Directories, file systems and special files

Suntae Hwang

Kookmin University

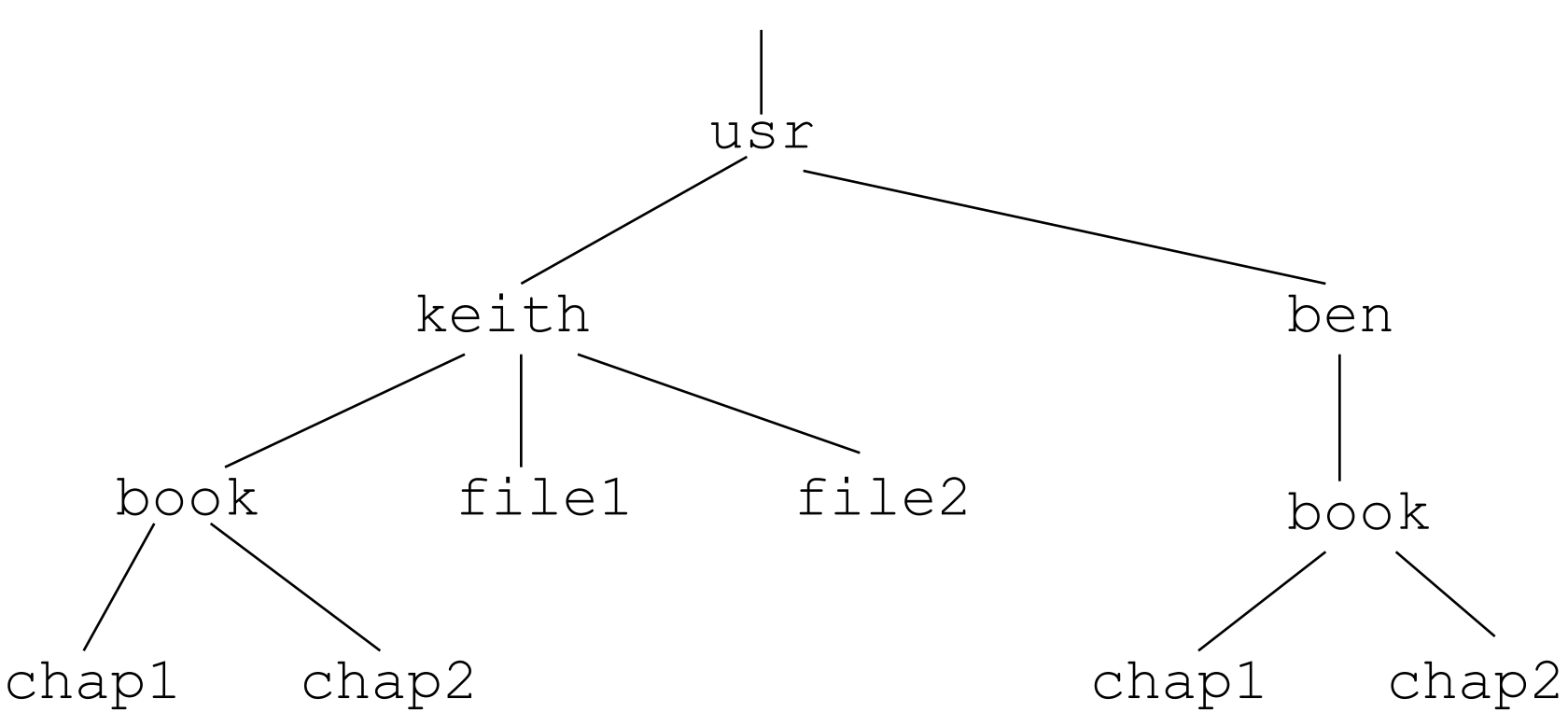
1 Introduction

- Directories
  - Repositories for file names
- File systems
  - Collections of directories and files
- Special files
  - UNIX extends the file concept to cover the peripheral devices connected to a system.

