SISTEMAS OPERATIVOS Grupo B - 14-dic-2016 – TEORIA PRUEBA TEMA 2

APELLIDOS:_____

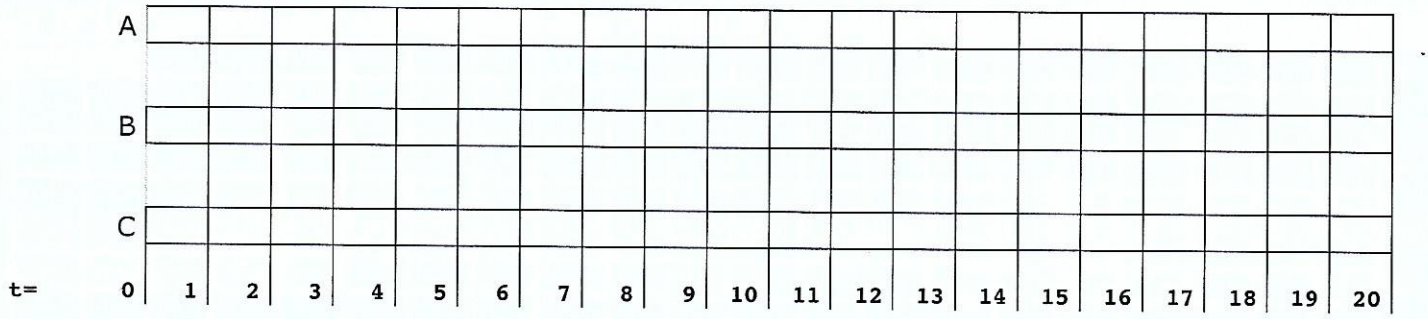
NOMBRE: _____ email (en mayúsculas): _____

NO ESTA PERMITIDO UTILIZAR MOVILES NI NIGÚN DISPOSITIVO DE COMUNICACION.
TODAS LAS RESPUESTAS DEBERAN ESTAR JUSTIFICADAS

1) [5*]Realide el diagrama de asignación de CPU para el ALGORITMO DE ASIGNACION DE CPU DE "ROUND ROBIN" O "BARRIDO CÍCLICO" con quantum = 2 milisegundos, sin desplazamiento (sin derecho preferente). En las figuras siguientes todas las unidades están dadas en milisegundos

	Creación		Cpu	Bloqueo	Cpu
A	3		1		
B	2		8		
C	0		3	2	1

Construya el diagrama de ocupación de la CPU:



Represente las colas de ejecutables tras cada momento en que haya actuado el planificador:
(represente por orden de antigüedad, el más antiguo más a la izquierda)

t= 0	t= 1	t= 2	t= 3	t= 4	t= 5	t= 6
t= 7	**t= 8**	**t= 9**	**t= 10**	**t= 11**	**t= 12**	**t= 13**
t= 14	**t= 15**	**t= 16**	**t= 17**	**t= 18**	**t= 19**	**t= 20**

2) [2*] Indique qué posibles eventos pueden provocar que un proceso deje de estar en estado ejecutable; indique si influye el tipo de algoritmo de asignación de CPU:

1.

2.

3.

4.

5.

3) [2*] Algoritmo Round Robin: para valores de Q muy grandes en relación al tamaño de las ráfagas de CPU, este algoritmo se comporta como ¿qué otro de los algoritmos estudiados?

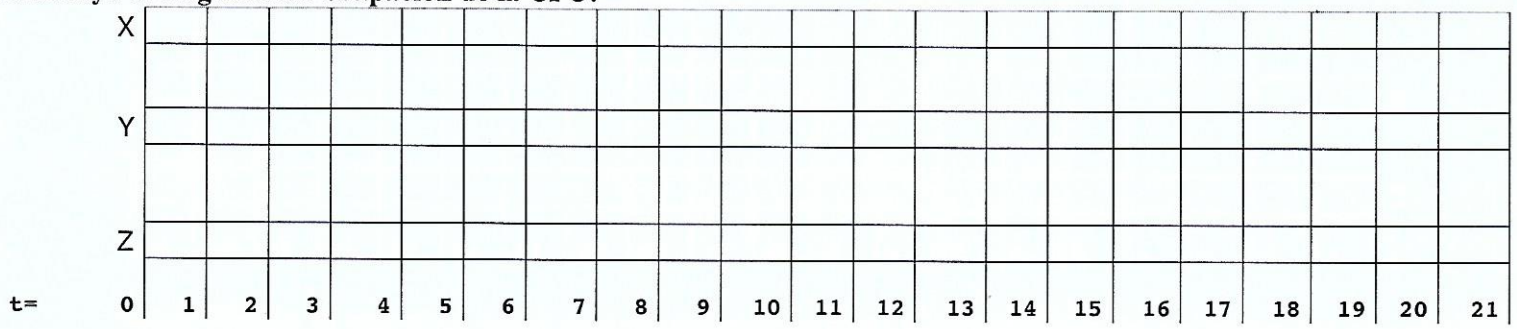
4) [10*] Realide el diagrama de asignación de CPU para el ALGORITMO DE ASIGNACION DE CPU DE "COLAS MULTIPLES CON TRASPASO" que se concreta así:

a) hay tres colas Round Robin o Barrido Cíclico: cola 1, cola 2 y cola 3 (1 es la más prioritaria o de más peso, 3 es la menos prioritaria)
b) los valores de los quantum son 2, 4 y 6 milisegundos respectivamente para cola 1, 2 y 3.
c) al crearse los procesos entran en la cola 1.
d) un proceso pasa de la cola 1 a la cola 2 cuando ha consumido 1 quantum completo en la cola 1
e) un proceso pasa de la cola 2 a la cola 3 cuando ha consumido 1 quantum completo en la cola 2.
f) cuando los procesos pasan a la cola 3 allí permanecen hasta que terminan.
g) cuando un proceso se desbloquea vuelve a la cola en que estaba antes de bloquearse.
h) no hay desplazamiento (sin derecho preferente)

En las figuras siguientes todas las unidades están dadas en milisegundos.

