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

Sistemas Operativos – Facultat d´Informàtica de Barcelona - UPC

Fecha: 17 de Enero de 2008 **Duración: 3 horas**

Notas: Las notas se publicarán el **Miércoles 23 de Enero** 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.

Ejercicio 1: (3 puntos)

Sea una arquitectura con un solo procesador. Tenemos un SO tipo UNIX, con un planificador a corto plazo que implementa Round Robin con prioridades.

En este SO tenemos ejecutándose un proceso P1 con un solo thread. Contesta a las siguientes preguntas, suponiendo que el proceso P1 está en RUNNING. Todo su código cabe en una página de memoria, que ya está cargada. Los datos ocupan más de 40 páginas.

1. ¿Puede P1 ejecutar una instrucción **read(a,&c,1);** y seguir en RUNNING? Si la respuesta es afirmativa, pon un ejemplo.

2. Ejecuto la instrucción **read(a,&c,1);**, el valor de la variable **a** es **1000**. ¿El proceso pasa a ZOMBIE?. Razona tu respuesta.

3. Ejecuto la instrucción **read(a,&c,1);**, habiendo ejecutado previamente, y con éxito, la llamada **a=open("f",O_RDONLY);** siendo **f** un fichero. Observo que el proceso pasa a BLOCKED. ¿Significa eso que estoy haciendo una entrada/salida sin gestor? Razona tu respuesta.

4. Ejecuto la instrucción **write(a,&c,1);**, habiendo ejecutado previamente, y con éxito, la instrucción **a=open("p",O_WRONLY);**, siendo **p** una *named pipe*. Indica bajo qué circunstancias puedo pasar a:

- BLOCKED

- ZOMBIE

5. Ejecuto la instrucción **z[i]=v[i]+w[i];** ¿puedo pasar al estado BLOCKED? Razona tu respuesta.

6. Ejecuto la instrucción **z[i]=v[i]+w[i];** ¿puedo pasar al estado ZOMBIE? Razona tu respuesta.

7. Estoy ejecutando una instrucción cualquiera. ¿En qué circunstancias una interrupción de reloj provocará que el proceso P1 …
… pase a READY?

… pase a BLOCKED?

… pase a ZOMBIE?

8. ¿Puede P1 recibir un SIGUSR1 de otro proceso? Si es así, ¿lo atiende al instante o espera a salir y volver a entrar en RUNNING?

Supongamos ahora que P1 tiene 3 threads (T1, T2 y T3). Contesta a las siguientes preguntas, suponiendo que el thread en ejecución es T1.

9. T1 ejecuta un **write(a,&c,1);** que provoca una operación de entrada / salida ¿El proceso pasa a READY? Razona tu respuesta.

10. T1 ejecuta un **sem_wait(&S1);** ¿el thread se bloquea? Razona tu respuesta.

11. Mientras se ejecuta una instrucción cualquiera de T1 se recibe un SIGSTOP. ¿Se bloquea T1 o todos los threads? Razona tu respuesta.

12. Si T1 ejecuta una instrucción **fork();**, ¿el proceso hijo tiene un único thread T1,o tiene 3 threads? Razona tu respuesta.

13. T1 ejecuta las llamadas **alarm(5); pause();** ¿Qué threads se bloquean? ¿Qué thread atiende el SIGALRM?

14. Supongamos un fichero **f** que no ha sido abierto desde el boot del sistema. T1 ejecuta un **a=open("f",O_RDONLY);** ¿Cuántas entradas nuevas hay en la tabla de canales, la tabla de ficheros abiertos y la tabla de inodos? Razona tu respuesta.

15. Supongamos un fichero **f** que no ha sido abierto desde el boot del sistema. T2 ejecuta un **a=open("f",O_RDONLY);** y T3 ejecuta un **b=open("f",O_RDONLY);** ¿Cuántas entradas nuevas hay en la tabla de canales, la tabla de ficheros abiertos y la tabla de inodos? Razona tu respuesta.

