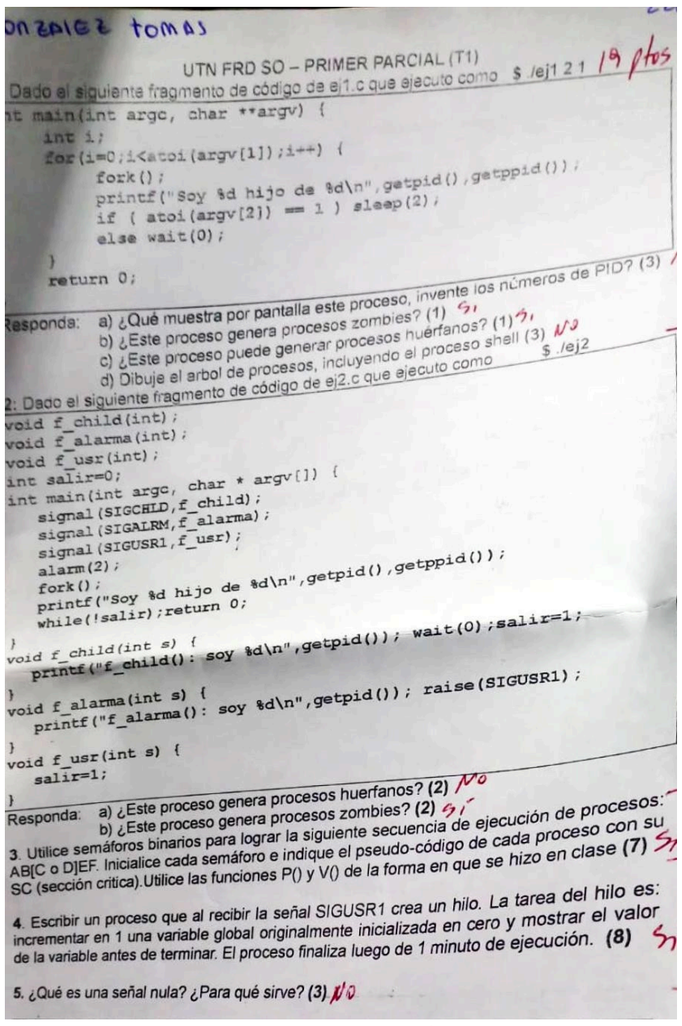
Parcial 12/05 T1

lunes, junio 05, 2023 2:38 PM



UTN FRD SO – PRIMER PARCIAL (T1)

Dado el siguiente fragmento de código de ej1.c que ejecuto como $ ./ej1 2 1 (9 ptos)

```
int main(int argc, char **argv) {
    int i;
    for(i=0;i<atoi(argv[1]);i++) {
        fork();
        printf("Soy %d hijo de %d\n",getpid(),getppid());
        if ( atoi(argv[2]) == 1 ) sleep(2);
        else wait(0);
    }
    return 0;
}
```

Responda: a) ¿Qué muestra por pantalla este proceso, invente los números de PID? (3)
b) ¿Este proceso genera procesos zombies? (1)
c) ¿Este proceso puede generar procesos huérfanos? (1)
d) Dibuje el arbol de procesos, incluyendo el proceso shell (3)

2: Dado el siguiente fragmento de código de ej2.c que ejecuto como $ ./ej2

```
void f_child(int);
void f_alarma(int);
void f_usr(int);
int salir=0;
int main(int argc, char * argv[]) {
    signal(SIGCHLD,f_child);
    signal(SIGALRM,f_alarma);
    signal(SIGUSR1,f_usr);
    alarm(2);
    fork();
    printf("Soy %d hijo de %d\n",getpid(),getppid());
    while(!salir);return 0;
}
void f_child(int s) {
    printf("f_child(): soy %d\n",getpid()); wait(0);salir=1;
}
void f_alarma(int s) {
    printf("f_alarma(): soy %d\n",getpid()); raise(SIGUSR1);
}
void f_usr(int s) {
    salir=1;
}
```

Responda: a) ¿Este proceso genera procesos huerfanos? (2)
b) ¿Este proceso genera procesos zombies? (2)

3. Utilice semáforos binarios para lograr la siguiente secuencia de ejecución de procesos: AB[C o D]EF. Inicialice cada semáforo e indique el pseudo-código de cada proceso con su SC (sección critica).Utilice las funciones P() y V() de la forma en que se hizo en clase (7)

4. Escribir un proceso que al recibir la señal SIGUSR1 crea un hilo. La tarea del hilo es: incrementar en 1 una variable global originalmente inicializada en cero y mostrar el valor de la variable antes de terminar. El proceso finaliza luego de 1 minuto de ejecución. (8)

