File System

Jo, Heeseung

리눅스 파일의 특징

파일

- 파일은 데이터 저장, 장치구동, 프로세스 간 통신 등에 사용
- 일반파일, 디렉토리, 특수파일

일반파일

- 텍스트 파일, 실행파일, 라이브러리, 이미지 등 사용하는 대부분의 파일

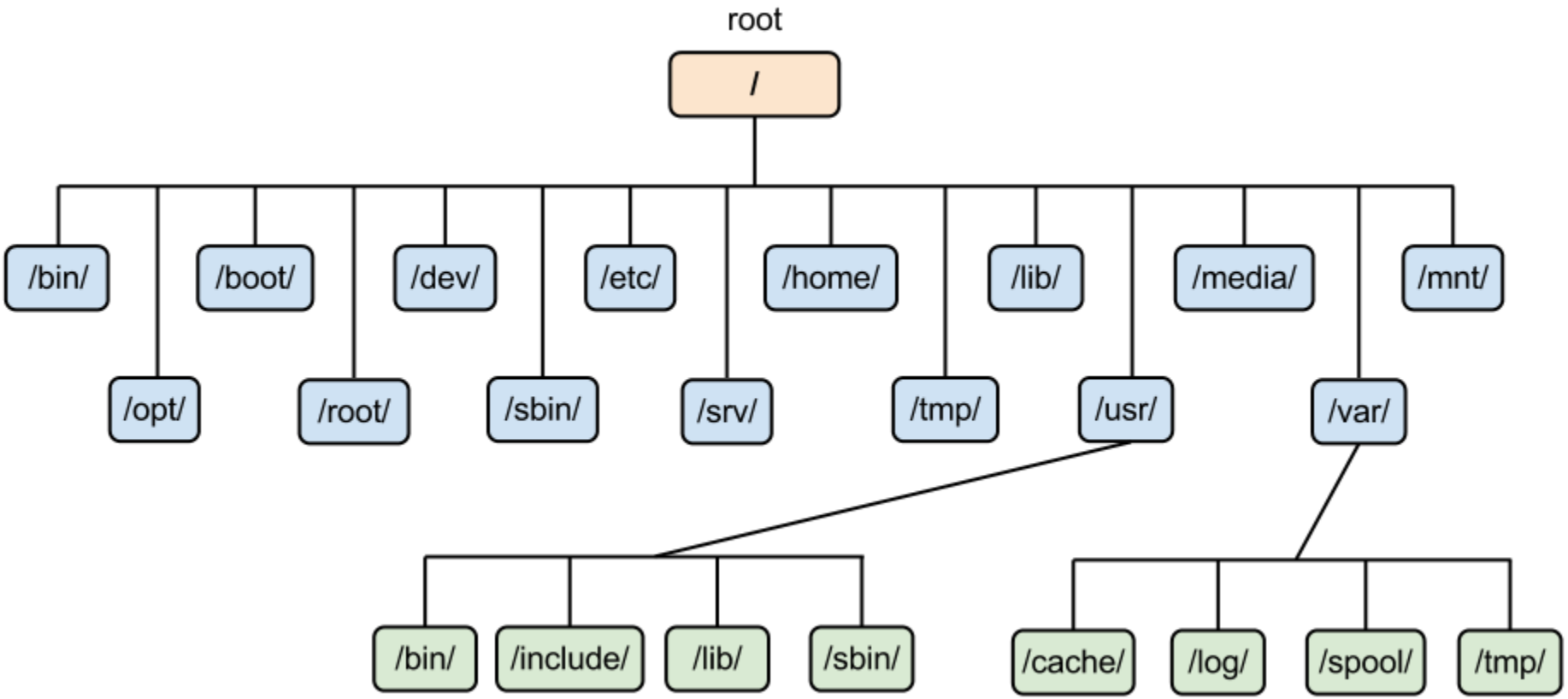
디렉토리

- 디렉토리도 파일로 취급
- 디렉토리에 속한 파일의 목록과 inode를 가진 파일

특수파일 - 장치파일

- 장치를 파일로 표현
- 예 : /dev/sda1

리눅스 파일의 특징


root
/
/bin/
/boot/
/dev/
/etc/
/home/
/lib/
/media/
/mnt/
/opt/
/root/
/sbin/
/srv/
/tmp/
/usr/
/var/
/bin/
/include/
/lib/
/sbin/
/cache/
/log/
/spool/
/tmp/

리눅스 파일의 특징

파일의 종류 구분

- ls -l 명령으로 파일의 종류 확인
  - 결과의 맨 앞 글자로 구분

```
# ls -l /usr/bin/vi
-r-xr-xr-x   5 root     bin        193968 2007  9월 14일 /usr/bin/vi
```

