Fundamentos de los Sistemas Operativos
Examen parcial, 3 de abril de 2017
Preguntas del test

1. ¿Cuál de estas funciones para el C de Linux se corresponde directamente con una llamada al sistema?
   a) puts()
   b) write()
   c) printf()
2. ¿Cuál es el objetivo principal de la multiprogramación?
   a) Sacarle más provecho a los recursos del sistema.
   b) Proporcionar una interfaz de usuario más universal.
   c) Aumentar la fiabilidad del sistema.
3. Los sistemas de tiempo compartido se apoyan en un componente de *hardware* característico. Este componente es:
   a) El despachador.
   b) El temporizador.
   c) Los registros base y límite.
4. ¿Qué tipo de planificación de procesos fue la predominante en los primeros sistemas informáticos?
   a) Tiempo compartido.
   b) Procesamiento por lotes.
   c) Tiempo real.
5. ¿Puede el *hardware* activar directamente al sistema operativo?
   a) Sí, por ejemplo mediante una interrupción.
   b) No, el sistema operativo sólo se activa mediante *software*.
   c) Sí, de hecho el sistema operativo solamente se puede activar por la acción externa de un periférico de E/S.
6. ¿Cuál de las siguientes afirmaciones acerca del núcleo de un sistema operativo es FALSA?
   a) Forma parte de la interfaz de usuario (CLI) del sistema operativo.
   b) Se carga en la memoria principal durante el arranque del sistema.
   c) Su arquitectura interna puede estar compuesta por varias capas con distintas responsabilidades.
7. Mientras el procesador está ejecutando un proceso de usuario, llega una interrupción procedente de un dispositivo de E/S. ¿Cómo reacciona el procesador?
   a) Conmuta a modo núcleo y ejecuta la rutina de servicio de interrupción correspondiente.
   b) Conmuta a modo usuario para que el código del programa pueda dialogar con la E/S.
   c) Ejecuta el código de la rutina de servicio de interrupción, en el modo de operación (usuario o núcleo) en el que se encontraba cuando se produjo la interrupción.
8. ¿Qué son las llamadas al sistema (*system calls*)?
   a) Son operaciones que el núcleo ofrece a través de una interfaz.
   b) Son llamadas en código máquina que hacen los dispositivos del hardware para conectar con el núcleo.
   c) Son mecanismos del núcleo para llamar a los dispositivos del hardware.
9. Un fichero ejecutable Linux para PC no puede ejecutarse sobre un núcleo Windows. ¿Cuál de estas posibles explicaciones NO es correcta?
   a) La API del núcleo de Linux es diferente a la del Windows.
   b) El formato del fichero Linux no es reconocible por el núcleo Windows.
   c) La arquitectura del código máquina del ejecutable Linux es incompatible con la del Windows.

Fundamentos de los Sistemas Operativos
Examen parcial, 3 de abril de 2017
Preguntas del test

10. Habitualmente, ¿cómo está almacenado en memoria principal el código del núcleo del sistema operativo?
    a) Cada uno de los procesos en ejecución tiene una copia del código del núcleo. Así cada proceso puede acceder con seguridad al núcleo.
    b) Hay una única copia en memoria y esta debe estar protegida de los accesos por parte de los procesos de usuario.
    c) El código del núcleo no debe residir en memoria principal de forma permanente. Sólo se traen a memoria principal los fragmentos que hacen falta en cada momento.
11. Los núcleos de Unix y Linux son casos de:
    a) núcleos monolíticos.
    b) micronúcleos.
    c) sistemas por capas.
12. ¿Cuál de estos algoritmos de planificación de CPU por definición no puede ser expulsivo?
    a) FCFS.
    b) Round Robin.
    c) SJF.
13. En un sistema interactivo como un PC o un móvil, ¿qué prioridad en la planificación deben tener los procesos intensivos en E/S respecto a los intensivos en CPU?
    a) Menos prioridad.
    b) Más prioridad.
    c) La misma prioridad.
14. ¿Para qué se usa el Planificador de Largo Plazo (PLP)?
    a) En los sistemas por lotes, para dosificar la cantidad de procesos que admitimos para ejecutar dentro del sistema.
    b) En los sistemas de tiempo real, para distinguir los procesos que tienen requisitos de plazos de ejecución cortos de aquellos que no tienen urgencia.
    c) En los sistemas interactivos, para encolar a aquellos procesos que no tengan requisitos de tiempos de respuesta cortos.
15. Los métodos multicolas de planificación de CPU:
    a) Manejan varias clases de procesos que se planifican según algoritmos diferentes.
    b) Resultan más apropiados para multiprocesadores que los métodos de una sola cola.
    c) Gestionan una cola de preparados y varias colas de espera por CPU.
16. En los sistemas multiprocesadores, la afinidad al procesador (*processor affinity*) consiste en lo siguiente:
    a) Cada proceso queda asignado continuamente al mismo procesador, para evitar ineficiencias con las cachés.
    b) Los procesos que son de una misma clase (ej. procesos intensivos en CPU) quedan asignados al mismo procesador, o grupo de procesadores.
    c) Cada procesador ejecuta aquellos procesos cuyo código máquina sea más afín a la arquitectura del procesador.
17. De los algoritmos de planificación de procesos que aquí se citan, ¿cuál es el más propenso a sufrir el llamado «efecto convoy»?
    a) FCFS.
    b) SJF.
    c) Multicolas.

