4장 파일의 관리

- 서론
- 예제 프로그램
- 함수
  - umask
  - access
  - chmod, fchmod
  - chown, fchown
  - link
  - rename
  - symlink
  - readlink
  - stat, fstat

파일의 정보를 관리하는 시스템 호출/표준 라이브러리 함수

함수	의미
umask	파일 생성 마스크를 설정한다.
access	파일에 대한 사용자의 접근 권한을 확인한다.
chmod/fchmod	파일에 대한 접근 권한을 변경한다.
chown/fchown	파일의 소유주와 그룹을 변경한다.
link	파일의 새로운 이름을 생성한다. (hard-link)
rename	파일의 이름이나 위치를 변경한다.
symlink	파일의 새로운 이름을 생성한다. (soft-link, symbolic link)
readlink	심볼형 링크의 값(실제 내용)을 읽어온다.
stat/fstat	파일의 상태 정보를 가져온다.

【예제】 예제 프로그램(1/3)


```
01 #include <unistd.h>
02 #include <fcntl.h>
03 #include <sys/types.h>
04 #include <sys/stat.h>
05 #include <stdio.h>

06 int main()
07 {
08     char *originalname = "test.txt";
09     char *hardfilename = "test.txt.hard";
10     char *softfilename = "test.txt.soft";
11
12     int filedes, retval;
13     mode_t oldmask;
14     char buffer[1024];
15     int  nread;
16     struct stat finfo;
```

14 버퍼

16 파일의 정보를 담기 위한 변수

【예제】 예제 프로그램(2/3)


```
17    oldmask = umask(0377);
18
19  filedes = open(originalname, O_RDWR |
    O_CREAT, 0755);
20    close(filedes);
21
22    if((retval = access(originalname, W_OK)) == -1)
23    {
24        printf("%s is not writable₩n", originalname);
25        chmod(originalname, 0644);
26    }
27
28    link(originalname, hardfilename);
29    symlink(originalname, softfilename);
30
31    rename(hardfilename, "newname.txt");
```

17 파일 생성시 적용되는 접근 권한 중 일부를 제한한다

20 파일을 생성한 후 바로 닫는다

22 읽기 권한을 가지고 있는지 확인한다.

25 지정한 파일의 권한을 변경한다.

28 하드 링크를 생성한다.

29 소프트 링크를 생성한다.

31 파일의 이름을 변경한다

표준입력 스트림

【예제】 예제 프로그램(3/3)


```
32     nread = readlink(softfilename, buffer, 1024);
33     write(1, buffer, nread);
34
35     stat(originalname, &finfo);
36     printf("₩n%s₩n", originalname);
37     printf("File mode   : %o₩n", finfo.st_mode);
38     printf("Files size  : %d₩n", finfo.st_size);
39     printf("Num of blocks  : %d₩n", finfo.st_blocks);
40 }
```

32 소프트 링크 파일의 실제 내용을 읽어서 표준 출력한다.

32 파일의 아이노드 정보를 가져와 그 일부를 표준 출력한다.

표준입력 스트림

실행 결과

```
$ ls -l test.*
찾지 못함
$ ex04-01
test.txt is not writable
test.txt
test.txt
File mode  : 100644
Files size  : 0
Num of blocks  : 0
$ ls -l test.*
-rw-r--r--   2 kimyh    graduate       0 Sep 26 15:46 test.txt
lrwxrwxrwx   1 kimyh    graduate       8 Sep 26 15:46 test.txt.soft -> test
.txt
$
```

새로운 파일을 생성할 때 적용하는 접근 권한 중 일부를 제한한다

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

mode_t umask(mode_t mask);
```

mask	파일에 대한 접근 권한으로 mask로 지정한 항목은 새로운 파일 생성 시 적용되지 않는다.
반환값	umask의 실행은 항상 성공하며 기존의 mask 값을 반환한다.

- **umask로 설정한 값은 파일의 접근 권한을 의미한다.**
- **새로운 파일을 생성할 때 umask로 등록한 권한은 생성되는 파일에 적용되지 않는다.**
- **umask의 설정은 프로세스가 실행 중인 동안만 유지되며 프로세스가 종료되면 원래의 값으로 복원된다.**

