COGNOMS: **FILA:**

NOM: **COLUMNA:**

Examen final SO. 12 de gener de 2006. Duració: 3 hores

Les notes sortiran el divendres 27 de gener a migdia al racó.
La revisió es farà entre el 30 i el 31 de gener. El lloc i hora es publicarà al racó amb les notes.
El problema 1 s'ha de lliurar en un full a part; els problemes 2, 3 i 4 s'han de lliurar en els mateixos fulls de l'enunciat.

Pregunta 1 (3 punts)

La siguiente tabla muestra los estados de los procesos en un sistema operativo derivado de UNIX:

Estado	Descripción
Ready	El proceso está en memoria listo para ejecutarse
Ready-Swapped	El proceso está listo para ejecutarse pero está en disco
Kernel Running	El proceso se está ejecutando pero en modo privilegiado. Esto sucede al: ejecutar una llamada al sistema, invocar el planificador o provocar un fallo de página
User Running	El proceso se está ejecutando
Blocked	El proceso está bloqueado
Blocked-Swapped	El proceso está bloqueado pero en disco
Page-pending	El proceso está bloqueado esperando la resolución de un fallo de página que ha provocado
Zombie	El proceso ha terminado y está esperando que se lea su estado de finalización

Un proceso no se puede ejecutar si no está en memoria. La política de planificación es apropiativa diferida y el algoritmo de planificación es Round-Robin.

a) Dibuja el grafo de estados de este sistema.

b) Numera cada una de las transiciones del grafo anterior. Explica, a continuación, qué o quién provoca cada una de las transiciones.

c) Dibuja un grafo de estados basado en el del apartado a), añadiendo lo necesario (estados y/o transiciones) para implementar SIGSTOP (signal de tratamiento inmediato que hace que un proceso se pare hasta que reciba SIGCONT). Explica las diferencias respecto al grafo del apartado a).

d) Dibuja un nuevo grafo con las especificaciones del apartado a) pero con una política de planificación apropiativa inmediata y algoritmo de planificación Round-Robin con prioridades. Explica las diferencias respecto al grafo del apartado a).

Pregunta 2 (1.5 punts)

Tenim un sistema de fitxers tipus UNIX. Sabent que:
/A/B/c és un soft-link cap a /A/D/e
/A/D/e és un soft-link cap a /A/B/f
/A/B/f és un fitxer de dades

Suposant que:
- Superbloc sempre està carregat a memòria
- Taula d'i_nodes està permanentment carregada a memòria
- Un directori necessita un sol bloc de dades
- Tenim una buffer cache de 4 entrades per a guardar blocs de dades
- Els buffers de la buffer cache es gestionen usant un algorisme LRU (es fa fora el que fa més temps que no ha estat utilitzat)

a) Especifica la sequència d'accessos a i_node i bloc de dades necessària per a fer un

c= open("/A/B/c", O_RDONLY)

Notació:
Usa I(nom) per a indicar l'accés a l'i_node corresponent a "nom"
Usa BD(nom) per a indicar l'accés al bloc de dades corresponent a "nom"

Número total d'accessos a i_nodes =
Número total d'accessos a blocs de dades =

b) Indica quins dels accessos a blocs de dades implicaran un accés a disc.

Notació:
Usa BD(nom) per a indicar l'acces al bloc de dades corresponent a "nom"

Accessos que impliquen acces a disc	**Accessos que reaprofiten blocs a la buffer cache**

COGNOMS: **FILA:**

NOM: **COLUMNA:**

Pregunta 3 (2.5 punts)

Contesta las siguientes preguntas de forma concisa.

a) ¿Qué diferencia hay entre un trap y una interrupción?

b) Cita dos mecanismos utilizados para pasar parámetros a un trap

c) ¿Para qué sirve el *spooling*?

d) ¿Por qué tenemos rutinas dependientes e independientes para acceder a los dispositivos?

e) ¿Qué diferencia hay entre la asignación indexada y la indexada multinivel?

f) Cita tres mecanismos para la gestión del espacio libre en un sistema de ficheros

g) ¿Implementa round-robin una política apropiativa? Justifica la respuesta.

h) ¿Cómo podemos sincronizar dos procesos usando *pipes* pero sin enviar ningún byte?

i) ¿Qué diferencia hay a nivel de recursos utilizados entre dos procesos o un proceso con dos flujos?

j) ¿Puede producirse un fallo de página de un acceso correcto? Justifica la respuesta.

COGNOMS: **FILA:**

NOM: **COLUMNA:**

Pregunta 4 (3 punts)

Vull fer un programa tal que, si s'invoca amb la següent comanda:
_$ myprog exe1 exe2 file1 file2
Fa l'equivalent a “exe1 file1 | exe2 > file2”.
S'han de tenir en compte les següents restriccions:

- El programa exe1 modifica file1, per tant s'ha de garantir que si hi hagués qualsevol problema creant els fills o la pipe, exe1 no hauria d'executar-se. Comenta breument (no més de 25 paraules) quina solució proposes.

- Si en 5 segons exe1 no ha acabat, els dos processos han de morir i no s'ha de generar file2 (no importa l'estat en que queda file1). Comenta breument com penses implementar aquesta restricció.

- Si el programa exe2 acaba amb exit(0), s'ha d'esborrar el fitxer file1. Quin procés esborrarà file1?

Per últim, escriu el codi complet de la teva solució (a aquest mateix full, al darrera).