	Creación	Cpu	Bloqueo	Cpu
X	2	4		
Y	3	1	2	1
Z	0	7	1	3

Construya el diagrama de ocupación de la CPU:



Represente las colas de ejecutables tras cada momento en que haya actuado el planificador:

	t= 0	t= 1	t= 2	t= 3	t= 4	t= 5	t= 6
Cola 1							
Cola 2							
Cola 3							

	t= 7	t= 8	t= 9	t= 10	t= 11	t= 12	t= 13
Cola 1							
Cola 2							
Cola 3							

	t= 14	t= 15	t= 16	t= 17	t= 18	t= 19	t= 20
Cola 1							
Cola 2							
Cola 3							

5) [2*] Situándonos en el algoritmo de planificación Round Robin con valor de quantum Q, ¿es posible que un proceso sea el que tiene asignada la CPU durante un tiempo mayor de Q?

3) [2*] Explique lo más detalladamente posible en cuáles de los algoritmos siguientes (Round Robin con quantum Q o FIFO), el tiempo de espera E es independiente del tiempo de cpu (llamémosle t) de la ráfaga:

6) Uno de los problemas que pueden producirse al diseñar un algoritmo de asignación de CPU es que un proceso acapare la CPU indefinidamente debido a que (por error o mala intención) tenga un bucle infinito en el que nunca se bloquea.
(a) [2*] ¿el algoritmo Round Robin evita este problema?

(b) [2*]¿el algoritmo "El más corto primero" evita este problema?

(c) [4*] En general, ¿qué característica debe cumplir un algoritmo de asignación de CPU para que se evite dicho problema?

7) Uno de los problemas que pueden producirse al diseñar un algoritmo de asignación de CPU es que un proceso quede sistemáticamente postergado durante largos periodos de tiempo debido a que se atiende antes a otros procesos que llegan.
(a) [2*]¿se evita este problema en el algoritmo de asignación de CPU Round Robin?

(b) [2*]¿se evita este problema en el algoritmo "El más corto primero"?

(c) [2*]¿se evita este problema en el algoritmo "Colas múltiples con traspaso" tal como se ha ejercitado en clase?

(d) [4*]¿cómo podría un algoritmo de asignación de CPU evitar ese problema?

8) [4*] Explique lo más detalladamente posible en cuáles de los algoritmos siguientes (Round Robin con quantum Q y FIFO), el tiempo de espera E es independiente del tiempo de CPU (llamémosle t) de la ráfaga:

9) Situándonos en el algoritmo de planificación Round Robin con valor de quantum Q,
(a)[1*]¿es posible que un proceso sea el que tiene asignada la CPU durante un tiempo mayor de Q?

(b)[1*]¿es posible que un proceso sea el que tiene asignada la CPU durante un tiempo menor de Q?

10) [2*] Explicar porqué es un buen criterio del planificador a largo plazo asegurar en la medida de lo posible que haya una mezcla de procesos con ráfagas de CPU cortas y procesos con ráfagas de CPU largas.

LAS SIGUIENTES PREGUNTAS SE SITUAN EN LA MATERIA SOBRE IMPLEMENTACIÓN EN LINUX

11) [2*] ¿donde se almacena el valor que un proceso pone como argumento en su llamada a `exit`?

12) [2*] Cuando un proceso realiza la llamada al sistema `exit` ¿puede que aún tenga hijos?

13) [4*] Consideremos que el proceso A hizo un `fork` en el pasado como resultado del cual se creó un proceso hijo que llamaremos B. ¿Puede darse el caso de que A y B tengan una página de memoria compartida?

14) [2*] Tenemos tres procesos cada uno perteneciente a una de las tres clases diferentes de planificación, explique en qué orden se ejecutarán los procesos.

15) [4*] En la clase de planificación CFS (neutra o limpia, Completely Fair Scheduling), el tiempo virtual vruntime que el kernel calcula para un proceso es ¿menor o mayor al aumentar el valor numérico de prioridad del proceso? ¿porqué?

16) [2*] ¿Mientras un proceso está ejecutando código de usuario podría tener su valor `preempt_count` asociado un valor >0?

17) [2*] Dentro de las funciones definidas en un programa de usuario, ¿cada vez que se retorna de una función (en el espacio de usuario) se chequea el flag TIF_NEED_RESCHED?

18) [2*] Explique qué repercusión tendría si un proceso pudiera asignar al elemento `preempt_count` asociado a su `task_struct` un valor >0.

19) [2*] ¿En qué estado está un proceso mientras ejecuta código kernel correspondiente a una llamada al sistema?

20) [4*] Nos situamos en el retorno a modo kernel en que, tras el tratamiento de una interrupción, se vuelve a ejecutar código en modo privilegiado, explique la actuación del kernel si el flag TIF_NEED_RESCHED está activo.

21) [4*] Eventos que pueden provocar la invocación del planificación principal (función `schedule`):

1.

2.

3.

4.

22) [2*] ¿Cuando se dice que el kernel "`vuelve a ser expropiativo`"?