Introducción a los Sistemas Operativos
Repaso 1er. Parcial Teórico

1. ¿El SO necesita tiempo de CPU?

Si, ya que es software y necesita tiempo de CPU.

2. ¿Pueden convivir en un mismo SO la modalidad Batch y la Interactiva?

Si, en un mismo SO pueden convivir los dos tipos de proceso. Si bien un proceso puede ser batch o interactivo, no puede ser los dos al mismo tiempo.

En los procesos interactivos, se interactua con un usuario, en cambio los procesos batch no tienen interacción.

3. ¿Puede un sistema monousuario ser multitarea?

Si, un usuario puede lanzar varios procesos.

4. ¿Puede un sistema multiusuario ser monotarea?

Poder puede, pero no tiene sentido.

Imaginen muchos usuarios pero una sola tarea en memoria, no tiene sentido dado que tengo que swapear cada vez que cambio de usuario.

5. ¿Los sistemas time sharing son una consecuencia de la multiprogramación?

Time sharing: Sistemas de tiempo compartido.

Esta relacionado a multiusuarios y tienen la posibilidad de tener un cachito de microprocesador cada uno. Por lo tanto el time sharing se puede hacer gracias a la multiprogramación.

No se puede implementar time sharing si no tengo multiprogramación (varios procesos en memoria).

6. ¿Puede un programa ejecutarse desde el disco?

No, tiene que estar cargado en memoria y tener la CPU.

7. ¿Puedo planificar el uso de la CPU si no cuento con memoria secundaria?

¿Para que sirve la memoria secundaria en uso de CPU? Para swapear.

¿Que pasa con los procesos nuevos que todabian no fueron cargados en memoria principal? Estan en disco.

No, ya que la memoria secundaria se utiliza para swapear y para los procesos en estado de nuevo.

En caso de sistemas de tiempo real los procesos ya están en memoria, porque hay una exigencia de tiempo que es muy fuerte.

8. La interrupción por clock impide que un proceso se apropie del procesador.

Si, se implementa normalmente a través de un clock y un contador. El SO da valor al contador que se decrementa con cada tick de reloj y al llegar a cero se produce la expulsión del proceso.

9. La interrupción es externa al proceso.

Verdadero, las interrupciones son externas al proceso, en cambio las interrupciones por software o TRAP o EXCEPCION no. Cuando ocurre una interrupción se ejecuta la rutina de interrupción (en modo KERNEL).

10. Un intento de acceder a una dirección ilegal, se trata como una excepción o trap.

Verdadero, el hardware lo detecta, se trata de una excepción.

11. Un proceso puede acceder al espacio de direcciones de otro proceso si esta en modo usuario.

No, un proceso en modo usuario puede acceder sólo a su espacio de direcciones.

Si esta en modo usuario va a estar delimitado por el registro base y registro limite.

Espacio de direcciones: Conjunto de direcciones al que el proceso puede acceder en modo usuario.

12. Una llamada al sistema (system call) genera un proceso del sistema operativo.

Falso, cuando se tiene que ejecutar instrucciones que están en el espacio de direcciones del kernel, el mismo proceso pasa a modo KERNEL (No se genera un nuevo proceso).

13. La llamada al sistema es la forma que tiene la aplicación de comunicarse con el sistema operativo.

Verdadero, ya que lo que no puede resolver el proceso dentro de su espacio de direcciones lo hace mediante una system call.

Por ejemplo si tengo la instrucción C = A + B, estando A y B en memoria ¿necesito la interacción del SO? No.

Si A y B estarían en memoria secundaria no se va a poder ejecutar esa instrucción, en ese caso si va a ver interacción por parte del SO.

14. Si tengo varios trabajos orientados a entrada/salida, las colas de solicitudes a los dispositivos estarán vacías.

Falso, como tengo trabajos orientados a I/O, las colas de I/O van a estar llenas.

Cuando hablamos de colas de procesos listos, colas de dispositivos de E/S, estamos hablando de colas de los PCB que están listos.

Nunca la PCB pasa a disco, aunque el proceso se swapee esta en memoria.

15. Buffer, es espacio en memoria, y spool, espacio en disco.

Los dispositivos tienen buffer, estamos hablando de buffer como almacenamiento temporal secundario.

Un spool es un cacho de disco.

16. Si varios usuarios mandan a imprimir a la misma impresora, los archivos se almacenan temporariamente en el buffer.

Falso, el buffer no es tan grande como para almacenar muchos archivos, se almacenan en el spool.

17. El sistema operativo permite al usuario abstraerse del hardware y su manejo.

Verdadero, las funciones básicas de un SO son:

- Manejo de INTERRUPCIONES
- Manejo de PROCESOS
- Manejo de MEMORIA

18. Es lo mismo el kernel que el sistema operativo?

Falso, es una porción del SO.