- 파일 종류 식별 문자

문자	파일의 종류
-	일반 파일
d	디렉토리
b	블록 장치 특수 파일
c	문자 장치 특수 파일
l	심볼릭 링크

리눅스 파일의 특징

파일의 구성 요소

- 파일명, inode, 데이터블록

파일명

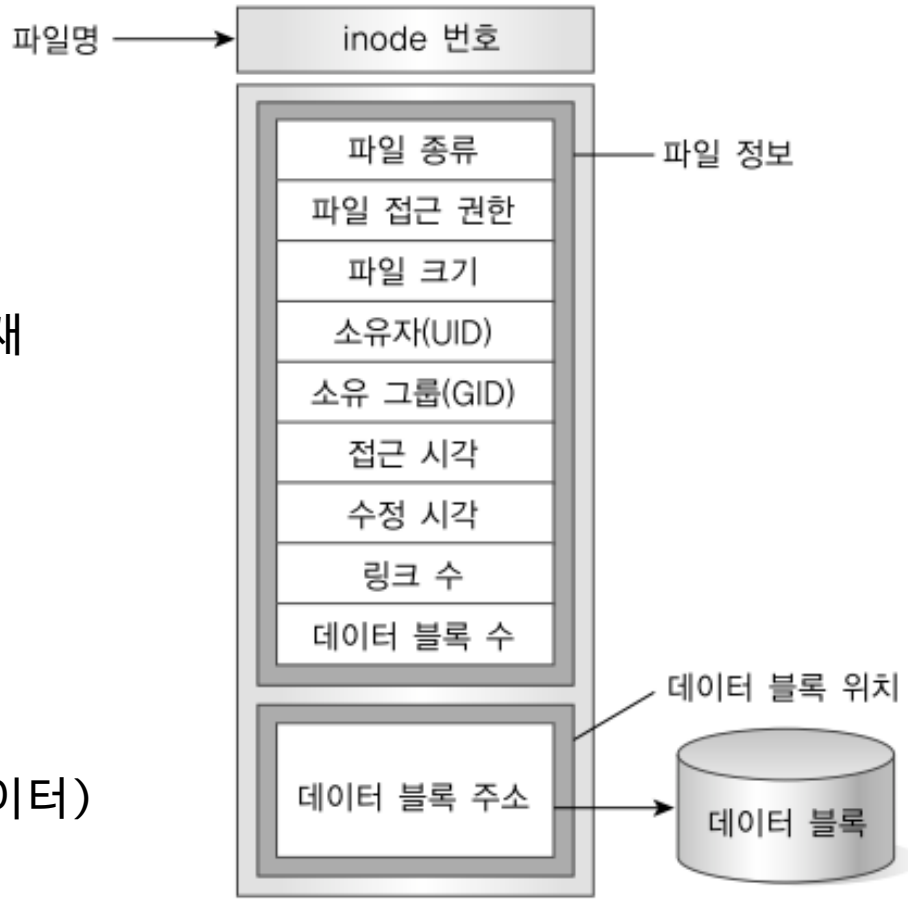
- 사용자가 파일에 접근할 때 사용
- 파일명과 관련된 inode가 반드시 존재
- 최대 255자까지 가능
- 대소문자를 구분
- '.' 으로 시작하면 숨김 파일

inode

- 번호로 표시
  - ls –i 명령으로 inode 번호 확인
- 파일 정보를 저장하는 부분 (메타데이터)
- 데이터 블록의 주소 저장하는 부분

데이터 블록

- 실제로 데이터가 저장되는 부분



파일 정보 검색

파일명으로 파일 정보 검색 : stat(2)

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *restrict path, struct stat *buf);
```

- inode에 저장된 파일정보 검색
- path에 검색할 파일의 경로 지정
  - 검색한 정보를 buf에 저장

