vecpids[i] = 0 ;
}

void  MostrarVector(pid_t vecpids[])
{
 int i ;
 for (i = 0 ; i < MAXPROCESS ; i++)
   printf(" vecpids[%d] = %d \n",i,vecpids[i]) ;
}

void CrearProceso(pid_t vecpids[])
{
 int i ;
 pid_t pid;
 for (i = 0 ; i < MAXPROCESS && vecpids[i] != 0; i++) ;
 if (i == MAXPROCESS)
     printf("\n ******** INTENTA SUPERAR MAXPROCESS ******** \n");
 else
    {
     pid = fork() ;
     if (pid == 0)
       {
         if ( execl("/home/pi/TallerC/Clase5/hijo",0) == - 1)
           printf(" \n ************** Hijo execl error ***************\n");
```

```
        }
       else
       {
        printf(" \n ********  Padre actualiza vecpids[%d] = %d ******** \n",i,pid);
        vecpids[i] = pid ;
       }
     }
}

void InformeProcesos(void)
{
     system("ps -l");
}


void TerminarTodosLosHijos(pid_t vecpids[])
{
     int i ;
     for (i = 0 ; i < MAXPROCESS; i++)
      {
       if (vecpids[i] != 0)
          kill(vecpids[i],SIGKILL);
      }
}

void TerminarUnHijo(pid_t vecpids[])
{
  int i ;
  pid_t pid ;
  printf("\n ------ Ingrese PID del proceso que quiere terminar ---> ");
  scanf("%d",&pid);
  for (i = 0 ; i < MAXPROCESS && vecpids[i] != pid; i++);
  if (i == MAXPROCESS)
     printf("\n ******** PID INGRESADO NO VALIDO ******** \n");
  else
     kill(vecpids[i],SIGKILL);
}

void EliminarZombies(pid_t vecpids[])
{
  pid_t pid ;
  while((pid = waitpid(-1,NULL,WNOHANG)) > 0)
   {
     printf(" \n ******** ELIMINA HIJO ZOMBIE PID = %d ******** \n",pid);
     int i ;
     for (i = 0 ; i < MAXPROCESS && vecpids[i] != pid; i++);
     vecpids[i] = 0;
   }
}
```

```
int Menu(void)
{
  int opcion;
  printf("1 - Mostrar el contenido del vector \n");
  printf("2 - Crear un proceso\n");
  printf("3 - Mostrar con el comando ps -l el informe de procesos \n");
  printf("4 - Terminar todos los hijos \n");
  printf("5 - Terminar un hijo \n");
  printf("6 - Eliminar zombies \n");
  printf("0 - Salir \n");
  printf(" Ingrese opcion ---> ");
  scanf("%d",&opcion);
  return opcion;
}
```