8 Manejo de archivos

Introducción

El sistema operativo UNIX® mantiene una interfaz sencilla para el manejo de archivos, de hecho se le utiliza también para el acceso a una gran cantidad de dispositivos de E/S, los cuales incluyen las tarjetas de red. Esta característica brinda una potente ventaja respecto a otros sistemas operativos, que exigen extensos conjuntos de funciones para llevar a cabo operaciones similares.

Por otra parte se cuenta con una característica adicional que se conoce como proyección de archivos. Esta operación permite reservar un área de memoria, la cual va a contener una parte o el total de un archivo, que originalmente se encuentra en el disco duro. Con esta operación se logra un acceso más rápido al archivo y para algunos programadores esta es la forma óptima para manejar archivos.

Ejemplo práctico

En el programa 8-1 se muestra el código de un programa que ejecuta una copia de un archivo, de manera similar a lo que realiza el comando cp (véase con `man`), y que utiliza llamadas al sistema POSIX.

Programa 8-1 Copia un archivo hacia otro destino

```
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>

char buffer[BUFSIZ]; /* Región de memoria para el
almacenamiento temporal de datos */

int main(int argc, char *argv[])
{
   int nbytes, origen, destino;

   if(argc != 3) {
      printf("Forma de uso: %s archivo_origen
archivo_destino\n", argv[0]);
```

continúa

```
        exit(-1);
    }
1 if((origen = open(argv[1], O_RDONLY)) == -1)
    {
        perror(argv[1]);
        exit(-1);
    }
    if((destino = open(argv[2], O_WRONLY|O_TRUNC|O_CREAT, 0666))
== -1)
    {
        perror(argv[2]);
        exit(-1);
    }
2 while((nbytes = read(origen, buffer, sizeof buffer)) > 0)
3     write(destino, buffer, nbytes);

4 close(destino);
  close(origen);
```

Línea 1 En UNIX cada archivo tiene asociado un número entero positivo que lo identifica, conocido como descriptor de archivo. Para obtener un descriptor de archivo asociado a un archivo en el disco duro se utiliza `open`. Esta llamada al sistema puede tener dos o tres parámetros. El primero especifica el nombre del archivo o su ruta absoluta, y el segundo mantiene un conjunto de banderas que especifican las acciones que deben llevarse a cabo cuando se abre un archivo. La bandera obligatoria es una de las tres siguientes:

`O_RDONLY`	Abre el archivo sólo para lectura.
`O_WRONLY`	Abre el archivo sólo para escritura.
`O_RDWR`	Abre el archivo para lectura y escritura.

Adicionalmente se pueden tener otras banderas, las cuales permiten realizar operaciones adicionales, siempre y cuando éstas se añadan con un operador de *bits* OR.

`O_APPEND`	Coloca los datos escritos al final del archivo.
`O_TRUNC`	Trunca el archivo a cero bytes.
`O_CREAT`	Crea el archivo si no existe, con los permisos que especifique el tercer parámetro de `open`.

El conjunto de permisos numéricos que especifica el tercer parámetro funciona de manera similar al que utiliza el comando `chmod` (veáse con el comando `man`).

Línea 2 La llamada al sistema `read` permite leer *bytes* de un archivo y tiene el siguiente prototipo:

```
size_t read(int fildes, void *buf, size_t nbytes);
```

el primer parámetro es el descriptor de archivo asociado con el archivo del cual se quiere hacer una lectura. El segundo parámetro es una dirección de memoria a partir de la cual se van a almacenar los *bytes* leídos provenientes del archivo. El tercero indica el número de *bytes* que se desean leer. Finalmente se devuelve el número de aquellos que se lograron leer. Note que no siempre el número de los que se desea leer es el número de bytes leídos, por ejemplo, en el caso de que el archivo contenga un menor número de *bytes* de los que se espera que existan.

Línea 3 La llamada al sistema `write` permite escribir datos en un archivo previamente abierto. Su prototipo es el siguiente:

```
size_t write(int fildes, const void *buf, size_t nbytes);
```

el primer parámetro de `write` es el descriptor de archivo correspondiente al archivo donde se va a escribir. El segundo, parámetro es un apuntador a una región de memoria, a partir de la cual se van a tomar los subsiguientes *bytes* para escribirlos en el archivo. El tercer parámetro indica el número de *bytes* que se desean escribir en el archivo. Finalmente, se devuelve el número de *bytes* que se lograron escribir.

Línea 4 Con la llamada al sistema `close` se termina la asociación entre el archivo y su descriptor de archivo, de manera que este último se encuentre disponible para su reutilización.

La llamada al sistema `lseek` permite ajustar un apuntador de lectura o escritura sobre el archivo. Es decir que podemos determinar a partir de donde se puede comenzar a leer o escribir en un archivo. Esta llamada tiene el siguiente prototipo:

```
off_t lseek(int fildes, off_t offset, int whence);
```

el primer parámetro es el descriptor de archivo, el parámetro `offset` se utiliza para especificar la posición contando "n" *bytes* a partir de lo que indique el tercer parámetro. Éste último especifica la posición a partir de la cual se comienza a contar `offset`, y tiene las siguientes opciones:

`SEEK_SET`	offset se cuenta a partir del inicio del archivo.
`SEEK_CUR`	offset se cuenta a partir de donde se encuentre posicionado actualmente el apuntador de lectura o escritura.
`SEEK_END`	offset se cuenta a partir del final del archivo.

Ejercicio 8.1 Mediante el uso de `lseek`, elabore un programa que invierta los *bytes* de un archivo, de modo que el último sea ahora el primero y el primero sea el último.