
 <p>Universidad Carlos III de Madrid</p>	<p><b>Departamento de Informática</b> <b>Grado en Ingeniería Informática</b> <b>Sistemas Operativos</b></p> <p><b>Prueba de Evaluación Continua</b> <b>3 de noviembre de 2010</b></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:**

**Teoría (4 puntos)**

- **[1 punto]** Describa las diferencias conceptuales existentes entre la comunicación de procesos usando tuberías y la comunicación usando memoria compartida.
  - **Comunicación solo posible entre “parientes” (en el caso de tuberías), en el caso de memoria compartida es posible compartir datos entre procesos sin “parentesco”.**
  - **Sincronización automática entre procesos usando tuberías, no posible usando memoria compartida (se utilizan semáforos externos para esta sincronización)**
- **[1 punto]** ¿Es posible usar la planificación SJF (*Shortest Job First*) en cualquier entorno? Razone la respuesta
  - **No ya que no se dispone, normalmente, del tiempo de servicio de un proceso (necesario para llevar a cabo la planificación). Solo se puede usar en entornos donde se conozca el tiempo de servicio de los procesos, como es el caso de algunos sistemas de procesamiento por lotes.**

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Prueba de Evaluación Continua 3 de noviembre de 2010</p>	
---	--	---

- **[1 punto]** ¿Qué es un *microkernel*? ¿En que tipo de SSOO se utiliza el concepto de *microkernel*?
- Es una pequeña parte del SO que realiza las siguientes funciones: interrupciones, gestión básica de procesos y memoria y servicios básicos de comunicación
- Se utiliza en los SSOO estructurados del tipo cliente/servidor.

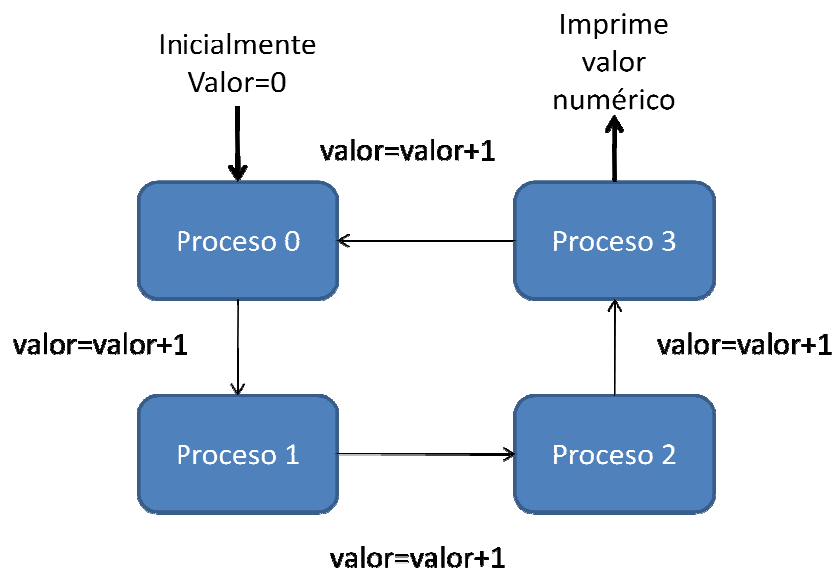
- **[1 punto]** ¿Que diferencia existe entre un *proceso huérfano* y un *proceso zombie*?

**Proceso zombie:** proceso que ha muerto, pero que sigue ocupando un espacio en la tabla de procesos.

**Proceso huérfano:** Proceso cuyo padre ha muerto. Son adoptados por el proceso init (ID del proceso = 1).



**Ejercicio 1** [3 puntos]:

**a)[2 punto]** Codifique , usando el lenguaje C, un programa en el que se creen 4 procesos que siguen el siguiente esquema:



1. El proceso principal debe crear 3 procesos hijos e inicializar un entero a 0. A continuación suma 1 al valor entero, para después enviarlo a través de una tubería al proceso 1.
2. El proceso N envía al proceso (N + 1) módulo 3, el valor entero recibido mas 1.
3. Si el proceso que recibe el valor es el proceso 3, este debe imprimir el valor recibido por pantalla.
4. Cuando valor sea igual a 1000 los procesos deben terminar y el proceso 0 (proceso padre) debe esperar al resto de los procesos.

En las comunicaciones, se utilizarán tuberías.