Si yo trabajara con el Kernel seria muy complicado, el Kernel es el que tiene las tareas esenciales de un SO, por ejemplo manejo de interrupciones, administración de la memoria, etc.

19. Las bibliotecas proveen módulos de sw para ser invocados en tiempo de compilación.

Falso, no tiene sentido ya que si las bibliotecas se actualizaran debería volver a compilar el programa, la invocación se hace en tiempo de ejecución.

20. La memoria principal es un recurso del tipo multiplexada en el espacio.

Verdadero.

Cuando hablamos de multiplexado, estamos hablando de como se comparte ese recurso, vamos a pensar que en un momento dado yo tomo la memoria, es un espacio que es compartido, en ese momento hay varios procesos por eso se dice que es un recurso multiplexado en el espacio. En cambio la CPU es un recurso multiplexado en el tiempo, porque en un momento dado la CPU esta usando un proceso solo.

21. El procesador en un sistema monoprocesador es un recurso del tipo multiplexada en el tiempo a cada proceso.

Verdadero.

22. Open (archivo) se implementa como una system call?

Verdadero, las system call se ejecutan en modo KERNEL, lo parámetros fundamentales del OPEN son: si EXISTE el archivo, y si el usuario tiene PERMISOS sobre el archivo.

System call READ: tiene 3 parámetros (de donde, donde se guarda, cantidad de bytes), el WRITE tiene los mismo parámetros.

23. Date se implementa como una system call?

Verdadero.

24. Un proceso tiene un stack en modo usuario y un stack en modo supervisor. Como no se usan a la vez, ocupan la misma dirección de memoria. (V o F)

Falso, principalmente el stack en modo usuario almacena direcciones en modo usuario, el stack en modo supervisor almacena direcciones en modo supervisor.

No pueden estar en la misma dirección de memoria.

25. El estado del proceso está en la PCB. (V o F)

Verdadero. La PCB contiene información asociada con cada proceso: Estado, Contenido del PC (program counter), Contenido de los Registros de la CPU.

26. Un proceso crea a otro mediante un system call. (V o F)

Verdadero, a traves de la System call FORK (devuelve 0 y el PID y el PPID) o CreateProcess en Windows.

27. La cola de procesos está en el disco. (V o F)

Falso, está en la memoria, y no están los procesos sino sus PCB.

28. Cuando un proceso se crea, está en disco. (V o F)

Si, en memoria está la PCB.

No necesariamente cuando se crea un proceso esta en memoria, tiene que ser elegido por el Scheduler de Long Term para ser llevado a memoria.

29. El proceso padre crea al hijo en su propio espacio de direcciones. (V o F)

Falso, cada uno tiene su propio espacio de direcciones, como su PCB y PID.

30. Las tablas de archivos que usa el proceso forma parte de su contexto. (V o F)

Verdadero. PCB, tablas de archivos, tablas de pagina, todo eso es contexto, forma parte de su ambiente de ejecución.

Un proceso esta formado por código, datos y stack.

31. La PCB se crea a partir que el proceso se carga en memoria. (V o F)

Falso, es lo que primero se crea.

La PCB es creada antes que el proceso sea llevado a memoria.

32. El proceso padre y el hijo comparten la PCB. (V o F)

Falso, la PCB es única de cada proceso.

33. Si no fuera por la E/S, los procesos no necesitarían system calls. (V o F)

Falso, las necesitan para comunicarse con el SO.

Por ejemplo hay sistem call para manejo de archivos.

34. Para que un proceso pueda acceder al espacio de direcciones de otro proceso, debe estar en modo supervisor. (V o F)

Verdadero.

35. El contexto de un proceso es lo mismo que su espacio de direcciones. (V o F)

Falso.

36. Para impedir que un proceso se apropie del procesador, existen las interrupciones por clock.

Verdadero.

37. Para implementar prioridad dinámica o aging, se tiene en cuenta: a) cuanto tiempo de CPU usó el proceso recientemente; b) cuanto tiempo de espera tiene acumulado.

En realidad se pueden usar cualquier de las dos cosas.

¿Que quiere decir esto? Antes la prioridad de los SO viejos era estática, la prioridad que se le daba al proceso dependía del usuario que lo había ejecutado, tenia el problema que si había un proceso de baja prioridad podría nunca llegar a ejecutarse, produciendo inanición, esto se soluciono con prioridad dinámica, una alternativa es a medida que el proceso esta en espera ya sea en cola de listos o dispositivos se le va aumentando la prioridad llegando a poder competir con los demás procesos (aumentarse la prioridad a los procesos que están en las colas) otra es bajarle la prioridad a los procesos que terminan de usar la CPU, ya sea porque se termino su quantum o porque quiso hacer E/S.

