System Programming

3. File IO (2): System Call

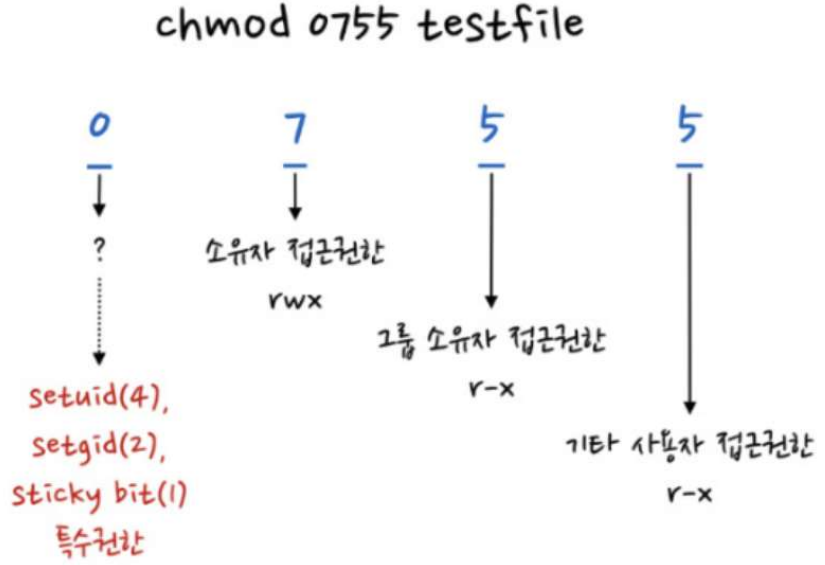
Seung-Ho Lim

Dept. of Computer & Electronic Systems Eng.

File permission attributes

파일종류	특수권한			소유자 접근권한			그룹 소유자 접근권한			기타 사용자 접근 권한		
-,d,c,b,s,l,p	4	2	1	4	2	1	4	2	1	4	2	1
	setuid	setgid	sticky bit	r	w	x	r	w	x	r	w	x

d: directory
c: character device file
b: block device file
s: socket
l: symbolic link



Process's Creator (1)

```
#include <sys/types.h>
#include <unistd.h>

uid_t getuid(void)   // process creator's uid
```

- return
  - user ID of the process if OK
  - -1 on error

```
#include <sys/types.h>
#include <unistd.h>

uid_t getgid(void)   // process creator's gid
```

- return
  - group ID of the process if OK
  - -1 on error

Process's Creator (2)

```
uid_t geteuid(void)
```

- return
  - effective user ID of the process if OK
  - -1 on error
- note
  - process's effective user id is used as a key for a kernel's protection system, and normally *uid* = *euid*,
  - but sometimes *euid* is a different one from *uid* for the dynamic protection system,

```
uid_t getegid(void)
```

- return
  - effective group ID of the process if OK
  - -1 on error

File's ID

- When a process is created, the user(creator)'s IDs are assigned to the process.
- But, in the following cases, a process's effective IDs are set to a **file owner**'s IDs
  - if we run a program file with S_ISUID (or S_ISGID) bit, the process's UID is not my UID, but the file owner's UID (or S_ISGID).
- How to set the bits (example)

```
$ chmod u+s a.out
$ chmod g+s a.out
```

- Example

```
$ ls -al /bin
-rwsr-xr-x  1 root root   26492 Dec 1  2017 mount
```

  - this mount command is executed, the mount process's UID is set to root, not the user. → i.e. with the root's authority, the mount will be run!

Sticky bit: S_ISVTX

- In a directory with sticky bit
  - users can make their own files or subdirectories to the directory
  - but, each file can be deleted only by its owner or supervisor.
- Example

```
$ chmod o+t /test
$ touch /test/file1
$ rm /test/file1 → OK!
$ touch /test/file1
```

```
....
(another user login)
$ touch /test/file2
$ rm /test/file2 → OK!
$ rm /test/myfile → Failed!
```

File access of a process

- File access (read/write/execute) is allowed in the following cases
  - if the effective UID of the process is 0 (supervisor)
  - if the effective UID of the process is equal to that of file owner, and if the access permission bit of owner is SET
  - if the effective GID of the process is equal to that of file owner, and if the access permission bit of group is SET
  - if other's access permission bit is SET

File's access permission

```
#include<unistd.h>

int access(const char *path, int amode);
```

- check if the process can access a file in the path
- parameter
  - *path* : path name
  - *amode* : access mode for the process to check

amode	meaning
R_OK	*READ permission check*
W_OK	*WRITE permission check*
X_OK	*Execute or Exploration permission check*
F_OK	*File existence check*

- return
  - 0 if OK
  - -1 on error

Permission check example

access-ex.c

```
#include<stdio.h>
#include<unistd.h>

int main(int argc, char *argv[])
{
        if(argc < 2) {
                perror("argument error");
                return 1;
        }
        if(access(argv[1], F_OK) == 0) {
                printf("%s : File Exists\n", argv[1]);
        if(access(argv[1], R_OK) == 0)
                printf("%s : Read\n", argv[1]);
        if(access(argv[1], W_OK) == 0)
                printf("%s : Write\n", argv[1]);
        if(access(argv[1], X_OK) == 0)
                printf("%s : Execute\n", argv[1]);
        }
        else printf("%s : NOT exist\n", argv[1]);
}
```

