/PARCIAL 1b)

① El sig. código representa la solución a la sig. problemática
Un proceso crea un cto de procesos hijos al mismo nivel (hermanos) y se le asigna a cu una tarea determinada, el proceso debe indicar a su proceso padre un estado de finalización de la tarea.
1 → exito 0 → no exito

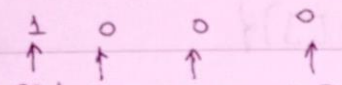
```
int tarea (int);
int main (int argc, char *argv[]){
    pid_t pid, pidTer;
    int status = 0;
    int np = atoi (argv[1]);

    while (np) {
        pid == fork();
        np--;
        If (pid == 0) exit( tarea (getpid()));
    }

// completar. para que no se hagan zombies ni huerfanos
   cada hijo debe tener un estado de finalizacion y PID.

    exit(0);
}

Int tarea (int num){
    sleep();
    print ("tarea %d \n", num);
    return num+2;
}
```

lo que falta

```
while (np) {
    pidter = wait (&status);
    printf = ("termino proceso %d y su estado fue %d \n",
      pidter ~~getpid()~~, status);
    np --;
}
```

→ status/256/WEXITSTATUS(status)

(2) Reescribo las funciones de los hilos para que el sig. código en lenguaje C permita que 3 hilos se sincronicen e impriman "soy hilo1" "soy hilo2" "soy hilo3".



Esquema (guía)

H1 (m1)	H2 (m2)	H3 (m3)
mutex m1 lock	mutex m2 lock	mutex m3 lock
Imprime H1	Imprime H2	Imprime H3
mutex m2 unlock	mutex m11 lock (VN)	mutex m1 unlock
mutex m11 lock		
Imprime H1		
mutex m3 unlock		

Programa

```
void Hilo2 (void);
void Hilo1 (void);
void Hilo3 (void);

Pthread_mutex_t m1 = pthread_mutex_Inicializer;
pthread_mutex_t m2 = pthread_mutex_Inicializer;
pthread_mutex_t m11 = pthread_mutex_inicializer;
pthread_mutex_t m3 = pthread_mutex_inicializer;
```

```
int main (void){
        pthread h1;
        pthread h2;
        pthread h3;

        pthread_mutex_lock (&m2);
        pthread_mutex_lock (&m11);
        pthread_mutex_lock( &m3);

pthread_create = (&h1, NULL, (void *)Hilo1, NULL );
pthread_create = (&h2, NULL, (void*) Hilo2 NULL);
pthread_create (&h3; NULL (void*) Hilo3, NULL);
Pthread_join (h2, NULL);
exit(1).
}
```

PARA QUE SE EJECUTE DE MANERA INFINITA.

```
Void Hilo1 (void){ ---------> while (1){
    pthread_mutex_lock (&m1);
      printf("soy hilo1 /n");
    pthread_mutex_unlock (&m2);
    Pthread_mutex_lock(&m11);
      Printf ("soy hilo1 /n");
    pthread_mutex_unlock(&m3);
}}

Void Hilo2 (void){ ---------> while (1){
    Pthread_mutex_lock(&m2);
      printf ("soy hilo2 /n");
    pthread_mutex_unlock(&m11);
}}

Void Hilo3 (void){
    Pthread_mutex_lock(&m3); ---------> while (1){
      printf ("soy hilo3 /n");
}} Pthread_mutex_unlock (&m1);
```

(3) un proceso padre se sincroniza con un hijo utilizando señales. Comienza el padre escribiendo "soy el padre". Luego escribe el hijo "soy el hijo"] repite infinitamente

- se debe completar el código.
- Además padre e hijo no deben recibir otra señal que no sea las dadas

↳ Bloqueo con sigblockmask
↳ sigprocmask.

```
void f-signalPadre(void);
void f-signalHijo(void);

Int main(void) {
    pid-t pid;
    pid=fork();
      If(pid==0) {
          signal(SIGUSR2, f-signalHijo);
          while(1) {
              kill(getppid(); SIGUSR1);
              Pause();
          }
      } else {
          signal(SIGUSR1, f-signalPadre);
          while (1) {
              Pause();
              kill(pid(); SIGUSR2);
          }
      }
    exit(0);
}

Void f-signalPadre(void) {
    Printf("soy el padre \n");
}
Void f-signalHijo(void) {
    printf("soy el hijo \n");
}
```

```
sigset_t set;
sigsetmtyset(&set);
sigaddset(&set, SIGUSR2);
sigprocmask(SIG_BLOCK, &set, NULL);
```

```
sigset_t set;
sigsetmtyset(&set);
sigaddset(&set, SIGUSR1);
sigprocmask(sy-block, &set, NULL);
```

se pausa primero porque necesito esperar la señal del hijo.

- P10:

⑤ ¿Qué es una interrupción?

Es un evento que impide el correcto flujo o traza de lo ejecutado por el procesador.

⑥ dif entre trap / interrupción

Una interrupción es causado por un evento externo al proceso en ejecución

ej: int E|S, reloj.

un trap está causado por un fallo en el proceso en ejecución

⑦ ¿Qué sucede si un proceso queda indefinidamente dentro de un S.C?

Si esto ocurre el proceso nunca libera los recursos que está utilizando y otro proceso no podrá hacer uso de estos (ABRAZO MORTAL).

- pg:
  - 4 razones que definen a un proceso
    - Una unidad que presta uso del procesador (hilo de ejec.)
    - Una entidad que tiene un estado actual, instrucciones, recursos asociados.
    - un programa en ejecución
    - Una instancia del computador.

- DIF ENTRE CLUSTER / SMP.

la dif es que en SMP se presentan las mismas unidades de memoria y E/S gestionadas por un SO.

y en un cluster c/u presenta su módulo de memoria y E/S.