5. ¿Qué es una señal nula? ¿Para qué sirve? (3)

```
Void Hilo 1(void);
Void h_sig1(int);
Void h_alarm(int);


int varglobal = 0;
int salir = 0;

Int main (void){

pthread &h1;
Signal(SIGUSR1, h_sig1);
Signal(ALARM, h_alarm);

while(!salir){
        alarm(60);
        pause();
        }

        return 0;
}

void Hilo1(void){
        Varglobal ++;
        printf("%d\n", varglobal);
}

void h_alarm(int s){
        salir = 1;
}

void h_sig1(int s){
        pthread_create(&h1, NULL,
        (void*)Hilo1, NULL);
        pthread_join(h1,NULL);
```

1).

```
$./ej1 2 1 12
for (i=0, i<atoi(argv[1]), i++)
    fork();
printf ("soy %d hijo de %d \n", getpid(), getppid());
    if (atoi(argv[2]) == 1)
        sleep(2),
else wait(0);
return 0,
```

A).

Soy 2000 hijo de 1900
Soy 2002 hijo de 2000

Soy 2000 hijo de 1900
Soy 2001 hijo de 2000

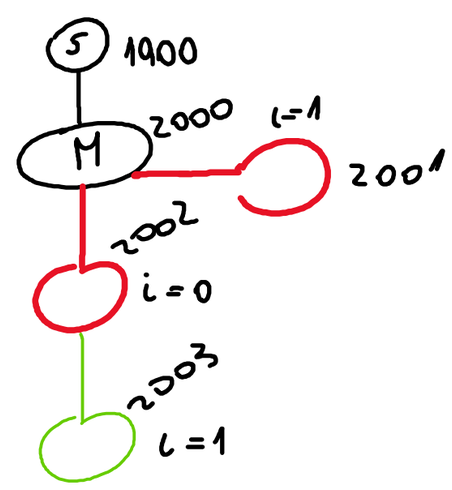
Soy 2002 hijo de 2000
Soy 2003 hijo de 2002

B).

El proceso genera procesos zombies ya que no se retorna la información del proceso hijo en caso de que el hijo finalice antes que el padre.
En caso de que el hijo quede huerfano la información será retornada a un proceso init.

C). El proceso puede generar procesos huérfanos ya que se hace un sleep de dos segundos y si el padre hace la espera y termina y el hijo todavía no finalizo este queda huérfano.

D).



2).

Flujo del Programa: El programa crea las funciones SIGUSR, SCHLD, ALARM, luego de esto el proceso padre ejecuta la alarma de 2 segundos y crea un proceso hijo. Después de esto tanto el padre como el hijo imprimen el printf. Como el padre hace la señal alarma se llama a la señal SIGUSR y se modifica el valor de la variable salir y sale del ciclo. En cambio el hijo nunca recibe la señal alarma entonces no cambia su valor de la variable salir y queda en el while generando una espera activa.

A). **¿Este proceso genera procesos huérfanos?**

El proceso genera procesos huérfanos ya que el padre va a terminar antes que el hijo ya que este queda en una espera activa.

B. **¿Este proceso genera procesos zombies?**

El proceso no será zombies ya que el proceso hijo le retornará su información al proceso init, ya que este queda huerfano

3).

A Wait(Sa)	B Wait(Sb)	C Wait(Scd)	D Wait(Scd)	E Wait(Se)	F Wait(Sf)
SC	SC	SC	SC	SC	SC
Signal(Sb)	Signal(Scd)	Signal (Se)	Signal(Se)	Signal(Sf)	Signal(Sa)

Inicialización:	1	0	0	0	0

4).

```
Void hilo1(int);
Void sigusr1(int);
Void alarma(int);

Contador = 0;
Salir = 0;

Int main (void){

    Signal(sigusr1, SIGUSR1);
    Signal (alarma, SIGALARM)
    pthread h1;

    raise(sigusr1); ? Para llamar a la función sigusr1
    alarm(60);
    while(!salir) pause();  //para que no se genere una espera activa
    return 0;
}

Void alarma(int s){
    salir = 1;
}

Void hilo1(int){
     Contador++;
     Printf("%d", contador);
}

Void sigusr1(int s){
    Pthread_create(%h1, NULL, (void)*hilo1, NULL);
    Pthread_join(hilo1, NULL);
}
```

5).

¿Qué es una señal nula?

La señal nula es aquella que presenta valor nulo y se utiliza para determinar si un proceso está en ejecución o no, como así también determinar si el proceso en cuestión no existe.