2 Directories: the user view

- 파일 이름의 집합으로서, 화일들을 논리적으로관련된 그룹으로 나누는 수단을 제공
- Home directory
- Subdirectory
- Inverted, hierarchical, tree-like structure

- Current working directory
  - 초기값은 home directory
  - cd 명령으로 변경
  - pwd : print working directory
  - 상대 경로를 찾을 때 찾기 시작하는 디렉토리




```
$cd /usr/keith
$pwd
/usr/keith
$cat book/chap1
$cat /usr/keith/book/chap1
$cat file1
$cat /usr/keith/file1
```

3 The implementation of a directory

- 디렉토리도 화일이다
  - open 으로 읽기를 위해 개방 가능
  - read, lseek, fstat, close 사용가능

- 그러나 제한이 있다.
  - creat/open 으로 생성되지 않는다
  - open에서 O_WRONLY, O_RDWR 이 설정되면 오류 발생
    errno: EISDIR

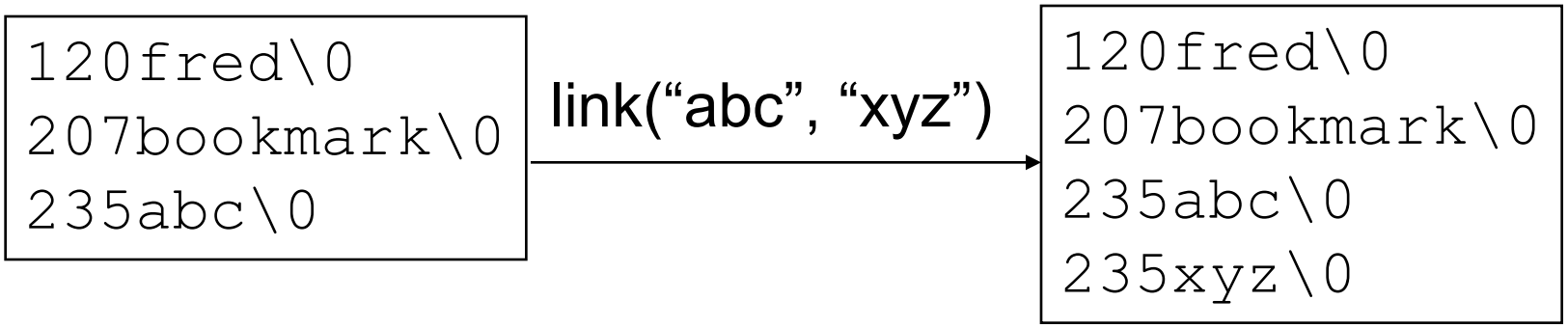
3 The implementation of a directory

- Directory entry
  - File or subdirectory
  - <inode number, file's name>
  - File name 은 고정 크기였다가 Berkeley UNIX 이 후에 가변 크기로 됨 → 시스템 의존적
  - inode: 한 파일을 유일하게 식별
  - stat, fstat의 대부분의 정보는 inode 구조에서 얻는다