```
mode_t oldmask;
oldmask = umask(037);
filedes = open("data.txt", O_CREAT, 0777);    /* 0777은 8진수 777을 의미 */
```

파일 생성 시 적용하는 초기 접근 권한	(0777) 111 111 111
umask로 제한한 접근 권한	& (~0037) 111 100 000
실제로 적용되는 초기 접근 권한	(0740) 111 100 000

- **파일을 생성하면서 설정한 초기 접근 권한 중에서 umask로 등록한 값은 제외된다.**
- **파일을 생성할 때 허용되지 않은 접근 권한을 실수로 선택하는 것을 방지할 수 있다.**

【예제 4-2】 ex04-02.c


```
01 /* 예제 프로그램 : ex04-02.c */
02 #include <unistd.h>
03 #include <sys/types.h>
04 #include <sys/stat.h>
05 int main()
06 {
07     int filedes;
08     mode_t oldmask;
09
10     oldmask = umask(023);
11     filedes = open("test.txt", O_CREAT, 0777);
12     close(filedes);
13 }
```

표준입력 스트림

```
$ ls -l test.dat
-rwxr-xr--   1 usp      student        0  3월 22일  21:13 test.txt*
```

지정한 파일에 대해서 특정 접근 권한을 가지고 있는지를 검사한다.

```
#include <unistd.h>

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

pathname	파일에 대한 경로이름이다.
mode	검사하려는 접근 권한으로 R_OK, W_OK, X_OK, F_OK를 사용할 수 있다.
반환값	access 호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- 파일의 접근 권한은 3가지 유형의 사용자에 대해서 설정되어 있다.
  - 파일의 소유주, 동일 그룹의 사용자, 기타 사용자
- access를 사용하여 검사하는 접근 권한은 access를 실행하는 프로세스와 지정한 파일 사이의 관계에 따라 다르다.

【예제 4-4】 ex04-04.c


```
01 /* 예제 프로그램 : ex04-04.c */
02 #include <stdio.h>
03 #include <stdlib.h>
04 #include <unistd.h>
05
06 int main()
07 {
08     char *filename = "test.txt";
09
10     if ( access(filename, R_OK) == -1 ) {
11         fprintf( stderr, "User cannot read file %s ₩n",
           filename);
12         exit(1);
13     }
14     printf("%s readable, proceeding ₩n", filename);
15
16     /* rest of program ... */
17 }
```

표준입력 스트림

예제 프로그램 : ex04-04.c 실행 결과

```
$ ls -l test.txt
-rw-r-----   1 usp      student        5  3월 22일  21:16 test.txt
$ id
uid=2040(usp) gid=200(student1)
$ ex04-04
test.txt readable, proceeding
...
$ id
uid=2045(honggildong) gid=201(student2)
$ ex04-04
User cannot read file test.txt
```

매크로 상수의 의미

이름	의미
R_OK	읽기 권한을 검사한다.
W_OK	쓰기 권한을 검사한다.
X_OF	실행 권한을 검사한다. (디렉터리일 경우 탐색 권한을 검사한다.)
F_OK	대상 파일이 존재하는지 검사한다.

파일의 접근 권한을 변경한다

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

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

path	파일에 대한 경로 이름이다.
filedes	개방된 파일의 파일 기술자이다.
mode	파일에 새롭게 적용하려는 접근 권한이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **chmod는 개방하지 않은 파일을 경로명으로 지정한다.**
- **fchmod는 개방한 파일을 파일 기술자로 지정한다.**

접근 권한의 표현

- 8진수 값
- 미리 정의된 매크로 상수

8진수 값	상수이름	의미
0400	S_IRUSR	소유자에 대한 읽기 권한
0200	S_IWUSR	소유자에 대한 쓰기 권한
0100	S_IXUSR	소유자에 대한 실행 권한
0040	S_IRGRP	그룹 사용자에 대한 읽기 권한
0020	S_IWGRP	그룹 사용자에 대한 쓰기 권한
0010	S_IXGRP	그룹 사용자에 대한 실행 권한
0004	S_IROTH	기타 사용자에 대한 읽기 권한
0002	S_IWOTH	기타 사용자에 대한 쓰기 권한
0001	S_IXOTH	기타 사용자에 대한 실행 권한

