

File I/O

part 2

한양대학교 소프트웨어학부

*Dept. of Division of Computer Science
Hanyang University*



다중 사용자 환경에서의 파일

- 허가와 파일모드
- permission: 상이한 사용자들이 파일에 접근할 권한
- 파일의 소유자가 permission 결정
- 사용자 유형
 - 파일의 소유자
 - 파일에 연관된 그룹과 같은 그룹에 속하는 사용자
 - 그룹에 속하지 않는 사용자
- 파일 permission의 세 가지 그룹
 - 파일을 읽기 (r)
 - 파일에 쓰기 (w)
 - 파일의 수행 (x)

다중 사용자 환경에서의 파일

- 파일 permission 변경

팔진수 값	상징형 모드	의 미
0400	S_IRUSR	소유자에게 읽기를 허가함
0200	S_IWUSR	소유자에게 쓰기를 허가함
0100	S_IXUSR	소유자에게 파일의 수행을 허가함
0040	S_IRGRP	그룹에 대해 읽기를 허가함
0020	S_IWGRP	그룹에 대해 쓰기를 허가함
0010	S_IXGRP	그룹에 대해 파일의 수행을 허가함
0004	S_IROTH	다른 모든 사용자에게 읽기를 허가함
0002	S_IWOTH	다른 모든 사용자에게 쓰기를 허가함
0001	S_IXOTH	다른 모든 사용자에게 파일 수행을 허가함

System Call – umask

사용법

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

mode_t umask(mode_t newmask);
```

umask	파일 허용권한을 위한 mask를 지정
newmask	8진수 형식, 다음 페이지 참조
Return value	항상 성공하며 이전에 설정된 mask값을 리턴

ex) umask 값을 022로 했을 경우 mode를 0666으로 했다면
 -> $0666 \& -022 = 0644 = \text{rw-r--r--}$ 권한을 가지는 파일 생성

newmask

자리수	값	의미
1	0	모든 사용자 권한이 허용된다
	4	사용자 읽기 권한이 금지된다
	2	사용자 쓰기 권한이 금지된다
	1	사용자 실행 권한이 금지된다
2	0	모든 그룹 권한이 허용된다
	4	그룹 읽기 권한이 금지된다
	2	그룹 쓰기 권한이 금지된다
	1	그룹 실행 권한이 금지된다
3	0	모든 기타 권한이 허용된다
	4	기타 읽기 권한이 금지된다
	2	기타 쓰기 권한이 금지된다
	1	기타 실행 권한이 금지된다

System Call – access

사용법

```
#include <unistd.h>

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

access

지정된 파일에 대해서 권한을 가지고 있는지 체크

int amode

R_OK : 읽기 여부 체크

W_OK : 쓰기 여부 체크

X_OK : 실행 여부 체크

F_OK : 파일의 존재 유무

Return value

[성공 시: 0]

[실패 시: -1]

Example - access

```

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4
5 int main(int argc, char **argv)
6 {
7     if (access(argv[1], F_OK) != 0)
8         perror("File not exist");
9     else
10        printf("File exist\n");
11
12     return 0;
13 }
```

```
[TA3@localhost lab6]$ ./access test.txt 존재하는 파일
File exist
[TA3@localhost lab6]$ ./access test.dox 존재하는 않는 파일
File not exist: No such file or directory
```



System Call – chmod

사용법

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

int chmod(const char *pathname, mode_t newmode);
```

chmod

파일의 모드를 newmode 모드로 변경한다.

mode_t newmode

새로운 파일 접근 모드 (팔진수, 상징형 모드)

Return value

[성공 시: 0]

[실패 시: -1]

Example - chmod

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5
6 int main(int argc, char **argv)
7 {
8     if (chmod(argv[1], S_IRUSR | S_IWUSR) < 0) {
9         perror("chmod error");
10        exit(0);
11    }
12 }
```

```
[TA3@localhost lab6]$ ls -l test.txt
-rw-r--r-- 1 TA3 TA3 10 2017-10-27 16:37 test.txt
[TA3@localhost lab6]$ ./chmod test.txt
[TA3@localhost lab6]$ ls -l test.txt
-rw----- 1 TA3 TA3 10 2017-10-27 16:37 test.txt
[TA3@localhost lab6]$
```



System Call – link

사용법

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

```
int link(const char *orginal_path, const char *new_path);
int unlink(const char *pathname);
```

link

존재하는 파일에 대해서 새로운 연결(하드 링크)을 만듬
만약 **newpath** 가 이미 존재하고 있다면, 덮어쓰지 않음

unlink

지명된 링크를 제거하고, 파일의 링크카운트를 하나 줄임

char *new_path :새로운 링크 또는 이름

Return value

[성공 시: 0]

[실패 시: -1]

System Call – symlink

사용법

```
#include <unistd.h>

int symlink(const char *oldpath, const char *newpath);
```

symlink

파일에 대한 심볼릭 링크를 만든다.

