# Primer Parcial Teórico – Repaso

## THREADS

1. Un thread no tiene program counter (PC) propio. Falso

   Cada thread tiene su propia TCB, esta misma contiene propio PC y registros. Además el PC del proceso va a ser el del último hilo que se ejecutó. El hilo es la unidad básica de ejecución.

2. Un context switch entre threads, no requiere un context switch de registros. Falso

   Cada thread tiene su propia TCB, esta misma contiene propio PC y registros. Para el profesor puede ser verdadero. Si el switch está entre hilos del mismo proceso, hay ciertos registros que no cambian, no debo cambiar todos haciendo más liviana la tarea. Igualmente es muy raro que no cambien NINGÚN registros (ya que el PC va a cambiar). No necesariamente cambian todos

3. Un hilo creado por un proceso tendrá su propio contexto. Verdadero

   Cada hilo creado por un proceso tendrá el mismo contexto pesado (ej. espacio de direcciones y recursos) del proceso y a su vez tendrán su propio contexto a nivel de hilo (ej. estado, stacks de usuario y kernel, variables propias)
   Cada hilo tiene su TCB que es propia del mismo (es inherente al hilo), el hilo tiene un contexto propio. El que tiene el contexto real es el proceso, un contexto general.

4. Un hilo creado por un proceso se ejecutará en el espacio de direcciones de este último. Verdadero

   Los hilos de un mismo proceso comparten el espacio de direcciones del proceso.

5. Un hilo es la unidad básica de uso de la CPU. Verdadero

Unidad básica de utilización de CPU: Hilos
Un proceso al menos siempre tiene un hilo.

6. Un hilo es la unidad de propiedad de recursos. Falso

   Unidad básica de asignación de RECURSOS: Proceso
   El sistema planifica el proceso, asigna recursos al proceso.

7. Dentro de un proceso, un hilo cuenta con un estado de ejecución. Verdadero

   Cada hilo tiene su propio estado de ejecución. Puede tener un hilo estado running y el proceso en espera.

8. Existe una PCB por proceso y por cada hilo que él cree. Falso

   Existe una PCB por proceso y una TCB por hilo. En hilos a nivel de usuario (ULT) el SO no ve la TCB.
   Un proceso no accede a la PCB y un hilo no accede a la TCB, debido a que están protegidas (espacio de dirección del Kernel). El que se encarga de que el hilo no acceda de forma directa a la TCB es la librería del hilo.
   Sea el hilo que sea la TCB va a existir, el tema es donde se maneja:
   - ULT - Librería hilo.
   - KLT - Kernel.

   El SO no ve que hay threads si se tiene ULT.

9. Cuando un proceso se swapea, los hilos quedan en memoria en estado de espera. Falso

   Cuando un proceso se swapea, se baja al disco todo el proceso incluido sus hilos. KLT se bloquea cada hilo individualmente, el ULT cada uno queda en su propio estado. Todo lo que le pase al proceso le pasa a los hilos contenidos en él.

10. En la administración de los hilos a nivel de usuario, interviene el kernel. Falso

Interviene la librería del hilo.

11. En los ULT, cada proceso se encarga de administrar sus hilos. Verdadero

    Se encarga de administrarlo la propia aplicación, es decir, el proceso.

12. La suspensión de un ULT provoca la suspensión del proceso. Falso

    Se pueden reemplazar llamadas al sistema bloqueantes por otras que no bloquean, durmiendo así solo al hilo. Se puede dormir solo el hilo. Si se usa una llamada al sistema bloqueante entonces se suspende también el proceso.

13. En los KLT, el context switch entre hilos, no provoca un cambio de modo. Falso

    Se necesitan cambios de modo de ejecución para la gestión. Que sea un KLT no quiere decir que siempre se ejecute en modo Kernel, se lo va a ejecutar así cuando sea necesario (una gran parte de su tiempo, pero no todo). Es como un cambio de contexto entre procesos.

