Advanced Programming in the UNIX Environment

Week 03, Segment 4: 
chmod(2) and chown(2)

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/
chmod(2), lchmod(2), fchmod(2)

```c
#include <sys/stat.h>
#include <fcntl.h>

int chmod(const char *path, mode_t mode);
int lchmod(const char *path, mode_t mode);
int fchmod(int fd, mode_t mode);
int fchmodat(int fd, const char *path, mode_t mode, int flag);
```

Returns: 0 if OK, -1 on error

Changes the permission bits on the file. Must be either euid 0 or euid == st_uid.

`mode` can be any of the bits from our discussion of st_mode as well as:

- S_ISUID – setuid
- S_ISGID – setgid
- S_ISVTX – sticky bit (aka “saved text”)
- S_IRWXU – user read, write and execute
- S_IRWXG – group read, write and execute
- S_IRWXO – other read, write and execute
perror("can't chmod file");
exit(EXIT_FAILURE);
}

/* set absolute mode to rw-r--r-- */
if (chmod("file1", S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH) == -1) {
  perror("can't chmod file1");
  exit(EXIT_FAILURE);
}

apue$ cc chmod.c
apue$ touch file file1
apue$ ls -l file file1
-rw------- 1 jschauma users 0 Sep 12 20:10 file
-rw------- 1 jschauma users 0 Sep 12 20:10 file1
apue$ ./a.out
apue$ ls -l file file1
--w----S---- 1 jschauma users 0 Sep 12 20:10 file
-rw-r--r-- 1 jschauma users 0 Sep 12 20:10 file1
apue$ chmod g+x file
apue$ ls -l file file1
--w----S---- 1 jschauma users 0 Sep 12 20:10 file
-rw-r--r-- 1 jschauma users 0 Sep 12 20:10 file1
apue$
# include <unistd.h>
#include <fcntl.h>

int chown(const char *path, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int fchownat(int fd, const char *path, uid_t owner, gid_t group, int flag);

Returns: 0 if OK, -1 on error

Changes st_uid and st_gid for a file. Generally requires euid 0. (Some SVR4’s let users chown their files to anybody. POSIX allows either, depending on _POSIX_CHOWN_RESTRICTED.)

owner or group can be -1 to indicate that it should remain the same.

Non-superusers can change the st gid field if both:

• euid == st_uid; and

• owner == st_uid and group == egid (or one of the supplementary group IDs)
chown(2), lchown(2), fchown(2)

CS631 - Advanced Programming in the UNIX Environment

apue$ ls -l file
-rw-r--r-- 1 jschauma wheel 0 Sep 12 21:58 file
apue$ ./a.out file
Successfully chowned file to 1000:-1.
Unable to chown(file, 1001, -1): Operation not permitted
Successfully chowned file to -1:100.
Unable to chown(file, -1, 4): Operation not permitted
apue$ ls -l file
-rw-r--r-- 1 jschauma users 0 Sep 12 21:58 file
apue$ ./a.out /etc/passwd
Unable to chown(/etc/passwd, 1000, -1): Operation not permitted
Unable to chown(/etc/passwd, 1001, -1): Operation not permitted
Unable to chown(/etc/passwd, -1, 100): Operation not permitted
Unable to chown(/etc/passwd, -1, 4): Operation not permitted
apue$ ls -l /etc/passwd
-rw-r--r-- 1 root wheel 1485 Sep 12 15:12 /etc/passwd
apue$ sudo ./a.out file
Successfully chowned file to 0:-1.
Successfully chowned file to 1001:-1.
Successfully chowned file to -1:100.
Successfully chowned file to -1:4.
apue$ ls -l file
-rw-r--r-- 1 fred tty 0 Sep 12 21:58 file
apue$
chmod(2) and chown(2)

chmod(2) and chown(2) consistently follow the semantics of the other calls we've seen.

Only root and the owner of a file can change its permissions.

Only root can change the owner of a file, but the owner may change the group ownership of a file.

Changing file permissions and ownerships has significant security implications.

Coming up next: default file ownership and permissions for newly created files.