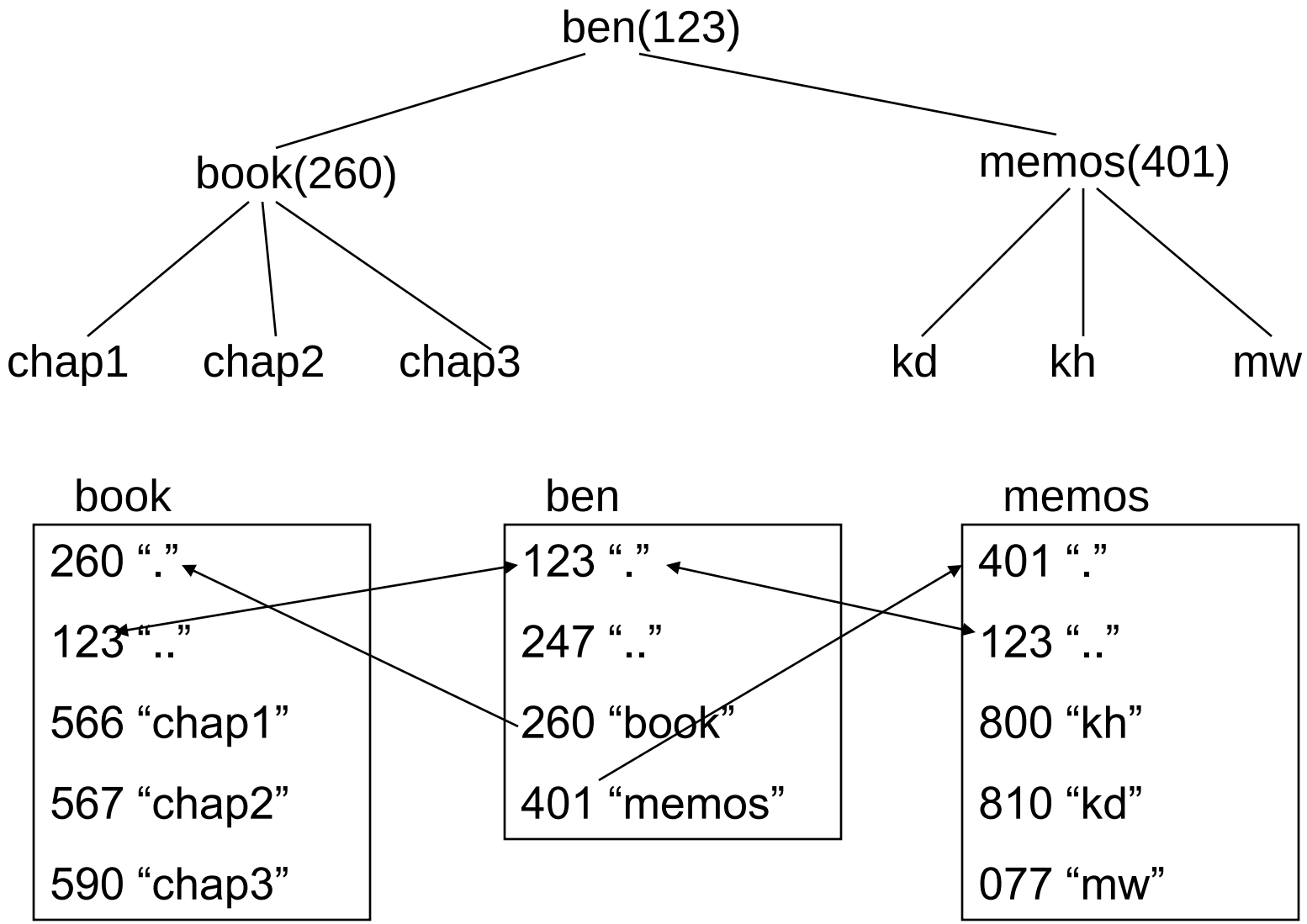
```
120fred\0
207bookmark\0
235abc\0
```

link **and** unlink **revisited**

- link
  - 새로운 파일 이름에 원래의 파일과 같은 inode number를 가지는 새로운 directory slot을 만든다.



Dot and double-dot



Directory permissions

- Read
  - 디렉토리 내에 있는 파일이나 부디렉토리의 이름을 리스트할 수 있다
  - 각 파일 자체의 정보를 읽는 것과는 별개이다.
- Write
  - 디렉토리 내의 파일을 제거하거나 새로운 파일을 만들 수 있게 한다.
  - 기존 파일의 내용을 변경하는 것을 뜻하지는 않는다. 그러나 기존 파일을 제거하고 같은 이름의 새로운 파일을 만들 수 는 있다.
- Execute(or search)
  - cd/chdir 등을 통해서 사용자가 디렉토리 내부로 들어가는 것을 허용한다.
  - 파일을 열거나 수행하기 위해서 사용자가 파일의 절대 경로에 명세된 각 디렉토리에 대해 탐색허가를 가져야만 한다.

```
[sthwang@linux LinuxProgramming]$ ls -l
합계 4
-rw-r-----  1 sthwang professor    0  9월 13 11:24 dir_t
drwxr-x---  2 sthwang professor 4096  9월 13 11:24 expenses
-rw-r-----  1 sthwang professor    0  9월 13 11:24 new
-rw-r-----  1 sthwang professor    0  9월 13 11:24 pwd_text
-rw-r-----  1 sthwang professor    0  9월 13 11:24 test.c
[sthwang@linux LinuxProgramming]$
[sthwang@linux LinuxProgramming]$ ls -ld
drwxr-xr-x  3 sthwang professor 4096  9월 13 11:24 .
[sthwang@linux LinuxProgramming]$
[sthwang@linux LinuxProgramming]$
```