## MULTIPROCESADORES

14. Los KLT, en un ambiente multiprocesador, pueden ejecutarse en distintos procesadores. Verdadero

15. En multiprocesadores, en la organización maestro esclavo, una syscall puede ser atendida en cualquiera de los procesadores. Falso

    Todas las syscalls van a una sola CPU.

16. En multiprocesadores, si cada CPU tiene su SO es posible que una CPU esté saturada y otras sin trabajo productivo. Verdadero

    Hay un desbalance de carga. Cada CPU tiene su propio scheduling. La instancia principal de SO dice que tarea va a cada CPU, y después a cada CPU schedulea propiamente.

17. En multiprocesadores, la técnica de SMP no requiere de exclusión mutua para el acceso a las estructuras del kernel. Falso

    Todas las CPUs comparten las estructuras del SO por lo que se debe garantizar su acceso exclusivo. Hay una sola copia de las estructuras del SO.

18. No existen diferencias en la planificación de procesos entre SO monoprocesadores y multiprocesadores. Falso

    Como hay varias CPU la complejidad aumenta en la distribución de tareas, el pasaje de mensajes y el acceso a memoria.

19. En un sistema distribuido, todos los SO de las diferentes computadoras que participan deben ser iguales. Falso

    Pueden ser diferentes SO ya que luego hay un middleware que se encarga de la homogeneización.

20. En las multicomputadoras, cada CPU tiene su memoria. Verdadero

21. En multicomputadoras, la comunicación entre procesos se realiza por:
    a) Memoria Compartida
    b) Pasajes de Mensajes
    c) RPC
    d) Ninguna

    Yendo a un extremo Pasaje de Mensajes generalmente es entre procesos de la misma máquina.
    Los procesos de una misma máquina se pueden comunicar a través de la memoria compartida.

22. En multicomputadoras, cada nodo puede correr un SO diferente. Falso

En realidad eso sería un sistema distribuido. Aunque es válido que tengan distinto SO (se puede) de acuerdo a la bibliografía las multicomputadoras son todas IGUALES incluso con el mismo SO.
Multicomputadoras → muchas computadoras funcionando como iguales.

23. Las computadoras que forman una Grid deben ser todas iguales. Falso

    Pueden tener distinto Hardware.
    GRID → Cluster de cluster. No todas tienen que ser iguales. Las de un cluster si son iguales, pero no todos los cluster son iguales.

24. El middleware es una capa de software entre el Hardware y el Sistema Operativo. Falso

    Es una capa de software por encima del SO.

## DEADLOCKS

25. Basta que una de las 4 condiciones de deadlock se cumpla, para que haya deadlock. Falso

    Se deben cumplir las 4.

26. La desventaja de usar algoritmos de prevención del deadlock, es que baja el grado de multiprogramación. Falso

    Cuando se habla de la técnica de prevenir, se habla vs. a la de evitar. Si se tiene prevenir o evitar, tiene mejor grado de multiprogramación la de prevenir.
    Prevenir → asigna los recursos y busca que no se cumplan las condiciones.
    Evitar → asigna cuidadosamente los recursos en base al estado que vaya quedando el sistema disminuyendo el grado de multiprogramación.
    Es falso respecto a la técnica de evitar.
    Es verdadero respecto a que no realiza nada.

27. En un esquema de una instancia por tipo de recurso, cuando se encuentra un ciclo en un grafo de alocación de recursos, la asignación de los recursos solicitados:
    a) puede poner al sistema en estado inseguro
    b) pone al sistema en estado inseguro.

28. Todos los estados inseguros son deadlock. Falso

    En estado inseguro hay posibilidad de que haya deadlock. No todo estado inseguro es deadlock. Si estoy en estado inseguro y a la larga no se hace nada hay deadlock.

29. El algoritmo del Banquero sirve para sistemas con múltiples instancias de cada recurso. Verdadero

30. Siempre que el grafo de recursos tiene ciclos, hay deadlock. Falso

    Si hay varias instancias por tipo de recurso hay posibilidad de deadlock pero no quiere decir que necesariamente haya.