매크로 상수 사용

- 비트별 OR 연산자(|)를 사용한다

```
mode_t mode;
mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
chmod("test.txt", mode);
```

- 위는 아래와 같다.

```
chmod("test.txt", 0644);
```

- 덧셈 연산자(+)를 잘못 사용한 경우

```
mode = S_IRUSR | S_IXUR;
mode = mode | S_IRUSR;   /*  실수로 S_IWUSR 대신에 S_IRUSR를 적음 */
…
mode = S_IRUSR + S_IXUR;
mode = mode + S_IRUR;
```

※잘못된 결과를 8진수로 표현해보고 실제로 어떻게 설정되는지 확인해보라.

【예제 4-5】 ex04-05.c


```
01 /* 예제 프로그램 : ex04-05.c */
02 #include <sys/types.h>
03 #include <sys/stat.h>
04
05 int main()
05 {
06     mode_t mode1, mode2;
07     mode1 = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
08     mode2 = 0644;
09
10     if(chmod("test1.txt", mode1) == -1)
11         exit(1);
12     if(chmode(test2.txt", mode2) == -1)
13         exit(1);
14
15     printf("rest of program ... ₩n");
16 }
```

표준입력 스트림

예제 프로그램 : ex04-05.c 실행 결과

```
$ ls -l test1.txt test2.txt
-r--------   1 usp      student        5  3월 22일  21:16 test1.txt
-r--------   1 usp      student        5  3월 22일  21:16 test2.txt
$ ex04-05
rest of program ...
$ ls -l test1.txt test2.txt
-rw-r--r--   1 usp      student        5  3월 22일  21:16 test1.txt
-rw-r--r--   1 usp      student        5  3월 22일  21:16 test2.txt
```

결과지정한 경로의 파일이나 이미 개방된 파일의 소유주를 변경한다.

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

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

path	파일에 대한 경로 이름이다.
fd	개방된 파일의 파일 기술자이다.
owner	새로운 소유주의 사용자 식별 번호(UID)이다.
group	새로운 소유주의 그룹 식별 번호(GID)이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

파일의 소유주를 변경하는 작업은 시스템 관리자만 수행할 수 있다.

【예제 4-6】 ex04-06.c


```
01 #include <sys/types.h>
02 #include <unistd.h>
03
04 int main()
05 {
06     if(chown("test.txt", 2045, 200) == -1)
07         exit(1);
08     printf("rest of program ...₩n");
09 }
```

표준입력 스트림

- **id 명령을 사용하여 사용자의 식별 번호와 그룹 식별 번호를 알 수 있다.**

```
$ id
uid=2040(usp) gid=200(student)
$ id honggildong
uid=2045(honggildong) gid=200(student)
```

