Ingeniería Informátiuca. Curso 3º.
Sistemas Operativos
Examen Final. PROBLEMAS.
7 de Septiembre de 2007

1. Dado el siguiente código

```
1:    int tubo[2];

2:    void* nuevo_hilo(int num) {
3:          int n=num;

4:          printf("Escribo %d \n",n);
5:          write(tubo[1],&n,4);
6:    }

7:    int main () {
8:          int pid;
9:          pthread_t tid;

10:         pipe(tubo);
11:         pid = fork();
12:         if (pid == 0) {
13:               int n;
14:               close(tubo[0]);
15:               for (n = 0; n < M; n++) {
16:                     if ((n%2) == 0)
17:                           pthread_create(&tid, NULL, nuevo_hilo,
n);
18:                     else
19:                           pthread_join(tid,NULL);
20:               }
21:               close(tubo[1]);
22:         }
23:         else {
24:               int n;
25:               close(tubo[1]);
26:               while (read(tubo[0],&n,4) != 0) {
27:                     printf("Leo %d \n", n);
28:               }
29:               close(tubo[0]);
30:         }
31:   }
```

a) (1.5 puntos) Especificar razonadamente el resultado de la ejecución del código para M=10.
b) (1 punto) ¿Qué ocurre si se intercambian las líneas 10 y 11 del código? (esto es, si ejecutamos **fork** antes de la llamada a **pipe**). Describir detalladamente la ejecución de cada proceso/hilo que se ejecute y sus condiciones de terminación.
c) (1 punto) Reescribir el código anterior usando semáforos generales y memoria compartida en lugar de tuberías para la sincronización y comunicación.
d) (1.5 puntos) Implementar las operaciones ***wait*** y ***post***, propias de semáforos generales, haciendo uso de tuberías.

2. Considerar el código de los siguientes procesos:

```
int fda;
Proceso A() {
  char buf[2048];
  char buf_w[2048]="cccc…";
        fda = open("tmp/faa");
//código (1.5ms)
1       read(fda,buf,200);
        //código (0.5 ms)
create_thread(,Hilo_Nuevo)
2       read(fda,buf,1600);
        //código (0.75 ms)
        lseek(fda,1600,SEEK_CUR);
3       read(fda,buf,600);
//código (0.5)
}

Hilo_Nuevo() {
        //código (0.75 ms)
4     read(fda,buf,1600);
        //código (1.25 ms)
5      write(fda,bu_wf,3000);
}
```

```
Proceso B() {

        char buf[2048];
        int  fdb;

        //código (0.5 ms)
        fdb = open("tmp/fbb");
6       read(fdb,buf,500);
        // código (1.25 ms)
        lseek(fdb,3500,SEEK_CUR);
7       read(fdb,buf,500);
        // código (0.5ms)
8       read(fdb,buf,1000);
    }
```

Se ha elegido **FAT** como sistemas de ficheros de la partición de trabajo. El tamaño de bloque (físico y lógico) de 1KB (1024 bytes). A continuación se muestra la información relevante del estado inicial del sistema de ficheros:

Estado inicial de la FAT: 1 3 eof 2 5 6 eof free .....

Contenido del fichero de directorio **tmp**:

Nombre	Primer_bloque	Tamaño	……..
faa	0	7500	
fbb	4	5200	

El planificador del SO utiliza una política **Round Robin** con un ***quanto* de tiempo de 1ms**.
Las llamadas al sistema read/write son **bloqueantes** y **tardan 1ms por cada bloque de físico** accedido.
Cuando un proceso finaliza una operación de entrada/salida, pasa al final de la cola de *Listos*. El proceso A comienza su ejecución un instante antes que el proceso B.

La gestión de hilos se realiza a nivel de *kernel*. Cuando se crea un hilo, el nuevo hilo se coloca al final de la cola de *Listos*, y el hilo *creador* continúa con su ejecución. Considere que las llamadas *open ,lseek,*y *create_thread* tardan **0ms** en ejecutarse.

Los accesos a los dos ficheros distintos pueden hacerse simultáneamente, pero los accesos a un mismo fichero deben ser atendidos secuencialmente.

Se pide:

a) (1,5 punto) Determinar el estado final de la FAT y de la entrada de directorio "**tmp**"
b) (1,5 punto) Realizar un diagrama temporal con la planificación de los hilos/procesos, indicando el estado de cada hilo.
c) (0,5 punto) Calcular los tiempos de *retorno* y de *espera* de cada hilo/proceso.
d) (1.5 punto) Para cada llamada a "read" y "write", indicar los bloques lógicos y físicos accedidos.

Ingeniería Informática. Curso 3º.
Sistemas Operativos
Examen Final. TEORIA.
7 de Septiembre de 2007

1. **[PROCESOS]**
Considerar el siguiente código:

```
main() {
      for (int i=0; i < N; i++) {
            if (fork() == 0) {
                  exec("cd",…);
                  printf("Hijo %i\n",i);
            }
            else {
                  printf("Padre %i\n",i);
            }
      }
}
```

Mostrar la salida por pantalla de este código para N = 4. Asumir que un proceso hijo siempre es más prioritario que su padre. Justificar su respuesta.

