



Prueba de Evaluación Continua 18 de octubre de 2010

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.	o aunoo
APELLIDOS:	
NOMBRE:	
NIA:	
GRUPO:	
Teoría (4 puntos)	

[1 punto] Indique las ventajas e inconvenientes de un sistema operativo monolítico.

[1 punto] Indique las fases necesarias para el arranque de un SO





Prueba de Evaluación Continua 18 de octubre de 2010

• [1 punto] Describa en que consiste la técnica Copy-on-Write

• [1 punto] Indique de que partes básicas se compone un proceso ligero y que comparte con el resto de los procesos ligeros.

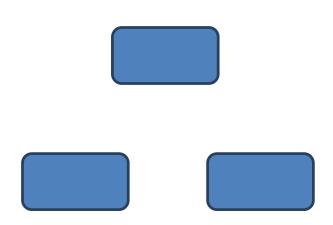




Prueba de Evaluación Continua 18 de octubre de 2010

Ejercicio 1 [3 puntos]:

a) [1 punto] Se pretende codificar un programa (en lenguaje C), que genere dos procesos hijos, siguiendo el siguiente esquema:



- 1. El proceso principal debe leer por teclado un número entero, y calcular si el valor es par o impar. Si el valor es 0, ir al punto 5.
- 2. Si el valor:
 - o es par, el proceso padre debe enviar un 1 al proceso hijo 0
 - o en caso contrario, enviará un 1 al proceso hijo 1.
- 3. El hijo que reciba la comunicación sumará 1 a un contador interno de números.
- 4. Volver al punto 1.
- 5. Enviar un 0 a ambos hijos para que terminen.

Complete el programa que aparece a continuación para que el programa funcione como se ha establecido.

En las comunicaciones, se utilizarán tuberías.





Prueba de Evaluación Continua 18 de octubre de 2010

#include <stdio.h></stdio.h>	int main (int argc, char *argv[]) {
#include <stdlib.h></stdlib.h>	impares = pares = 0;
#include <unistd.h></unistd.h>	
int impares, pares, valor;	
in imparos, paros, vaior,	if () /* codigo del hijo */
void hijo0(){	hijo0();
int valor;	}else{
impares=-1;	if () { /* codigo del hijo */
	hijo1();
	}else{
	while (0 ¡= scanf("%d",&valor)){
	if (valor == 0)
do{	break;
	valor = 1;
	if (valor%2 != 0){
impar++;	
}while(valor !=0);	}else{
printf("impares %d\n", impares	
);	}
	}
}	valor = 0;
void hijo1(){	
int valor;	
pares=-1;	,
	}
	}
	return 0;
	}
do{	
par++;	
}while(valor !=0);	
printf("pares %d\n", pares);	
}	





Prueba de Evaluación Continua 18 de octubre de 2010





Prueba de Evaluación Continua 18 de octubre de 2010

) [1 punto] Indique que modificaciones se deben realizar sobre el código anterior para que el proceso padre capture la señal SIGINT (ctrl+c). En este caso, el programa padre enviará un 0 a cada uno de los hijos y estos imprimirán el número de valores pares e impares leídos.

Los procesos hijos ignoran la señal SIGINT en todo momento.





Prueba de Evaluación Continua 18 de octubre de 2010

c) [1 punto] Indique que modificaciones se deben realizar sobre el código para que el proceso padre lea los datos a través de un fichero de texto, llamado f1.txt, que contiene valores enteros (redirección de entrada estándar). Una vez que se han leído todos los valores por fichero, el programa padre enviará un 0 a cada uno de los hijos y estos imprimirán el número de valores pares e impares leídos.

En las indicaciones se ha de poner donde se realizan las operaciones





Prueba de Evaluación Continua 18 de octubre de 2010

Ejercicio 2 [3 puntos] Un sistema operativo utiliza un planificador cíclico (round-robin). 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	Prioridades
Α	0	5	3
В	1	7	3
С	2	5	2
D	4	3	2
E	6	4	1

Las prioridades son inversas al valor que tienen. Así un proceso con prioridad 1 es prioritario respecto a otro con prioridad 2 o 3.

Se pide rellenar las siguientes tablas en los siguientes casos:

- a) Política de planificación round-robin con rodaja de 1 [1 puntos]
- b) Política de planificación round-robin con rodaja de 4 [1 puntos]
- c) Política de planificación SJF (Shortest Job First) (No expulsivo) [1 puntos]

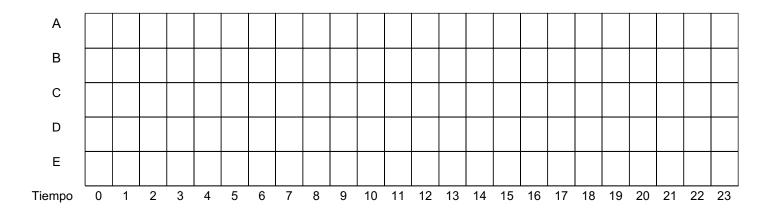
NOTA: Si la rodaja de ejecución de un proceso termina en el mismo instante que llega un nuevo proceso al sistema, entonces el nuevo proceso se coloca en la cola de listos para ejecutar antes que el proceso que le expira la rodaja.





Prueba de Evaluación Continua 18 de octubre de 2010

a) Política de planificación round-robin con rodaja de 1



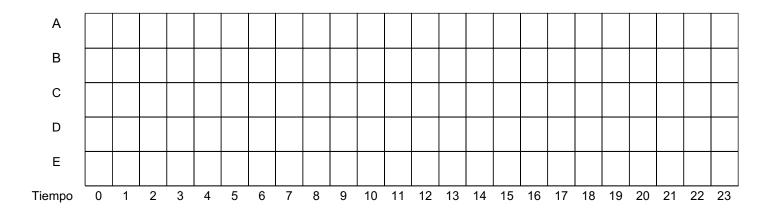
Proceso	Tiempo de finalización	Tiempo de	Tiempo de	Tiempo de	Tiempo de
		retorno	servicio	espera	retorno
					normalizado
Α					
В					
С					
D					
E					
Va	alores medios				





Prueba de Evaluación Continua 18 de octubre de 2010

b) Política de planificación round-robin con rodaja de 4



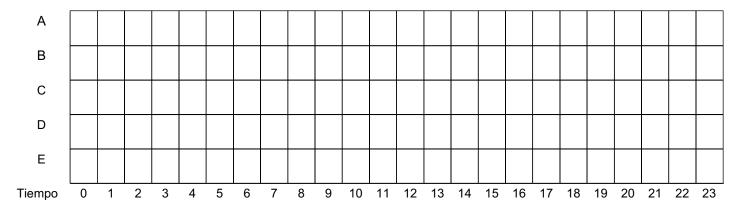
Proceso	Tiempo de finalización	Tiempo de	Tiempo de	Tiempo de	Tiempo de
		retorno	servicio	espera	retorno
					normalizado
Α					
В					
С					
D					
E					
V	alores medios				





Prueba de Evaluación Continua 18 de octubre de 2010

c) Política de planificación SJF (Shortest Job First)



Proceso	Tiempo de finalización	Tiempo de retorno	Tiempo de servicio	Tiempo de espera	Tiempo de retorno normalizado
А					
В					
С					
D					
E					
V	alores medios				