Repaso primer parcial teórico

ROSA → nicolas
AMARILLO → el de la mañana

1. ¿El SO necesita tiempo de CPU?
si, porque es software
tiempo improductivo mientras no se ejecuta ningun proceso

2. ¿Pueden convivir en un mismo SO procesos batch y procesos interactivos?
si
proceso batch: no necesita inter con el usuario, proceso de calculo
proc int: interactúa con el usuario
pueden convivir claramente

3. ¿Puede un sistema monousuario ser multitarea?
si
multitarea no es multiprocesador

monotarea ser multiusuario?
no

4. ¿Puede un sistema multiusuario ser monotarea?

si puede, pero no debe porque no sirve

5. ¿Puede un programa ejecutarse desde el disco?
no
lo que se ejecuta es el proceso
además, necesita memoria
al momento de ejecutarse se crea un entorno llamado proceso

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

si
lo puedo planificar si tengo sistema basado en ram
la necesidad de una memoria secundaria es vital
una vez que están en la memoria podría prescindir de memoria secundaria
no podría swapear

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

8. Las interrupciones son externas a los procesos.
si,
la excepción causada por lo que está ejecutándose

9. Un intento de acceder a una dirección ilegal, se trata como un trap.
si
esta causado por una instruccion que causa la int x software
el origen del trap es lo que se esta ejecutando
el trap lo detecta el hardware

10. Un proceso puede acceder al espacio de direcciones de otro proceso si esta en modo usuario.
no
en modo usuario no puede

si el proceso invoca system call hay cambio de modo pero no de contexto
en modo kernel sí

areas de memoria compartida: el proceso tiene esas áreas dentro de sus tabla de pagina por la que se considera el espacio de direcciones propia

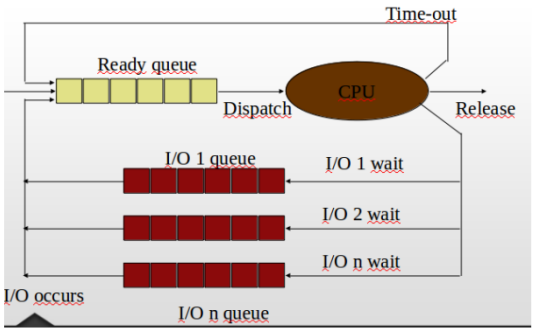
11. Una llamada al sistema (system call) genera la creación de un proceso del sistema operativo para atender la llamada.
no
los proceso son lo que usan en so para ejecutar el programa
no es una entidad proceso sistema operativo

no necesariamente porque
kernel q no es monolítico es kernel rodeado de procesos

el system call se ejecuta en el mismo proceso

12. Las llamadas al sistema son la forma que tienen las aplicaciones de comunicarse con el sistema operativo.
si
lo q no puede lo resuelve el so, se lo pide por system call

13. Si tengo muchos procesos orientados a entrada/salida, las colas de solicitudes a los dispositivos de E/S estarán vacías.



no
procesos encolados solicitando dispositivos

14. El sistema operativo permite al usuario abstraerse del hardware y su manejo.
si
funcion del so

15. ¿Es lo mismo el kernel que el sistema operativo?
no
so tiene conjunto de herramientas que hace amigable al usuario, facilidad pra pedir instrucciones y obtener respuestas
la shell se encarga de hacerlo con más facilidad
ACOMPANAN AL KERNEL

16. La memoria principal es un recurso del tipo multiplexada en el espacio.
si
compartido
en un momento dado hay espacio compartido que acceden varios procesos a la vez

17. El procesador en un sistema monoprocesador es un recurso del tipo multiplexado en el tiempo a cada proceso.
si
en el tiempo en un momento determinado un proceso está en la cpu

18. Date se implementa con una system call?

si
ya que se vale del tik del reloj y esto lo hace en modo privilegiado

estoy accediendo al clock en modo kernel (el so puede acceder ahi), no necesariamente tiene que ocurrir una interrupcion