 <p>Universidad Carlos III de Madrid</p>	<p><b>Departamento de Informática</b> <b>Grado en Ingeniería Informática</b> <b>Sistemas Operativos</b></p> <p><b>Prueba de Evaluación Continua</b> <b>3 de noviembre de 2010</b></p>	
---	---	---

```
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define MAX 4
#define LIMITE 1000

int main(){
    int tub[MAX][2], valor=0, proc;
    int i,pid,status;

    for (i=0;i<MAX;i++)
        pipe(tub[i]);

    for (i=1;i<MAX;i++){
        proc=i;
        if ((pid=fork())==0){
            break;
        }else{
            proc = 0;
        }
    }



    if(proc==0)
        write(tub[proc][1],&valor,sizeof(int));

    while(valor < LIMITE){
        read(tub[(proc+MAX-1)%MAX][0],&valor,sizeof(int));
        valor ++;
        if (proc == (MAX-1)) printf("%d\n",valor);
        write(tub[proc][1],&valor,sizeof(int));
    }

    for (i=0;i<MAX;i++){
        close(tub[i][0]);
        close(tub[i][1]);
    }

    if (proc == 0)
        for (i=1;i<MAX;i++)
            wait(&status);

    exit(proc);
}
```

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Prueba de Evaluación Continua 3 de noviembre de 2010</p>	
---	--	---

**b) [0,5 punto]** Indique que modificaciones se deben realizar sobre el código anterior para que el proceso padre evite que los procesos anteriores queden zombies usando la señal SIGCHLD.

**Incorporar en el padre:**



**signal(SIGCHLD , sigchld\_handler);**

**void sigchld\_handler(void){**

**int status;**

**wait(&status);**

**}**

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Prueba de Evaluación Continua 3 de noviembre de 2010</p>	
---	--	---

c) [0,5 puntos] Indique que modificaciones se deben realizar sobre el código para que el proceso 3 imprima los valores a un fichero de texto llamado ***f1.txt***, ***usando redirecciones a fichero***.

En el proceso 3 (N-1):

***close (1);***

***creat("f1.txt", 0666);***

**Ejercicio 2** [3 puntos] En un sistema operativo, 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	A		B		C		D	
T. de llegada	0		1		2		3	
	CPU	E/S	CPU	E/S	CPU	E/S	CPU	E/S
	2		4		3		5	
		5		1		5		
	1		2		3			
				1				
			1					

Se pide rellenar las siguientes tablas en los siguientes casos:

- a) Política de planificación FIFO [1 punto]
- b) Política de planificación *round-robin* con rodaja de 1 [1 punto]
- c) Política de planificación *round-robin* con rodaja de 3 [1 punto]

NOTA1: Si la rodaja de ejecución de un proceso termina en el mismo instante que llega un nuevo proceso al sistema, entonces el nuevo proceso se coloca en la cola de listos para ejecutar antes que el proceso que le expira la rodaja.

NOTA2: Si la rodaja de ejecución de un proceso termina en el mismo instante que un proceso termina una operación de E/S, entonces el proceso que estaba realizando la operación de E/S se coloca en la cola de listos para ejecutar antes que el proceso que le expira la rodaja.

NOTA3: Las operaciones de E/S no son concurrentes (política FIFO).

a) Política de planificación *FIFO*

A	CPU	2	1											1												
	E/S			5	4	3	2	1																		
B	CPU			4	3	2	1								2	1				1						
	E/S							1									1						1			
C	CPU							3	2	1							3	2	1							
	E/S										5	4	3	2	1						3	2	1			
D	CPU										5	4	3	2	1											
	E/S																									
Tiempo		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24	25



b) Política de planificación *round-robin* con rodaja de 1

A	CPU	2				1							1													
	E/S						5	4	3	2	1															
B	CPU		4				3			2				1			2			1			1			
	E/S																1				1					
C	CPU			3				2			1						3		2		1					
	E/S											5	4	3	2	1										
D	CPU				5				4			3			2	1										
	E/S																									
Tiempo		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24	25

c) Política de planificación *round-robin* con rodaja de 3

A	CPU	2	1											1												
	E/S			5	4	3	2	1																		
B	CPU			4	3	2								1				2	1		1					
	E/S														1					1						
C	CPU						3	2	1							3	2	1								
	E/S									5	4	3	2	1												
D	CPU									5	4	3	2	1												
	E/S																									
Tiempo		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24	25