Examen Parcial de Sistemas Operativos - Marzo de 2015

Grado en Ingeniería Informática

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 1:50 horas.
  * No se pueden utilizar libros ni apuntes
  * Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

NOMBRE: GRUPO:

Ejercicio 1 (2 puntos) . Autotest.

Responda a las preguntas del autotest en los cuadros adjuntos indicando la letra de la respuesta válida. Recuerde que por cada 3 fallos se quita un punto. No contestadas no penalizan.

1	2	3	4	5	6	7	8	9	10

1.- Dado un proceso que ha creado un pipe, ¿con cuál de estos procesos podría potencialmente comunicarse a través del mismo?
A) Con su proceso padre.
B) Con un proceso nieto.
C) Con un proceso hermano.
D) Con cualquier proceso del sistema.

2.- ¿Que hace la función int kill(pid_t pid, int sig) ?
A) Envía una señal de terminar al proceso "pid"
B) Termina el proceso que llama a la función y envía la señal "sig" al proceso "pid"
C) Envía al proceso "pid" la señal "sig"
D) Fuerza al proceso "pid" a terminar

3.- ¿Cuál es la definición más adecuada de un proceso zombie?
A) Aquel que ejecuta exit() , pero su padre no hizo wait()
B) Aquel que su padre terminó antes que él
C) Aquel que no tiene descendientes
D) Aquel proceso cuyo código está siendo recogido por su padre

4.- ¿Qué NO comparten los threads de un mismo proceso?
A) Los descriptores de ficheros abiertos.
B) El código.
C) Los datos globales al proceso.
D) El puntero de pila.

5.- ¿Cuál de los siguientes algoritmos de planificación del procesador produce menos cambios de contexto?

A) FIFO (FCFS) expulsivo.
B) FIFO (FCFS) no expulsivo.
C) El del menor tiempo restante (Shortest Remaining Time) .
D) Round-Robin.

6.- ¿Qué es cierto acerca del código de un mandato del intérprete de mandatos de un Sistema Operativo?
A) Ejecuta en modo núcleo (kernel) .
B) Siempre ejecuta con derechos de superusuario.
C) Ejecuta en modo usuario.
D) Operan directamente con controladores de dispositivos.

7.- ¿Cuantas API puede ofrecer un SO simultáneamente?
A) No puede ofrecer simultáneamente APIs
B) 3
C) 10
D) Varias

8.- ¿Cual de las siguiente afirmaciones es falsa?
A) Al terminar un proceso los recursos asignados tienen que ser liberados manualmente
B) El kernel comunica al proceso padre de la terminación del proceso
C) Un proceso puede terminar voluntaria e involuntariamente
D) Todas las anteriores

9.- El orden de fases de arranque de un sistema operativo es:
A) Iniciador ROM, Parte residente de SO, Cargador de SO, Normal ejecución de SO
B) Cargador de SO, Parte residente de SO, Iniciador ROM, Normal ejecución de SO
C) Iniciador ROM, Cargador de SO, Parte residente de SO, Normal ejecución de SO
D) Ninguna de las anteriores es correcta

10.- ¿Cuál puede ser una desventaja de un sistema operativo estructurado en cliente/servidor?
A) Se puede extender de una forma sencilla a un modelo distribuido
B) Es muy flexible respecto a otros sistemas operativos
C) Hay una sobrecarga en la ejecución de los servicios
D) Todas son desventajas de un sistema operativo estructurado en cliente/servidor

Ejercicio 2 (4 puntos) .

a) Indicar la llamada al sistema para crear una tubería. Explicar para qué sirve y cómo funciona. Haga un esquema.

b) Realizar un programa en C que cree dos hijos que se comunicarán utilizando tuberías (las que sean necesarias). Al empezar la ejecución del hijo 1, éste escribirá el número 1 en tubería.
El hijo 2 deberá: leer de la tubería en la que ha escrito el hijo 1, mostrar el número

recibido por pantalla con el mensaje Hijo 2: recibido número N, aumentar el número recibo en una unidad y mandar el número incrementado al hijo 1.
El hijo 1 deberá leer de la tubería en la que ha escrito el hijo 2, mostrar el número recibido por pantalla con el mensaje Hijo 2: recibido número N, aumentar el número recibo en una unidad y mandar el número incrementado al hijo 2.
Estos envíos continuarán hasta que el padre reciba 5 veces la señal SIGINT. En ese momento enviará la señal SIGUSR1 a ambos hijos y esperará su finalización.
Los hijos, cuando reciban la señal SIGUSR1 se asegurarán de que no quedan números en la tubería y finalizarán.

