AMPLIACIÓN DE SISTEMAS OPERATIVOS (71023016) **Enero 2024**

Material permitido: **Ninguno** Tiempo: **120 minutos** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71023016/

1. Conteste **razonadamente** a las siguientes apartados:

a) *(1 p)* Describir la gestión del área de intercambio en los SOBUNIX.

b) *(1 p)* Defina los siguientes conceptos de los SOBUNIX: núcleo no expropiable, núcleo expropiable y puntos de expropiación.

c) *(1 p)* En Linux, ¿para qué se utiliza la estructura `rq`? ¿qué información contiene?

d) *(1 p)* Enumerar y comentar brevemente las estructuras de datos utilizadas en Windows para implementar la seguridad.

2. *(2 p)* Dibujar un diagrama que represente la *estructura en una partición de disco* de un sistema de archivos UFS. Explicar la información contenida en cada uno de sus componentes.

3. En un SOBUNIX la máscara de modo del archivo ordinario expresada en octal es `5321`:

a) *(1 p)* Escriba la máscara de modo simbólica asociada a este archivo.

b) *(1 p)* ¿Qué orden se debe escribir en un intérprete de comandos para que la máscara simbólica del archivo pasase a ser `-rwSrw-rwT`?

Material permitido: **Ninguno** Tiempo: **120 minutos** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71023016/

4. *(2 p)* En la Figura 1 se muestra el código C del programa `f24`.

```
#include <signal.h>
void f1(int sig);
main()
{
   int a,b;
   signal(SIGUSR1,SIG_DFL);

   if(fork()==0)
   {
      signal(SIGUSR1,SIG_IGN);
      pause();
      exit(3);
   }
   else
   {
      a=wait(&b);
      printf("\n %d %d \n",a,b);
   }
}

void f1(int sig)
{
    printf("\nMensaje B\n");
}
```

Figura 1 – Código programa `f24`

Supóngase que al invocar este programa desde la línea de ordenes de un intérprete de comandos se crea un proceso con PID=5601 y que la asignación de los PIDs de los procesos hijos, si se llegaran a crear, se realizaría incrementando en una unidad el PID del proceso padre. Conteste **razonadamente** a los siguientes apartados:

a) *(1 p)* Explique el significado de la llamada al sistema

```
signal(SIGUSR1,SIG_DFL);
```

que aparece en el código del programa `f24`.

b) *(1 p)* Explicar el funcionamiento de `f24` si se escriben consecutivamente las siguientes tres órdenes en un interprete de comandos de un SOBUNIX:

1) `f24 &`
2) `kill -SIGUSR2 5601`
3) `kill -SIGUSR1 5602`

AMPLIACIÓN DE SISTEMAS OPERATIVOS (Cód. 71023016)
Solución Examen Enero 2024

Solución Ejercicio 1

a) La implementación de la técnica de gestión de memoria mediante demanda de página requiere la existencia de un espacio de almacenamiento en memoria secundaria en el que copiar aquellas páginas de procesos activos cargadas en la memoria principal que son seleccionadas para ser reemplazadas. A dicho espacio se le denomina *área de intercambio*. En los SOBUNIX el área de intercambio está formada típicamente por una o varias particiones de discos no formateadas a alto nivel y a las que se accede mediante operaciones de E/S en bruto.

Las páginas pertenecientes a la mayoría de las regiones del espacio de direcciones virtuales de un proceso (datos no inicializados, montículo, pila, archivos mapeados en memoria con el indicador `MAP_PRIVATE`, etc) son copiadas en el área de intercambio si son seleccionadas para ser reemplazadas.

Las páginas de código de los procesos o de las librerías compartidas no son copiadas en el área de intercambio. Estas páginas son generadas desde el archivo ejecutable correspondiente si es necesario cargarlas de nuevo en la memoria principal debido a un fallo de página. Lo mismo ocurre con las páginas seleccionadas para ser reemplazadas pertenecientes a archivos mapeados con el indicador `MAP_SHARED`. En este caso, además, si las páginas han sido modificadas son salvadas en los bloques de disco del archivo.