Fundamentos de los Sistemas Operativos
Examen parcial, 3 de abril de 2017
Preguntas del test

18. Tenemos en la cola de preparados una carga de trabajo de 10 procesos, cada uno listo para ejecutar una ráfaga de 5 milisegundos en el procesador. Tenemos dos planificadores Round Robin, uno con Q=1 milisegundo y otro con Q=3 milisegundos. ¿Cuál provocará un mayor tiempo de espera medio para esta carga?
    a) El de Q=1 milisegundo.
    b) El de Q=3 milisegundos.
    c) No habrá diferencia significativa entre ambos.
19. En un sistema con un solo procesador tenemos exactamente tres hilos en la cola de preparados, cuyas siguientes ráfagas de CPU se sabe que van a ser de 5, 3 y 8 milisegundos. No conocemos el algoritmo de planificación de CPU. ¿Cuál de las siguientes afirmaciones sería cierta en cualquier caso?
    a) La suma de los tiempos de retorno de las tres ráfagas no será inferior a 27 mseg.
    b) La suma de los tiempos de espera de las tres ráfagas será superior a 13 mseg.
    c) Al menos una de las ráfagas tendrá un tiempo de espera no superior a 3 mseg.
20. ¿Por qué hace falta exigir a un algoritmo de sección crítica que cumpla la propiedad de progreso (*liveness*)?
    a) Porque así descartamos algoritmos que aunque garantizan exclusión mutua, no garantizan que la sección crítica se llegue a usar alguna vez.
    b) Porque tenemos que descartar algoritmos que tengan riesgo de postergación indefinida de un proceso frente al resto.
    c) Porque no podemos considerar totalmente válido un algoritmo que no sea extensible a más de dos procesos.
21. ¿Qué es una sección crítica?
    a) Una zona de código en la que se accede a datos compartidos y que debe ser ejecutada en exclusión mutua.
    b) Una sección de datos compartidos que necesita ser controlada por el núcleo del sistema operativo.
    c) Un proceso ligero o hilo (*thread*) que debe ejecutarse en exclusión mutua con respecto al resto.
22. Inhibir las interrupciones no se considera una técnica universal para gestionar secciones críticas. ¿Cuál de estos argumentos es válido para apoyar tal afirmación?
    a) Esta técnica es inviable en sistemas multiprocesadores.
    b) Esta técnica es inviable si utilizamos cachés para la memoria principal.
    c) Esta técnica unicamente sirve cuando compiten sólo dos procesos por la sección crítica.
23. ¿Qué característica peculiar tiene la instrucción *test-and-set*?
    a) Se ejecuta de forma atómica.
    b) Sólo funciona en multiprocesadores.
    c) Ejecuta dos acciones de forma simultánea (en paralelo).
24. Una diferencia entre el interbloqueo y la inanición es:
    a) La inanición puede afectar a un solo proceso, mientras que el interbloqueo siempre afecta a un conjunto de procesos.
    b) La inanición está causada por problemas en los algoritmos internos del sistema operativo, mientras que el interbloqueo está causado por problemas en el código de las aplicaciones.
    c) La inanición ocurre en el contexto del acceso a un recurso, mientras que el interbloqueo ocurre en el contexto del acceso a una variable compartida.