Linux Programming
6장. 파일 시스템 (2/2)

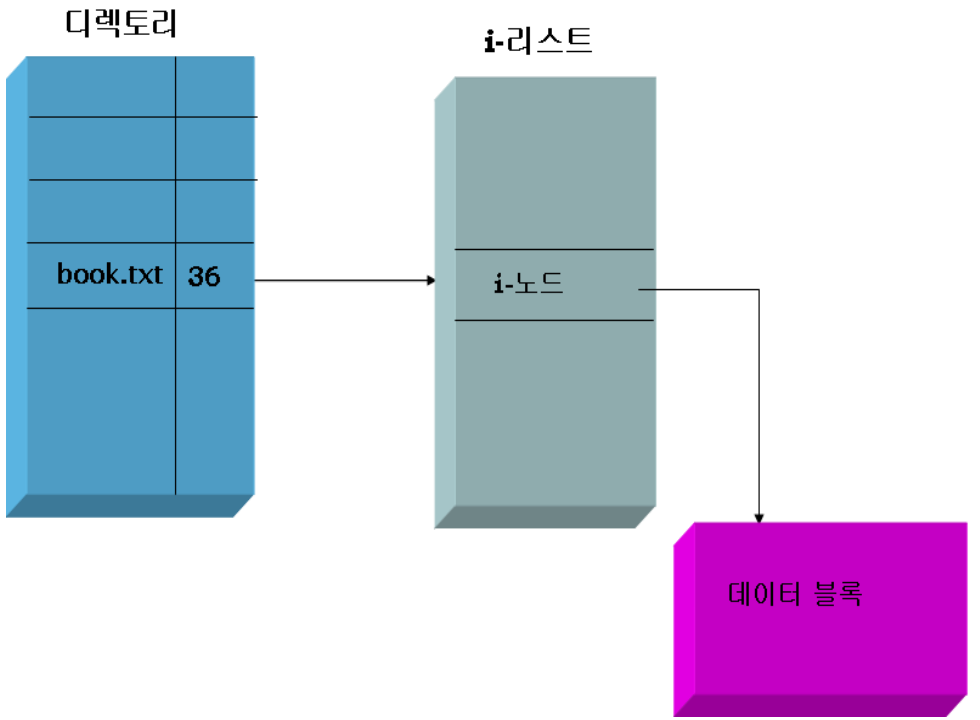
sisong@ut.ac.kr
한국교통대학교 컴퓨터공학전공
송석일

6.4 디렉터리

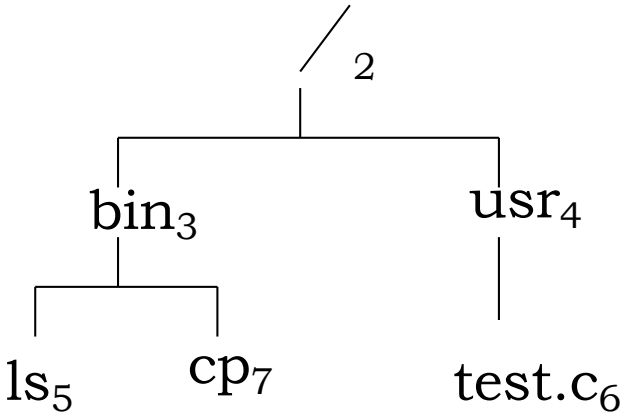
디렉토리 구현

- 디렉토리 파일의 내용
  - 다수의 디렉터리 엔트리로 구성됨

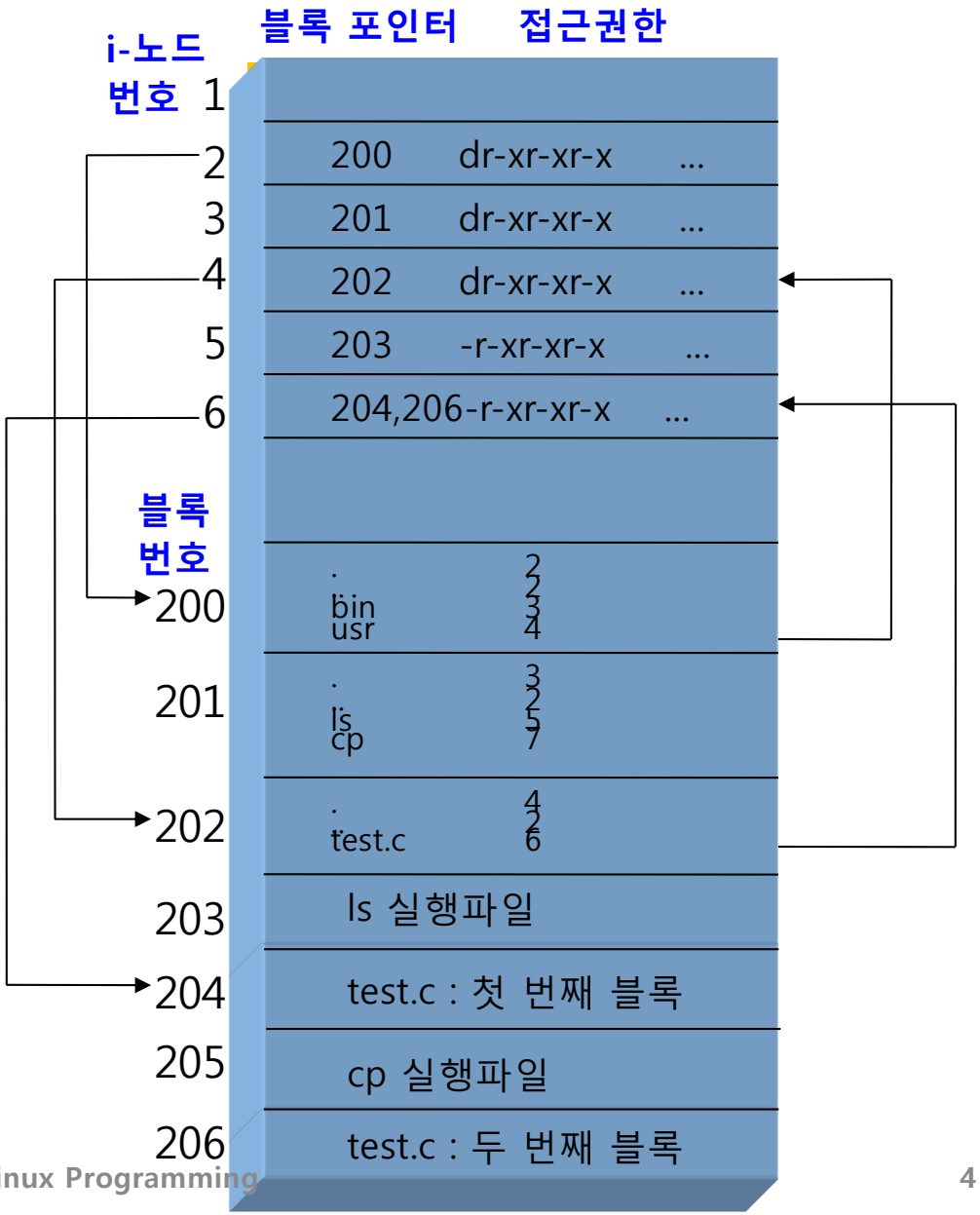
```
#include <dirent.h>
struct dirent {
   ino_t d_ino; // i-노드 번호
   char d_name[NAME_MAX + 1];
   // 파일 이름
}
```



디렉토리 구현



/usr/test.c



디렉터리 리스트

- opendir()
  - 디렉터리 열기 함수
  - DIR 포인터(열린 디렉터리를 가리키는 포인터) 반환
- readdir()
  - 디렉터리 읽기 함수

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir (const char *path);
```

- path 디렉터리를 열고 성공하면 DIR 구조체 포인터를, 실패하면 NULL을 반환

```
struct dirent *readdir(DIR *dp);
```

- 한 번에 디렉터리 엔트리를 하나씩 읽어서 반환

디렉터리 리스트: list1.c