Cuando se crea una región del espacio de direcciones virtuales de un proceso que no sea de código se reserva el espacio suficiente en el área de intercambio para poder almacenar todas sus páginas. Una página *i* perteneciente a una determinada región de un proceso activo X se copia en el área de intercambio la primera vez que ésta es seleccionada por el escáner de páginas para ser reemplazada. Si posteriormente la ejecución del proceso X produce un fallo de página asociado a la página *i* el núcleo comprueba si existe una copia de la página en el área de intercambio. En caso afirmativo la página es copiada en memoria principal desde el área de intercambio. Más tarde si la página *i* vuelve a ser seleccionada para ser reemplazada de la memoria principal solo se copiará de nuevo en el área de intercambio si su contenido ha sido modificado (el bit *m* de su entrada de la tabla de páginas está activo), es decir, si la copia de la página *i* en memoria principal difiere de la copia de la página *i* en el área de intercambio.

El núcleo mantiene una *lista de espacio libre en el área de intercambio* que consulta cuando necesita asignar espacio del área de intercambio. También mantiene alguna clase de *mapa de intercambio* para poder localizar rápidamente las páginas almacenadas en el área de intercambio. La implementación y mantenimiento de estas estructuras depende de cada SOBUNIX

b) Un núcleo se dice que es *no expropiable* si a una unidad planificable (procesos o hilos del núcleo) ejecutándose en modo núcleo no se le puede expropiar el uso del procesador. Sólo se le puede expropiar en los siguientes casos: para tratar las interrupciones, cuando regresa a modo usuario o cuando entra en el estado dormido a la espera de que ocurra algún evento. Ejemplos de SOBUNIX de núcleo no expropiable son SVR3 y BSD 4.3. Este tipo de núcleos son más propensos a sufrir el problema de la inversión de prioridad, sin embargo reducen el número de mecanismos de sincronización usados en el núcleo.

Por el contrario, un núcleo se dice que *expropiable* si a una unidad planificable ejecutándose en modo núcleo se le puede expropiar el uso del procesador. En este caso, conviene tener presente que con objeto de garantizar la integridad de las estructuras del núcleo existen algunas regiones de código del núcleo en las que no está permitida la expropiación. A dichas regiones se les denomina

puntos de no expropiación. Un ejemplo de SOBUNIX de núcleo expropiable es Solaris. La implementación de un núcleo expropiable requiere que las estructuras del núcleo estén protegidas por mecanismos de sincronización.

Algunos SOBUNIX, como por ejemplo SVR4, aplican una medida intermedia que consiste en permitir la expropiación del núcleo sólo en determinados puntos del código del núcleo denominados *puntos de expropiación*. En estos puntos se garantiza que las estructuras de datos del núcleo se encuentran en un estado consistente. En estos sistemas una unidad planificable ejecutándose en modo núcleo solo puede ser expropiada cuando alcanza un punto de expropiación.

c) A partir de la versión 2.6.23 del núcleo de Linux el planificador O(1) fue sustituido por el planificador CFS (*Completely Fair Scheduling*, planificación completamente justa), En este planificador a cada procesador existente se le asocia una `rq` que contiene los siguientes campos:

- `cfs`. Es una estructura de tipo `cfs_rq` que permite implementar el árbol rojo-negro en que se organizan las tareas en el estado ejecutándose pertenecientes a la clase justa completamente.
- `rt`. Es una estructura de tipo `rt_rq` que permite implementar las 100 colas FIFO, una por cada nivel de prioridad en el rango [0, 99], en que se organizan las tareas en el estado ejecutándose pertenecientes a la clase de tiempo real.

d) La implementación de la seguridad en Windows se basa principalmente en el uso de las siguientes estructuras de datos:

- *Ficha de acceso* (access token). Cuando un usuario se autentifica Windows asigna al primer proceso creado para el usuario, el proceso `explorer.exe` que implementa el escritorio, una ficha de acceso que es heredada por todos los procesos que cree dicho proceso inicial debido a la interacción del usuario con el escritorio.