심볼릭 링크 **newpath**가 이미 존재하면 덮어쓰지 않음

Return value

[성공 시: 0]

[실패 시: -1]

주의할 점

oldpath 파일 삭제 : 심볼형 링크 파일 사용불가

Example – link (1/2)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char* argv[])
6 {
7     if (argc != 4) {
8         printf("Usage: ./link [original file] [copy file (hard)] [copy file(symbolic)]");
9         return -1;
10    }
11
12    if (access(argv[1], F_OK) != 0) {
13        printf("original file not exists\n");
14        return -1;
15    }
16
17    if (access(argv[2], F_OK) == 0) {
18        printf("file already exists\n");
19        return -1;
20    }
21
22    if (access(argv[3], F_OK) == 0) {
23        printf("file already exists\n");
24        return -1;
25    }
26
27    if (link(argv[1], argv[2]) == -1)
28        perror("hard link error");
29    if (symlink(argv[1], argv[3]) == -1)
30        perror("symbolic link error");
31
32    return 0;
33 }
```

Example – link (2/2)

```
[TA3@localhost lab6]$ ./link test.txt test.hardlink test.symlin
[TA3@localhost lab6]$ ls -li
합계 44
41422698 -rwxr-xr-x 1 TA3 TA3 6730 2017-10-27 16:33 access
41422702 -rw-r--r-- 1 TA3 TA3 203 2017-10-27 16:38 access.c
41422699 -rwxr-xr-x 1 TA3 TA3 6712 2017-10-27 17:04 chmod
41422704 -rw-r--r-- 1 TA3 TA3 208 2017-10-27 17:09 chmod.c
41422700 -rwxr-xr-x 1 TA3 TA3 7443 2017-10-27 17:16 link
41422705 -rw-r--r-- 1 TA3 TA3 629 2017-10-27 17:16 link.c
41422701 -rw----- 2 TA3 TA3 10 2017-10-27 16:37 test.hardlink
41422703 lrwxrwxrwx 1 TA3 TA3 8 2017-10-27 17:18 test.symlin -> test.txt
41422701 -rw----- 2 TA3 TA3 10 2017-10-27 16:37 test.txt
[TA3@localhost lab6]$
```



System Call – rename

사용법

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

```
int rename(const char *oldpathname,const char *newpathname)
```

rename

파일의 이름이나 위치를 변경

Return value

[성공 시: 0]

[실패 시: -1]

System Call – stat & fstat

사용법

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

int stat(const char *pathname , struct stat *buf);
int fstat(int filedes , struct stat *buf);
```

stat, fstat

파일의 상태를 얻어온다.

성공적으로 수행될 경우 파일의 정보를 stat구조체에 복사
struct stat *buf

stat 구조를 가리키는 포인터

Return value

[성공 시: 0]

[실패 시: -1]

struct stat의 구성

```
struct stat {
    dev_t          st_dev;      /* 파일이 들어있는 논리적 장치를 기술 */
    ino_t          st_ino;      /* 파일의 inode번호 */
    mode_t         st_mode;     /* 파일 모드 */
    nlink_t        st_nlink;    /* 하드링크 심볼 링크의 수 */
    uid_t          st_uid;      /* 파일 사용자 식별번호 */
    gid_t          st_gid;      /* 파일 그룹 식별번호 */
    dev_t          st_rdev;     /* 논리적 장치의 type */
    off_t          st_size;     /* 파일의 크기(byte로 표시) */
    blksize_t      st_blksize;   /* 파일 시스템 고유의 I/O 블록 크기 */
    blkcnt_t       st_blocks;    /* 파일에 할당된 물리적 파일 시스템 블록의 수 */
    time_t         st_atime;    /* 파일의 자료가 마지막으로 읽혔던 시간 */
    time_t         st_mtime;    /* 파일의 자료가 마지막으로 변경된 시간 */
    time_t         st_ctime;    /* stat 구조체가 변경된 시간*/
};
```

Example – stat (1/3)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <pwd.h>
7 #include <grp.h>
8
9 int main(int argc, char* argv[])
10 {
11     char *file_name;
12     struct stat file_info;
13     struct passwd *my_passwd;
14     struct group *my_group;
15
16     mode_t file_mode;
17
18     if (argc != 2) {
19         printf("Usage: ./file_info [file name]\n");
20         exit(-1);
21     }
22
23     file_name = argv[1];
24
25     if (stat(file_name, &file_info) == -1) {
26         perror("Error: ");
27         exit(-1);
28     }
29
30     file_mode = file_info.st_mode;
31     printf("file name: %s\n", file_name);
32     printf("=====\\n");
33     printf("file type: ");
```

Example – stat (2/3)

```
35     if (S_ISREG(file_mode))
36         printf("regular file\n");
37
38     else if (S_ISLNK(file_mode))
39         printf("symbolic link\n");
40
41     else if (S_ISDIR(file_mode))
42         printf("directory\n");
43
44     else if (S_ISCHR(file_mode))
45         printf("character device\n");
46
47     else if (S_ISBLK(file_mode))
48         printf("block device\n");
49
50     else if (S_ISFIFO(file_mode))
51         printf("FIFO\n");
52
53     else if (S_ISSOCK(file_mode))
54         printf("socket\n");
55
56     my_passwd = getpwuid(file_info.st_uid);
57     my_group = getgrgid(file_info.st_gid);
58     printf("OWNER: %s\n", my_passwd->pw_name);
59     printf("GROUP: %s\n", my_group->gr_name);
60     printf("FILE SIZE IS: %d\n", file_info.st_size);
61     printf("LAST READ TIME: %d\n", file_info.st_atime);
62     printf("LAST MODIFY TIME: %d\n", file_info.st_mtime);
63     printf("NUMBER OF HARD/SYMBOLIC LINK: %d\n", file_info.st_nlink);
64
65     return 0;
66 }
```

Example – stat (3/3)

```
[TA3@localhost lab6]$ ./stat test.txt
file name: test.txt
=====
file type: regular file
OWNER: TA3
GROUP: TA3
FILE SIZE IS: 10
LAST READ TIME: 1509089839
LAST MODIFY TIME: 1509089839
NUMBER OF HARD/SYMBOLIC LINK: 2
[TA3@localhost lab6]$
```

Q & A

- Thank you :)