Solución

a) La llamada al sistema para la creación de una tubería es pipe(int mitubería []) que debe recibir un array de 2 casillas (declarado como int mituberia[2]). En la casilla 0 el SO colocará el número de descriptor correspondiente a la lectura de datos de la tubería y en la casilla [0] el correspondiente a la escritura de datos en la tubería.

Es decir, en condiciones normales y si no se ha realizado ninguna otra apertura de tuberías ni de ficheros, etc, en la posición mitubería[0] se almacenará el número 3 y en mitubería[1] se almacenaría el 4. Luego se utilizarán dichos descriptores para leer (read(mitubería[0],)) y escribir (write (mituberia[1], .) en la tubería.

b)

```
#include varios .

int fd1[2],fd2[2]; //Necesitamos 2 tuberias

int contSIGINT=0; //contador senales SIGINT recibidas

int fin=0;  //para finalizar los hijos


void contarSIGINT (int s){

  contSIGINT++;

}

void bloquearSIGINT(){}

void finalizar (int s){

  fin=1;

}

void hijo1(){

   int i=1;

   char cad[10];
```

```
  signal (SIGUSR1, finalizar);
  signal (SIGINT, bloquearSIGINT);  //No se considerará en el examen como un error no ponerlo
  /* El hijo 1 escribe en la tuberia fd1 y lee de la tuberia fd2 cierra los otros descriptores */
  close (fd1[0]);
  close (fd2[1]);
  while (!fin){
    sprintf(cad, "%d", i);
    /* El hijo escribe en la tuberia */
    write (fd1[1], cad, sizeof (cad));
    read (fd2[0], cad, sizeof (cad));
    i=atoi (cad);
    printf ("Hijo 1: recibido numero: %d\n", i);
    i++;
  }
 // Cierro fd1 y me aseguro de que no quedan datos en la tuberia
  close (fd1[1]);
  read (fd2[0], cad, sizeof (cad));
  printf ("Fin hijo 1\n");
  exit (0);
}


void hijo2(){
   int i=1;
   char cad[10];


   signal (SIGUSR1, finalizar);
   signal (SIGINT, bloquearSIGINT); //No se considerará en el examen como un error no ponerlo
   /* El hijo 2 escribe en la tuberia fd2 y lee de la tuberia fd1
     cierra los otros descriptores */
   close (fd2[0]);
```

```
  close (fd1[1]);
  while (!fin){
    /* El hijo 2 lee de la tuberia y despues escribe*/
    read (fd1[0], cad, sizeof (cad));
    i=atoi (cad);
    printf ("Hijo 2, recibido numero:%d\n", i);
    i++;
    sprintf(cad, "%d", i);
    write (fd2[1], cad, sizeof (cad));
  }
 // Cierro fd2 y me aseguro de que no quedan datos en la tuberia
  close (fd2[1]);
  read (fd1[0], cad, sizeof (cad));
  printf ("Fin hijo 2\n");
  exit (0);
}
main () {
            pid_t     pidHijo1, pidHijo2;

            /* Crea las tuberia  */
            pipe (fd1);
            pipe (fd2);

            /*Crea los hijos*/
            if ((pidHijo1 = fork ())== -1) {
              perror ("fork 1 ");
              exit (1);
            }
            if (pidHijo1 == 0 )  hijo1();

      if ((pidHijo2 = fork ())== -1) {
```

```
        perror ("fork 2");
        exit (1);
      }
      if (pidHijo2 == 0 )  hijo2();

      /*Cierra las tuberias*/
      close (fd1[0]);
      close (fd1[1]);
      close (fd2[0]);
      close (fd2[1]);
      /*asocia el manejador de la senal SIGINT*/
      signal (SIGINT, contarSIGINT);
      while (contSIGINT<5)   pause();
       kill (pidHijo1, SIGUSR1);
       kill (pidHijo2, SIGUSR1);
      wait();
      wait();
      return (0);
}
```

Ejercicio 3 (4 puntos) .

En un determinado sistema con política de planificación apropiativa se utiliza una rodaja de tiempo de 100 milisegundos. El tiempo promedio para realizar un cambio de contexto de de 10 microsegundos.

A este sistema llegan 3 procesos:

- El proceso A, llega en el instante de tiempo t=0. Este proceso realiza cómputo durante 200 milisegundos, seguido de una fase de entrada/salida de 300 milisegundos y una fase de cómputo de de 100 milisegundos.
- El proceso B, llega en el instante de tiempo t=50 milisegundos. Este proceso realiza cómputo inicial durante 100 milisegundos y una tarea que repite dos veces. En cada iteración realiza una fase de entrada/salida de 150 milisegundos, seguida de cómputo durante 50 milisegundos y otra fase de entrada/salida de 100 milisegundos.

