Test SITR Temas: Planificación, Sincronización, Comunicación entre Procesos, Relojes, Señales, Temporizadores (TestSITR_T4-T9)

Temas:

-Planificación
-Sincronización y Comunicación entre Procesos
-Funciones POSIX: Semáforos, mutex, memoria compartida
-Relojes POSIX
-Señales POSIX
-Temporizadores POSIX: tareas periódicas

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

---

1. **El planificador realiza la siguiente tarea:**

    1. Cambia de contexto
    2. Expulsa el proceso actual en ejecución e introduce un nuevo proceso en el uso de la CPU
    3. Decide cual es el siguiente proceso preparado que debe usar la CPU
    4. Ninguna de las anteriores

2. **Los objetivos de un planificador son:**

    1. Maximizar el tiempo de retorno de los procesos
    2. Minimizar el coeficiente de utilización de la CPU
    3. Minimizar el tiempo de espera
    4. Todas las anteriores

3. **El cambio de contexto es realizado por:**

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

4. **En un algoritmo de planificación expulsivo la decisión de planificación se realiza:**

    1. Cuando un proceso pasa de estado de ejecución a estado suspendido
    2. Cuando un proceso pasa de estado de ejecución a estado preparado
    3. Cuando un proceso pasa de estado suspendido a estado preparado
    4. Todas las anteriores

5. **El algoritmo de planificación FCFS:**

    1. Es expulsivo
    2. Minimiza el tiempo de espera
    3. Optimiza la utilización de la CPU
    4. El tiempo de respuesta presenta muy poca varianza

6. El algoritmo de planificación por prioridades:

1. Las prioridades dinámicas provocan el problema de inanición en procesos de baja prioridad
2. Puede ser expulsivo y no expulsivo
3. Los procesos largos son penalizados
4. Todas las anteriores

7. El algoritmo de planificación mediante clases de prioridades:

1. Utiliza un único algoritmo de planificación basado en prioridades
2. Utiliza un algoritmo SJF para planificar entre colas
3. Los procesos pueden cambiar dinámicamente entre las distintas colas de planificación
4. La cola de procesos preparados está formada por múltiples colas utilizando diferentes algoritmos

8. POSIX define al menos las siguientes políticas de planificación:

1. Planificación basada en prioridades (FIFO y Round Robin, Tiempo compartido sin prioridades
2. FCFS, SJF, Round Robin
3. Colas de prioridades, SRTF, Round Robin
4. FCFS, Round Robin, SRTF

9. Una condición de carrera se produce cuando:

1. Se inclumple la seriabilidad
2. La ejecución de la tareas concurrente genera resultados diferentes en cada ejecución
3. Todas las anteriores
4. Ninguna de las anteriores

10. El problema de sincronización de tareas concurrentes mediante una sección crítica plantea:

1. Una parte del código que debe ejecutarse con máxima prioridad
2. Una sección de código que solo puede estar siendo ejecutada concurrentemente por una tarea
3. Una sección de código que solo puede ser ejecutada por una de las tareas
4. Ninguna de las anteriores

11. La operación V de un semáforo:

1. Incrementa el contador del semáforo y, si el resultado es positivo, despierta a proceso en la cola
2. Decrementa el contador del semáforo y, si el resultado es negativo suspende el proceso en la cola asociada
3. Incrementa el contador del semáforo y, si el resultado es negativo o cero, despierta a proceso en la cola del semáforo
4. Decrementa el contador del semáforo y suspende el proceso en la cola asociada

12. El contador de un semáforo POSIX indica:

1. El número de procesos que pueden acceder a la sección crítica sin suspenderse si es positivo
2. El número de procesos suspendidos en la cola asociada al semáforo si es positivo
3. El número de procesos que pueden acceder a la sección crítica sin suspenderse si es negativo
4. Ninguna de las anteriores

13. Para implementar una sección crítica mediante semáforos se deben usar las operaciones P,V del siguiente modo:

1. Operación V en la sección de entrada, operación P en la sección de salida
2. Operación P en la sección crítica, operación V en la sección restante
3. Operación P en la sección de entrada, operación V en la sección de salida
4. Ninguna de las anteriores

14. El interbloqueo entre procesos con recursos compartidos mediante semáforos se produce cuando:

1. Ninguno de los procesos accede a la sección crítica y quedan estado suspendido
2. Un proceso de mayor prioridad está en estado suspendido a la espera de uno de menor prioridad
3. Varios procesos acceden a dos recursos compartidos con dos semáforos de forma anidada no atómica
4. Todas las anteriores

15. La operación V es implementada en POSIX mediante la llamada al sistema:

1. sem_wait()
2. sem_post()
3. sem_trywait()
4. sem_getvalue()