```
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;
    time_t      st_atime;
    time_t      st_mtime;
    time_t      st_ctime;
    blksize_t   st_blksize;
    blkcnt_t    st_blocks;
    char        st_fstype[_ST_FSTYPSZ];
};
```

파일명으로 inode 정보 검색하기

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <stdio.h>
04
05  int main(void) {
06     struct stat buf;
07
08     stat("unix.txt", &buf);
09
10     printf("Inode = %d\n", (int)buf.st_ino);
11     printf("Mode = %o\n", (unsigned int)buf.st_mode);
12     printf("Nlink = %o\n",(unsigned int) buf.st_nlink);
13     printf("UID = %d\n", (int)buf.st_uid);
14     printf("GID = %d\n", (int)buf.st_gid);
15     printf("SIZE = %d\n", (int)buf.st_size);
16     printf("Atime = %d\n", (int)buf.st_atime);
17     printf("Mtime = %d\n", (int)buf.st_mtime);
18     printf("Ctime = %d\n", (int)buf.st_ctime);
19     printf("Blksize = %d\n", (int)buf.st_blksize);
20     printf("Blocks = %d\n", (int)buf.st_blocks);
21     //printf("FStype = %s\n", buf.st_fstype);
22
23     return 0;
24  }
```

```
# ex3_1.out
Inode = 192
Mode = 100644
Nlink = 1
UID = 0
GID = 1
SIZE = 24
Atime = 1231397228
Mtime = 1231397228
Ctime = 1231397228
Blksize = 8192
Blocks = 2
```

UNIX time:
1970/1/1 00:00:00 (UTC)
부터 경과한 초

파일 정보 검색

파일 기술자로 파일 정보 검색 : fstat(2)

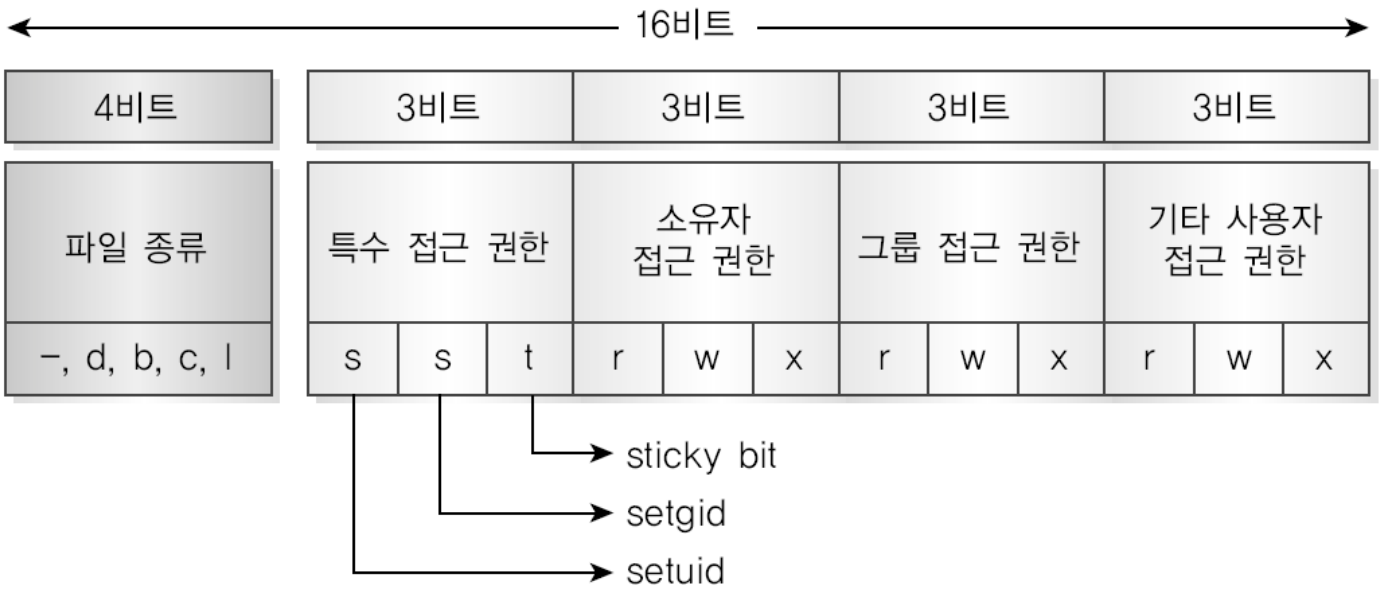
```
#include <sys/types.h>
#include <sys/stat.h>
int fstat(int fd, struct stat *buf);
```

