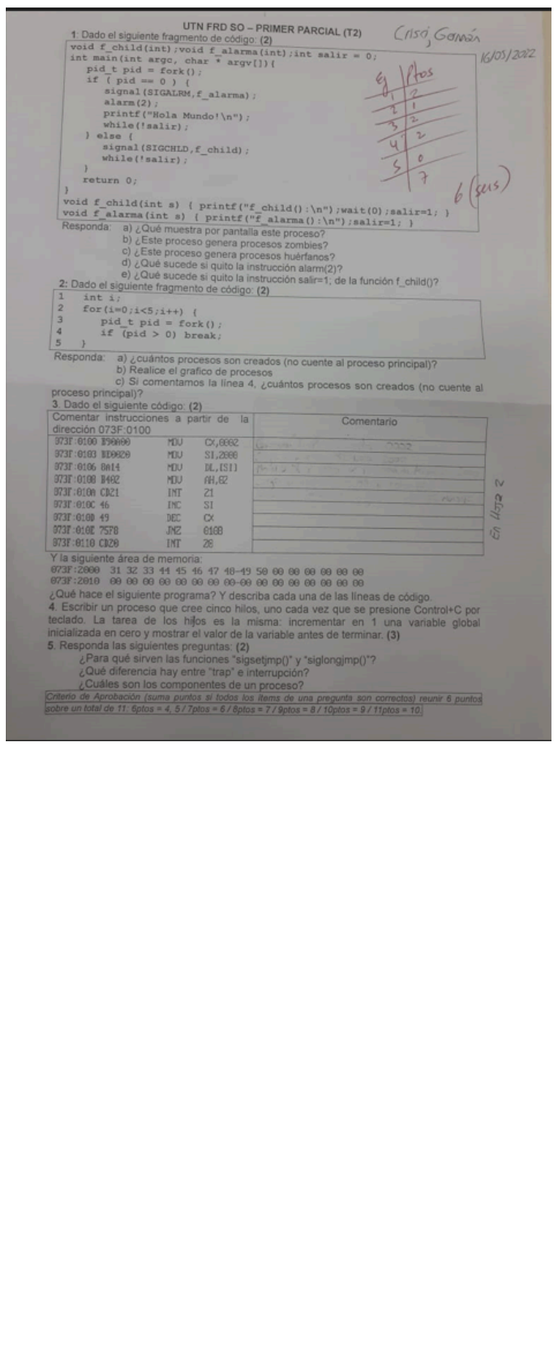
Parcial 16-05-2022



UTN FRD SO – PRIMER PARCIAL (T2)

1: Dado el siguiente fragmento de código: (2)

```
void f_child(int);void f_alarma(int);int salir = 0;
int main(int argc, char * argv[]){
  pid_t pid = fork();
  if ( pid == 0 ) {
     signal(SIGALRM,f_alarma);
     alarm(2);
     printf("Hola Mundo!\n");
     while(!salir);
  } else {
     signal(SIGCHLD,f_child);
     while(!salir);
  }
  return 0;
}
void f_child(int s) { printf("f_child():\n");wait(0);salir=1; }
void f_alarma(int s) { printf("f_alarma():\n");salir=1; }
```

Responda: a) ¿Qué muestra por pantalla este proceso?
b) ¿Este proceso genera procesos zombies?
c) ¿Este proceso genera procesos huérfanos?
d) ¿Qué sucede si quito la instrucción alarm(2)?
e) ¿Qué sucede si quito la instrucción salir=1, de la función f_child()?

2: Dado el siguiente fragmento de código: (2)

```
1  int i;
2  for(i=0;i<5;i++) {
3     pid_t pid = fork();
4     if (pid > 0) break;
5  }
```

Responda: a) ¿cuántos procesos son creados (no cuente al proceso principal)?
b) Realice el grafico de procesos
c) Si comentamos la línea 4, ¿cuántos procesos son creados (no cuente al proceso principal)?

3. Dado el siguiente código: (2)

Comentar instrucciones a partir de la dirección 073F:0100			Comentario
073F:0100 B90800	MOV	CX,0002	
073F:0103 BE0020	MOV	SI,2000	
073F:0106 8A14	MOV	DL,[SI]	
073F:0108 B402	MOV	AH,02	
073F:010A CD21	INT	21	
073F:010C 46	INC	SI	
073F:010D 49	DEC	CX	
073F:010E 75F8	JNZ	0108	
073F:0110 CD20	INT	20	

Y la siguiente área de memoria:
073F:2000 31 32 33 44 45 46 47 48-49 50 00 00 00 00 00 00
073F:2010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

¿Qué hace el siguiente programa? Y describa cada una de las líneas de código.

4. Escribir un proceso que cree cinco hilos, uno cada vez que se presione Control+C por teclado. La tarea de los hijos es la misma: incrementar en 1 una variable global inicializada en cero y mostrar el valor de la variable antes de terminar. (3)

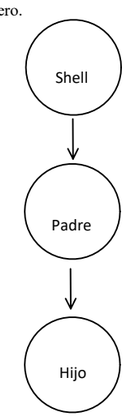
5. Responda las siguientes preguntas: (2)
¿Para qué sirven las funciones "sigsetjmp()" y "siglongjmp()"?
¿Qué diferencia hay entre "trap" e interrupción?
¿Cuáles son los componentes de un proceso?

