Unidades Didácticas 4-6: Ejercicio de feedback 2

Sistemas operativos

Para aplicar los conocimientos adquiridos en la asignatura se propone la realización de este ejercicio feedback que corresponde a los contenidos desarrollados en las Unidades Didácticas 4, 5 y 6.

1. Objetivo

El objetivo del ejercicio es que el estudiante comience a utilizar las técnicas de programación concurrente para el desarrollo de aplicaciones de usuario. Para ello, tendrá que hacer uso de los tipos de datos y llamadas al sistema que establece la API del sistema operativo. Además, se pretende que los estudiantes apliquen los elementos tratados sobre memoria virtual para determinar el contenido de las tablas de páginas de los procesos y también para trasladar direcciones de memoria lógicas a sus correspondientes direcciones físicas. Finalmente, el ejercicio persigue la puesta en práctica de las funciones para la gestión y manejo de archivos y directorios que ofrece el sistema operativo.

2. Enunciado Ejercicio 1

Realice los siguientes apartados escribiendo el código necesario en lenguaje C:

2.1. Apartado 1

Se declarará un tipo de datos (*hebval_t*) correspondiente a una estructura con dos campos enteros: un identificador (*id*) correspondiente a un valor para identificar a una hebra y un valor asociado a ella (*val)*.

Se definirán las constantes simbólicas:

- N: Número de posiciones del array compartido (10)
- NHP: Número de hebras productoras (35)

Se declarará un array para almacenar N estructuras de tipo *hebval_t*.

2.2. Apartado 2

Se escribirá una función **hebraProductora** que recibirá por parámetro un puntero a una estructura como la definida en 1. La hebra copiará la estructura en la posición que corresponda del array (será necesario utilizar un puntero de inserción compartido). Si tras insertar el elemento recibido en el array, se detecta que éste está lleno:

1) No se permitirá nuevas inserciones en el array.
2) El puntero de inserción volverá a la posición 0.
3) Se activará el mecanismo de sincronización correspondiente para que pueda ejecutar la **hebraConsumidora.**

2.3. Apartado 3

Se escribirá una función **hebraConsumidora** que se activará cuando el array esté lleno y mostrará por pantalla todos los elementos del array en el formato *[[id 0, val 0], [id 1, val 1], …, [id N-1, val N-1]]*:

La hebraConsumidora determinará el final de la ejecución de todas las hebras tras mostrar los elementos insertados en el array por todas las hebras productoras.

2.4. Apartado 4

Se escribirá una función *main()* donde:

1) Se declararán los arrays para las hebras y semáforos necesarios.
2) Se inicializarán todas las estructuras del array con el valor -1 en ambos campos
3) Se lanzarán las NHP hebrasProductoras. La estructura pasada como argumento a cada hebra llevará un valor secuencial distinto en ambos campos (id y val). La primera hebra el valor 0, la segunda el valor 1, etc.
4) Se lanzará la hebraConsumidora
5) Se esperará por la terminación de todas las hebras

3. Enunciado Ejercicio 2

Supongamos un sistema que utiliza memoria virtual con paginación. Sabiendo que la memoria física cuenta con 64 marcos de página y que en las direcciones lógicas los 7 bits menos significativos corresponden a la posición de una palabra dentro de una página, obtener:

3.1. Apartado 1

1) Tamaño total de la memoria física en posiciones.
2) Tamaño en palabras de los marcos de página.
3) Número máximo de páginas de que puede constar un proceso.

3.2. Apartado 2

Supongamos la siguiente asignación de procesos a la memoria.

0	A0
1	A1
2	B0
3	B1
4	B2
5	C0
6	C1
7	C2
8	A2
9	A3
10	B3
11	B4

1) Obtener la tabla de páginas de los procesos A, B y C.

3.3. Apartado 3

Suponiendo que el tamaño del proceso A es de 512 posiciones y el del tamaño C de 384 posiciones, obtener:

1) La dirección virtual y física correspondientes a la última dirección lógica del proceso A.
2) La dirección física correspondiente a la dirección virtual 130 del proceso A.
3) La primera y última dirección del proceso C.

4. Enunciado Ejercicio 3

Escriba el código necesario de acuerdo con las siguientes indicaciones:

4.1. Apartado 1

Escriba un programa que reciba 4 parámetros: el primero corresponderá a un directorio y los 3 siguientes serán archivos. El programa buscará los dos primeros archivos en el directorio especificado y con ellos creará un nuevo archivo resultado de concatenarlos. El nuevo archivo llevará se tendrá el nombre que indique el tercer parámetro

4.2. Apartado 2

Escriba un programa que reciba por parámetro el nombre de un directorio. El programa mostrará por pantalla un listado con los nombres de sus archivos, y el número de inodo que correspondiente a cada uno. Además, para cada uno de los inodos mostrados, se indicará el número total de enlaces duros que lo referencian.

5. Instrucciones de entrega

Para la entrega de la propuesta de solución al ejercicio feedback se seguirán las siguientes indicaciones:

- **Se subirá un único archivo comprimido de nombre <NPAlumno>.zip**
- **En el archivo comprimido se incluirán 2 archivos de código fuente, ej1.c y ej3.c, con la propuesta de solución a los Ejercicios 1 y 3. También se incluirá una memoria en formato PDF que recoja los elementos más importantes relativos al desarrollo del código de los Ejercicios 1 y 3, así como las correspondientes capturas de pantalla con los resultados de su compilación y ejecución. En la memoria, también se adjuntará la propuesta de solución al Ejercicio 2.**
- **Dentro de los archivos ej1.c y ej3.c se inclurirán comentarios internos para indicar los distintos apartados y subapartados. A modo de ejemplo:**

```
//Apartado 2
//Apartado 2.1
```

UAX Universidad Alfonso X el Sabio

G R A C I A S

UAX.COM