지정한 파일에 대한 하드 링크(link)와 소프트 링크(symlink)를 생성한다

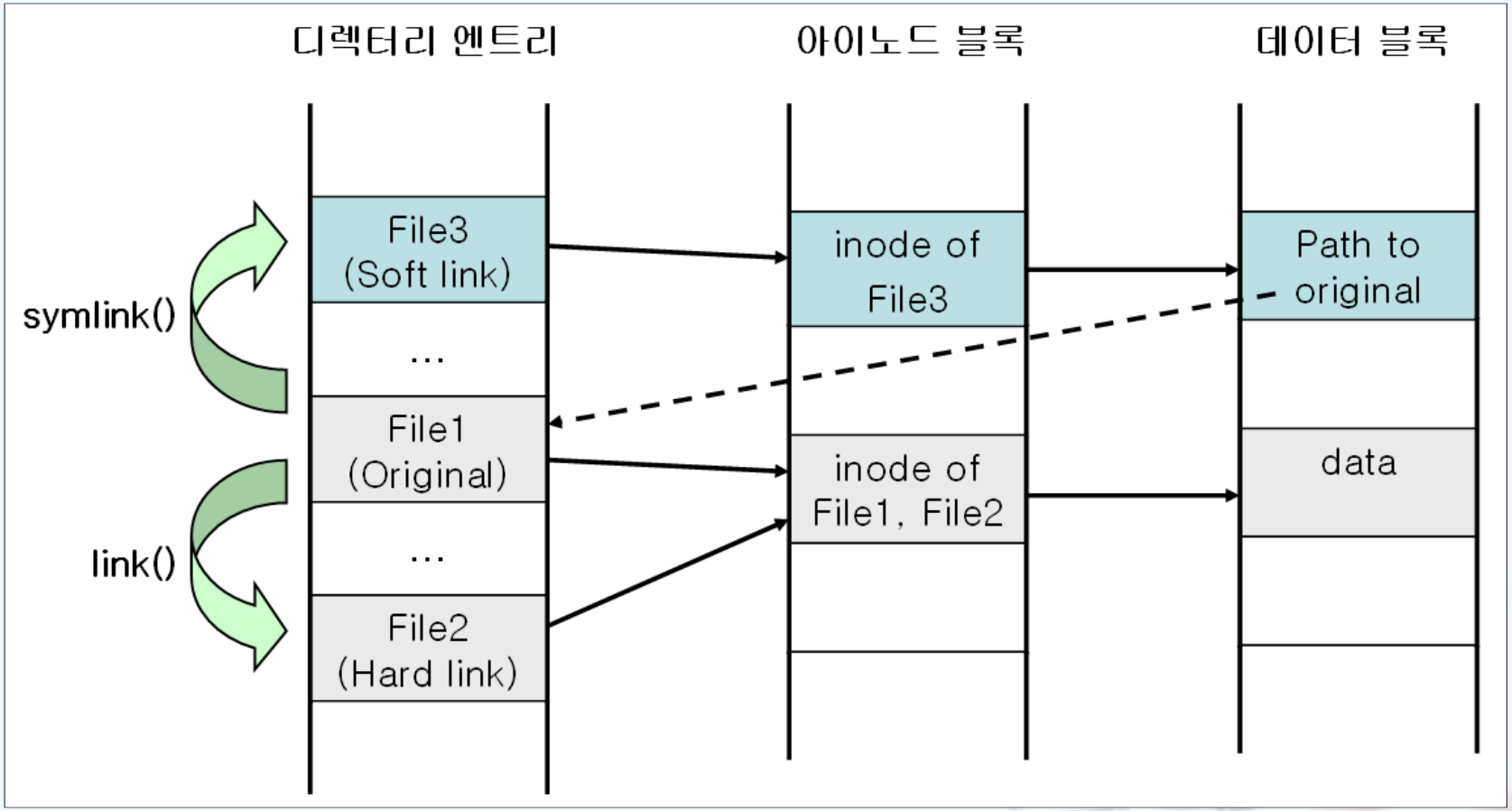
```
#include <unistd.h>

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

oldpath	원본 파일의 경로 이름이다.
newpath	하드 링크/소프트 링크의 경로 이름이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- 하드 링크는 지정한 파일에 새로운 이름을 하나 더 추가해주는 것이다.
- 소프트 링크는 MS Windows의 바로 가기 아이콘처럼 지정한 파일을 가리키는 새로운 특수한 파일을 생성한다.

하드 링크와 소프트 링크의 비교



【예제 4-7,8】 ex04-07,08.c

예제 프로그램 : ex04-07.c

```
#include <unistd.h>

int main(int argc, char *argv[])
{
    if(link(argv[1], argv[2]))
        printf("hard-link failed₩n");
}
```

예제 프로그램 : ex04-08.c

```
#include <unistd.h>

main(int argc, char *argv[])
{
    if(symlink(argv[1], argv[2]))
        printf("soft-link failed₩n");
}
```

