NOMBRE Y APELLIDOS: _____NIA:_____

Examen de Sistemas Operativos
1º Parcial
19 de Octubre de 2009

NOTAS:

* La fecha de publicación de las notas, así como de revisión se notificarán por Aula Global
* Para la realización del presente examen se dispondrá de 1 hora y 20 minutos.
* El examen se contesta en las hojas dadas con el enunciado.
* No se pueden utilizar libros ni apuntes, ni usar móvil (o similar)
* Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

---

Ejercicio 1 (2 puntos). Dado el programa que se muestra a continuación:

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

main () {
  int pid,i;

  printf("INICIO\n");
  for (i=0; i<2; i++){
    pid=fork();
    if (pid == 0){
       sleep (1);
       printf("UNO\n");
       pid=fork();
       if (pid == 0) {
         printf("TRES\n");
         exit (0);
       }
    }
    else {
       sleep (2);
       printf(":DOS\n");
    }
  }
}
```

A- Indicar cuántas veces aparecen las palabras UNO, DOS y TRES en pantalla y en que momentos de la ejecución, tomando como 0 el momento en el que se escribe INICIO

B- Indicar la jerarquía de procesos creados utilizando la notación padre, hijo, nieto, bisnieto y mostrando claramente las relaciones jerárquicas y el orden de creación de los procesos dentro de su mismo nivel.

Ejercicio 2 (5 puntos). Se desea realizar una aplicación que pida números al usuario y simultáneamente escriba una secuencia de números que empiece en el recibido.
El programa debe pedir un número por pantalla y después crear un hijo.

El hijo debe escribir la secuencia de números enteros desde el número recibido hacia los números mayores. El hijo finalizará su ejecución cuando reciba la señal SIGUSR1 y escribirá en pantalla FINAL POR ORDEN.

NOMBRE Y APELLIDOS: _____NIA:_____

El padre deberá seguir pidiendo números mientras los hijos ejecutan su tarea. Cada vez que lea un nuevo número deberá mandar la señal SIGUSR1 al hijo anterior para que finalice (si no es el primer número recibido) y crear uno nuevo para que haga lo indicado para el hijo.
En cada momento no puede haber más de 1 hijo en ejecución.
El padre finalizará cuando hayan pasado 60 seg. desde que empezó su ejecución. Antes de finalizar deberá finalizar los hijos creados y esperar su finalización.

Ejercicio 3 (3 puntos). Responder brevemente a las siguientes preguntas:

A- ¿Qué es un trap?

B- ¿Qué es el BCP? ¿Qué información almacena?

C- ¿Cuándo se produce un cambio de contexto involuntario de un proceso?

D- Indica y explica 3 medidas de valoración de la planificación de procesos.

NOMBRE Y APELLIDOS: _____NIA:_____

Soluciones

Ej1:
La palabra UNO aparece 3 veces
La palabra DOS aparece 3 veces
La palabra TRES aparece 3 veces

0:INICIO
1:UNO
1:TRES
2:DOS
2:UNO
2:TRES
3:UNO
3:TRES
3:DOS
4:DOS

La jerarquía creada es:
-padre
--hijo0
----nieto1
----nieto2 exámenes
------bisnieto2
--hijo1
----nieto2

Ej2

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

int seguir = 1;

void terminarPadre (int s) {
  seguir = 0;
  printf ("Alarma\n");
}
void terminarHijo (int s) {
  seguir = 0;
  printf ("FINAL POR ORDEN");
}
hijo(int i){
  struct sigaction act;

  act.sa_handler = terminarHijo;
  sigemptyset(&act.sa_mask);
  act.sa_flags = 0;
  sigaction(SIGUSR1, &act, NULL);
  while (seguir ){
    printf ("%d\n",i);
    i++;
    sleep(1);
```

NOMBRE Y APELLIDOS: _____NIA:_____

```
  }
  exit(0);
}
main () {
  int n,pid=0,numhijos=0,i;
  struct sigaction act;

  act.sa_handler = terminarPadre;
  sigemptyset(&act.sa_mask);
  act.sa_flags = 0;
  sigaction(SIGALRM, &act, NULL);
  alarm (60);

  do {
    printf ("Introducir nzmero :");
    scanf ("%d", &n);
    if (pid != 0) kill (pid, SIGUSR1);
    if ( seguir) {
      pid=fork();
      switch (pid) {
        case -1: printf ("Error en la creacisn del hijo");
                 break;
        case 0 : hijo(n);
                 break;
        default: numhijos++;
      }
    }
  } while ( seguir );
  printf ("Espero por %d hijos \n", numhijos);
  i=0;
  while (i<numhijos)
    if (wait (NULL)>0) i++ ;
}
```

Ej3:

A- ¿Qué es un trap?

TRAP: los traps son interrupciones sw invocadas por el usuario desde el programa . El so pasará a ejecutar el manejador del trap asociado (su rutina de atención a la interrupción o ISR).Pueden ser invocadas expresamente con instrucciones trap o ser provocadas por situaciones inesperadas (división por 0)

B- ¿Qué es el BCP? ¿Qué información almacena?

Bloque de control de Procesos (BCP): Cada entrada de la tabla que mantiene la información sobre un proceso.
En el BCP se mantiene casi toda la información sobre un proceso.

- Información de identificación.
- Estado del procesador.
- Información de control del proceso

Algunos elementos de información se mantienen fuera por motivos de implementación.

NOMBRE Y APELLIDOS: _____NIA:_____

C- ¿Cuándo se produce un cambio de contexto involuntario de un proceso?

SO quita de la CPU al proceso
En ejecución → listo
Ejemplos: fin de rodaja de ejecución o pasa a listo proceso bloqueado de mayor prioridad
¿Motivo? ⇒ Reparto del uso del procesador

D- Indica y explica 3 medidas de valoración de la planificación de procesos.

- Utilización de CPU:
  - Porcentaje de tiempo que se usa la CPU.
  - Objetivo: Maximizar.
- Productividad:
  - Número de trabajos terminados por unidad de tiempo.
  - Objetivo: Maximizar.
- Tiempo de retorno (T_q)
  - Tiempo que está un proceso en el sistema. Instante final (T_f) menos instante inicial (T_i).
  - Objetivo: Minimizar.
- Tiempo de servicio (T_s):
  - Tiempo dedicado a tareas productivas (cpu, entrada/salida). $T_s = T_{CPU} + T_{E/S}$
- Tiempo de espera (T_e):
  - Tiempo que un proceso pasa en colas de espera.
    $T_e = T_q \quad T_s$
- Tiempo de retorno normalizado (T_n):
  - Razón entre tiempo de retorno y tiempo de servicio.
    $T_n = T_q / T_s$
  - Indica el retardo experimentado.