19. 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)
no estan en el mismo lugar
cuenta 2 stack
si tiene 1 no puede ejecutar mas de 2 procesos a la vez (MONOUSUARIO)
falso

20. El estado del proceso está en la PCB. (V o F)
si
cuando se esta ejecutando pcb estado de running
“este se ejecuta en tal cpu”

21. Un proceso crea a otro mediante un system call. (V o F)
VERDADERO
si a través de fork
le sigue execve

fork (crea otro proceso igual)

– - - FORK - se ejecuta la línea que le sigue en el padre - - -	copia exactamente igual - - - -

si hago execve
reemplazo por imagen en el hijo (es lo que hace el intérprete de comandos)
reemplaza todo lo del padre

– - - FORK	copia exactamente igual

- no se ejecuta la linea que siguen - - esto no se llega a ejecutar nunca -	- si se ejecuta esto en el hijo - - -

0 al hijo pid al padre

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

no, la cola de procesos esta en la memoria ram
en la cola de procesos guardo las pcb

23. Cuando un proceso se crea, está en disco. (V o F)
NO, el proceso no esta en disco, el programa esta en disco
el proceso estan en memoria

falso porque el proceso esta en memoria ram
en disco esta el programa
en memoria tengo imagen de proceso

otra pregunta
cuando un proceso se crea parte de el esta en el disco (PAGINACION X DEMANDA), ESTO NO ENTRA

24. El proceso padre crea al hijo en su propio espacio de direcciones. (V o F)
no

a traves de una system call se crea el proceso que se ejecuta en modo kernel
estando en modo kernel, el padre no es el que crea el proceso sino que el so crea el proceso (por eso es que lo crea en un espacio de direcciones nuevo)

25. Las tablas de archivos correspondientes a los archivos abiertos que está usando un proceso, forman parte de su contexto. (V o F)
si
el contexto del proceso es el conjunto de estructuras que usa para el proceso, para que despues el proceso pueda acceder

el proceso NO accede a su contexto ya que está afuera de su espacio de direcciones
si pudiese puede cambiar la prioridad

TPICA
donde se almacena el contexto de un proceso EN LA PCB
NO EN EL ESPACIO DE DIRECCIONES

26. La PCB se crea a partir que el proceso se carga en memoria. (V o F)
no
long term decide el loader hace
short term decide el dispatcher lo hace
swap el medium term

se crea cuando esta en estado de new

27. Luego de la system call fork, el proceso padre y el proceso hijo comparten la PCB. (V o F)
no
la pcb que se crea tiene pid nuevo

el hijo y el padre son totalmente indepnedientes, el hijo es una copia en el sentido del mismo contenido

se crea contexto indenediente

28. Si no fuera por la E/S, los procesos no necesitarían system calls. (V o F)
no
hay system call q no requiere e/s

29. En modo supervisor, es posible acceder al espacio de direcciones de cualquier proceso. (V o F)
si
verdadero

30. El contexto de un proceso es lo mismo que su espacio de direcciones.(V o F)
no
el espacio de direcciones esta apuntado en el contexto osea guarda un puntero de la tabla de direcciones

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

b

tiempo de espera

depende del criterio o del algoritmo usado
las dos serian correcta ya que no solo existen los algoritmos q dimos

32. Un cambio de modo involucra un cambio de contexto
no
ejemplo el clock cambia de modo y no contexto
falso puede mismo proceso pasar a ejecutarse en mod kernel

33. Un cambio de contexto involucra un cambio de modo.
si – tiene que pasar a modo usuario antes de darle la cpu al q viene
verdadero
el cambio de contexto lo hace el dispatcher y es kernel (TENGO Q HACER CONTEST SWITCH)

34. Es lo mismo cambio de contexto que cambio de proceso?
si
si es lo mismo la frma correcta de llamarlo es cambio de contexto

35. Es lo mismo cambio de contexto que cambio de modo?
no
NO

36. Un fork exitoso produce cambios en la PCB del padre pues se almacena .... del hijo.
el pid
EL proceso padre tien lista con los pid de los hijos

