DISEÑO Y ADMINISTRACIÓN DE SISTEMAS OPERATIVOS **Febrero 2013**

Material permitido: **NINGUNO.** Tiempo: **2 horas.** N2	**Aviso 1:** Todas las respuestas deben estar razonadas. **Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.** **Aviso 3: No use Tipp-ex** o similares (atasca el escáner).

ESTE EXÁMEN CONSTA DE 5 PREGUNTAS

1. Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas.

i) *(1 p)* La ejecución de los procesos en un sistema UNIX está dividida en dos modos de ejecución: un modo de mayor privilegio denominado *modo núcleo o supervisor* y otro modo de menor privilegio denominado *modo usuario*.

ii) *(1 p)* Las interrupciones son atendidas en modo usuario dentro del contexto del proceso que se encuentra actualmente en ejecución, aunque dicha interrupción no tenga nada que ver con la ejecución de dicho proceso.

Solución:

i) VERDADERO (Páginas 12 y 13 del libro base).

Con el objetivo de poder implementar una protección eficiente del espacio de direcciones de memoria asociado al núcleo y de los espacios de direcciones asociados a cada proceso, la ejecución de los procesos en un sistema UNIX está dividida en dos modos de ejecución: un modo de mayor privilegio denominado modo núcleo o supervisor y otro modo de menor privilegio denominado modo usuario.

Un proceso ejecutándose en modo usuario sólo puede acceder a unas partes de su propio espacio de direcciones (código, datos y pila). Sin embargo, no puede acceder a otras partes de su propio espacio de direcciones, como aquellas reservadas para estructuras de datos asociadas al proceso usadas por el núcleo.

Tampoco puede acceder al espacio de direcciones de otros procesos o del mismo núcleo. De esta forma se evita una posible corrupción de los mismos.

Por otra parte, un proceso ejecutándose en modo núcleo puede acceder a su propio espacio de direcciones al completo y al espacio de direcciones del núcleo, pero no puede acceder al espacio de direcciones de otros procesos. Debe quedar claro que cuando se dice que un proceso se está ejecutando en modo núcleo, en realidad el que se está ejecutando es el núcleo pero en el nombre del proceso. Por ejemplo, cuando un proceso en modo usuario realiza una llamada al sistema está pidiendo al núcleo que realice en su nombre determinadas operaciones con el hardware de la máquina.

ii) FALSA (Página 136 del libro base).

Las interrupciones (hardware o software) son atendidas en modo NÚCLEO dentro del contexto del proceso que se encuentra actualmente en ejecución, aunque dicha interrupción no tenga nada que ver con la ejecución de dicho proceso.

Tanto las interrupciones (hardware o software) como las excepciones son tratadas en modo núcleo por determinadas rutinas del núcleo, no por los procesos. Este es el mecanismo por el cual el sistema operativo controla los procesos. La única forma de pasar de modo usuario a modo núcleo es mediante una interrupción o excepción y cada vez que esto ocurre el sistema operativo toma el control del procesador evitando que los procesos se ejecuten en modo núcleo.

2. *(1.5 p)* Explica las principales ventajas y desventajas que presentan las tuberías FIFO frente a las tuberías sin nombre.

Solución:

Los ficheros FIFO poseen las siguientes ventajas sobre las tuberías sin nombre:

- Tienen un nombre en el sistema de archivo.
- Pueden ser accedidos por procesos sin ninguna relación familiar.
- Son persistentes, es decir, continúan existiendo hasta que un proceso los desenlaza explícitamente usando la llamando al sistema unlink. Por tanto son útiles para mantener datos que deban sobrevivir a los usuarios activos.

Asimismo, poseen las siguientes desventajas:

- Deben ser borrados de forma explícita cuando no son usados.
- Son menos seguros que las tuberías, puesto que cualquier proceso con los privilegios adecuados puede acceder a ellos.
- Son difíciles de configurar y consumen más recursos.

3. *(2 p)* ¿qué es un sistema de ficheros transaccional?

Solución.

Un sistema de ficheros es transaccional si utiliza transacciones para implementar las operaciones en el disco. Al menos con las estructuras más importantes como los nodos-i y los directorios.

Las transacciones consisten en tratar las escrituras como operaciones atómicas, de tal forma que una vez acabado el proceso o bien la escritura se ha completado o bien no se ha realizado ningún cambio en absoluto.

De este modo se garantiza la consistencia del sistema de archivos cuando la operación de escritura se ve interrumpida por algún tipo de error como puede ser una caída de tensión.

4. (*2 p*) El ladrón de páginas de un cierto sistema UNIX debe transferir a un dispositivo de intercambio 25 páginas del proceso A, 100 páginas del proceso B, 50 páginas del proceso C y 75 páginas del proceso D. En una operación de escritura puede transferir 30 páginas al dispositivo de intercambio. Determinar la secuencia de operaciones de intercambio de páginas que tendría lugar si el ladrón de páginas examina las páginas de los procesos en el orden C, B, D y A.

Solución:

La secuencia de operaciones es la siguiente

1) El ladrón de páginas asigna espacio para 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso C. Le quedan por transferir 20 páginas del proceso C, 100 páginas del proceso B, 75 páginas del proceso D y 25 páginas del proceso A.
2) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 20 páginas del proceso C y 10 páginas del proceso B. Luego ha transferido todas las páginas del proceso C y le quedan por transferir 90 páginas del proceso B, 75 páginas del proceso D y 25 páginas del proceso A.
3) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso B. Le quedan por transferir 60 páginas del proceso B, 75 páginas del proceso D y 25 páginas del proceso A.
4) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso B. Le quedan por transferir 30 páginas del proceso B, 75 páginas del proceso D y 25 páginas del proceso A.
5) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso B. Luego ha transferido todas las páginas del proceso B. Le quedan por transferir 75 páginas del proceso D y 25 páginas del proceso A.

6) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso D. Le quedan por transferir 45 páginas del proceso D y 25 páginas del proceso A.

7) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 30 páginas del proceso D. Le quedan 15 páginas del proceso D y 25 páginas del proceso A.

8) El ladrón de páginas asigna espacio para otras 30 páginas y transfiere al dispositivo de intercambio 15 páginas del proceso D y 15 páginas del proceso C. Luego ha transferido todas las páginas del proceso D. Le quedan por transferir 10 páginas del proceso A.

9) El ladrón de páginas guarda las 10 páginas que le restan por intercambiar del proceso A en la lista de páginas de intercambio y no las intercambiará hasta que la lista este llena.

5. Conteste razonadamente a los siguientes apartados:
a) (*1 p*) Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa.
b) (*1.5 p*) Explicar el funcionamiento del programa mostrando la salida en pantalla.

```
    #include <sys/ipc.h>
    #include <sys/msg.h>
    #include <string.h>
    #include <stdio.h>
    #include <stdlib.h>
    void main(void)
    {
         int pid, msqid;
         struct{
              long tipo;
              char cadena[50];
              } mensaje;
         int longitud=sizeof(mensaje)-sizeof(mensaje.tipo);
         key_t llave;
[1]      creat("./archivo",0777);
[2]      llave=ftok("archivo", 'M');
[3]      msqid=msgget(llave, IPC_CREAT | 0600);
         if (msqid==-1){exit(1);}
[4]      if ((pid=fork())==-1)
              {
[5]                exit(1);
              }
              else if (pid==0)
              {
[6]                mensaje.tipo=2;
                   strcpy(mensaje.cadena,"dos");
[7]                msgsnd(msqid, &mensaje,longitud,0);
                   mensaje.tipo=1;
                   strcpy(mensaje.cadena,"uno");
                   msgsnd(msqid, &mensaje,longitud,0);
              }
              else
              {
[8]                msgrcv(msqid, &mensaje,longitud,1,0);
                   printf("mensaje %s \n",mensaje.cadena);
                   msgrcv(msqid, &mensaje,longitud,2,0);
                   printf("mensaje %s \n",mensaje.cadena);
              }
    }
```

Solución:

a) El significado de las sentencias numeradas es el siguiente:

[1]: La llamada al sistema `creat` crea un fichero llamado "`archivo`" en el directorio actual con permisos de lectura escritura para el usuario que ejecuta el programa (que es el propietario del fichero recién creado). Si el archivo ya existía simplemente borra su contenido.

[2]: La llamada al sistema `ftok` crea una llave asociada al fichero "archivo" y la letra M.

[3]: La llamada al sistema `msgget` crea una cola de mensajes asociada a la llave anterior y devuelve el identificador dela cola `msquid`.

[4]: La llamada al sistema `fork` crea un proceso hijo que es una copia idéntica al proceso padre. `fork` devuelve un argumento de salida distinto en función de si el código que se ejecuta es el padre o el hijo. Dicho resultado se almacena en la variable `pid`. La sentencia `if` comprueba el valor devuelto para comprobar si ha habido un error (`pid`=-1) y en caso afirmativo ejecuta [5].

[5]: La llamada al sistema `exit` finaliza el programa devolviendo un código de condición `1`.

[6]: Se establece el tipo de mensaje a `2`.

[7]: La llamada al sistema `msgsnd` envía a través de la cola de mensajes cuyo identificador es `msquid` el contenido de la estructura "mensaje".

[8]: La llamada al sistema `msgrcv` queda a la espera de la cola de mensajes y cuando llega un mensaje a la cola de mensajes de tipo `1` lo extrae copiando su contenido en la estructura "`mensaje`".

b) El funcionamiento del programa es el siguiente:

En primer lugar se crea un archivo vacío, dicho archivo se utiliza para crear una llave y con ella se crea una cola de mensajes cuyo identificador es `msquid`. Si aparece un error durante la creación de la cola `msqid` tomaría el valor -1 y se saldría del programa (`exit(1))`.

A continuación el programa se divide en un proceso padre y un proceso hijo. Si en la división se produce algún problema el valor devuelto por `fork` será -1 ejecutándose la línea [5] y terminando el programa. Si no hay ningún error el padre y el hijo realizan las siguientes acciones:

HIJO: envía un mensaje de tipo "`2`" con contenido "`dos`" y a continuación envía un mensaje de tipo "`1`" con contenido "`uno`" y finaliza.

PADRE: Espera la recepción de un mensaje de tipo "`1`". Cuando llega muestra el contenido por pantalla. A continuación espera un mensaje de tipo "`2`", realiza la misma operación y finaliza.
La salida devuelta por el programa es la siguiente:

```
mensaje uno
mensaje dos
```

NOTA: Cuando se extrae un mensaje de la cola, el SO saca el mensaje **del tipo solicitado** que haya llegado antes a la cola. De este modo aunque primero se envía el mensaje que contiene la cadena "`dos`" y después el que contiene la cadena "`uno`", el proceso padre los saca en el orden contrario.