Universidad Carlos III de Madrid	**Departamento de Informática** **Grado en Ingeniería Informática** **Sistemas Operativos** **Prueba de Evaluación Continua** **22 de octubre de 2009**	

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:

Ejercicio 1 [0,5 puntos]: ¿Qué elementos debe tener un procesador para que se pueda utilizar planificación apropiativa?

Señal del reloj.

Ejercicio 2 [0,5 puntos]: Indique una ventaja del uso de bibliotecas dinámicas.

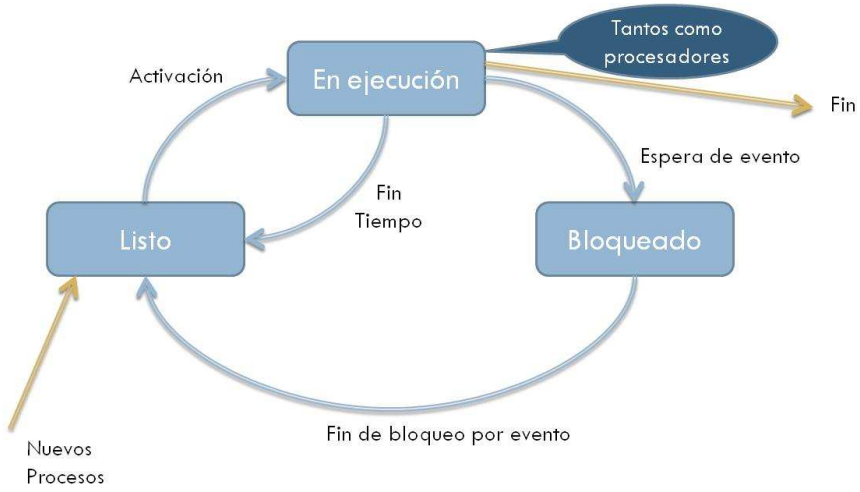
Cambio de las funciones sin necesidad de recompilar los ejecutables
Menor consumo de memoría.

Ejercicio 3 [0,5 puntos]: ¿En qué nivel de planificación (corto plazo, medio plazo o largo plazo) se gestiona la expulsión de procesos al área de swap?
Medio plazo.

Ejercicio 4 [0,5 puntos]: Enumere las dos razones para colocar un elemento de información **fuera** del BCP (bloque de control de procesos) de un determinado proceso.

Optimización en el acceso a recursos compartidos
Reducción de memoria.
Compartición de datos entre procesos.

Ejercicio 5 [1 puntos]: Dibuje el diagrama de estados del ciclo básico de un proceso, para el caso de un único procesador, incluyendo exclusivamente los estados que afectan a la planificación a corto plazo.



Ejercicio 6 [0,5 puntos]: Indique un ejemplo de caso en que se produce un cambio de contexto voluntario.

En una operación de E/S.
En la finalización de un proceso.

Ejercicio 7 [0,5 puntos]: Si un programa que utiliza hilos no realiza llamadas al sistema bloqueantes, ¿qué es mejor usar hilos de usuario (ULT) o hilos de kernel (KLT)?
Hilos de usuario ya que el uso de hilos de kernel suponen un coste en tiempo al realizar los cambios de contexto entre el espacio de usuario y el espacio de kernel.

Ejercicio 8 [3 puntos]: Dado el siguiente código, responder a las preguntas que aparecen a continuación:

```
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#define MAX 1
int espera = 10;
void controlador ();
int main (int contargs, char *args[]){
    pid_t pid;
    int i=0;
    signal (SIGCHLD, controlador);

    for (i=0;i<MAX;i++){
       pid = fork ();
       if (pid == 0){
         while(1){
           sleep (1);
           printf("Soy %d \n", i);
         }
         exit(0);
       }
     }

     sleep (espera);
     signal (SIGCHLD, SIG_IGN);
     kill (pid, SIGINT);
     exit (0);
 }
 void controlador (){
     int id, est;
     id = wait (&est);
     exit (0);
 }
```

a) Describa el funcionamiento del programa
b) ¿Qué pasaría si MAX tuviera un valor de 10?
c) Modifique el programa para que funcione para cualquier valor de MAX.

a) El proceso padre crea un hijo, y después de 10 segundos le manda una señal para que termine su ejecución. Si se tiene en cuenta `signal (SIGCHLD, SIG_IGN);` el programa padre termina. En caso contrario, cuando el hijo muera, se ejecutará el código de la función controlador.

b) Si MAX es igual a 10, se crearán 10 procesos hijos de los que solo 1 terminará.