37. En el mecanismo de manejo de memoria con particiones, el espacio de direcciones de un proceso está delimitado por los registros ..... y .....
base y limite

38. El fork devuelve dos valores: ... al proceso hijo y ..... al proceso padre.

padre + (se creo y recibe pid) o - si no se crea
hijo 0
un valor mayor a 0 al proceso padre q es el pid del hijo

39. Un acceso no autorizado a memoria es detectado por:
a) El S.O. b) El Hardware c) No puede detectarse
b

El hardware DETECTA
el que actua es el so decide si aborta el proceso
la mmu detecta

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

41. 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

a,b,c,d
guardo todas!!!!!!!!!
lo dijo nicolas (el de la tarde)
no solo el PC y dos registros, sino que IMAGEN DEL PROCESO

d
se guarda estado del proceso nada mas (el de la mañana)

y registro q estan el stack pointer, donde hay punteros
las tablas no se guardan, solo los punteros
ante cambio de contexto no necesariamente cambian

42. El chequeo de la existencia de una interrupción se realiza entre los pasos de “Fetch” y “Execute” de cada ciclo de instrucción
falso
(falta el graficooooooooooooo)

43. El vector de interrupciones siempre debe estar en memoria
si

44. Un system call fork, provocará cambio de contexto

FALSO no necesariamente, la idea es que el proceso padre siga
PODRÍA EVENTUALMENTE ante situación determinada, si el scheduler toma la decisión de ejecutar el otro puede pasar

no porque crea las estructuras para tener proceso
el fork no lo produce
fork en modo kernel

kill wait son servicios no implican cambio de contexto

cambio de modo si

45. Un proceso swappeado en estado listo (ready to run) no compite por CPU.
VERDADERO
esta swapeado no listo
(ready to run swapped)

(diagrama completo)

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

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

f
el dispatcher

48. Cuando a un proceso se le termina su quantum, pasa a estado de espera.
F, vuelve a la cola de listos

pasa a estado de listo

49. El scheduler de medium term es quien decide el cambio entre nuevo y ready.
F
falso, lo decide el long term
el medium term, swap in swap out

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

falso, selecciona
el dispatcher es el que verdaderamente hace pasarlo

51. En la planificación de CPU se trata de maximizar la productividad y minimizar el tiempo de respuesta.
si
buscas que los procesos ni bien llegan se ejecuten y ser productivo

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

si, verdadero (incluye todo)
desde que la llegada

53. 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.

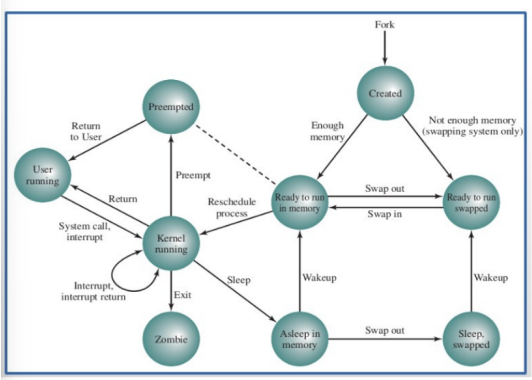
verdadero espera evento se decidio swapearlo, queda ahi y el medium term va a tomar la decision de pasarlo a memoria principal

si
listo es swapeado

espera - swapeado es diferente

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

(vas creando la imagen del proceso en disco) (podes hacerlo sin memoria ram) despues la subis a la memoria



55. Un proceso puede pasar de esperar en memoria secundaria a esperar en memoria principal.
FALSO - NI

podria porque la transicion existe pero no tiene sentido

si el so no tiene nada q procesar podriiiiiiiiiiia ser

56. El scheduler de medium term maneja el grado de multiprogramación.
VERDADERO PORQUE INFLUYE

el que maneja es el long term
el medium lo controla a través de la tecnica de swapear procesos (asi reduce), logra que temporalmente se reduzca

