UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

1	2	3	test	extra	NOTA

Fundamentos de los Sistemas Operativos
Convocatoria ordinaria - 23 de mayo de 2017
Primera parte: temas 1, 2 y 3

Nombre y apellidos	**DNI/NIE**
SOLUCIONES	

DURACIÓN: *Dispones de dos horas para realizar el examen.*

Lee las instrucciones para el test en la hoja correspondiente.

1 (1'25 puntos) Responde a cada apartado. Usa como máximo 100 palabras por respuesta.

- En la planificación de procesos, ¿cuál es el problema que tienen los algoritmos basados en prioridades? ¿Cómo se puede abordar ese problema?

Pauta de la respuesta → Riesgo de inanición o postergación indefinida de los procesos con baja prioridad. Se soluciona con envejecimiento (*aging*).

- ¿Qué beneficios aportan los métodos de planificación multicola respecto a los de una sola cola?

Pauta de la respuesta → Podemos dar un tratamiento diferente a procesos de distinto tipo, ej. interactivos, por lotes, de tiempo real, de mucha prioridad, etc. Cada uno con una política de planificación distinta.

2 (1'50 puntos) Explica cada uno de estos aspectos sobre la técnica de multiprogramación. Desarrolla cada apartado en menos de 150 palabras.

- Cómo consiguió la multiprogramación aumentar el rendimiento de los sistemas con un solo procesador.

Pauta de la respuesta → evitando que la CPU se quede ociosa si el proceso en curso queda bloqueado; se aprovecha y se ejecutan fragmentos de procesos que estén dispuestos a ejecutar instrucciones; se solapa la ejecución de instrucciones en CPU con la realización de operaciones de E/S

- Qué necesidades de protección y seguridad introdujo la multiprogramación, especialmente en el ámbito de la gestión de la memoria.

Pauta de la respuesta → si tenemos varios procesos en ejecución, estos normalmente estarán compartiendo la memoria principal. Dada esta configuración, hay que evitar que un proceso acceda por error o malicia al espacio de memoria de otro. Además, hay que evitar que si un proceso se bloquea, o daña al SO, arrastre con él al conjunto de procesos del sistema.

3 (1'25 puntos) En una carretera tenemos un puente controlado por un sistema de sensores. Cada sensor se encarga del extremo de un carril y ejecuta un proceso independiente que detecta cuándo entra o sale un vehículo del puente. Cada evento modifica un contador compartido por todos los sensores. En total hay seis sensores (tres carriles, un sensor en cada extremo). Además de los procesos sensores, un proceso monitor se encarga de consultar periódicamente el contador y alerta si se alcanza el límite máximo de vehículos dentro del puente.

El sistema software se implementa en lenguaje C. El código sigue este esquema (la variable "contador" está inicializada a cero):

Sensor de entrada	Sensor de salida	Monitor

`while (true) {`
`  ... esperar una entrada`
`  contador++;`
`}`
	`while (true) {`
	`  ... esperar una salida`
	`  contador--;`
	`}`
		`while (true) {`
		`  sleep(1);`
		`  if (contador>MAX) {`
		`  alerta();`
		`  }`
		`}`

El sistema se instala y se observa que funciona bien durante un tiempo, pero al cabo de unas horas el monitor no detecta cuándo se alcanza el límite máximo. Haciendo trazas se observa que la variable "contador" acaba adquiriendo valores incorrectos. Incluso a veces se observan valores que no deberían darse nunca, por ejemplo un -1. Los dispositivos electrónicos se han comprobado y estos sí que funcionan perfectamente, no parece haber errores físicos de lectura.

¿Qué explicación puede tener este comportamiento del algoritmo? ¿Por qué el contador adquiere valores incorrectos? ¿Hay alguna solución algorítmica al problema?

El problema viene de las modificaciones concurrentes a la variable **contador**, que no están protegidas para que se ejecuten de forma atómica. Si dos procesos incrementan o decrementan al mismo tiempo **contador**, se puede perder una de las dos operaciones.

Existen formas algorítmicas de remediar este problema, y es conseguir que las modificaciones a **contador** sean atómicas, lo cual se puede conseguir con cualquier algoritmo válido de sección crítica. O con un semáforo tipo *mutex*, si disponemos de esa herramienta.

(extra) También se puede atenuar el problema si cada sensor utiliza una variable propia, ej. **entrada1, entrada2, entrada3, salida1, salida2, salida3**. Y el monitor sume y reste los valores de esas seis variables para ofrecer el balance general de entradas y salidas. El problema de esta técnica es que con el tiempo los contadores se saldrían del rango de valores del tipo manejado. Además, no soluciona del todo el problema, porque como el monitor no lee todos los valores al mismo tiempo, la suma puede dar un resultado que no se corresponda con la realidad.