**Afirmaciones verdaderas cuando se ejecuta el comando pthread\_create():**

* Es posible configurar el tamaño de stack y la prioridad del hilo en el segundo argumenta de la llamada.
* Si la creación es exitosa, el hilo ejecuta la función indicada como tercer argumento.
* Si la creación del hilo no es exitosa, retorna el número de error.
* Si no hubo error, almacena en un buffer indicado por el primer argumento el TID.

rc = pthread\_create(&hilo[0], NULL, hola , (void \*)(&t) );

**RTS**

**Indique lo correcto respecto a RTS (Real Time System)**

* El determinismo en RTS es poder cumplir predeciblemente requisitos temporales
* Los RTS no tienen que ejecutarse rápidamente.
* Los algoritmos de supervisión se ejecutan con periodos generalmente mayores a los algoritmos de control.
* En el nivel de software interface de usuario, no hay restricciones temporales estrictas.

**Implementación foreground/background (primer plano/Segundo plano) para RTS.**

* La latencia en estos sistemas dependerá de lo que demore la ejecución de las rutinas de atención de interrupción.
* Pueden tener incoherencia si los datos se comparten entre rutinas de interrupción y tareas de primer plano.
* La latencia en estos sistemas está definida por las rutinas de atención de interrupción.

**Una lo que corresponda:**

* **Un RTOS no expropiativo**: Puede tener incoherencia solo si se comparten datos entre ISR y tareas de primer plano.
* **Un sistema background/foreground respect de un scan loop**: disminuye la latencia de las tareas de segundo plano
* **Un sistema scan loop:** tiene el mayor de los rendimientos.
* **Un RTOS expropiativo respecto a un RTOS no expropiativo**: disminuye la latencia de las tareas de primer plano.

**RTOS expropiativo:**

* El uso correcto de semáforos o colas de mensaje permite evitar problemas de incoherencia.
* Este tipo de RTOS permite disminuir el tiempo de latencia de foreground respecto a los RTOS no apropiativos.

**RTOS no apropiativos**

* Tienen mayor latencia las tareas de foreground que en los RTOS apropiativos.

**Implantación de RTS con procesamiento secuencial.**

* Si se necesita un periodo de muestreo (Ts) determinado, se adiciona un timer y una tarea, siempre que Ts sea menor al tiempo que demoran las tareas que componen el bucle.
* En caso que una tarea del bucle con periodo mayor a Ts y tiempo de ejecución largo, es posible dividirla en dos o más, para que se cumpla el Ts.

**Tarea A se está ejecutando en FreeRTOS**

* Si la tarea B que es mas prioritaria que A se pasa a estado listo, no es necesario esperar el quantum para realizar la conmutación y ejecutar la tarea B.
* Si la tarea B que es de igual prioridad que A se pasa a estado listo, se debe esperar el quantum o que A se bloquee para realizar la conmutación y ejecutar la tarea B.

**Seleccione lo correcto respecto a ISR**

* Las ISR no deben invocar a funciones del RTOS que conmuten de tareas.

**Memoria Virtual**

**Algoritmos de reemplazo de página para memoria virtual:**

* El algoritmo Working Set se implementa con una lista circular y los bits M y R.
* El algoritmo aging es una alternativa de LRU, sin utilizar hardware especial.
* El algoritmo NRU es más fácil de implementar que el algoritmo working set.
* El algoritmo wsclock es una mejora del algoritmo working set.
* El algoritmo optimo es imposible de implementar, pero se usa con fines comparativos.
* El algoritmo aging es una modificación en software del LRU.

**Segmentación para memoria virtual**

* Cada segmento constituye un espacio de direcciones independiente
* Desacopla la relación 1 a 1 entre el bus de direcciones y la memoria física.
* Para acceder a una dirección física solo es necesario indicar el nombre del segmento y el offset.

**Técnica de paginación para memoria virtual**

* Permite ejecutar programas mas grandes que la memoria física.
* El bus de direcciones del procesador no esta directamente conectado a la memoria física.
* Solo crea un espacio de direcciones para cada programa en ejecución.
* EL bus de direcciones del procesador no está directamente conectado a la memoria física.

**Gestión de memoria con particiones fijas:**

* Puede existir fragmentación interna.
* Para asignar particiones se pueden usar cola única o múltiples colas.

**Administración de memoria con mapas de bits, para particiones variables:**

* Cuanto menor sea la unidad de asignación, menor será la fragmentación interna.
* La fragmentación externa NO depende del tamaño de la unidad de asignación de memoria.
* NO Corresponde:
  + La fragmentación externa depende del tamaño de la unidad de asignación de memoria.
  + Cuanto mayor sea la unidad de asignación de memoria, mayor será el mapa de bits.
  + Es rápida la asignación de memoria y lenta la liberación de memoria.
  + La asignación y liberación de memoria demoran el mismo tiempo.