4 Programming with directories

- dirent
  - <dirent.h>

    ino_t d_ino; /* inode number */

    char d_name[]; /* filename, null terminated */
  - d_ino 가 0인 것은 빈 slot을 나타낸다

- Creating and removing directories

  int mkdir(const char *pathname, mode_t mode);

  int rmdir(const char *pathname);

```
int retval
retval = mkdir("/tmp/dir1", 0777);
```

- Opening and closing directories

  DIR *opendir(cont char *dirname);

  Int closedir(DIR *dirptr);

- Reading directories

  struct dirent *readdir(DIR *dirptr);

  void rewinddir(DIR *dirptr);

Example: my_double_ls

```
#include <dirent.h>

int my_double_ls(const char *name)
{
    struct dirent *d;
    DIR *dp;

    /* open the directory and check for failure */
    if((dp=opendir(name)) == NULL)
        return (-1);

    /* continue looping through the directory,
     * printing out the directory entry name as long
     * as the inode number is valid
     */
    while(d = readdir(dp))    {
        if(d->d_ino != 0)
            printf("%s\n", d->d_name);
    }
```

```
/* now go back to the beginning of the directory ... */
    rewinddir(dp);

    /* ... and print out the directory again */
    while(d = readdir(dp))
    {
        if(d->d_ino != 0)
            printf("%s\n",d->d_name);
    }

    closedir(dp);
    return (0);
}
```

```
.
..
fred
bookmark
abc
.
..
fred
bookmark
abc
```

Example: find_entry

```
#include <stdio.h>   /* for NULL */
#include <dirent.h>
#include <string.h>  /* for string function */

int match(const char *, const char *);

char *find_entry(char *dirname, char *suffix, int cont)
{
  static DIR *dp=NULL;
  struct dirent *d;

  if(dp == NULL || cont == 0) {
    if(dp != NULL)
      closedir(dp);
    if((dp = opendir(dirname)) == NULL)
      return (NULL);
    }
```

```
while(d = readdir(dp))  {
    if(d->d_ino == 0)
      continue;
    if(match(d->d_name, suffix))
      return (d->d_name);
    }
  closedir(dp);
  dp = NULL;
  return (NULL);
}

int match(const char *s1, const char *s2)
{
  int diff = strlen(s1) - strlen(s2);

  if(strlen(s1) > strlen(s2))
    return (strcmp(&s1[diff], s2) == 0);
  else
    return (0);
}
```

4 Programming with directories (cont.)

- The current working directory
  - 로그인한 각 사용자는 current working directory에서 작업을 시작한다.
  - 상대경로의 시작점
  - 각 프로그램의 수행은 자신의 current working directory를 가지고 있다. (shell도…)
- Changing directories with chdir
  - 프로세스의 current working directory를 변경하는데 해당 프로세스의 것만 변경한다.

```
fd1=open("/usr/ben/abc",O_RDONLY);
fd2=open("/usr/ben/xyz",_RDWR);
```

```
chdir("/usr/ben");
fd1=open("abc",O_RDONLY);
fd2=open("xyz",_RDWR);
```

보다 효율적이다

Finding the name of the current working directory

```
/* my_pwd -- print working directory */

#include <stdio.h>
#include <unistd.h>
#define VERYBIG 200

void my_pwd(void)
{
  char dirname[VERYBIG];

  if(getcwd(dirname, VERYBIG) == NULL)
    perror("getcwd error");
  else
    printf("%s₩n", dirname);
}
```

마지막 '\0'도 들어갈 만큼 충분한 크기이어야 한다.

Walking a directory tree : ftw

```
#include <ftw.h>
int ftw(const char *path, int (*func)(), int depth);
```

- recursive tree walk을 시작하는 path를 지정
- depths는 ftw에서 사용되는 file descriptor의 수를 제어
- fun()은 3개의 인수를 가진다.

  int func(const char *name, const struct stat *sptr, int type)
- type은 다음과 같다.
  - FTW_F : file
  - FTW_D : directory
  - FTW_DNR : 읽을 수 없는 directory
  - FTW_SL : symbolic link
  - FTW_NS : not a symbolic link, stat이 성공적으로 수행되지 못할 객체
- 트리 산책은 트리의 leaf에 도달하거나 ftw에서 오류가 발생할 때 까지 계속된다.
- 위의 func이 0이 아닌 값을 돌려주어도 산책은 종료된다.

