Universidad Carlos III de Madrid	**Departamento de Informática** **Grado en Ingeniería Informática** **Sistemas Operativos** **Prueba de Evaluación Continua** **17 de marzo de 2009**	ARCOS

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- Dispone de una hora y cuarenta y cinco 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.

NOMBRE:

APELLIDOS:

NIA:

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

Ejercicio 2 [0,5 puntos] : Marque las opciones correctas en la siguiente tabla:

	SO Monolítico	SO Estructurados por capas	SO cliente/servidor
Más difícil distribuir las funciones			
OS/2			
Más fácil de extender			
Mantenimiento más complejo			
Mínima parte en modo supervisor			
Windows NT			
UNIX			
MacOS			
El menos eficiente			

Ejercicio 3 [0,5 puntos] : ¿Qué mecanismos se pueden usar para realizar el paso de parámetros a una llamada al sistema operativo?

Ejercicio 4 [1 puntos] : Dibuje el diagrama de estados del ciclo básico de un proceso, para el caso de un único procesador.

Ejercicio 5 [0,5 puntos] : ¿Por qué se suele utilizar para representar la tabla de procesos un vector (array) de tamaño fijo?

Ejercicio 6 [0,5 puntos] : ¿De qué tipo son los únicos cambios de contexto que se producen en un sistema con planificación no apropiativa?

Ejercicio 7 [0,5 puntos] : ¿Qué información no es compartida entre los distintos hilos (threads) de un mismo proceso?

Ejercicio 8 [3 puntos] : Se dispone de un archivo ("ordenes.txt") con una lista de órdenes a ejecutar. Para tratar dicho archivo se dispone de la biblioteca orden, cuyo archivo de cabecera se muestra a continuación:

```
/* Archivo: orden.h */
int abre_ordenes(char * nombre);
int num_ordenes(int fd);
int lee_ordenes(int fd, char ** vec);
```

En dicha biblioteca, la función abre_ordenes(), toma una cadena de caracteres y devuelve el descriptor del fichero abierto. Si no se puede abrir el fichero, la función devuelve un número negativo.

La función num_ordenes(), toma un descriptor de fichero abierto y devuelve un número entero que es el número de ordenes que hay en le archivo de ordenes abierto. Si se produce un error, devuelve un número negativo.

Por último, la función lee_oerdenes(), toma un descriptor de ficheros y un vector de cadenas de tamaño suficiente y almacena las ordenes del fichero en el vector de cadenas. Devuelve el valor cero si se ejecuta correctamente o un número negativo, si se produce algún error.

Se desea construir un programa que lea un fichero llamado "ordenes.txt" y lance la ejecución en paralelo de todos los programas especificados en dicho fichero. Si algún programa tarda más de 10 segundos en ejecutarse, deberá cancelarse su ejecución.

NOTA: La función waitpid(), tiene la siguiente interfaz:

```
pid_t waitpid(pid_t pid, int *status, int options);
```

Donde:

- pid es el pid del proceso hijo sobre el que se desea esperar.
- status puede ser el puntero nulo (NULL).
- Options puede tomar el valor WNOHANG para indicar que wait vuelva inmediatamente.
- Si se usa la opción WNOHANG, la función devuelve 0 si el hijo no ha terminado o el pid del hijo, si éste ha terminado.

Ejercicio 8 [3 puntos] : En un determinado sistema operativo los procesos se ejecutan con planificación apropiativa y política de planificación cíclica (round-robin), siendo la rodaja de tiempo de 100 milisegundos. Un cambio de contexto requiere 5 microsegundos.

En la siguiente tabla se especifica para cada proceso, su tiempo de llegada y el tiempo que necesitan para ejecutarse.

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

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.

Suponga ahora que se modifica la rodaja de tiempo a 200 milisegundos y repita los cálculos de los puntos 1, 2, 3, 4, 5 y 6.

¿Puede concluir algo de los resultados?