APELLIDOS:	**DNI:**
NOMBRE:	
FILA:	**COLUMNA:**

Sistemas Operativos – Facultat d´Informàtica de Barcelona - UPC

Fecha: 11 de Junio de 2007 **Duración: 3 horas**

Notas: Las notas se publicarán el **Miércoles 20 de Junio** en el RACÓ a las 18:00 horas.

Revisión: La fecha, hora y lugar exactos de la revisión del examen se publicarán junto con las notas. Estad atentos.

ATENCIÓN: Las preguntas se tienen que contestar en las mismas hojas de examen utilizando el espacio reservado a tal efecto. Asegúrate de poner NOMBRE y APELLIDOS, DNI, fila y columna en cada una de las hojas. El examen se tiene que entregar en bolígrafo negro o azul. Se pueden utilizar el documento con las llamadas al sistema.

Exercici 1: (3 punts)

Respon **breument** a les següents preguntes:

a) El procés que està en RUNNING executa una instrucció “v[i]=2;” i passa a BLOCKED. Què pot haver provocat aquest canvi d’estat?

b) El procés que està en RUNNING executa una instrucció “v[i]=2;” i passa a READY. Què pot haver provocat aquest canvi d’estat?

c) El procés que està en RUNNING executa una instrucció “v[i]=2;” i passa a ZOMBIE. Què pot haver provocat aquest canvi d’estat?

d) Si hi ha un fallo al TLB, implica que haig d’anar a buscar una pàgina a l’àrea de swap?

e) Enumera els principals problemes del sistema FAT.

f) Un procés executa una instrucció *sem_wait(S1)*, a quin/s estat/s pot passar?

g) Dos threads poden intercanviar-se informació amb una named_pipe?

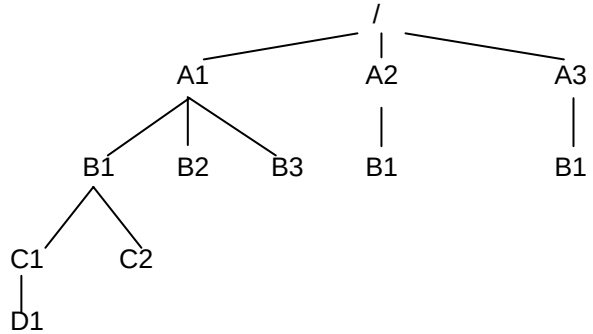
h) Descriu els camps de la Taula de Fitxers Oberts, indicant perquè serveixen cadascun d'ells.

i) Perquè serveix un gestor d'entrada/sortida ?

j) Què aporta la tècnica "aging" a un planificador a curt termini amb prioritats?

Exercici 2: (2 punts)

Dado el siguiente árbol de directorios:



En el cual:
/A1/B1/C2 es un soft-link a /A1/B3
/A2/B1 es un hard-link a /A3/B1
/A1/B1/C1/D1, /A1/B2 y /A1/B3 son ficheros de datos de 1 bloque
/A3/B1 es un fichero de datos de 2 bloques

APELLIDOS:	**DNI:**
NOMBRE:	
FILA:	**COLUMNA:**

a) Dibuja los i-nodos (según la figura) y los bloques de datos que creas convenientes, en un sistema de ficheros con índices multinivel. Supón que el Inodo de la raíz del sistema de ficheros es el 2 y el resto de Inodos está libre. Los datos de todos los ficheros están situados a partir del bloque 1. Los Inodos contienen: tipo de fichero, índices a los bloques de datos y número de referencias.

b) Haz lo mismo que en el apartado a) suponiendo que ahora nuestro sistema de ficheros utiliza una FAT. ¿Existe algún problema? En el caso de que exista, coméntalo brevemente así como su solución.

Exercici 3 (2 Punts)

Volem fer un codi que faci el següent:
_$ processar fitxer lletra

El programa crearà dos fills (anomenem-los A i B) i cadascun crearà dos threads (anomenem-los 1 i 2).

- Els dos threads de A i B llegiran el fitxer caràcter a caràcter. Cada cop que algun dels dos threads trobi el caràcter "*lletra*" incrementarà un comptador propi de cada procés.
- Cada thread te un comptador local que porta el numero de caràcters llegits per cada thread. Per tant, aquest comptador s'incrementarà per cada caràcter llegit.
- Quan el thread arribi al final del fitxer, mostrarà per pantalla el numero de caràcters llegits.
- Un cop acabats de executar els threads, el procés mostrarà per pantalla el numero de caràcters "*lletra*" trobats pels seus threads.
- El procés pare comprovarà si el numero de caràcters "*lletra*" trobats per A es igual al de B.

```
 1: int fd, cont_lletra=0;
 2: char lletra;
 3:
 4: void *lector(void *arg)
 5: {
 6:    sem_t Sllegir, Slletra;
 7:    char c;
 8:    int cont_caracters = 0;
 9:
10:    while (read(fd, c, 1)>0){
11:        sem_wait(&Sllegir);
12:        cont_caracters++;
13:        if (c==lletra){
14:            sem_post(&Slletra);
15:            cont_lletra++;
16:            sem_wait(&Slletra);
17:        }
18:        sem_post(&Sllegir);
19:    }
20:    printf("Thread %d ha llegit %d caràcters\n", getpid (), cont caracters);
21:    pthread_exit(0);
22: }
23:
24: int main(int argc, char **argv)
25: {
26:    pthread_t threads[2];
27:    int aux[2], PID[2], status;
28:
29:    lletra=argv[2][0];
30:    fd=open(argv[1], O_RDONLY);
31:    for (i=0; i<2; i++){
32:        if((PID[i]=fork())==0){
33:            sem_init(&Slletra, 0, 0);
34:            sem_init(&Sllegir, 0, 1);
35:            pthread_create(&threads[0], NULL, lector, NULL);
36:            pthread_create(&threads[1], NULL, lector, NULL);
37:            aux[i]=cont_lletra;
38:            printf("Procés(%d) ha trobat %d aparicions de %s\n", i, aux[i], lletra);
39:            exit(0);
40:        }
41:    }
42:    pthread_join(threads[0],&status);
43:    pthread_join(threads[1],&status);
44:    close(fd);
45:    waitpid(PID[1], &status, 0);
46:    if (aux[0] != aux[1]) printf("Processat incorrecte!!!\n");
47:    else printf("Processat OK\n");
48:    wait(&status);
49:    return 0;
50: }
```