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

main(int argc, char **argv)
{
  int list(const char *, const struct stat *, int);

  if(argc == 1)
    ftw(".", list, 1);
  else
    ftw(argv[1], list, 1);
  exit(0);
}
```

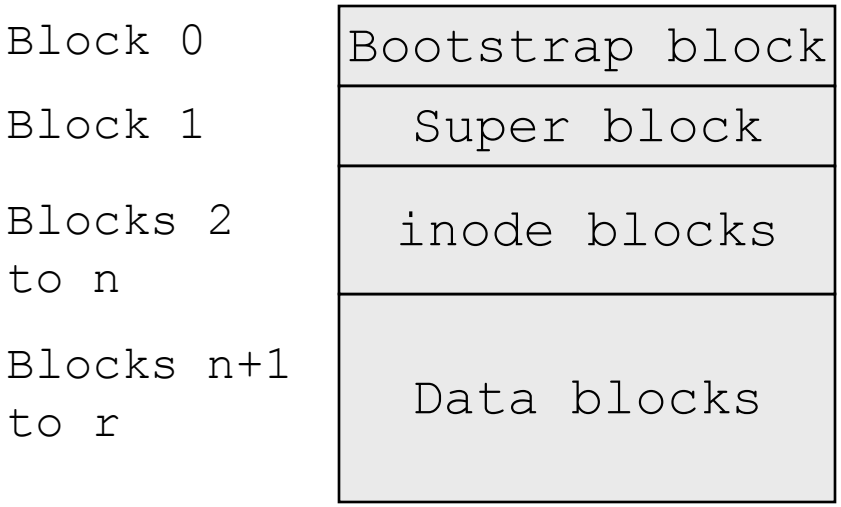
```
./list                * 0755
./file1                 0644
./subdir              * 0777
./subdir/another        0644
./subdir/subdir2      * 0755
./subdir/yetanother     0644
```

```
int list(const char *name, const struct stat *status, int type)
{
  /* if the stat call failed, just return */
  if(type == FTW_NS)
    return 0;
  /*
   * otherwise print object name,
   * permissions and "*" postfix
   * if object is directory or symbolic link
   */
  if(type == FTW_F)
    printf("%-30s\t0%3o\n", name, status->st_mode&0777);
  else
    printf("%-30s*\t0%3o\n", name, status->st_mode&0777);

  return 0;
}
```

5 UNIX file systems

- File → directory → file system
- Demountable volume
  - 계층 트리의 한 부분을 트리 구조의 어느 위치에든 동적으로 추가 가능
- File system 정보는 disk partition에 저장
  - Disk partition은 UNIX device file (special file)이다.
- 전통적으로 4개의 논리 부분으로 나뉜다.



- Bootstrap block
  - 논리적 첫 블록, 물리적으로 디스크 분할이 시작 되는 곳
  - 시스템 시동시 UNIX를 load하기 위한 하드웨어 고유의 부트 프로그램을 포함
- Super block
  - 중요한 정보를 저장

    파일 시스템의 총크기(r), inode block의 수(n-2), 파일 시스템이 갱신된 날자, 시간 등
  - 2개의 free list
    - Chain of free inode numbers
    - Chain of free data block numbers
  - Mount 된 파일 시스템은 위의 2개의 free list 중 일부를 메모리에 탑재

- Inode의 크기는 시스템 의존적: 64bytes or 128bytes
- Inode block이나 Data block 중 어느 하나만 부족하여도 파일 시스템은 공간이 모자라게 된다.
  - 오늘날에 파일 시스템은 크기 변경이 가능하며 inode들은 동적으로 할당된다.
- Inode number 는 file system 내에서만 유일하기 때문에 link하는 것이 불가능하다
  - Symbolic link는 가능하다.

- Caching: **sync** and **fsync**
  - sync : 모든 메모리 버퍼를 디스크로 dump
  - fsync : 특정 파일과 연관된 자료와 속성만을 dump
  - 주기적으로 sync를 호출하는 데몬이 있다.
    - 통상 30초 간격

6 UNIX device files

- 장치 파일
  - 시스템에 부착된 주변장치는 파일 이름으로 접근
  - 디스크 파티션도 장치 파일로 표현
  - /dev 디렉토리

    $cat fred > /dev/lp

    $cat fred > /dev/rmt0