```
1.  #include <sys/types.h>
2.  #include <sys/stat.h>
3.  #include <dirent.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  /* 디렉터리 내의 파일 이름들을 리스트한다. */
7.  int main(int argc, char **argv)
8.  {
9.    DIR *dp;
10.   char *dir;
11.   struct dirent *d;
12.   struct stat st;
13.   char path[BUFSIZ+1];
14.   if (argc == 1)   dir = "."; // 현재 디렉터리를 대상으로
15.   else dir = argv[1];
16.   if ((dp = opendir(dir)) == NULL) // 디렉터리 열기
17.     perror(dir);
18.   while ((d = readdir(dp)) != NULL) // 각 디렉터리 엔트리에 대해
19.     printf("%s %lu \n", d->d_name, d->d_ino); // 파일 이름, i-노드 번호 출력
20.   closedir(dp);
21.   exit(0);
22. }
```

```
$ list1 .
list1.c 12059349
.. 12059625
list1 12059357
. 12059348
```

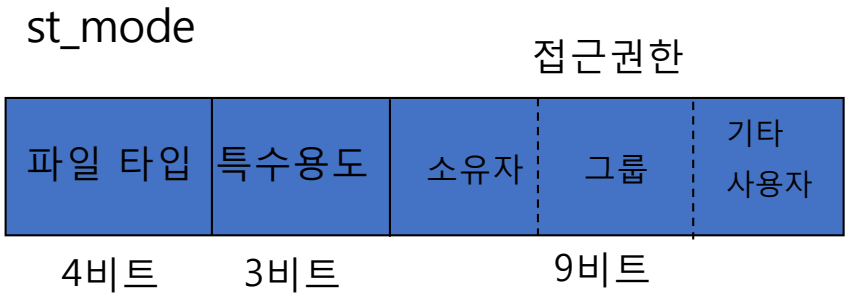
파일 이름과 크기 출력

```
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
/* 디렉터리 내의 파일 이름들을 리스트한다. */
int main(int argc, char **argv)
{
  DIR *dp;
  char *dir;
  struct dirent *d;
  struct stat st;
  char path[BUFSIZ+1];
  if (argc == 1)   dir = "."; // 현재 디렉터리를 대상으로
  else dir = argv[1];
  if ((dp = opendir(dir)) == NULL) // 디렉터리 열기
    perror(dir);
  while ((d = readdir(dp)) != NULL) {          //디렉터리 내의 각 파일
   sprintf(path, "%s/%s", dir, d->d_name);   // 파일경로명 만들기
   if (lstat(path, &st) < 0)                 // 파일 상태 정보 가져오기
     perror(path);
   printf("%5d %s", st->st_size, d->name);   // 블록 수, 파일 이름 출력
   putchar('\n');
  }
}
```

st_mode

- lstat() 시스템 호출
  - 파일 타입과 접근권한 정보는 st->st_mode 필드에 함께 저장됨.

- st_mode 필드
  - 4비트: 파일 타입
  - 3비트: 특수용도
  - 9비트: 접근권한
    - 3비트: 파일 소유자의 접근권한
    - 3비트: 그룹의 접근권한
    - 3비트: 기타 사용자의 접근권한



디렉터리 리스트 프로그램

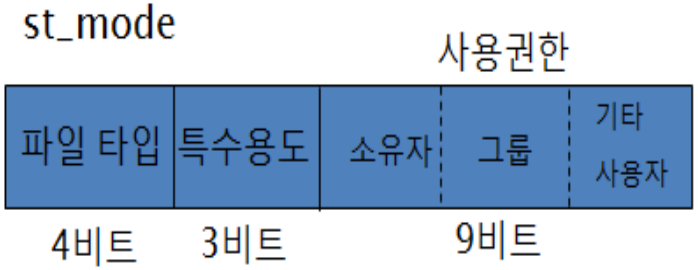
- list2.c
  - ls –l 명령어처럼 파일의 모든 상태 정보를 프린트

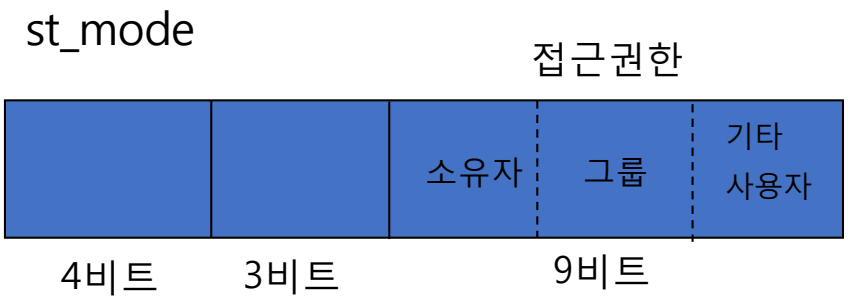
- 프로그램 구성
  - main() 메인 프로그램
  - printStat() 파일 상태 정보 프린트
  - type() 파일 타입 반환
  - perm() 파일 접근권한 반환