  Una ficha de acceso es una estructura que contiene, entre otros, los siguientes datos:

  - *Identificador de seguridad del usuario al que pertenece el proceso*. En Windows cada usuario tiene asociado un identificador de seguridad (Security IDentifier, SID), que lo identifica de forma univoca.
  - *Grupos*. Especifica los grupos a los que pertenece el usuario.
  - *Lista de control de acceso discrecional*. Es la lista de control de acceso asignada por defecto a los objetos creados por el proceso.
  - *Privilegios*. Permiten proporcionar a un proceso permisos para realizar ciertas tareas reservadas para el administrador, como por ejemplo apagar el computador o el acceso a ciertos archivos.
- *Descriptor de seguridad*. En Windows cuando se crea un objeto se le asigna un descriptor de seguridad que es una estructura de datos que especifica las operaciones que pueden realizar los diferentes usuarios sobre el objeto. Contiene, entre otros, los siguientes datos: SID del usuario propietario del objeto, SID del grupo propietario del objeto y un puntero a la *lista de acceso discrecional* (Discretionary Access Control List, DACL) asociada al objeto. La DACL del objeto contiene una o varias entradas.

Solución Ejercicio 2

Un sistema de archivos UFS presenta la siguiente estructura en la partición de disco donde se crea (ver Figura 1):