```
#include <fcntl.h>

Main()
{
  int i,fd

  fd = open(“/dev/tty00”,O_WRONLY);
  for(i=0; i<100; i++)
      write(fd,”x”,1);

  close(fd);
}
```

Block and character device files

- Block device
  - 디스크, 자기테이프 등
  - 장치와 커널 사이의 자료 이동은 표준크기 블록 단위로
  - Random access
  - 커널내에서 이 장치들에 대한 접근은 고도로 구조화된 커널 자료구조와 루틴 집합에 의해 제어
- Character device
  - 단말기 라인, 모뎀라인, 프린터 등
  - 자료전송은 임의 길이의 바이트열로

- File system
  - Block device에서만 존재
- Raw device
  - 이 장치에 빠른 접근을 위해서 연관된 character device
  - mkfs, fsck 같은 utility는 raw device를 이용

- Block device switch table & Character device switch table
  - 특정 주변 장치를 구동시키기 위하여 주변장치와 device-specific code를 연관 시켜주는 configuration table
  - 커널 안의 table
  - 장치 파일의 inode에 저장된 major device number를 사용하여 색인됨
  - Minor device number: 1개 이상의 주변장치 포트를 제공하는 장치들에 대해서 어떤 포트가 접근되고 있는가를 식별

- 주변장치를 access하는 순서
  - Read/write가 device file의 inode에 접근
  - Inode의flag를 확인하여 시스템 장치의 종류 파악 (block or character)
  - Inode에서 major device number 추출
  - Device switch table에서 major number로 색인하여 device driver routine을 찾아 호출
  - Minor device number로 포트를 식별

The stat structure revisited

- Structure stat 의 두 필드
  - st_mode: 장치 파일이면 permission에 첨가하여 장치 종류를 명시

    Block device : S_IFBLK, 060000

    Chareacter device: S_IFCHR, 020000
  - st_rdev: major & minor device number 포함

```
[sthwang@linux dev]$ ls -l /dev/tty0
crw--w----  1 root root 4, 0  2월 24  2004 /dev/tty0
```

또는

```
If (S_ISCHAR(buf.st_mode))
  printf("It's a character device\n");
Else
  printf("It's not\n");
```

File system information

```
#include <sys/statvfs.h>
int statvfs(const char *path, struct statvfs *buf);
int fstatvfs(int fd, struct statvfs *buf);
```

Struct statvfs의 필드

- f_bsize: 나은 성능을 얻기 위한 시스템 블록의 크기
- f_frsize: 기본 파일 시스템 블록 크기 (파일 시스템 생성시 지정)
- f_blocks: f_frsize의 단위로 표시한 블록 수
- f_bfree: 전체 free block 수
- f_bavail: 특권을 갖지 않은 프로세스가 사용가능한 free block수
- f_files: 전체 inode 수
- f_ffree: 전체 free inode 수

fsys -- print file system information

```
main(int argc, char **argv)
{
  struct statvfs buf;

  if(argc != 2)
  {
    fprintf("stderr, "usage: fsys filename\n");
    exit(1);
  }
  if(statvfs(argv[1], &buf) != 0)
  {
    fprintf("stderr, "statvfs error\n");
    exit(2);
  }
  printf("%s:\tfree blocks %d\tfree inodes %d\n",
                    argv[1], buf.f_bfree, buf.f_ffree);
  exit(0);
}
```

File and directory limits: pathconf **and** fpathconf

- 일부 시스템 제한은 파일과 디렉토리에 따라 변한다.
  - 하나의 시스템이 여러 개의 파일 시스템 유형을 지원할 수 있으므로
- 디렉토리 별로 이들 제한을 질의
  - pathconf
  - fpathconf
- System-wide limits
  - <limits.h>에 기술
  - sysconf로 질의

lookup -- displays the settings of file limits

```
typedef struct{
  int val;
  char *name;
} Table;

main()
{
  Table *tb;
  static Table options[] = {
        { _PC_LINK_MAX, "Maximum number of links"},
        { _PC_NAME_MAX, "Maximum length of a filename"},
        { _PC_PATH_MAX, "Maximum length of pathname"},
        {-1, NULL}
  };

  for(tb=options; tb->name != NULL; tb++)
    printf("%-28.28s\t%ld\n", tb->name,
                              pathconf("/tmp", tb->val));
}
```