```
1.  #include <sys/types.h>
2.  #include <sys/stat.h>
3.  #include <dirent.h>
4.  #include <pwd.h>
5.  #include <grp.h>
6.  #include <stdio.h>
7.  #include <stdlib.h>
8.  #include <string.h>
9.  #include <time.h>

10. char type(mode_t);
11. char *perm(mode_t);
12. void printStat(char*, char*, struct stat*);

13. /* 디렉토리 내용을 자세히 리스트 */
14. int main(int argc, char **argv)
15. {
16.     DIR *dp;
17.     char *dir;
18.     struct stat st;
19.     struct dirent *d;
20.     char path[BUFSIZ+1];

21.     if (argc == 1)
22.         dir = ".";
23.     else dir = argv[1];

24.     if ((dp = opendir(dir)) == NULL)  // 디렉토리 열기
25.         perror(dir);
26.     while ((d = readdir(dp)) != NULL) {// 디렉토리 내 각 파일에 대해
27.         sprintf(path, "%s/%s", dir, d->d_name); // 파일 경로명 만들기
28.         if (lstat(path, &st) < 0)          // 파일 상태 정보 가져오기
29.             perror(path);
30.         else
31.             printStat(path, d->d_name, &st);  // 상태 정보 출력
32.     }

33.     closedir(dp);
34.     exit(0);
35. }

36. /* 파일 상태 정보를 출력 */
37. void printStat(char *pathname, char *file, struct stat *st) {
38.     printf("%5d ", st->st_blocks);
39.     printf("%c%s ", type(st->st_mode), perm(st->st_mode));
40.     printf("%3d ", st->st_nlink);
41.     printf("%s %s ", getpwuid(st->st_uid)->pw_name,
42.             getgrgid(st->st_gid)->gr_name);
43.     printf("%9d ", st->st_size);
44.     printf("%.12s ", ctime(&st->st_mtime)+4);
45.     printf("%s\n", file);
46. }
```

```
67.  /* 파일 타입을 반환 */
68.  char type(mode_t mode) {
69.      if (S_ISREG(mode))
70.          return('-');
71.      if (S_ISDIR(mode))
72.          return('d');
73.      if (S_ISCHR(mode))
74.          return('c');
75.      if (S_ISBLK(mode))
76.          return('b');
77.      if (S_ISLNK(mode))
78.          return('l');
79.      if (S_ISFIFO(mode))
80.          return('p');
81.      if (S_ISSOCK(mode))
82.          return('s');
83.  }
```

```
84.  /* 파일 허가권을 반환 */
85.  char* perm(mode_t mode) {
86.      int i;
87.      static char perms[10];
88.      strcpy(perms, "---------");

89.      for (i=0; i < 3; i++) {
90.          if (mode & (S_IRUSR >> i*3))
91.              perms[i*3] = 'r';
92.          if (mode & (S_IWUSR >> i*3))
93.              perms[i*3+1] = 'w';
94.          if (mode & (S_IXUSR >> i*3))
95.              perms[i*3+2] = 'x';
96.      }
97.     return(perms);
98.  }
```