```
$ ls -l test.txt
-rw-r--r--   1 usp      student        5 Sep 26 14:16 test.txt
$ ex04-07 test.txt test.txt.hard-link
$ ex04-08 test.txt test.txt.soft-link
$ ls -l test.txt*
-rw-r--r--   2 usp      student       17 Sep 26 14:16 test.txt
-rw-r--r--   2 usp      student       17 Sep 26 14:16 test.txt.hard-link
lrwxrwxrwx   1 usp      student        8 Sep 26 14:16 test.txt.soft-link ->
test.txt
$ chmod 600 test.txt.hard-link
$ ls -l test.txt*
-rw-------   2 usp      student       17 Sep 26 14:16 test.txt
-rw-------   2 usp      student       17 Sep 26 14:16 test.txt.hard-link
lrwxrwxrwx   1 usp      student        8 Sep 26 14:16 test.txt.soft-link ->
test.txt
$ chmod 644 test.txt.soft-link
$ ls -l test.txt*
-rw-r--r--   2 usp      student       17 Sep 26 14:16 test.txt
-rw-r--r--   2 usp      student       17 Sep 26 14:16 test.txt.hard-link
lrwxrwxrwx   1 usp      student        8 Sep 26 14:16 test.txt.soft-link ->
test.txt
$
```

소프트 링크 파일의 실제 내용을 읽는다

```
#include <unistd.h>

int readlink(const char *path, char *buf, size_t bufsize);
```

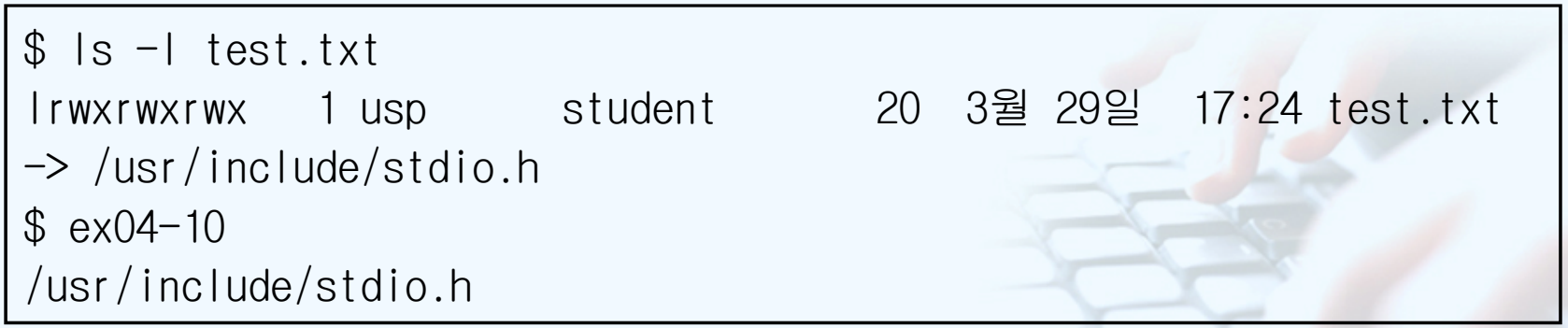
path	소프트 링크에 대한 경로 이름이다.
buf	소프트 링크의 실제 내용을 담을 공간이다.
bufsize	buf의 크기이다.
반환값	읽기가 성공하면 buf에 저장한 바이트 수를 반환하며 실패할 경우 -1을 반환한다.

- **소프트 링크는 원본 파일의 경로 이름을 저장하고 있고, readlink는 이 경로 이름을 읽어온다.**

【예제 4-9】 ex04-09.c


```
01 #include <stdio.h>
02 #include <unistd.h>
03
04 int main()
05 {
06         char buffer[1024];
07         int nread
08         nread = readlink("test.txt", buffer, 1024);
09         write(1, buffer, nread);
10 }
```

표준입력 스트림



```
$ ls -l test.txt
lrwxrwxrwx    1 usp      student        20  3월 29일  17:24 test.txt
-> /usr/include/stdio.h
$ ex04-10
/usr/include/stdio.h
```

지정한 경로의 파일 이름을 새로운 이름으로 변경한다.