- El proceso C, llega en el instante de tiempo t= 150 milisegundos. Este proceso realiza una tarea exclusivamente de cómputo durante 400 milisegundos.

a) En el caso general en que se tienen n procesos en ejecución y ninguno realiza entrada/salida ¿Qué porcentaje de tiempo se usaría como máximo para cambios de contexto?

b) ¿Cuál sería el tamaño mínimo de la rodaja de tiempo si se está dispuesto a admitir un sobrecoste por rodaja de tiempo de hasta el 5%?

c) ¿En general puede ocurrir que el tiempo de servicio de un proceso sea mayor que su tiempo de consumo de CPU? Razone su respuesta.

d) Determine cómo sería la ejecución de estos procesos en el caso de una planificación no apropiativa FCFS, rellenando una tabla como la siguiente. Ignore los tiempos derivados de cambios de contexto y arranque y finalización de procesos:

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0						
B	50						
C	150						

e) Determine cómo sería la ejecución de estos procesos en el caso de una planificación apropiativa Round-Robin (cíclica), rellenando una tabla como la siguiente. Ignore los tiempos derivados de cambios de contexto y arranque y finalización de procesos:

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0						
B	50						
C	150						

f) Compare el tiempo medio de espera y el tiempo medio de retorno normalizado. ¿Qué conclusiones pueden derivarse de los resultados?

SOLUCIÓN:

a) Si se considera el que tiempo de una rodaje de tiempo es tr y el tiempo de un cambio de contexto es tc. La tasa de desaprovechamiento d, sería:

d= tc / (tr+tc)

En este caso particular:

d = 10 / (100000+10) = 0.000099

Es decir, 0.0099%

b) Sustituyendo en la expresión anterior:

0.05 = 10 / (tr+10)
0.05* tr + 0.5 = 10
tr = 9.5/0.05 = 190 microsegundos

c) El tiempo de servicio viene dado por el tiempo dedicado a CPU más el tiempo dedicado a entrada/salida. Por tanto cualquier proceso que haga entrada/salida tendrá un tiempo de servicio mayor que el tiempo de consumo de CPU

d) En este caso el primer proceso es en ejecutarse es el proceso A, cuando termina se ejecuta B y luego C.

El tiempo de servicio es el tiempo de CPU más el de E/S, por tanto se tiene:

- TS(A) = 200 + 300 + 100 = 600
- TS(B) = 100 + 2 * (150 + 50 + 100) = 700
- TS(C) = 400

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0	600	0	600	600	0	600/600=1
B	50	700	600	1300	1250	550	1250/700=1.79
C	150	400	1300	1700	1550	1150	1550/400=3.88

e)

Instante	En ejecución	Listos	Bloqueados	Cómputo Pendiente	Entrada/Salida Pendiente	Fina
0	A			A(200)		
50	A	B		A(150), B(100)		
100	B	A		A(100), B(100)		
150	B	A, C		A(100), B(50)		
200	A	C	B	A(100), C(400)	B(150)	
300	C		B, A	C(400)	B(50), A(300)	
350	C	B	A	C(350), B(50)	A(250)	
400	B	C	A	C(300), B(50)	A(200)	
450	C		A, B	C(300)	A(150), B(100)	

600	C	A	B	C(150), A(100)	B(100)	
650	A	C	B	C(100), A(100)	B(50)	
700	A	C, B		C(100), A(50), B (50)		
750	C	B		C(100), B(50)		A
850	B			B(50)		C
900	<nulo>		B		B(100)	
1000	<nulo>					B

Por tanto:

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0	600	0	750	750	150	750/600=1.25
B	50	700	100	1000	950	250	950/700=1.36
C	150	400	300	850	700	300	700/400=1.75

f) En el caso de planificación FCFS, se tiene:

- Tiempo medio de espera: (0+550+1150)/3 = 933.33
- Tiempo medio de retorno normalizado: (1+1.79+3,88)/3 = 2.22

En el caso de planificación cíclica se tiene:

- Tiempo medio de espera: (150+250+300)/3 = 233.33
- Tiempo medio de retorno normalizado: (1.25+1.36+1.75)/3= 1.45

La planificación cíclica reduce notablemente, en este caso, los tiempos que los procesos pasan esperando en colas. Esto se debe a que se aprovechan los tiempos de entrada salida de unos procesos para realizar tareas de cómputo en otros proceso. De hecho solamente al final, el procesador está libre durante 100 milisegundos, teniéndose un aprovechamiento de la CPU del 90%.

Por la misma razón los tiempos de retorno normalizados son también muy inferiores.