38. Es lo mismo cambio de contexto que cambio de proceso?

No, no es lo mismo, un cambio de proceso genera un cambio de contexto.

39. Es lo mismo cambio de contexto que cambio de modo?

No, un cambio de contexto es cuando se esta ejecutando un proceso y quiere ejecutarse otro proceso, un cambio de modo es pasar de modo usuario a modo supervisor o viceversa.

40. Un cambio de contexto involucra un cambio de modo.

Si. ¿Quien hace el cambio de contexto? El dispacher, hace 3 acciones, guarda el estado del proceso que se está ejecutando en la PCB, toma la información de la PCB del proceso entrante y la carga en la CPU, cambio de modo (de modo supervisor a modo usuario) y salto a la instrucción a ejecutar.

41. Un cambio de modo involucra un cambio de contexto

No, cuando paso a ejecutar una system call cambio de modo pero no de contexto.

42. Un fork exitoso produce cambios en la PCB del padre pues se almacena .... del

hijo.

Fork: System call que genera un proceso hijo.

Se almacena la PID del hijo.

43. El espacio de direcciones de un proceso está delimitado por los registros ..... y .....

Registro base y registro limite.

44. EL fork devuelve dos valores: ... al proceso hijo y ..... al proceso padre.

Al padre le devuelve la PID del hijo y al hijo le devuelve 0. ¿Porque devuelve 0? Porque el hijo heredo un montón de información del padre, ya sabe la PID del padre.

45. El acceso no autorizados por parte de un proceso a una posición de memoria es detectado por:
a) El S.O.
b) El Hardware
c) No es detectado nunca

b)

46. Las Systems Calls se ejecutan en “Modo Privilegiado”. V o F

Verdadero.

47. Ante un cambio de contexto, indique cuáles de estos elementos se guarda en la PCB:
a)tabla de páginas;
b)pila de usuario;
c)tabla de archivos abiertos;
d) estado del proceso

Se guarda la dirección de tabla de pagina, dirección de pila de usuario, dirección de tabla de archivos abiertos y estado del proceso, sino debería variar de tamaño.

48. El chequeo de la existencia de una interrupción se realiza entre los pasos de “Fetch” y “Execute” de cada ciclo de instrucción.

Falso, cuando termina el Execute.

49. El vector de interrupciones siempre debe estar en memoria.

Verdadero.

En el vector de interrupciones están las direcciones de las rutinas de atención de interrupciones.

50. Un system call fork, provocará cambio de contexto.

Verdadero. Puede ser que haya lugar en la memoria o no, si hay lugar en la memoria puede haber un cambio de modo.

51. Un proceso swappeado en estado listo (ready to run) no compite por CPU.

Verdadero, ya que al estar en disco no puede ser ejecutado.

Si compitiera por CPU y ganara, el proceso esta en memoria secundaria y para ejecutarse tiene que estar en memoria. ¿En que moemnto un proceso swapeado sube a memoria principal? Al llegar a determinada prioridad se lo sube a memoria.

52. El scheduler de short term se ejecuta con menos frecuencia que el de long term.

Falso, el scheduler de short term es el que se ejecuta con mayor frecuencia.

53. El cambio de contexto lo hace el scheduler de long term.

Falso. Lo hace el dispacher.

54. Cuando un proceso se crea, se pone en la cola de procesos, hasta que es elegido por el scheduler de long term.

Falso.

Cuando un proceso se crea su PCB se pone en la cola de procesos nuevos, hasta que es elegido por el scheduler de long term.

55. Cuando a un proceso se le termina su quantum, pasa a estado de espera.

Falso, pasa a estado de listo.

56. El scheduler de medium term es quien decide el cambio entre nuevo y ready.

Falso. El scheduler de medium term maneja la selección de los procesos que intervienen en el swaping.

Hay 5 modulos que intervienen en la planificación, los 3 scheduler, dispacher y loader.

57. El scheduler de short term es quien hace pasar al proceso de estado ready a running.

Verdadero. ¿Porque es el que decide que pase de estado de ready a running? Porque es el que elije en la cola de listos a quien se le da la CPU.

58. En la planificación de CPU se trata de maximizar la productividad, minimizar el tiempo de respuesta.

Verdadero.

59. El tiempo de retorno, es el tiempo desde que se inicia hasta que termina, sumando cpu, espera en colas.

Verdadero.

60. Supongamos que un proceso está en espera swappeado y se cumple el evento por el que estaba esperando. El proceso queda en estado de listo en memoria secundaria.

Verdadero.

¿Puedo swapear un proceso que se esta ejecutando? No.