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

Ejercicio 2 (3 puntos)

Tenemos un sistema de ficheros en el cual solamente existe el directorio raíz. Desde este directorio, ejecutamos la siguiente lista de comandos:

```
> mkdir A
> mkdir B
> cd B
> echo "Hello " > fic1
> echo "world" > fic2
> cat fic1 fic2 > /A/fic3
> link /A/fic3 fic3
```

a) Dibuja el grafo de directorios después de ejecutar estos comandos. Supón que todo se ha ejecutado correctamente.

b) Suponiendo que el sistema de ficheros es como el de Unix, dibuja a nivel de Inodos y bloques de datos el grafo de directorios resultante. El único Inodo que está ocupado antes de empezar a ejecutar estos comandos es el del directorio raíz, cuyo número es el 2 y que tiene asociado el bloque de datos 1. El resto de Inodos y bloques están libres y se asignan consecutivamente. La información que contiene un Inodo es: tipo de fichero, bloques de datos asociados y número de referencias.

c) Enumera los accesos que realiza el comando *cat* a disco durante su ejecución. Explica brevemente el motivo de por qué realiza los accesos.

d) Escribe un programa llamado *inverso.c* que, dado el nombre de un fichero como parámetro, cree un segundo fichero, cuyo nombre también se ha pasado como parámetro, cuyo contenido sea el inverso del contenido del primer fichero. En ningún momento se puede hacer suposiciones del tamaño máximo del fichero. *Inverso.c* tiene que funcionar de forma eficiente: tiene que realizar el mínimo número de llamadas al sistema como sea posible. No es necesario que incluyas el código de comprobación de errores.

e) Lista las llamadas al sistema, poniendo el valor de sus parámetros y su valor de retorno, que ejecuta tu implementación de inverso.c al ser ejecutado de la siguiente forma:
> inverso /B/fic2 /A/2fic

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

Ejercicio 3 (1 punto)

En un ordenador tenemos tres threads ejecutándose de forma paralela y cuyas regiones críticas están indicadas como Operaciones1, Operaciones2, Operaciones3, Operaciones4 y Operaciones5.

Añade al codígo de los threads siguientes, la gestión de semáforos que creas conveniente para que el orden de ejecución sea: Operaciones1, Operaciones2, Operaciones3, Operaciones4 y Operaciones5.
Las dos Operaciones3 de los dos threads se tienen que ejecutar antes que Operaciones4 del Thread 2. Aunque, en este caso, no importa en qué orden se ejecuten Operaciones3.
Indica aparte la inicialización de los semáforos que utilices.

INICIALIZACIONES:

Thread1	Thread2	Thread3
Operaciones1	Operaciones3	Operaciones3
Operaciones2	Operaciones4	Operaciones5

Ejercicio 4 (1 punto)

¿Qué valores tendrán x e y después de la ejecución de los siguientes tres threads que se ejecutan concurrentemente y comparten las variables? Razona tu respuesta indicando en qué orden se ejecutan las instrucciones de cada thread teniendo en cuenta que no se conoce el orden de creación de los threads y no sabemos que planificador a corto plazo está implementado en el sistema operativo.

Inicializaciones:
x=1;
y=4;
sem_init (&S1, 1);
sem_init (&S2, 0);
sem_init (&S3, 1);

Thread A	Thread B	Thread C

sem_wait(S2);
sem_wait(S3);
x = y * 2;
y = y + 1;
sem_signal(S3);
	sem_wait(S1);
	sem_wait(S3);
	x = x + 1;
	y = 8 + x;
	sem_signal(S2);
	sem_signal(S3);
		sem_wait(S1);
		sem_wait(S3);
		x = y + 2;
		y = x * 4;
		sem_signal(S3);
		sem_signal(S1);

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

Ejercicio 5 (2 puntos)

Estamos diseñando el ciclo de vida de un proceso dentro de un sistema operativo. Sabemos que este sistema operativo ofrece signals. Además, como será un SO sencillo, no dispondrá de gestores. Por último, sería interesante poder distinguir cuando un proceso está ejecutando código de usuario (user running) y cuando de sistema (kernel running).

Nuestro planificador es no apropiativo.

a) Dibuja el ciclo de vida del proceso junto con las transiciones de cambio de estado. Toma las decisiones que creas convenientes, justificándolas, para realizar este ciclo de vida. Además, numera las transiciones e indica qué las causan.

b) Enumera las causas que hacen que un proceso pase de user running a kernel running y posteriormente, otra vez, a user running.

c) Enumera las llamadas al sistema que obligan a un proceso a pasar de kernel running a blocked.

d) ¿Cómo cambia el ciclo de vida si el planificador es apropiativo diferido?

e) Al ejecutar una instrucción como:

```
if ((e=read(fd, &c, 1))>0)
```

¿en qué punto preciso pasa el proceso de user running a kernel running? Describe todos los pasos que se tienen que hacer para este cambio de estado.