Default permission change

```
#include<sys/types.h>
#include<sys/stat.h>

mode_t umask(mode_t cmask);
```

- By default
  - by default, a file's permission is set to 0666 (rw-rw-rw)
  - by default, a directory's permission is set to 0777 (rwxrwxrwx)
- umask() changes the default permission
  - set *unmask* which masks off (i.e. not permits)
  - e.g. if umask is 0022, a new file permission is set to 0644
- parameter
  - *cmask* : new umask
- return
  - previous umask

```
$ umask
  0022
$ touch test
$ ls -al test
-rw-r--r--  root root .....
```

umask value

- OR'ed combination of these modes

mode	meaning
S_IRWXU	owner RWX
S_IRUSR	owner R
S_IWUSR	owner W
S_IXUSR	owner X
S_IRWXG	group RWX
S_IRGRP	group R
S_IWGRP	group W
S_IXGRP	group X
S_IRWXO	others RWX
S_IROTH	others R
S_IWOTH	others W
S_IXOTH	others X

File permission change

```
#include<sys/types.h>
#include<sys/stat.h>

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

- change the permission mode of a file in the path
  - file owner or supervisor(root) can do this
- parameters
  - *path* : path name of a file
  - *mode* : the new access permission to change
    - mode is also OR'ed combination of the access modes (see previous)
  - *fd* : file descriptor
- return
  - 0 if OK, -1 on error

Permission change example (1)

chmod-ex.c

```
#include <sys/stat.h>
#include <sys/types.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
        struct stat statbuf;
        if(argc != 2) {
                perror("argument error");
                return 1;
        }
        if (lstat(argv[1], &statbuf) < 0) {
                perror("lstat");
                return 2;
        }
        if (S_ISREG(statbuf.st_mode)) {
                if(chmod(argv[1], (statbuf.st_mode & ~S_IXGRP)) < 0) {
                        perror("chmod");
                        return 3;
                }
        }
        else printf("%s is not requlat file\n", argv[1]);
}
```

Permission change example (2)

- Run & Results

```
$ ls -ld myfile mydir
drwx------ 2 root root 512 Jul 15 15 : 13 mydir
-rwx--x--- 1 root root 0   Jul 15 15 : 12 myfile

$ ./a.out myfile
$ ./a.out mydir
mydir is not a regular file

$ ls -ld myfile mydir
drwx------ 2 root root 512 Jul 15 15 : 13 mydir
-rwx------ 1 root root 0   Jul 15 15 : 12 myfile

$
```

Ownership change

```
#include <unistd.h>
#include <sys/types.h>

int chown(const char *path, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
                                        // doesn't follow links
int fchown(int fd, uid_t owner, gid_t group);
```

- parameters
  - *path* : path name of a file
  - *owner* : owner's UID
  - *group* : group's GID
  - *fd* : file descriptor
- return
  - 0 if OK, -1 on error

Ownership change example (1)

chown-ex.c

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

int main(int argc, char *argv[])
{
        int owner_id, group_id, filedes ;

        if(argc != 5) {
                perror("argument error");
                return 1;
        }
        owner_id = atoi(argv[3]);
        group_id = atoi(argv[4]);
        if (strcmp(argv[1], "chown") == 0) {
                if (chown(argv[2], owner_id, group_id)) {
                        perror("chown");
                        return 2;
                }
                printf("chown %s to %s, %s\n", argv[2], argv[3], argv[4]);
        }
```

Ownership change example (2)

chown-ex.c

```
        else if (strcmp(argv[1], "fchown") == 0) {
                filedes = open(argv[2], O_RDWR);
                if (fchown(filedes, owner_id, group_id)) {
                        perror("chown");
                        return 3;
                }
               printf("fchown %s to %s, %s\n", argv[2], argv[3], argv[4]);
        }
        else if (strcmp(argv[1], "lchown") == 0) {
                if (lchown(argv[2], owner_id, group_id)) {
                        perror("lchown");
                        return 4;
                }
               printf("lchown %s to %s, %s\n", argv[2], argv[3], argv[4]);
        }
}
```

Ownership change example (3)

- *Run & Results*

```
$ ls -l my*
drwxr-xr-x 2      oskernel   oskernel   512  Jul 9 21 : 20   mydir/
-rw-r--r– 1       oskernel   oskernel   0    Jul 7 15 : 37   myfile
lrwxrwxrwx 1      oskernel   oskernel   13   Jul 9 21 : 20   mylink -> ~mydir/myfile1

$ id -u cisc
1703

$id -g cisc
511

$ ./a.out chown myfile 1703 511
chown myfile to 1703, 511

$ ls -l my*
drwxr-xr-x 2   oskernel   oskernel    512    Jul 9 21 : 20   mydir/
-rw-r--r-- 1   cisc          cisc       0    Jul 7 15 : 37   myfile
lrwxrwxrwx 1   oskernel   oskernel     13    Jul 9 21 : 20   mylink -> ~mydir/myfile1
```

HW

- No HW for FileIO(2)