2. **[SINCRONIZACION Y COMUNICACIÓN]**
¿Qué tres condiciones debe cumplir toda solución válida al problema de la sección crítica?. Justificar si la siguiente solución resuelve el problema de la sección crítica para **2** procesos:

```
bool flag[2]={false,false};
…
while (true) {
      …
      flag[i]=true;
      while (flag[j]) {
            flag[i]=false;
            while (flag[j]) { };
            flag[i] = true;
      }
      <Sección crítica>
      flag[i]=false;
      <Resto de código (NO sección crítica) >
}
```

donde `flag` es una variable compartida por los dos procesos,
`i` y `j` son constantes locales a cada proceso. Para el primer proceso, i=0, j=1. Para el segundo proceso i=1, j=0;

3. **[MEMORIA]**
a) ¿Cómo se llama la técnica utilizada para guardar en disco las imágenes de aquellos procesos que actualmente no están activos, liberando así espacio de memoria para otros procesos en ejecución?
b) ¿En cuál de estas operaciones puede o debe intervenir el kernel del sistema operativo?
- Cargar un fichero ejecutable en memoria
- Seleccionar una página para que sea sustituida
- Traducir una dirección virtual a una física
- Todas las anteriores

c) ¿Cuál de estas afirmaciones es cierta?
- El tamaño de un marco de página no tiene por qué coincidir con el tamaño de página lógica
- La paginación por demanda exige un hardware con bit de referencia
- La segmentación soluciona completamente el problema de fragmentación
- La TLB se puede usar con paginación y también con segmentación

d) ¿Por qué la memoria virtual no es adecuada para un sistema de tiempo real?

e) Señale qué elemento es responsable de cada una de estas acciones (SO: Sistema Operativo, o MMU: Unidad de Gestión de Memoria)

Rellenar la tabla de páginas
Activar los bits de validez
Activar los bits de referencia
Activar los bits de modificación
Generar la excepción "fallo de página"
Seleccionar las páginas a sustituir
Traducir una dirección virtual a física

4. **[ENTRADA/SALIDA]**

a) ¿Cómo se llama el módulo del sistema operativo que hace de interfaz entre el controlador hardware de un dispositivo de entrada/salida y los programas que hacen uso del dispositivo?

b) ¿Cómo se llama el subsistema que proporciona la interfaz a los programas de usuario para que estos puedan realizar transferencias de entrada/salida de manera uniforme?

c) ¿Cómo se llama el tiempo transcurrido desde que el brazo del disco está posicionado en el cilindro especificado por una petición de disco hasta que el sector especificado pasa por debajo de la cabeza de lectura/escritura?

d) ¿Cuál de estas afirmaciones es cierta y por qué? Un manejador de dispositivo

… accede al h/w indirectamente, ya que se ejecuta en modo usuario
… ofrece sus servicios en forma de llamadas al sistema
… se ajusta a un interfaz de uso impuesta por el diseñador del SO
… si es de caracteres, utiliza una cache para optimizar accesos a la E/S

e) Supongamos un sistema en el que las pistas de las peticiones de disco se distribuye uniformemente. ¿Qué política de planificación de disco es la más conveniente: C-SCAN, SCAN, SSTF, o FIFO?

5. [**SISTEMA DE FICHEROS**]

a) ¿Qué tipo de asignación presenta fragmentación externa: contigua, enlazada o indexada?

b) Supuesto un sistema de ficheros que utiliza asignación indexada y representación de espacio libre mediante mapa de bits, qué supondría el aumento del tamaño de bloque:

… disminución de la fragmentación interna?
… aumento de la fragmentación externa?
… disminución del tamaño máximo que puede alcanzar un fichero?
… disminución del tamaño del mapa de bits?

c) ¿Qué problema se plantea con la operación de eliminar un fichero en un sistema que soporta estructura de directorios de grafo acíclico? ¿Cómo se suele solucionar?

d) Dado el siguiente conjunto de operaciones sobre ficheros, ¿Cuáles se implementan como llamadas al sistema en LINUX y cuáles no?

Crear un fichero
Eliminar un fichero
Hacer una copia de un fichero a otro
Leer datos de un fichero
Escribir en un fichero
Listar el contenido de un directorio
Renombrar un fichero
Obtener el número de líneas de un fichero de texto
Cambiar los permisos de acceso de un fichero

e) Entre las principales estructuras de datos asociadas a la gestión de ficheros están las tablas TDA (descriptores de ficheros abiertos, TIIP (intermedia de i-nodos y posiciones) y TINM (i-nodos en memoria). ¿Cuántas hay de cada una de ellas? ¿Qué campos de información contiene la TIIP? ¿Si se prescindiese de la TIIP, ¿dónde colocarías su información, en tablas TDA o en tablas TIM, y por qué?