Criterio de Aprobación (suma puntos si todos los ítems de una pregunta son correctos) reunir 6 puntos sobre un total de 11: 6ptos = 4, 5 / 7ptos = 6 / 8ptos = 7 / 9ptos = 8 / 10ptos = 9 / 11ptos = 10

1. SEÑALES

Flujo del programa: el programa presenta dos señales SIGCHLD y la Alarm. Se crea un hijo que presenta un pid_t
Luego si es hijo entra al if y realiza la función alarma que imprime Hola Mundo, luego hace la función alarma imprime "f_alarma(): " y cambia el valor de la variable salir del hijo a 1 y sale ya que la variable del hijo se modifico
En cambio si es padre realiza el else llama a la función SIGCHLD se imprime "f_child()" se hace un wait (0) y se modifica el valor de salir a 1, el padre sale del ciclo while ya que la variable salir es distinto de cero.

Diagrama de procesos



A. ¿Qué muestra por pantalla este proceso?

El proceso muestra por pantalla
Hola Mundo! => cuando hace la instrucción alarma pasa a la siguiente línea y hace hola mundo
f_alarma
f_schld

B. ¿Este proceso genera zombies?

Este proceso no genera zombies ya que el padre con la función SIGCHLD hace un wait(0) que permite retornar la información del proceso hijo.

C. ¿Este proceso genera huérfanos?

Este proceso no genera procesos huérfanos ya que el proceso hijo realiza su proceso antes que el padre por lo tanto el padre nunca realizará el proceso antes que el hijo.

D. ¿Qué sucede si quitamos la instrucción alarm(2)?

Si quitamos la instrucción alarm(2) el proceso hijo no llamaría a la señal alarm y no se cambiaria el valor de la variable salir de esta manera el hijo queda en una espera activa ya que no se modifica el valor de la variable salir.

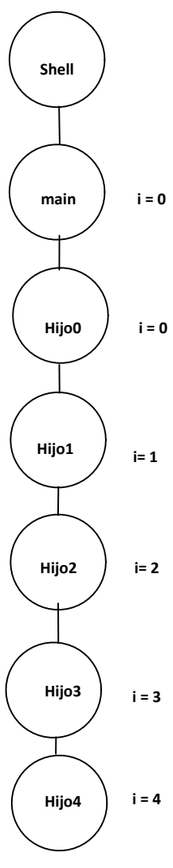
E. ¿Qué sucede si quito la instrucción salir = 1, de la función f_child()?

Si quitamos esa instrucción la variable salir no cambiaría su valor y por lo tanto quedará en una espera activa.

2. PROCESOS

A. ¿Cuántos procesos son creados?
Son creados 5 procesos sin contar el shell

B.



C. 2^n; $2^5 = 32$ sin contar el shell ≥ 31

3. ASSEMBLER

Muevo la posición 0002 al registro Cx
Muevo la posición 2000 al registro SI
Muevo el contenido de SI al registro bajo de Dx
Muevo la función 02 al registro alto AH. Función 02 para mostrar en pantalla
Interrupción 21
Incremento en 1 el puntero del registro SI
Decremento en 1 el puntero del registro CX
Hago un salto a la instrucción 0100
Finalización del Programa

El programa imprime 31-31-33-44-45-46-47-48-49-50 (VER EN TABLA ASCII)
Muestra en pantalla lo guardado en la memoria 1,2,3,D,E,F,G,H,I,P

4. HILOS

Como pide usar el Control + C debemos usar la función Sigint, como se espera la finalización del proceso se usa el PAUSE();

Declaro las señal sigint y el hilo

```
Void func_sigint(int);
Void Hilo1(int);

Contador = 0; //variable global
```

Declaro la función main

```
Int main (void){

        Pthread h1;
        Signal(SIGINT, func_signint);

        //ciclo de creación de hilos

        For(i=0. i<5, i++){

                Printf("presione control C para crear un hilo/n");
                Pause();
        }

        Return 0;
}
```

Función Hilo

```
Void hilo1(int){

        Contador ++;
        Printf("%d", contador);

}
```

Función Sigint (crea los hilos)

```
Void func_sigint(int s){
```

```
    pthread.create(&h1, null, (void)*Hilo1, null);
    pthread_join(hilo1,null); (espera del hilo)
}
```

5. TEORÍA

¿Para qué sirven las funciones sigsetjmp() y siglongjmp()?

Ambas funciones sirven para realizar saltos entre las líneas de código no necesariamente locales.

Sigsetjmp() lo que hace es determinar a qué línea de código se saltará cuando se realice el salto con **Siglongjmp()**.

Siempre se debe realizar un Sigsetjmp() antes de realizar un Siglongjmp().

¿Qué diferencia hay entre trap e interrupción?

Trap se refiere a una forma de interrupción que se da dentro del proceso interrumpido asociado a un error o excepción.

Interrupción se da en base a algo externo al proceso en ejecución.

Componentes de un proceso

1. Datos (variables / buffers)
2. Programa Ejecutable
3. Contexto de Ejecución