¿Tiene sentido que el proceso que esta en espera en memoria secundaria vuelva a memoria principal? No, para que lo voy a llevar a memoria si no lo puedo ejecutar.

¿Tiene sentido un proceso que esta en memoria secundaria llevarlo a memoria principal? Si.

61. Según el diagrama visto: puede un proceso pasar del estado de nuevo (creado) a listo swappeado? SI - NO

No.

62. Un proceso puede pasar de esperar en memoria secundaria a esperar en memoria real.

No, no tiene sentido.

63. El scheduler de medium term maneja el grado de multiprogramación.

No lo maneja, disminuye o aumenta el grado de multiprogramación (cantidad de procesos que hay en memoria).

64. El disco permitió implementar la planificación de procesos.

Si.

65. Cuando a un proceso se le termina su quantum, pasa a estado de espera.

Falso, pasa a estado de listo.

66. En un sistema monoprocesador, cuando se ejecuta la rutina de interrupciones, el resto de los procesos quedan en espera.

En espera o en estado de listo.

Si yo tengo un solo procesador y en ese momento se esta ejecutando la rutina de acción de interrupciones, no se puede ejecutar otro proceso, el proceso que se estaba ejecutando queda en estado de suspendido.

67. En un ambiente multiprocesador; a) cada procesador tiene su propia cola de procesos. b) hay una cola de procesos para todos los procesos

En realidad se podría implementar cualquier de las dos modalidades.

68. Priorizar los procesos cortos a los largos elimina el efecto convoy.

Verdadero.

Si permito que se ejecuten todos los procesos largos, voy a tener la cola de listos con muchos procesos cortos, debería penalizar a los largos dejando que se ejecuten los procesos chicos.

69. Conviene que un proceso en espera swappeado retorne a memoria principal?

No, no tiene sentido porque no se puede ejecutar.

70. En un ambiente interactivo y batch, que se maneja con colas multiples con retroalimentación... conviene usar algoritmos apropiativos?

Verdadero. Tiene que haber algoritmos apropiativos (permite quitarle la CPU a un proceso) para poder sacar los procesos batch.

71. Indique cuál es la combinación que representa la sucesión de actividades que realiza el dispatcher:
a)Cambio de contexto;
b) Cambio de Modo;
c) Salto a primer instrucción a ejecutar;
d) Carga en memoria del proceso elegido

a) b) y c), d) no porque puede ser que el proceso ya este en memoria.

72. Indique que puede ocurrir cuando tengo demasiados procesos orientados a I/O:
a) Se incrementa el uso de CPU;
b) se saturan las colas de dispositivo;

b)

74. Cuando se carga un proceso en memoria, se hace en modo usuario.

Falso.

75. En la administración particionada de memoria, no puede haber swapping.

Falso.

76. En la administración particionada de memoria hay fragmentación interna.

Hay que aclarar en que partición, fijas o dinámicas, en las particiones fijas se puede producir fragmentación interna y en la dinámicas fragmentación externa.

77. En la estrategia best fit, se elige de entre las particiones disponibles, la que produzca menor fragmentación interna.

Falso, es fragmentación externa.

78. En la estrategia worst fit, se elige de entre las particiones disponibles, la que produzca mayor fragmentación interna.

Falso, se elige entre las particiones disponibles la que produce mayor fragmentación externa.

En la FIJAS: Se produce mayor fragmentación interna.

En las DINAMICAS: No hay fragmentación interna porque se produce un hueco para otro proceso.

79. La más rápida de las estrategias de asignación de particiones de memoria, es la first fit.

Verdadero, o la NEXT FIT si se implementa como una lista circular.

80. Resolver la dirección en el momento de la carga, exigirá que el proceso se ejecute siempre en el mismo lugar de la memoria.

Verdadero, lo que permite que el proceso pueda ir a memoria secundaria es la resolución al momento de la ejecución.

81. Si se resuelven las direcciones en el momento de la compilación y quieren cambiarse, se debe recompilar.

Verdadero, porque se están utilizando direcciones absolutas.

RECORDAR

PCB: No se guardan los stacks (pilas), sino que se guardan PUNTEROS a esas pilas sino serian muy grandes de las PCB.

RESOLVER UNA DIRECCION: Es pasar de una dirección lógica o virtual, a una física o absoluta. De esto se encarga la MMU en el medio de la transformación esta la CACHE DE MEMORIA.

Ciclo de interrupción: Se chequea después del FET.

82. En la Administración particionada fija, el grado de multiprogramación lo dá la cantidad de particiones.

Verdadero, en las particiones fijas hay un proceso por partición, si tengo 5 particiones fijas tengo 5 procesos por partición.

83. La resolución de direcciones en el momento de la carga facilita la administración paginada.

Falso.