
 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Prueba de Evaluación Continua 19 de Marzo de 2013-A</p>	
---	---	---

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- Dispone de 90 minutos para realizar la prueba.
- No se podrán utilizar libros ni apuntes, ni calculadoras de ningún tipo.
- Los teléfonos móviles deberán permanecer desconectados durante la prueba (apagados, no silenciados).
- Solamente se corregirán los ejercicios contestados con bolígrafo. Por favor no utilice lápiz.

APELLIDOS:

NOMBRE:

NIA:

GRUPO:

Ejercicio 1: Responda a las siguientes preguntas de teoría (4 puntos)

- [1 punto] Indique en qué situaciones un proceso puede realizar las siguientes transiciones de estado:
Listo -> Bloqueado en primer plano
Ejecución -> Fin de proceso.

Listo -> Bloqueado en primer plano

Esta transición no puede darse, si un proceso está listo pero no se ejecuta no puede pasar a bloqueado, podría pasar al estado ejecutando, o bien, a Ejecución -> Fin de proceso

Si el proceso termina su ejecución, si el proceso recibe una señal que hace que este muera, por ejemplo, por una operación errónea como dividir por cero o una señal kill, ctrl+C, etc

- [1 punto] Se desea diseñar un Sistema Operativo en el que prime la terminación de trabajos sobre cualquier otra medida, indique qué algoritmo de planificación sería más conveniente y porqué entre los siguientes FCFS, SJF, RR q=2.

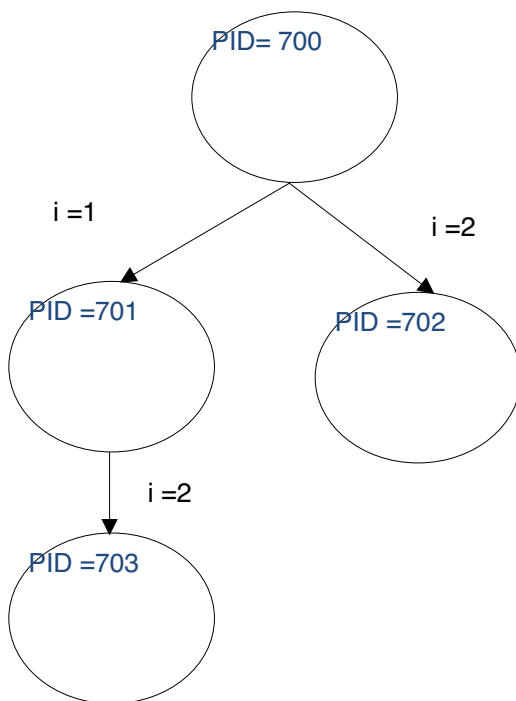
Elegiría SJF con expulsión, ya que este algoritmo de planificación garantiza que se ejecuten el mayor número de trabajos en el menor tiempo posible, aunque para aplicarlo tendríamos que saber a priori el tiempo de ejecución de los procesos y por otro lado los trabajos largos se verían penalizados e incluso podrían llegar a no ejecutarse.

- [1 punto] Cuando se produce un cambio de contexto de un proceso de forma involuntaria

Quando se produce una excepción: división por cero, violación de segmento
Quando el proceso es abortado por el usuario (ctrl-c) u otro proceso (kill), es decir, señales que no pueden manejar o ignorar.
Quando se acaba la rodaja de tiempo en Round Robin.

- [1 punto] Suponiendo que el PID de este proceso es 700, indicar la jerarquía de procesos generada identificando cada proceso con un valor PID consecutivo al del padre, y escribir los mensajes que aparecerían en pantalla

```
for (i=1; i<3; i++){
    pid=fork();
    if (pid==0){ //Hijo
        printf ("Hijo %d, pid =%d, ppid=%d\n",i, getpid(),getppid());
    }
    else
        printf ("El padre ha creado %d hijos \n",i);
}
exit (0);
```



Hijo 1, pid=701, ppid=700

El padre ha creado 1 hijos

Hijo 2, pid=702, ppid=700

El padre ha creado 2 hijos

Hijo 2, pid=703, ppid=701

El padre ha creado 2 hijos

NOTA Esta es una de las posibles soluciones, puesto que el orden de ejecución puede variar

Ejercicio 2 [3 puntos]

Un sistema operativo utiliza un planificador. En un instante determinado no hay ningún trabajo en ejecución y se desean ejecutar trabajos cuyos tiempos de llegada al sistema son los siguientes:

Proceso	Tiempo de llegada al sistema	Tiempo de ejecución	Prioridad
A	0	2	1
B	0,999999999	5	0
C	1,999999999	6	0
D	1,999999999	2	1
E	3,999999999	4	1

Para los cálculos redondear los tiempos de llegada al valor inmediatamente superior . Sabiendo la prioridad más alta es 1. Se pide rellenar las tablas en los siguientes casos:

- Política de planificación SJF (Shortest Job First)
- Política de planificación Round-Robin con rodaja de 2.

Para las dos posibilidades, se pide:

- Determine el momento de finalización de cada proceso.
- Determine el tiempo que cada proceso ha estado en el sistema (tiempo de retorno).
- Determine el tiempo de servicio y el tiempo de espera de cada proceso.

a) Política de planificación SJF (Shortest Job First) (usando un esquema apropiativo)



A																									
B																									
C																									
D																									
E																									
Tiempo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	

Proceso	Tiempo de finalización	Tiempo de retorno	Tiempo de servicio	Tiempo de espera
A	2	2	2	0
B	13	11	5	6
C	19	17	6	11
D	4	2	2	0
E	8	4	4	0

b) Política de planificación round-robin con rodaja de 2

A																									
B																									
C																									
D																									
E																									
Tiempo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	

Proceso	Momento de finalización	Tiempo de retorno	Tiempo de servicio	Tiempo de espera
A	2	2	2	0
B	17	15	5	10
C	19	17	6	11
D	4	2	2	0
E	8	4	4	0

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Prueba de Evaluación Continua 19 de Marzo de 2013-A</p>	
---	---	---

Ejercicio 3 [3 puntos]:

Dibuje el esquema de procesos y codifique, usando el lenguaje C, un programa que permita ejecutar el siguiente mandato: `cat | sort < f1`

El fichero `f1` es un fichero de texto que ya existe.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
Int main (int argc, char *argv[])
{
    Int fd[2];
    Int fid;
    pipe(fd);
    pid=fork();
    switch (pid)
    {
        case -1:
            perror (error al crear proceso hijo);
            break;
        case 0:
            fid= open (f1, O_RDONLY );
            close (0);
            dup(fid);
            close(1);
            dup(fd[1]);
            close(fd[0]);
            close(fd[1]);
            execlp(cat,cat,NULL);
            exit(-1)
        default:
            close (0);
            dup(fd[0]);
            close(fd[0]);
            close(fd[1]);
            execlp(sort,sort,NULL);
            exit(-1)
    }
}
```