```
#include <stdio.h>

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

oldpath	이름을 바꾸려는 파일의 경로 이름이다.
newpath	파일의 새로운 이름이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **oldpath와 newpath가 동일하게 지정될 경우 성공으로 간주한다.**
- **newpath로 지정한 파일이 이미 존재할 경우 이를 삭제한 후에 이름을 변경한다.**
- **newpath로 지정한 파일이 이미 존재하며 만약 이 파일이 디렉터리라면 디렉터리가 비어있으면 호출이 성공하고 비어있지 않으면 실패한다.**

【예제 4-10】 ex04-10.c

```
01 #include <stdio.h>
02
03 int main(int argc, char *argv[])
04 {
05     if(argc != 3)
06         exit(1);
07     if(rename(argv[1], argv[2]) == 0)
08         printf("성공!₩n");
09     else
10         printf("실패!₩n");
11 }
```

표준입력 스트림

지정한 파일에 대한 상세한 정보를 알아온다

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

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

filename	파일의 경로 이름이다.
filedes	개방된 파일의 파일 기술자이다.
buf	파일의 정보를 담기 위한 struct stat 타입 구조체의 포인터이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **지정한 파일의 아이노드 블록에 저장된 정보를 읽어온다.**

struct stat 구조체

```
struct stat {
    dev_t           st_dev
    ino_t           st_ino
    mode_t          st_mode
    nlink_t         st_nlink
    uid_t           st_uid
    gid_t           st_gid
    dev_t           st_rdev
    off_t           st_size
    timestruc_t     st_atim
    timestruc_t     st_mtim
    timestruc_t     st_ctim
    blksize_t       st_blksize
    blkcnt_t        st_blocks
    char            st_fstype[_ST_FSTYPSZ];
};
```

아이노드 블록에 저장된 각 정보는 데이터 형식이 다르므로 구조체형으로 결과를 구한다.

struct stat 구조체 멤버 변수의 의미

멤버 변수		의미
dev_t	st_dev	디렉터리 엔트리를 포함하고 있는 논리적인 디바이스의 식별번호
ino_t	st_ino	아이노드 블록의 번호
mode_t	st_mode	파일의 접근 권한
nlink_t	st_nlink	하드 링크 계수(카운터)
uid_t	st_uid	파일 소유자의 사용자 식별번호
gid_t	st_gid	파일 소유자의 그룹 식별번호
off_t	st_size	바이트 단위의 파일의 크기
timestruc_t	st_atim	마지막으로 파일이 사용된 시간
timestruc_t	st_mtim	마지막으로 데이터 블록이 수정된 시간
timestruc_t	st_ctim	마지막으로 파일의 상태가 수정된 시간
blkcnt_t	st_blocks	파일 저장을 위해 할당된 데이터 블록의 수
char	st_fstype[_ST_FSTYPSZ];	파일 시스템의 종류

【예제 4-11】 ex04-11.c


```
01 #include <stdio.h>
02 #include <string.h>
03 #include <sys/types.h>
04 #include <sys/stat.h>
05
06 int main(int argc, char *argv[])
07 {
08     struct stat finfo
09     char fname[1024];
10
11     if(argc > 1)
12         strcpy(fname, argv[1]);
13     else
14         strcpy(fname, argv[0]);
15
16     if(stat(fname, &finfo) == -1) {
17         fprintf(stderr, "Couldn't stat %s ₩n", fname);
18         exit(1);
19     }
```

표준입력 스트림

```
20      printf("%s ₩n", fname);
21      printf("ID of device: %d ₩n", finfo.st_dev);
22      printf("Inode number: %d ₩n", finfo.st_ino);
23      printf("File mode   : %o ₩n", finfo.st_mode);
24      printf("Num of links: %d ₩n", finfo.st_nlink);
25      printf("User ID     : %d ₩n", finfo.st_uid);
26      printf("Group ID    : %d ₩n", finfo.st_gid);
27      printf("Files size  : %d ₩n", finfo.st_size);
28      printf("Last access time : %u ₩n", finfo.st_atim);
29      printf("Last modify time : %u ₩n", finfo.st_mtim);
30      printf("Last stat change : %u ₩n", finfo.st_ctim);
31      printf("I/O Block size : %d ₩n", finfo.st_blksize);
32      printf("Num of blocks  : %d ₩n", finfo.st_blocks);
33      printf("File system : %s ₩n", finfo.st_fstype);
34  }
```

표준입력 스트림