57. El disco permitió implementar la planificación de procesos.
si
disco como medio de almacenamiento si
verdadero

58. En un sistema monoprocesador, cuando se atiende una interrupción (se ejecuta una rutina de manejo de interrupciones) todos los procesos quedan en espera.

si
verdadero, ya que tiene que ejecutar el codigo de la interrupcion, mientras tanto los procesos estan en estado de ready o alguno en running

59. En un ambiente con procesos interactivos y batch, que maneja colas multinivel. ¿Conviene usar algoritmos apropiativos?
VERDADERO
si conviene pero te puede llevar a INANICION (con cuidado de esto conviene)

60. Indique cuál es la combinación que representa la sucesión de actividades que realiza el dispatcher:
a) Cambio de contexto; c) Salto a primer/próxima instrucción a ejecutar; (carga en el pc la direccion que debería cargar, el valor del proceso que va a entrar)
b) Cambio de Modo; d) Carga en memoria del proceso elegido

cambio de modo (dispatcher es kernel)
cargamos en memoria el proceso elegido
cambio de contexto
salto a la prox instruccion
b
d

a
c

61. Indique que puede ocurrir cuando solamente se tienen muchos procesos orientados a I/O:
a) Se incrementa el uso de CPU;
b) Se saturan las colas de dispositivo;
b.

62. Cuando se carga un proceso en memoria, se hace en modo usuario.
f
falso ya que el proceso en memoria lo carga el so

63. La dirección que se carga en el PC es una dirección física
f
hablamos de direcciones logicas (por eso se traducen)

64. En las particiones dinámicas siempre es mejor la opción worst fit para la asignación de particiones.
worst fit – VERDADERO
en cambio…
best fit (hueco q mejor calza), hueco muy chico

de la dinamica es la worst fit
fija – next

65. ¿Quién resuelve una dirección en la que interviene el contenido del registro de reubicación y una dirección lógica?
MMU

66. ¿Cual es la ventaja de la paginación pura con respecto a cargar todo el proceso en memoria de forma contigua? Analizar ventajas y desventajas
todo paginado o todo continuo

ventajas:

- no tenes q encontrar espacio contiguo
- es mas probable que tengas hueco libre
- no tenes fragmentacion externa
- es mas sencillo

desventajas:

fragmentacion

- Ventajas de la paginación pura:
  - No hay fragmentación externa.
  - Mejor aprovechamiento del espacio de memoria.
  - Flexibilidad al no requerir que todo el proceso esté en memoria para ejecutarse.
- Desventajas:
  - Puede haber fragmentación interna.
  - Complejidad en la gestión de la memoria.
  - Necesidad de hardware adicional para la traducción de direcciones

67. En paginación los procesos utilizan direcciones lógicas que son necesarias traducir a direcciones físicas.

V

68. Qué información es necesario guardar en el entrada de la tabla de páginas en la paginación pura

Marco y pagina
nro de marco que esta cargado en la pagina
bit de modificacion podriiiiiiiiia
bit de ...

NOTAS DE CLASE

direccion ilegal → con la direccion que se tiene nos e puede accedera esa direccion, segmento que no es valido

MULTITAREA ES LO mismo que multiprogramacion

—----------particionada de memoria hay fragmentacion interna

system call→ los procesos piden servicios al kernel

la direc de retorno es apilada en el stack de modo usuario, con call
lo hace el hardware

libreria glibc en modo usuario la que te crea el proceso
kernel tiene el manejador de interrupciones, nro tiene el syscall handler
2 pilas, modo usuario y modo kernel

proceso nace y muere en memoria principal
lo que swapea es el contenido del proceso

PCB
mantiene los punteros a esas estructuras
cambio de estado no implica cambio de tabla
la pcb tiene puntero a la tabla de paginas
no se guarda la tabla de archivos abiertos, sino q tiene puntero
el cambio de contexto no implica que cambien las tablas

execv no se puede ejecutar si no hubo fork antes

CONTEXT SWITCH
pid no cambia
espacio de direcciones si
pc cambia

particiones fijas no hya grado de multiprogramación variable (misma cant de marcos)