```
#define S_IRUSR 00400
#define S_IWUSR 00200
#define S_IXUSR 00100
```

```
$ list2 .
 8 drwxrwxr-x  2 lect lect 4096  Dec 21 13:43 .
 8 drwxrwxr-x 16 lect lect 4096  Aug 30 18:55 ..
 8 -rw-r--r--   1 lect lect  781  Mar 21 17:13 list1.c
 8 -rw-r--r--   1 lect lect 2178  Mar 28 14:25 list2.c
24 -rwxrwxr-x  1 lect lect 8775  Mar 21 17:14 list1
32 -rwxrwxr-x  1 lect lect 13376 Dec 21 13:43 list2
...
```

디렉터리 생성

- mkdir() 시스템 호출
  - path가 나타내는 새로운 디렉터리 생성
  - "." 와 ".." 파일은 자동 생성

```
#include <sys/types.h>
#include <sys/stat.h>
int mkdir (const char *path, mode_t mode );
```

- 새로운 디렉터리 만들기에 성공하면 0, 실패하면 -1을 반환한다.

디렉터리 삭제

- rmdir() 시스템 호출
  - path가 나타내는 디렉터리가 비어 있으면 삭제

```
#include <unistd.h>

int rmdir (const char *path);
```

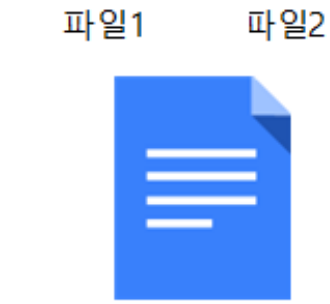
- 디렉터리가 비어 있으면 삭제
- 성공하면 0, 실패하면 -1을 반환

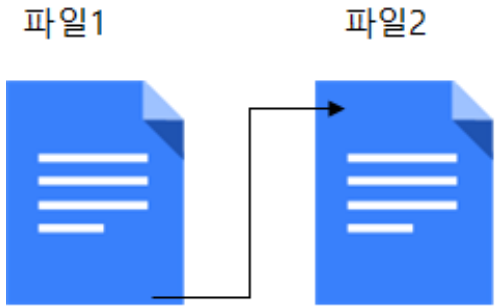
6.5 링크

하드 링크 vs 심볼릭 링크

- 하드 링크(hard link)
  - 파일 시스템 내의 i-노드를 가리키는 파일
  - 같은 파일 시스템 내에서만 사용

- 심볼릭 링크(symbolic link)
  - 소프트 링크(soft link)
  - 실제 파일의 경로명 저장하고 있는 링크
  - 파일에 대한 간접 포인터 역할 (바로가기)
  - 다른 파일 시스템에 있는 파일 링크 가능





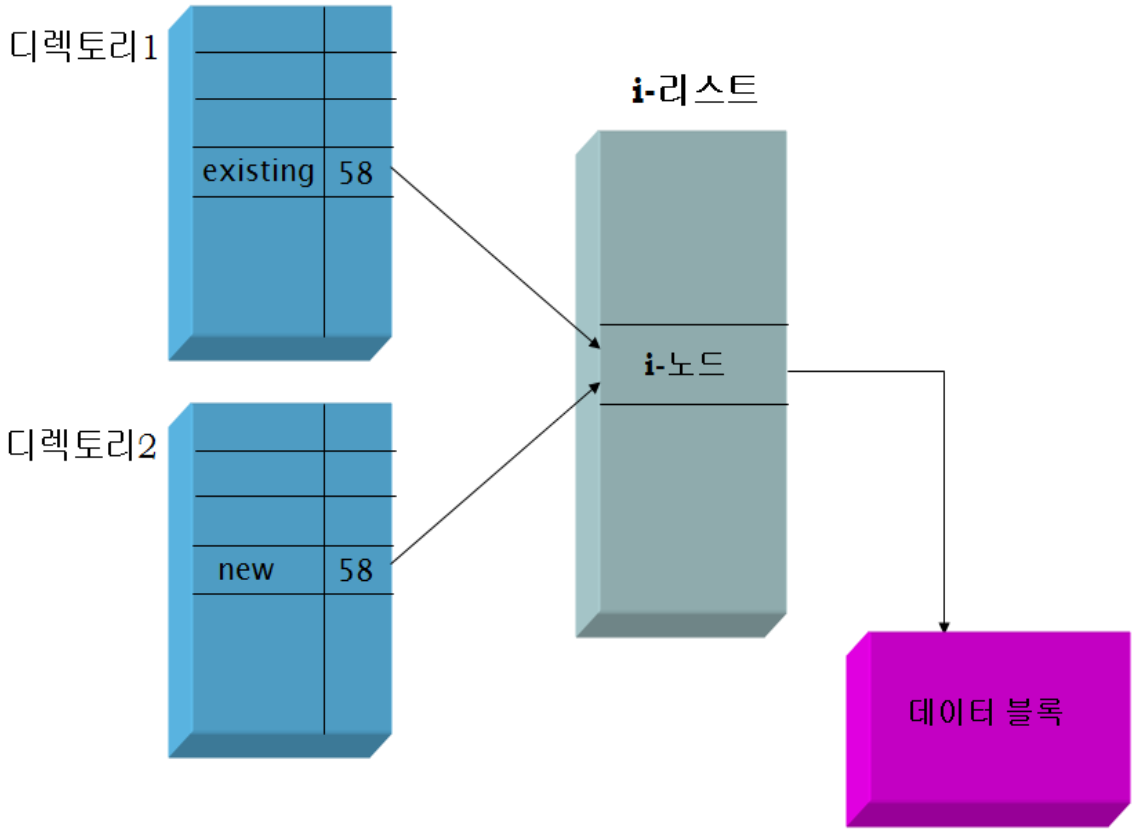
링크

- 링크는 기존 파일에 대한 또 다른 이름
- 하드 링크, 심볼릭(소프트) 링크