c) Se proponen 2 soluciones:

```c
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#define MAX 100
int espera = 10;
pid_t pid[MAX];

void controlador ();
int main (int contargs, char *args[]){
    int i=0;
    signal (SIGCHLD, controlador);
    for (i=0;i<MAX;i++){
       pid[i] = fork ();
       if (pid[i] == 0){
         while(1){
           sleep (1);
           printf("Soy %d \n", i);
         }
         exit(0);
       }
    }
    sleep (espera);
    signal (SIGCHLD, SIG_IGN);
    for (i=0;i<MAX;i++){
       kill (pid[i], SIGINT);
    }
    exit (0);
 }
 void controlador (){
     int id, est, i;
     for (i=0;i<MAX;i++){
       id = wait (&est);
     }
     exit (0);
 }
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#define MAX 100
int espera = 10;
void controlador ();

int main (int contargs, char *args[]){
    pid_t pid;
    int i=0;
    signal (SIGCHLD, controlador);
    for (i=0;i<MAX;i++){
       pid = fork ();
       if (pid == 0){
         while(1){
           sleep (1);
           printf("Soy %d \n", i);
         }
         exit(0);
       }
       sleep (espera);
       signal (SIGCHLD, SIG_IGN);
       kill (pid, SIGINT);
    }
    exit (0);
 }
 void controlador (){
     int id, est;
     id = wait (&est);
     exit (0);
 }
```

Ejercicio 9 [3 puntos]: En un determinado sistema operativo los procesos se ejecutan con planificación apropiativa y política de planificación cíclica (round-robin).

En la siguiente tabla se especifica para cada proceso, su tiempo de llegada y el tiempo que necesitan para ejecutarse. Todos los procesos realizan exclusivamente tareas de cálculo.

Proceso	Tiempo de llegada	Tiempo de ejecución
P1	0	500
P2	100	300
P3	300	400
P4	600	1000
P5	700	600

Se desea evaluar las diferencias que se producirán al variar la longitud de la rodaja de tiempo, considerándose valores de 200 y 500 milisegundos.

Para las dos posibilidades, se pide:

1. Determine el tiempo de finalización de cada proceso.
2. Determine el tiempo que cada proceso ha estado en el sistema (tiempo de retorno).
3. Determine el tiempo de servicio y el tiempo de espera de cada proceso.
4. Determine el tiempo de retorno normalizado
5. Determine el tiempo medio de espera.
6. Determine el tiempo medio de retorno normalizado.

¿Puede concluir algo de los resultados?

T (rodaja 200)	**CPU**	**COLA**
0	**P1<500>**	
100	**P1<400>**	**P2<300>**
200	**P2<300>**	**P1<300>**
300	**P2<200>**	**P1<300>, P3<400>**
400	**P1<300>**	**P3<400>, P2<100>**
600	**P3<400>**	**P2<100>, P4<1000>, P1<100>**
800	**P2<100>**	**P4<1000>, P1<100>, P5<600>, P3<200>**
900 – Fin P2	**P4<1000>**	**P1<100>, P5<600>, P3<200>**
1100	**P1<100>**	**P5<600>, P3<200>, P4<800>**
1200 – Fin P1	**P5<600>**	**P3<200>, P4<800>**
1400	**P3<200>**	**P4<800>, P5<400>**
1600 – Fin P3	**P4<800>**	**P5<400>**
1800	**P5<400>**	**P4<600>**
2000	**P4<600>**	**P5<200>**
2200	**P5<200>**	**P4<400>**
2400 – Fin P5	**P4<400>**	
2800 – Fin P4		

Proceso	Ti	Tf	Tq	Ts	Te	Tq norm
P1	0	1200	1200	500	700	1200/500 = 2,4
P2	100	900	800	300	500	800/300 = 2,67
P3	300	1600	1300	400	900	1300/400 = 3,25
P4	600	2800	2200	1000	1200	2200/ 1000 = 2,2
P5	700	2400	1700	600	1100	1700/600 = 2,83
Promedio					880	2,67

T (rodaja 500)	CPU	COLA
0	P1<500>	
100	P1<400>	P2<300>
300	P1<300>	P2<300>, P3<400>
500 – Fin P1	P2<300>	P3<400>
600	P2<200>	P3<400>, P4<1000>
700	P2<100>	P3<400>, P4<1000>, P5<600>
800 – Fin P2	P3<400>	P4<1000>, P5<600>
1200 – Fin P3	P4<1000>	P5<600>
1700	P5<600>	P4<500>
2200	P4<500>	P5<100>
2700 – Fin P4	P5<100>	
2800 – Fin P5		

Proceso	Ti	Tf	Tq	Ts	Te	Tq norm
P1	0	500	500	500	0	500/500 = 1
P2	100	800	700	300	400	700/300 = 2,33
P3	300	1200	900	400	500	900/400 = 2,25
P4	600	2700	2100	1000	1100	2100/1000 = 2,1
P5	700	2800	2100	600	1500	2100/600 = 3,5
Promedio					700	2,23