

Práctica 2.2: Sistema de Ficheros

Objetivos

En esta práctica se revisan las funciones del sistema básicas para manejar un sistema de ficheros, referentes a la creación de ficheros y directorios, duplicación de descriptores, obtención de información de ficheros o el uso de cerrojos.

Contenidos

- Preparación del entorno para la práctica
- Creación y atributos de ficheros
- Redirecciones y duplicación de descriptores
- Cerrojos de ficheros
- Directorios

Preparación del entorno para la práctica

La realización de esta práctica únicamente requiere del entorno de desarrollo (compilador, editores y utilidades de depuración). Estas herramientas están disponibles en las máquinas virtuales de la asignatura y en la máquina física de los puestos del laboratorio.

En la realización de las prácticas se puede usar cualquier editor gráfico o de terminal. Además se puede usar tanto el lenguaje C (compilador gcc) como C++ (compilador g++). Si fuera necesario compilar varios ficheros se recomienda el uso de alguna herramienta para la compilación de proyectos como make. Finalmente, el depurador recomendado en las prácticas es gdb. **No está permitido** el uso de IDEs como Eclipse.

Creación y atributos de ficheros

El i-nodo de un fichero guarda diferentes atributos de éste, como por ejemplo el propietario, permisos de acceso, tamaño o los tiempos de acceso, modificación y creación. En esta sección veremos las llamadas al sistema más importantes para consultar y fijar estos atributos así como las herramientas del sistema para su gestión.

Ejercicio 1. La herramienta principal para consultar el contenido y atributos básicos de un fichero es ls. Consultar la página de manual y estudiar el uso de las opciones -a -l -d -h -i -R -l -F y --color. Estudiar el significado de la salida en cada caso.

Ejercicio 2. El *modo* de un fichero es <tipo><rw_x_propietario><rw_x_grupo><rw_x_resto>:

- tipo: - fichero ordinario; d directorio; l enlace; c dispositivo carácter; b dispositivo bloque; p FIFO; s socket
- rw_x: r lectura (4); w escritura (2); x ejecución (1)

Comprobar los permisos de algunos directorios (con ls -ld).

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized, blue, serif font with a shadow effect, set against a light blue and orange background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejercicio 5. Escribir un programa que, usando `open(2)`, cree un fichero con los permisos `rw-r--r-x`. Comprobar el resultado y las características del fichero con la orden `ls`.

Ejercicio 6. Cuando se crea un fichero, los permisos por defecto se derivan de la máscara de usuario (`umask`). El comando interno de la *shell* `umask` permite consultar y fijar esta máscara. Usando este comando, fijar la máscara de forma que los nuevos ficheros no tengan permiso de escritura para el grupo y no tengan ningún permiso para otros. Comprobar el funcionamiento con los comandos `touch`, `mkdir` y `ls`.

Ejercicio 7. Modificar el ejercicio 5 para que, antes de crear el fichero, se fije la máscara igual que en el ejercicio 6. Comprobar el resultado con el comando `ls`. Comprobar que la máscara del proceso padre (la *shell*) no cambia.

Ejercicio 8. El comando `ls` puede mostrar el i-nodo con la opción `-i`. El resto de información del i-nodo puede obtenerse usando el comando `stat`. Consultar las opciones del comando y comprobar su funcionamiento.

Ejercicio 9. Escribir un programa que emule el comportamiento del comando `stat` y muestre:

- El número *major* y *minor* asociado al dispositivo.
- El número de i-nodo del fichero.
- El tipo de fichero (directorio, enlace simbólico o fichero ordinario).
- La hora en la que se accedió el fichero por última vez. ¿Qué diferencia hay entre `st_mtime` y `st_ctime`?

Ejercicio 10. Los enlaces se crean con la orden `ln`:

- Con la opción `-s`, se crea un enlace simbólico. Crear un enlace simbólico a un fichero ordinario y otro a un directorio. Comprobar el resultado con `ls -l` y `ls -li`. Determinar el i-nodo de cada fichero.
- Repetir el apartado anterior con enlaces rígidos. Determinar los i-nodos de los ficheros y las propiedades con `stat` (observar el atributo número de enlaces).
- ¿Qué sucede cuando se borra uno de los enlaces rígidos? ¿Qué sucede si se borra uno de los enlaces simbólicos? ¿Y si se borra el fichero original?

Ejercicio 11. `link(2)` y `symlink(2)` crean enlaces rígidos y simbólicos, respectivamente. Escribir un programa que reciba una ruta a un fichero como argumento. Si la ruta es un fichero regular, creará un enlace simbólico y rígido con el mismo nombre terminado en `.sym` y `.hard`, respectivamente. Comprobar el resultado con la orden `ls`.

Redirecciones y duplicación de descriptores

La *shell* proporciona operadores (`>`, `>&`, `>>`) que permiten redirigir un fichero a otro, ver los ejercicios propuestos en la práctica opcional. Esta funcionalidad se implementa mediante `dup(2)` y `dup2(2)`.

Ejercicio 12. Escribir un programa que redija la salida estándar de un fichero que sube cosas...

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cerrosos de ficheros

El sistema de ficheros ofrece cerrosos de ficheros consultivos.

Ejercicio 14. El estado y cerrosos de fichero en uso en el sistema se pueden consultar en el fichero `/proc/locks`. Estudiar el contenido de este fichero.

Ejercicio 15. Escribir un programa que consulte y muestre en pantalla el estado del cerrojo sobre un fichero usando `lockf(3)`. El programa mostrará el estado del cerrojo (bloqueado o desbloqueado). Además:

- Si está desbloqueado, fijará un cerrojo y escribirá la hora actual. Después suspenderá su ejecución durante 30 segundos (con `sleep(3)`) y a continuación liberará el cerrojo.
- Si está bloqueado, terminará el programa.

Ejercicio 16 (Opcional). El comando `flock` proporciona funcionalidad de cerrosos antiguos BSD en guiones *shell*. Consultar la página de manual y el funcionamiento del comando.

Directorios

Ejercicio 17. Escribir un programa que cumpla las siguientes especificaciones:

- El programa tiene un único argumento que es la ruta a un directorio. El programa debe comprobar la corrección del argumento.
- El programa recorrerá las entradas del directorio de forma que:
 - Si es un fichero normal, escribirá el nombre.
 - Si es un directorio, escribirá el nombre seguido del carácter `/`.
 - Si es un enlace simbólico, escribirá su nombre seguido de `->` y el nombre del fichero enlazado. Usar `readlink(2)` y dimensionar adecuadamente el *buffer*.
 - Si el fichero es ejecutable, escribirá el nombre seguido del carácter `*`.
- Al final de la lista el programa escribirá el tamaño total que ocupan los ficheros (no directorios) en kilobytes.

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the rest of the text. The logo is set against a background of a light blue and orange gradient with a subtle shadow effect.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70