**Algoritmos de ubicación de memoria, para gestión de memoria con particiones variables:**

* El mejor provoca más desperdicio de memoria que el primer ajuste.
* El primer ajuste es más rápido para asignar memoria que el peor ajuste.
* El siguiente ajuste tiene rendimiento un poco peor que el primer ajuste.
* El mejor ajuste provoca mas desperdicio de memoria que el primer ajuste.
* El siguiente ajuste tiene rendimiento mejor que el mejor ajuste.
* El algoritmo del mejor ajuste se puede mejorar con listas separadas y ordenadas de huecos y procesos.
* El mejor ajuste provoca mas desperdicio de memoria que el primer ajuste.
* EL primer ajuste es mas rápido para asignar memoria que el peor ajuste.

**HILOS**

**Recursos compartidos por los hilos de un mismo proceso.**

* Señales y manejadores de señales
* Variables globales
* Espacio de direcciones

**Afirmaciones verdaderas respecto al uso de hilos.**

* Ante un error en tiempo de ejecución en el código del hilo, se termina el proceso al que pertenece el hilo.
* Permiten el paralelismo real en computadoras de más de un núcleo.
* La ejecución de hilos es separada y los recursos son compartidos.
* Puede mejorar la velocidad de ejecución en sistemas multiprocesador.

**Recurso compartido por los hilos de un mismo proceso.**

* Espacio de direcciones (space address)
* Memoria dinámica (heap)

**Planificador**

**Planificación de procesos.**

* La necesidad de los planificadores surge para poder tener en memoria mas de un proceso a la vez.
* Al poder tener en memoria mas de un proceso a la vez, es necesario el uso de planificadores.

**Algoritmo de planificación Fiar-Share**

* Es un algoritmo expropiativo
* Es un sistema proporcional para los usuarios

**Algoritmo de planificación Round-Robin**

* No usa técnicas de envejecimiento
* Si a una tarea en ejecución se le termina el quantum, va al final de la FIFO
* Es un algoritmo equitativo.
* No es necesario que las tareas estén disponibles en el momento inicial.
* Si a una tarea en ejecución se le termina el quantum, va al final de la FIFO.

**Algoritmo de planificación Shortest Remaining Time Next**

* Es la versión expropiativa de shortest job first
* Se debe conocer de antemano el tiepo de ejecución de los trabajos.
* Favorece la ejecución de tareas cortas.

**Multi threading**

* En la implementación de hilos de manera hibrida: Cada hilo de nivel kernel tiene algún conjunto de hilos de nivel usuario
* En la implementación de hilos modo kernel: Cuando un hilo se bloquea, no bloquea al resto de los hilos del proceso.
* En la implementación de hilos en modo usuario: La creación de nuevos hilos es más rápida.

**Algoritmo shortest process next**

* Usa una aproximación basada en registración del comportamiento anterior.
* Es fácil de implementar, sumando el nuevo valor a la estimación actual y desplazando un bit a la derecha.
* No es necesario tener todos los procesos en tiempo inicial T0.

**Algoritmo de planificación por prioridad con múltiples colas**

* Cuando los procesos se convierten interactivos, se les asigna la mayor prioridad
* Disminuye la cantidad de cambios de contexto.
* Se asigna distinta cantidad de quantums a cada nivel de prioridad.

**La inversión de prioridad:**

* Se puede dar entre dos tareas de distinta prioridad que utilicen un semáforo binario compartido.
* Se puede dar entre dos tareas de distinta prioridad que utilicen una cola de mensaje compartida.
* Siempre aumenta la latencia de la tarea más prioritaria.

**Condiciones necesarias para que exista deadlock entre dos o más eventos.**

* Exclusión mutua de los eventos
* Espera circular entre los eventos
* Sin expropiación de los eventos.

**Indique lo correcto referido a concurrencia del evento A respecto del evento B**

* El evento A y el evento B pueden ejecutarse simultáneamente en caso de usar sistemas multitarea.
* No es necesario forzar la situación de concurrencia, ni usar mutex, barrier o semáforos.

**Indique lo correcto referido a punto de encuentro del evento A respecto del evento B**

* El evento A debe esperar la ocurrencia del evento B, e inversamente
* Para forzar la situación de punto de encuentro (para dos eventos) se pueden usar dos semáforos, ambos inicialmente en 0.

**Indique lo correcto referido a exclusión mutua del evento A respecto del evento B**

* Para forzar la situación de exclusión mutua (para dos eventos) se puede usar un semáforo, inicialmente en 1.
* El evento A no debe ocurrir en el mismo momento que el evento B

**Indique lo correcto respecto a serialización del evento A respecto del evento B**

* El evento A debe esperar la ocurrencia del evento B.
* Para forzar la situación de serialización se puede usar un mutex, inicialmente en 0.