1. ¿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 (paralela).
2. Tenemos una variable entera **x** que actualmente vale **1**. Dos hilos acceden simultáneamente a **x**. Uno de ellos ejecuta la sentencia **x=x+1** y el otro la sentencia **x=x-1**. Una vez que se ejecuten esas sentencias, ¿cuál será el valor de **x**?
   a) 1.
   b) 0, 1 o 2.
   c) 0 o 2.
3. ¿Cuál es el problema de utilizar espera activa (*busy waiting*) para resolver la sincronización entre procesos?
   a) Se utiliza improductivamente tiempo del procesador.
   b) No se resuelve del todo la sincronización, por ejemplo si varios hilos ejecutan justo al mismo tiempo la espera activa.
   c) No funciona en sistemas multiprocesadores de memoria compartida.
4. Si utilizamos la biblioteca *pthreads* y queremos que un hilo espere por la finalización de otro, ¿qué función deberíamos invocar?
   a) pthread_wait()
   b) pthread_join()
   c) pthread_exit()
5. ¿Cuál es la limitación que tiene el algoritmo de Peterson para secciones críticas?
   a) Sólo sirve para dos procesos.
   b) Depende de la existencia de un sistema de interrupciones.
   c) Sólo funciona con hilos concurrentes, no paralelos.
6. ¿Cuál es la utilidad de la llamada «afinidad al procesador» (*processor affinity*) en la planificación de sistemas multiprocesadores?
   a) Aprovechar que el procesador que está ejecutando un hilo mantiene en caché bloques de memoria de ese hilo.
   b) Asociar a cada hilo el procesador que es más afín a la arquitectura de su código máquina.
   c) Lograr que cada hilo se ejecute el mayor tiempo posible en un procesador que no esté muy cargado de trabajo.
7. Tenemos diez procesos en la cola de preparados, cada uno de los cuales quiere ejecutar una ráfaga de CPU de 2 milisegundos. Suponiendo que los planificamos con un Round-Robin con Q=1 milisegundo, ¿cuánto tiempo tardará el conjunto de estas ráfagas en finalizar?
   a) 20 milisegundos.
   b) 10 milisegundos.
   c) 30 milisegundos.
8. ¿Cuál de estos algoritmos de planificación de procesos requiere de un temporizador?
   a) FCFS.
   b) Round-Robin.
   c) SJF expulsivo.
9. Tenemos tres procesos en la cola de preparados. El primero quiere ejecutarse durante 3 mseg., otro durante 2 mseg. y el último 5 mseg. Sabemos que dentro de 3 mseg. llegará un proceso que pretende ejecutarse durante 1 mseg. Si la CPU actualmente está libre, ¿qué algoritmo de planificación producirá más cambios de contexto para esta carga de trabajo?
   a) FCFS.
   b) Round-Robin con Q=2mseg.
   c) SJF expulsivo.

10. ¿Qué transición de estados normalmente NO se da en un proceso?
    a) de «preparado» a «bloqueado».
    b) de «bloqueado» a «terminado».
    c) de «en ejecución» a «preparado».
11. ¿Cómo puede eliminarse el riesgo de inanición en los algoritmos de planificación de CPU basados en prioridades?
    a) Subiendo la prioridad de los procesos en cola a medida que aumenta su tiempo de espera.
    b) Expulsando de la CPU al proceso que está en ejecución si lleva mucho tiempo acaparando el recurso.
    c) No es posible eliminar por completo el riesgo de inanición en esta clase de algoritmos.
12. ¿Cuál es la diferencia entre un proceso «pesado» y un hilo (*thread*)?
    a) Un proceso pesado es una aplicación en ejecución, mientras que un hilo es una actividad concurrente dentro de una aplicación.
    b) Un proceso pesado puede contener un hilo o ninguno, mientras que un hilo puede contener uno o más procesos pesados.
    c) Un proceso pesado corre siempre en modo usuario, mientras que un hilo puede correr en modo usuario o modo supervisor/modo núcleo.
13. Al analizar algoritmos de planificación de CPU, el tiempo de retorno corresponde:
    a) Al tiempo que el proceso espera hasta que se le concede el procesador.
    b) Al tiempo que transcurre desde que un proceso se crea hasta que se completa su ejecución.
    c) A la suma de los tiempos en los que el procesador aguarda en la cola de preparados.
14. Para implementar de forma eficaz un esquema de protección de memoria basado en la pareja de registros base y límite, ¿cuál de estas características debe estar presente en el hardware?
    a) Un temporizador controlable por software.
    b) Un procesador con dos modos de operación (usuario/sistema).
    c) Una jerarquía de memorias de al menos dos niveles.
15. El núcleo del sistema operativo:
    a) Está cargado en memoria principal de forma permanente.
    b) Contiene los programas del sistema.
    c) Virtualiza las distintas interfaces que puede ofrecer el sistema operativo.
16. ¿Cuál de estos sistemas de procesamiento es menos apropiado para una consola de juegos tipo PlayStation o Xbox?
    a) Sistema de tiempo compartido.
    b) Sistema de tiempo real crítico.
    c) Sistema de procesamiento por lotes.
17. Al ofrecer una API uniforme para acceder a la entrada/salida, el sistema operativo consigue:
    a) Abstraer a los desarrolladores de los detalles concretos de los periféricos.
    b) Forzar a la industria informática a fabricar periféricos con una interfaz uniforme.
    c) Hacer homogénea la velocidad de acceso a los datos almacenados en los periféricos.
18. ¿El *hardware* puede activar directamente al sistema operativo?
    a) Sí, por ejemplo mediante una interrupción.
    b) No, el sistema operativo sólo se activa mediante *software*.
    c) No, todos los eventos del *hardware* deben pasar por el núcleo.
19. ¿Cuál de estos servicios resulta imprescindible en cualquier sistema operativo?
    a) Cargador de programas (*program loader*).
    b) Multiprogramación (multiprogramming).
    c) Sistema de archivos (*file system*).

20. ¿Qué mecanismo se utiliza para que el sistema operativo conozca que un proceso ha terminado su ejecución?
    a) El proceso invoca a una llamada al sistema específica para avisar de que finaliza.
    b) El contador de programa del proceso alcanza la última instrucción.
    c) Transcurre un tiempo prefijado sin que el proceso haya ejecutado instrucciones en la CPU.
21. Las llamadas al sistema sirven de interfaz entre:
    a) Los programas de usuario y el núcleo.
    b) Los programas de usuario y los programas del sistema.
    c) El núcleo y el sistema de interrupciones.
22. Si dispusiéramos de memoria infinita, ¿qué servicio dejaría de tener sentido?
    a) La gestión de zonas de la memoria libres y ocupadas.
    b) La protección del área de memoria ocupada por el sistema operativo.
    c) La memoria virtual paginada.
23. El sistema operativo...
    a) consume tiempo de CPU a costa de los programas de usuario.
    b) sirve de interfaz entre el procesador y los periféricos.
    c) interpreta las instrucciones en código máquina de los programas de usuario.
24. ¿Cuál de estas funcionalidades debe apoyarse necesariamente en un procesador con modo dual de operación (modo núcleo/modo usuario)?
    a) Protección de zonas de memoria.
    b) Multiprogramación.
    c) Sincronización entre procesos.