APELLIDOS:	**DNI:**
NOMBRE:	
FILA:	**COLUMNA:**

Exemple de execució: *_$ processar pepe a*

```
Thread A1 ha llegit 7 caràcters
Thread A2 ha llegit 3 caràcters
Procés(A) ha trobat 5 aparicions de a
Thread B1 ha llegit 4 caràcters
Thread B2 ha llegit 6 caràcters
Procés(B) ha trobat 5 aparicions de a
Processat OK
```

Contingut fitxer "**pepe**":
abadafahaj

a) Aquest codi conté 10 errors. Enumera'ls tot indicant per cada un la línia on es troba, quin es l'error i com el solucionaries.

Error 1:

Error 2:

Error 3:

Error 4:

Error 5:

Error 6:

Error 7:

Error 8:

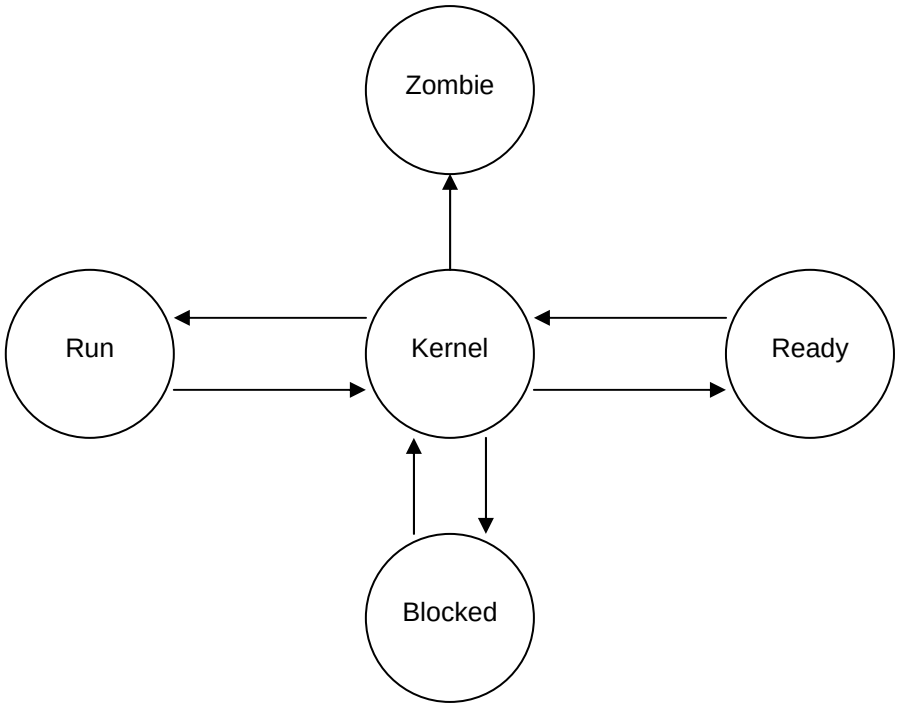
Error 9:

Error 10:

b) Aconseguiríem alguna millora si comuniquem els processos A i B amb el pare a traves de una pipe? Raona breument la teva resposta.

Ejercicio 4 (3 puntos):

Tenemos un sistema operativo en el que los procesos tienen el siguiente ciclo de vida:



Los estados Ready, Run, Blocked y Zombie funcionan exactamente igual que los vistos en clase. Un proceso está en Kernel cuando ejecuta código del sistema.

APELLIDOS:	**DNI:**
NOMBRE:	
FILA:	**COLUMNA:**

Contesta razonadamente a las siguientes preguntas sobre este ciclo de vida.

a) Enumera las acciones que fuerzan a un proceso a pasar de Run a Kernel.

b) Razona en qué estado se realizan las siguientes acciones:

-Ejecución de código de usuario:

-Encolar petición a un gestor de una entrada/salida:

-Preparar los parámetros de una llamada al sistema:

-Evaluación de los signals de baja prioridad que estén pendientes:

c) ¿Qué tipos de políticas y algoritmos de planificación se pueden aplicar al anterior ciclo de vida sin modificarlo?

d) Suponiendo que nuestro ordenador solamente tienen un procesador, ¿en qué estado un proceso puede recibir signals? ¿Y si tenemos más de un procesador en nuestro ordenador?