Ejercicios Módulo 2. Manejo de ficheros

Ampliación de Sistemas Operativos. Curso 2017-2018

Cuestiones

Creación de ficheros Ejecuta la orden `touch prueba`. ¿Con qué permisos se ha creado el nuevo fichero?. Ahora ejecuta `umask` y comprueba qué máscara se está usando para fijar los permisos de un fichero nuevo? ¿Qué argumentos estará usando `touch` en su llamada a `open()` para que se reproduzca el comportamiento observado?. Modifica la máscara para que `touch` cree ficheros con permisos de lectura y escritura para usuario y grupo, pero sólo de lectura para el resto.

Tamaño ficheros Crea un nuevo fichero usando la orden

```
echo "Hola mundo!!"  > nuevoFichero
```

¿Qué tamaño tiene ese fichero? Usa la orden:

```
stat -c " %s %b %B %o" nuevoFichero
```

¿Qué significa cada número de los que aparecen? ¿Cuánto ocupa realmente el fichero en disco?

Enlaces Crea un enlace simbólico (`man ls`) al fichero `nuevoFichero` llamado `miLink`. ¿Cuánto ocupa el fichero del enlace? Crea un nuevo enlace simbólico al mismo fichero pero usando la ruta completa hasta `nuevoFichero`. ¿Cuánto ocupa éste? Explica la diferencia. ¿Coincide alguno de los tamaños con el del fichero original?

Nodos-i Utiliza `ls -li` para mostrar la información de los ficheros del directorio actual, incluyendo su número de nodo-i. Los enlaces simbólicos creados anteriormente, ¿tienen el mismo número de nodo-i que el fichero al que apuntan? Crea ahora un enlace rígido al mismo fichero `nuevoFichero`. ¿Coincide su número de nodo-i con el de alguno de los anteriores?

Redirección Considera nuevamente la orden

```
echo "Hola mundo!!"  > nuevoFichero
```

¿Qué efecto tiene el símbolo > ? ¿Qué código estará ejecutando `bash` para conseguir el comportamiento observado. Investiga como redireccionar la salida de error estándar en `bash`.

Apertura y creación de ficheros

open.c Escribe un programa que simplemente abra un fichero existente utilizando la función `open`. Varía los `flags` de apertura, y a continuación intenta responder a las siguientes preguntas (consulta la página de manual para responder a alguna de las preguntas):

1. ¿Qué combinación de flags para la función `open` serían equivalentes a la invocación de la función `creat`?
2. ¿Qué flag o combinación de flags serían necesarios para que la función `open` devolviese un error si se intenta crear un fichero que ya existe?
3. ¿Bajo qué circunstancia la función `open` devolverá un error de tipo `ENOENT`? En esa circunstancia, ¿qué flag evitaría dicho error?
4. ¿Bajo qué circunstancia o circunstancias (combinación de `flags` y tipo de fichero), la función `open` devolverá un error de tipo `EISDIR`?
5. ¿Bajo qué circunstancia la función `open` devolverá un error de tipo `ENAMETOOLONG`?

Propiedades de un fichero

stat.c Completa el programa llamado `stat.c` que, dado un determinado fichero, muestre por pantalla información relativa a él que ofrezca la función `stat`: dispositivo, número de inodo, permisos del usuario y tipo (al menos si es fichero regular, directorio o enlace simbólico), tamaño total (en bytes), número de bloques utilizados, e información temporal –acceso y modificación– con formato legible para el usuario.

Descriptores de ficheros

dup.c Completa el programa para que escriba en un fichero la frase `Hola, mundo`. No está permitido utilizar directamente la llamada al sistema `write` ni la función `fwrite()` (se debe mantener la llamada a `printf()` tal y como está). Propón dos alternativas distintas para resolver el problema.

cat.c Escribe una función llamada `copy`, que no reciba ningún parámetro. Dicha función leerá continuamente un carácter desde el descriptor de fichero `0`, y escribirá en el descriptor de fichero `1` cada cadena de caracteres introducida.

Invocar dicha función desde `main()` para mostrar por pantalla el contenido de un determinado fichero.

Desplazamiento en ficheros

lseek.c Ejecuta ese fichero que crea un agujero (*hole*) de un determinado tamaño en un fichero. Abre el fichero con un editor de textos y comprueba el resultado. Utiliza el comando`hexdump` para mostrar el contenido del fichero.

Lectura/escritura en ficheros

rw.c Escribe un programa que copien fichero usando las llamadas al sistema `read`/`write`. Usa `times` para temporizar y ejecuta varias veces el código con el mismo fichero origen aumentando el número de bytes leidos/escritos en cada llamada al sistema. Para ello usa una macro `TAM_BLOQUE` cuyo valor se especificará en tiempo de compilación usando el flag `-D` (prueba con valores de `TAM_BLOQUE` de 1,64,512,1024 y 4096, usando como entrada un fichero de varios MB como por ejemplo `/boot/initrd.img-3.2.0-4-amd64`). ¿Varia significativamente el tiempo de copia?

Enlaces

lee_enlace.c Escribe un programa con nombre `lee_enlace.c` que, para un determinado enlace simbólico, muestre por pantalla el nombre del fichero al que apunta dicho enlace. ¿Qué se almacena en el campo `st_size` del inodo correspondiente a dicho fichero? Ten especial precaución en la reserva de espacio de memoria para la cadena que contendrá el nombre del fichero destino.

links.c Implementa un programa que cree un fichero, y a continuación un enlace simbólico y otro duro que apunten a él. Extrae el tamaño de cada uno de dichos ficheros. ¿Son diferentes en ambos casos? ¿Por qué?

Manejo de directorios

mi_ls.c Escribe un programa llamado `mi_ls.c` que, para un determinado directorio, muestre por pantalla su contenido (nombres de los ficheros que lo componen).

mi_ls_bis.c A partir del programa anterior, crea un nuevo programa que, además, muestre por pantalla la información de los permisos de cada fichero tal y como la muestra el comando `ls -l`

Entrega

La entrega consistirá en un solo fichero comprimido (`.tgz`) que contendrá:

- Todos los ficheros fuentes completos (NO debe incluirse ningún fichero objeto o ELF).
- El fichero *makefile*.
- Un fichero de texto o PDF con las respuestas a las preguntas planteadas en cada ejercicio, así como una copia de las fuentes de cada apartado completado.