Introducción a Sistemas Operativos: Ficheros

Clips xxx
Francisco J Ballesteros

1. Cambiando los metadatos

Los metadatos cambian muchas veces simplemente con operar sobre el fichero para cambiar los datos. Por ejemplo, el tamaño del fichero cambia si lo haces crecer utilizando `write`, al igual que las fechas de acceso y modificación.

Aunque ya podemos vaciar un fichero utilizando `open` y hacerlo crecer utilizando `seek` y `write`, en ocasiones querremos truncar un fichero o cambiar su tamaño a un número dado de bytes. Esto puede hacerse con *truncate(2)*, que ajusta el tamaño al número indicado. Si el fichero era más grande, se tiran los bytes del final que sobran. Si el fichero era más pequeño, se hace crecer con ceros. La llamada puede utilizarse como en

```
if (truncate("myfile", 1024) < 0) {
    // truncate ha fallado...
}
```

que dejaría `myfile` con 1024 bytes exactamente. Como podrás suponer, existe *ftruncate(2)* que trunca el fichero indicado por un descriptor de fichero en lugar de por un path.

En realidad, `truncate` cambia el tamaño en los metadatos y si ello require liberar bloques en disco para los datos que ya no se usan, así lo hace.

Los permisos pueden cambiarse con la llamada *chmod(2)*, que es la que utiliza el comando *chmod(1)* que hemos usado anteriormente. Aunque esta vez sí diremos que también tienes *fchmod(2)*, en el futuro no mencionaremos más las funciones que operan con descriptores, ya estás acostumbrado a leer el manual y sabes cómo encontrarlas y usarlas.

Este programa ajusta los permisos de los ficheros que se pasan como argumento al valor indicado por el primer argumento, similar a *chmod(1)*.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <err.h>
```

```
int
main(int argc, char* argv[])
{
    long mode;
    int i, sts;

    if (argc < 3) {
        fprintf(stderr, "usage: %s mode file...\n", argv[0]);
        exit(1);
    }
    mode = strtol(argv[1], NULL, 8);
    sts = 0;
    for (i = 2; i < argc; i++) {
        if (chmod(argv[i], mode) < 0) {
            warn("chmod: %s", argv[i]);
            sts = 1;
        }
    }
    exit(sts);
}
```

Podemos utilizarlo como puedes ver:

```
unix$ chm 664 chm.c
unix$ ls -l chm.c
-rw-rw-r--  1 nemo  staff  452 Aug 21 11:59 chm.c
unix$
```

El propietario y el grupo a que pertenece un fichero se puede cambiar como se describe en *chown(2)*, como en:

```
if (fchown(fd, newuid, newgid) < 0) {
    // fchown ha fallado
}
```

¿Y si tienes un path en lugar de un descriptor? ¿Cómo lo harás?

Las fechas de acceso y modificación de un fichero pueden cambiarse con *utimes(2)*. Por ejemplo, este program es similar a *touch(1)*, aunque nunca crea los ficheros si no existen.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <err.h>

int
main(int argc, char* argv[])
{
    int i, sts;
    struct timeval tv[2];

    if (argc < 2) {
        fprintf(stderr, "usage: %s file...\n", argv[0]);
        exit(1);
    }
    if (gettimeofday(&tv[0], NULL) < 0) {
        err(1, "gettimeofday");
    }
    tv[1] = tv[0];
    sts = 0;
    for (i = 1; i < argc; i++) {
        // could use just NULL instead of tv.
        if (utimes(argv[i], tv) < 0) {
            warn("utimes: %s", argv[i]);
            sts = 1;
        }
    }
    exit(sts);
}
```

Aquí hemos utilizado *gettimeofday(2)* para pedir a UNIX la fecha y hora actual y cambiamos los tiempos de los ficheros justo a ese momento. El mismo efecto podría haberse conseguido utilizando "`NULL`" como argumento (que hace que `utimes` use la fecha actual).