16. La inicialización de un semáforo nombrado es implementada en POSIX mediante la llamada al sistema:

1. sem_init()
2. sem_trywait()
3. sem_open()
4. sem_create()

17. El acceso a una sección crítica mediante mutex POSIX se realiza:

1. pthread_mutex_lock() en la sección de entrada, y pthread_mutex_unlock() en la sección de salida
2. pthread_mutex_wait() en la sección de entrada, y pthread_mutex_post() en la sección de salida
3. pthread_mutex_unlock() en la sección de entrada, y pthread_mutex_lock() en la sección de salida
4. Ninguna de las anteriores

18. La operación P de mutex es implementada en POSIX mediante la llamada al sistema:

1. sem_lock()
2. sem_trylock()
3. Todas las anteriores
4. Ninguna de las anteriores

19. La creación e inicialización de una región de memoria compartida es implementada en POSIX mediante la llamada al sistema:

1. shm_init()
2. shmat()
3. shmget()
4. shmctl()

20. La comunicación entre procesos mediante colas de mensajes de forma síncrona, establece un canal con:

1. Capacidad finita para almacenar mensajes no leídos
2. Capacidad cero para almacenar mensajes no leídos
3. Notificación mediante del número de mensajes recibidos
4. Control encolado por prioridades

21. El uso de métricas de tiempo densas está ligado a:

1. Sistemas distribuidos con relojes locales
2. Sistemas síncronos, en los cuales los eventos ocurren en intervalos de tiempo preestablecidos
3. Sistemas que exhiben un comportamiento totalmente discreto
4. Todas las anteriores

22. Las propiedades de un reloj ideal son:

1. Estabilidad, resolución, exactitud
2. Resolución, época, precisión
3. Uniformidad, resolución, viveza
4. Todas las anteriores

23. Las propiedades estáticas de un reloj real son:

1. Granularidad, exactitud, estabilidad
2. Resolución, rango de valores
3. Uniformidad, resolución, viveza
4. Resolución, época, precisión

24. La distancia en tiempo real entre dos tiempos de reloj consecutivos se define como:

1. Exactitud
2. Resolución
3. Granularidad
4. Todas las anteriores

25. La estructura *timespec* que especifica el tiempo de reloj en formato POSIX contiene:

1. Un campo de tipo entero con el número de segundos y un campo de tipo entero largo con el número de nanosegundos desde el encendido del equipo
2. Un campo con el número de segundos y un campo con el número de microsegundos transcurridos
3. Dos campos: número de segundos y nanosegundos transcurridos desde la época POSIX
4. Cuatro campos: horas, minutos, segundos, nanosegundos transcurridos

26. Se dice que una señal POSIX es atrapada cuando

1. Se ejecuta el manejador de señal cuando se deposita
2. El proceso asociado emprende una acción con base a ella
3. La señal está bloqueada pendiente de ser depositada
4. Se ignora el manejador cuando es depositada

27. La llamada al sistema POSIX para enviar una señal de tiempo real a un proceso es:

1. kill()
2. sigsend()
3. sigqueue()
4. Todas las anteriores

28. Las señales POSIX de tiempo real:

1. Se ejecutan por orden de llegada sin prioridades
2. Se ejecutan según su prioridad con la señal de mayor número primero
3. Se ejecutan según su prioridad con la señal de menor número primero
4. Se ejecutan de forma asíncrona con los intervalos de reloj

29. La llamada al sistema *sigwait*() permite a un thread quedar bloqueado esperando por la aparición de una señal. Esta función realiza las siguientes tareas:

1. Atómicamente bloquea el conjunto de señales pasada como parámetro, pasa el thread a estado suspendido esperando por la señal
2. Cuando la señal aparece boquea el conjunto de señales pasado como parámetro, y pasa el thread a estado preparado
3. Cuando la señal aparece desbloquea el conjunto de señales pasado como parámetro y ejecuta el manejador asociado a la señal
4. Todas las anteriores

30. La estructura *itimerspec* permite asignar el valor de un temporizador, para ello contiene los siguientes campos:

1. Un solo campo con el valor inicial del temporizador
2. Un campo con el valor inicial del temporizador, y un campo con el valor del incremento del mismo
3. Tres campos con el valor inicial del temporizador, el valor a recargar cuando expire el temporizador, y la resolución del reloj de temporización
4. Un campo con el valor inicial del temporizador, y un campo con el valor a recargar cuando expire el temporizador

---

SOLUCIONES:

1) 3	11) 3	21) 1
2) 3	12) 1	22) 3
3) 2	13) 3	23) 2
4) 4	14) 4	24) 3
5) 3	15) 2	25) 3
6) 2	16) 3	26) 1
7) 4	17) 1	27) 3
8) 1	18) 4	28) 3
9) 3	19) 3	29) 2
10) 2	20) 2	30) 4