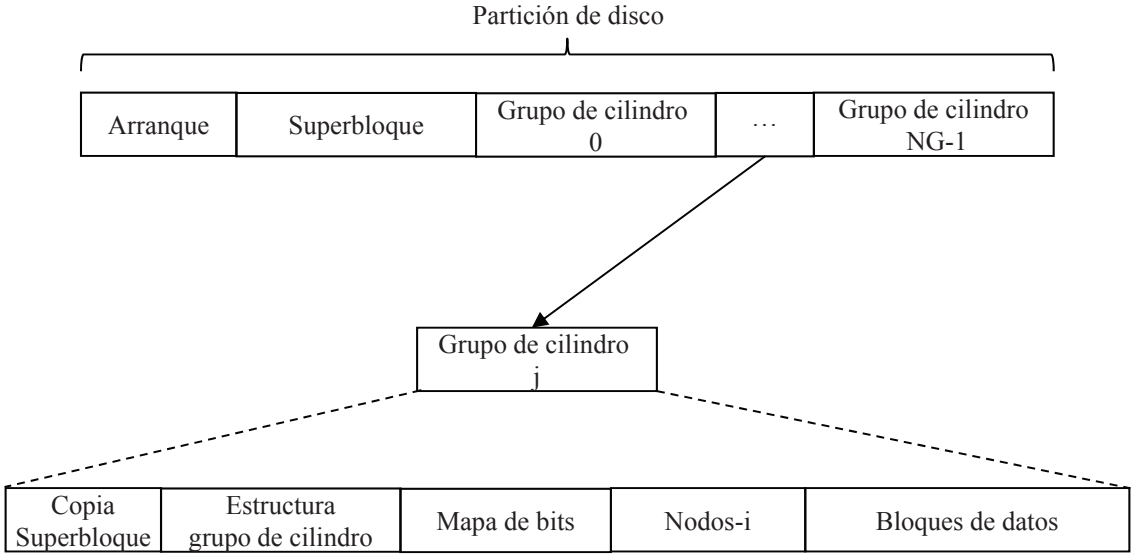

Figura 1 – Estructura de un sistema de archivos UFS

- *Área de arranque*. Se sitúa al principio de la partición. Contiene el código necesario para arrancar el SOBUNIX si dicha partición se usa como partición activa.
- *Superbloque*. Se sitúa a continuación del área de arranque. Es una estructura de datos que contiene información estadística y administrativa del sistema de archivo, como por ejemplo: un número mágico que identifica al sistema como de tipo UFS, el número, el tamaño y la localización de los grupos de cilindros, el tamaño de bloque de datos que se utiliza, el número total de bloques de datos y de nodos-i, etc. Esta información se configura al crear el sistema de archivos en la partición.
- *Grupos de cilindros*. Un grupo de cilindros está formado por un conjunto de cilindros contiguos. El número de grupos de cilindros (NG) y el número de cilindros que contiene un grupo son configurados cuando se crea el sistema de archivos UFS. Cada grupo de cilindro contiene la siguiente información:
  - *Una copia del superbloque*. La información que contiene el superbloque es fundamental para poder operar con el sistema de archivos. En previsión de la aparición de posibles errores en el disco el superbloque se replica en cada grupo de cilindros. La ubicación de la copia del superbloque en cada grupo de cilindros se elige cuidadosamente para evitar que todas las copias se encuentren en la misma superficie o plato del disco.
  - *Estructura de grupo de cilindros*. Contiene información estadística y administrativa del grupo de cilindros.
  - *Mapas de bits*. Especifica cuales son los bloques, fragmentos y nodos-i libres en el grupo de cilindros.
  - *Nodos-i del grupo de cilindros*. Cada nodo-i tiene un tamaño de 128 bytes y almacena los atributos del archivo. También almacena la localización de los bloques de datos de un archivo. En concreto contiene 15 direcciones de bloques de disco de 32 bits cada. Las 12 primeras direcciones son las de los 12 primeros bloques de datos del archivo, también denominados

bloques directos. Las tres direcciones restantes localizan a un bloque de indirección simple, un bloque de indirección doble y un bloque de indirección triple, respectivamente.

– *Bloques de datos del grupo de cilindros*. Contienen los datos propiamente dichos de los archivos.

Solución Ejercicio 3

a) La máscara de modo simbólica asociada a este archivo, cuyo nombre no se especifica en el enunciado, es:

```
--ws-w--t
```

b) La máscara de modo simbólica `-rwSrw-rwT` es equivalente a la máscara de modo binaria

```
101 110 110 110
```

que en octal es `5666`. Luego un posible comando a utilizar es:

```
chmod 5666 nombre_archivo
```

Solución Ejercicio 4

a) La llamada al sistema `signal(SIGUSR1,SIG_DFL)` permite especificar la acción que debe realizar el núcleo cuando reciba una señal `SIGUSR1` para el proceso que invoca esta llamada. En este caso, como el segundo argumento es `SIG_DFL`, se especifica que la acción que debe realizar el núcleo es la acción por defecto asociada a dicha señal, es decir, terminar el proceso.

b) 1) Al escribir la orden `f24 &` se comienza a ejecutar el programa `f24` en segundo plano. Supóngase que a la ejecución de dicho programa se le asocia el proceso A, por el enunciado se sabe que su PID es 5601.

Al ejecutar el proceso A en primer lugar se invoca a la llamada al sistema `signal` para especificar que cuando el proceso reciba la señal `SIGUSR1` la acción que debe realizar el núcleo es la acción por defecto asociada a dicha señal, es decir, terminar el proceso. En segundo lugar se invoca a la llamada al sistema `fork` para crear un proceso hijo B, cuyo PID sería 5602. Como `fork` devuelve el PID del proceso hijo al padre, entonces no se cumple la condición `if` y se invoca a la llamada al sistema `wait` que suspende la ejecución del proceso A hasta que finalice su proceso hijo B.

Asimismo cuando el proceso hijo B sea planificado invocará a la llamada al sistema `signal` para especificar que cuando reciba una señal `SIGUSR1` se debe ignorar. A continuación invoca a la llamada al sistema `pause` que hace que el proceso B quede a la espera de una señal que no ignore o que no tenga bloqueada.

2) Al escribir la orden `kill -SIGUSR2 5601` se envía una señal `SIGUSR2` al proceso con PID 5601, es decir, al proceso A. Como no se ha especificado ninguna acción para esta señal se ejecuta la acción por defecto, que es terminar el proceso A.

3) Finalmente al escribir la orden `kill -SIGUSR1 5602` se envía ahora una señal `SIGUSR1` al proceso B. Como este tipo de señal se configuró para ser ignorada, el proceso B permanece bloqueado en espera de la llegada de una señal que no ignore o que no tenga bloqueada.