- fd로 지정한 파일의 정보를 검색하여 buf에 저장

파일 접근권한 제어

stat구조체의 st_mode항목에 파일의 종류와 접근권한 저장

st_mode 값의 구조



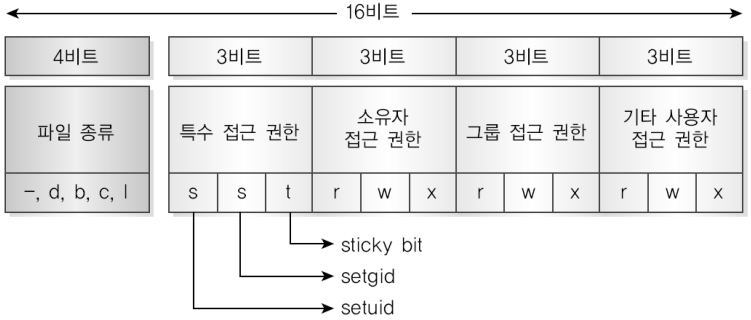
[그림 3-3] st_mode의 비트 구조

- sticky bit: 디렉토리에 설정. 누구나 파일 생성. 삭제는 본인만 가능.
- setuid: 실행시, 일시적으로 파일 소유자의 권한으로 실행
- setgid: 실행시, 일시적으로 파일 소유 그룹의 권한으로 실행

파일 접근권한 제어

chmod 명령어

- 파일/디렉토리의 접근 권한을 변경



[그림 3-3] st_mode의 비트 구조

- chmod 755 test.out
  - st_mode의 하위 9비트를 111 101 101 로 변경
- chmod -R 644 testdir1
  - testdir1을 포함한 모든 하위 파일/디렉토리를 644로 변경

파일 접근권한 제어

파일 확장자

- Linux/Unix에서는 파일의 확장자는 아무런 의미가 없음
- a.exe b.jpg 등 확장자는 이름의 일부일 뿐임

실행 파일? or not?

- 접근 권한의 x bit으로만 구분됨

./a.out

- 현재 디렉토리 밑의 a.out 파일을 실행하는 의미
- a.out에 x bit 권한이 없으면 실행 안됨

링크 파일 생성

링크

- 이미 있는 파일이나 디렉토리에 접근할 수 있는 새로운 이름
- 같은 파일/디렉토리지만 여러 이름으로 접근
- 하드 링크 : 기존 파일과 동일한 inode 사용, inode에 링크 개수 증가
  - ln original.txt link.txt
- 심볼릭 링크 : 기존 파일에 접근하는 다른 파일 생성(다른 inode 사용)
  - ln -s original.txt link.txt

하드링크 생성 : link(2)

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

- 두 경로는 같은 파일시스템에 존재해야 함

link 함수 사용하기

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <unistd.h>
04  #include <stdio.h>
05
06  int main(void) {
07      struct stat buf;
08
09      stat("unix.txt", &buf);
10      printf("Before Link Count = %d\n", (int)buf.st_nlink);
11
12      link("unix.txt", "unix.ln");
13
14      stat("unix.txt", &buf);
15      printf("After Link Count = %d\n", (int)buf.st_nlink);
16
17      return 0;
18  }
```

```
# ls -l unix*
-rwxrwx---   1 root    other  24  1월   8일   15:47 unix.txt
# ex3_8.out
Before Link Count = 1
After Link Count = 2
# ls -l unix*
-rwxrwx---   2 root    other  24  1월   8일   15:47 unix.ln
-rwxrwx---   2 root    other  24  1월   8일   15:47 unix.txt
```

링크 파일 생성

심볼릭 링크 생성 : symlink(2)

```
#include <unistd.h>
int symlink(const char *name1, const char *name2);
```

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <unistd.h>
04
05  int main(void) {
06      symlink("unix.txt", "unix.sym");
07
08      return 0;
09  }
```

```
# ls -l unix*
-rwxrwx---   2 root     other         24  1월   8일   15:47 unix.ln
-rwxrwx---   2 root     other         24  1월   8일   15:47 unix.txt
# ex3_9.out
# ls -l unix*
-rwxrwx---   2 root     other         24  1월   8일   15:47 unix.ln
lrwxrwxrwx   1 root     other          8  1월 11일   18:48 unix.sym ->
unix.txt
-rwxrwx---   2 root     other         24  1월   8일   15:47 unix.txt
```

디렉토리 관련 함수

디렉토리 생성: mkdir(2)

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

- path에 지정한 디렉토리를 mode 권한에 따라 생성

디렉토리 삭제: rmdir(2)

```
#include <unistd.h>
int rmdir(const char *path);
```

디렉토리명 변경: rename(2)

```
#include <stdio.h>
int rename(const char *old, const char *new);
```

디렉토리 생성/삭제/이름 변경하기

```
01  #include <sys/stat.h>
02  #include <unistd.h>
03  #include <stdlib.h>
04  #include <stdio.h>
05
06  int main(void) {
07      if (mkdir("serv", 0755) == -1) {
08          perror("serv");
09          exit(1);
10      }
11
12      if (mkdir("prog", 0755) == -1) {
13          perror("prog");
14          exit(1);
15      }
16
17      if (rename("serv", "server") == -1) {
18          perror("server");
19          exit(1);
20  }
21
```

serv -> server로 변경

디렉토리 생성/삭제/이름 변경하기

```
22      if (rmdir("prog") == -1) {
23          perror("prog");
24          exit(1);
25      }
26
27      return 0;
28  }
```

prog는 생성했다 삭제

```
# ex3_13.out
# ls -l
drwxr-xr-x   2 root other 512  1월 12일   18:06 server
```

