Test SITR Temas: Introducción, Concurrencia Procesos y Threads (TestSITR_T1-T3)

Temas:
- Introducción a los Sistemas de Tiempo Real
- Concurrencia: Procesos y Threads
- Funciones POSIX: Procesos y Threads

El test es de respuesta única, se debe marcar una sola de las respuestas.

---

1. **Algunas de las propiedades de los sistemas de tiempo real son:**

    1. Estrictos, con parada segura, controlados por eventos.
    2. Concurrentes, tolerantes a fallos, manejo de estímulos externos.
    3. Empotrados, muy rápidos, con una arquitectura controlada por tiempo.
    4. Ninguna de las anteriores.

2. **Los sistemas de tiempo real se pueden clasificar en:**

    1. Estrictos y no estrictos
    2. Controlados por eventos o por tiempo
    3. Todas las anteriores
    4. Ninguna de las anteriores

3. **Algunos lenguajes de programación específicos de las aplicaciones de tiempo real que incorporan la concurrencia son :**

    1. ADA95, C, Pascal
    2. C, Pascal, Java
    3. Modula-2, PEARL ADA95
    4. C, C++

4. **Un sistema operativo multitarea permite:**

    1. Ejecutar multiples tareas al mismo tiempo en una CPU
    2. Ejecutar de forma concurrente multiples tareas en una CPU
    3. Ejecutar tareas a nivel de nucleo y nivel de usuario
    4. Ninguna de las anteriores

5. **La estructura de un sistema operativo de tiempo real establece varias capas:**

    1. Nivel de Núcleo, y nivel de Usuario
    2. Nivel de Aplicación, y nivel de Planificación
    3. Nivel de Usuario donde se ejecutan las tareas críticas del sistema , y nivel de Núcleo donde se ejecutan las tareas de bajo nivel
    4. Nivel de Núcleo monotarea y nivel de Usuario multitarea.

6. **Un proceso contiene los siguientes recursos:**

    1. Código, Datos, Stack, Ficheros, Planificable
    2. Recursos de planificación, sin datos
    3. Código máquina en un fichero
    4. Solo datos de variables globales, stack para parámetros y variables locales

7. **Un proceso que está esperando que se le asigne un procesador está en estado:**

    1. Suspendido
    2. En ejecución
    3. Preparado
    4. Terminado

8. **El bloque de control del proceso (PCB) permite al sistema operativo:**

    1. Realizar el cambio de contexto
    2. Cambiar la imagen del proceso
    3. Crear un nuevo proceso
    4. Todas las anteriores

9. **Un proceso termina cuando:**

    1. Termina su última línea de código
    2. Realiza la llamada al sistema exit()
    3. Otro proceso con privilegios envia la señál kill
    4. Todas las anteriores

10. **Un thread (hilo de ejecución) es:**

    1. Una entidad planificable con recursos independientes (registros, memoria, ficheros)
    2. Una función dentro de un programa
    3. Un puntero a una función
    4. Una secuencia de código planificable de forma independiente dentro de un proceso

11. **Las ventajas del uso de threads frente a los procesos para implementar la concurrencia son:**

    1. Disminuir el tiempo de cómputo de las tareas
    2. Disminuye el tiempo de respuesta
    3. Acceso más eficiente a los recursos del núcleo
    4. Todas las anteriores

12. El modelo de control de threads a nivel de núcleo:

1. Mantiene una entidad planificable a nivel de núcleo por cada thread
2. Cuando un thread ejecuta una llamada del sistema bloqueante, el resto de threads no son bloqueados
3. Pueden aprovechar al paralelismo de sistema con múltiples CPUs
4. Todas las anteriores

13. Los modelos de control de threads son los siguientes:

1. Nivel de Usuario, Nivel de Núcleo, Control Híbrido
2. Nivel de Núcleo, Nivel Planificación, Modelo Híbrido
3. Nivel de Proceso, Nivel de Thread, Nivel de Núcleo
4. Ninguna de las anteriores

14. Las ventajas del control de threads a nivel de Núcleo frente al controla a nivel de Usuario son:

1. El cambio de contexto es más rápido
2. Puede aprovechar el paralelismo hardware del sistema
3. Los threads se pueden planificar de forma independiente evitando que el thread esté en estado suspendido si está esperando por una E/S
4. Todas las anteriores

15. Una cola de planificación es:

1. Una lista de programas esperando que se les asigne recursos
2. Una lista de procesos esperando en cada estado
3. Una lista de recursos usados por los procesos del sistema
4. Una lista con todos los procesos esperando por cualquier recurso del sistema

16. El cambio de contexto es realizado por:

1. El planificador
2. El despachador
3. La aplicación del usuario
4. El proceso init

17. La llamada al sistema *fork*():

1. Crea un nuevo proceso hijo devolviendo la función el pid del padre
2. Crea una copia del padre compartiendo el mismo espacio de memoria
3. Crea un nuevo proceso hijo devolviendo la función al proceso creador el PID del hijo
4. Todas las anteriores

18. La llamada al sistema *wait*() sirve para:

1. Esperar la terminación de proceso hijo especificado como parámetro
2. Espera a la terminación de una proceso hijo
3. Provoca que un proceso que zombie al no liberar sus recursos
4. Todas las anteriores

19. La librería POSIX para gestión de threads permite crear y gestionar hilos de ejecución mediante objetos atributos que se caracterizan por:

1. Una vez creados no pueden modificarse
2. Cada thread está ligado a un objeto atributo y cualquier cambio en él afecta al thread ya creado
3. Cada thread está ligado a un objeto atributo en la creación. Si el el objeto se modifica posteriormente no afecta a los threads ya creados
4. Cada thread está siempre ligado a un objeto atributo único e independiente del resto de threads

20. Un thread POSIX en estado desconectado (detached):

1. No puede comunicar datos a otros threads del proceso
2. No puede ser esperada su terminación por el resto de threads
3. No puede acceder a los recursos de sistema a través de llamadas del sistema
4. Todas las anteriores

SOLUCIONES:

1) 2	11) 2	
2) 3	12) 4	
3) 3	13) 1	
4) 2	14) 2	
5) 1	15) 2	
6) 1	16) 2	
7) 3	17) 3	
8) 1	18) 2	
9) 4	19) 2	
10) 4	20) 2	