```
#include <unistd.h>
int link(char *existing, char *new);
int unlink(char *path);
```

링크의 구현

- link() 시스템 호출 예
  - 기존 파일 existing에 대한 새로운 이름 new (하드 링크) 생성



link.c, unlink.c

link.c

```
1.  #include <unistd.h>
2.  int main(int argc, char *argv[ ])
3.  {
4.    if (link(argv[1], argv[2]) == -1) {
5.      exit(1);
6.    }
7.    exit(0);
8.  }
```

unlink.c

```
1.  #include <unistd.h>
2.  main(int argc, char *argv[ ])
3.  {
4.    int unlink( );
5.    if (unlink(argv[1]) == -1 {
6.      perror(argv[1]);
7.      exit(1);
8.    }
9.    exit(0);
10. }
```

심볼릭 링크

- 심볼릭 링크 (소프트 링크) 생성

int symlink (const char *actualpath, const char *sympath);

- 심볼릭 링크를 만드는데 성공하면 0, 실패하면 -1을 반환

```
1.  #include <unistd.h>
2.  int main(int argc, char *argv[ ])
3.  {
4.    if (symlink(argv[1], argv[2]) == -1) {
5.      exit(1);
6.    }
7.    exit(0);
8.  }
```

```
$ slink /usr/bin/gcc cc
$ ls -l cc
2 lrwxrwxrwx 1 lect lect 7 4월 8 19:58 cc -> /usr/bin/gcc
```

심볼릭 링크 내용 읽기

```
#include <unistd.h>
int readlink (const char *path, char *buf, size_t bufsize);
```

- path 심볼릭 링크의 실제 내용을 읽어서 buf에 저장
- 성공하면 buf에 저장한 바이트 수를 반환하며 실패하면 –1을 반환

심볼릭 링크 내용 확인: rlink.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  int main(int argc, char *argv[ ])
5.  {
6.      char buffer[1024];
7.      int nread;
8.      nread = readlink(argv[1], buffer, 1024);
9.      if (nread > 0) {
10.         write(1, buffer, nread);
11.         exit(0);
12.     } else {
13.         fprintf(stderr, "오류 : 해당 링크 없음\n");
14.         exit(1);
15.     }
16. }
```

```
$ rlink cc
/usr/bin/gcc
```