디렉토리 관련 함수

현재 작업 디렉토리 위치 : getcwd(3)

```
#include <unistd.h>
char *getcwd(char *buf, size_t size);
```

- size는 buf의 크기
- 현재 작업 디렉토리 위치를 알려주는 명령은 pwd, 함수는 getcwd

디렉토리 이동: chdir(2)

```
#include <unistd.h>
int chdir(const char *path);
```

디렉토리 정보 검색

```
typedef struct dirent {
    ino_t           d_ino;
    off_t           d_off;
    unsigned short  d_reclen;
    char            d_name[NAME_MAX+1];
} dirent_t;
```

디렉토리 열기: opendir(3)

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

- 성공하면 열린 디렉토리를 가리키는 DIR 포인터를 리턴

디렉토리 닫기: closedir(3)

```
#include <sys/types.h>
#include <dirent.h>
int closedir(DIR *dirp);
```

디렉토리 정보 읽기: readdir(3)

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir(DIR *dirp);
```

- 디렉토리의 내용을 한 번에 하나씩 읽음
- 순서는 정해지지 않음 (일반적으로 생성된 순서)

디렉토리 열고 정보 읽기

```
01  #include <dirent.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      DIR *dp;
07      struct dirent *dent;
08
09      if ((dp = opendir("serv")) == NULL) {
10          perror("opendir: serv");
11          exit(1);
12      }
13
14      while ((dent = readdir(dp))) {
15          printf("Name : %s  ", dent->d_name);
16          printf("Inode : %d\n", (int)dent->d_ino);
17      }
18
19      closedir(dp);
20
21      return 0;
22  }
```

```
# ex3_15.out
Name : .  Inode : 208
Name : ..  Inode : 189
```

```
typedef struct dirent {
    ino_t             d_ino;
    off_t             d_off;
    unsigned short    d_reclen;
    char              d_name[NAME_MAX+1];
} dirent_t;
```

디렉토리 항목의 상세 정보 검색하기

```
01 #include <sys/types.h>
02 #include <sys/stat.h>
03 #include <dirent.h>
04 #include <stdlib.h>
05 #include <stdio.h>
06
07  int main(void) {
08      DIR *dp;
09      struct dirent *dent;
10      struct stat sbuf;
11      char path[BUFSIZ];
12
13      if ((dp = opendir("serv")) == NULL) {
14          perror("opendir: serv");
15          exit(1);
16      }
18      while ((dent = readdir(dp))) {
19         if (dent->d_name[0] == '.') continue;
20         else break;
21      }
22
23      sprintf(path, "serv/%s", dent->d_name);
24      stat(path, &sbuf);
```

디렉토리의 항목을 읽고
다시 stat 함수로 상세 정보 검색

디렉토리 항목의 상세 정보 검색하기

```
25
26      printf("Name : %s\n", dent->d_name);
27      printf("Inode(dirent) : %d\n", (int)dent->d_ino);
28      printf("Inode(stat) : %d\n", (int)sbuf.st_ino);
29      printf("Mode : %o\n", (unsigned int)sbuf.st_mode);
30      printf("Uid : %d\n", (int)sbuf.st_uid);
31
32      closedir(dp);
33
34      return 0;
35  }
```

```
# ls -ai serv
208 .   189 ..   213 main.c

# ex3_16.out
Name : main.c
Inode(dirent) : 213
Inode(stat) : 213
Mode : 100644
Uid : 0
```

디렉토리 정보 검색

디렉토리 오프셋: telldir(3), seekdir(3), rewinddir(3)

```
#include <dirent.h>
long telldir(DIR *dirp);
void seekdir(DIR *dirp, long loc);
void rewinddir(DIR *dirp);
```

- telldir : 디렉토리 오프셋의 현재 위치를 알려줌
- seekdir : 디렉토리 오프셋을 loc에 지정한 위치로 이동
- rewinddir : 디렉토리 오프셋을 디렉토의 시작인 0으로 이동

Exercise

Ex.

- 특정 디렉토리 내의 모든 파일 이름과 inode 번호를 출력하는 프로그램을 작성

```
./a.out /tmp/dirtest
25067538 .
25067521 ..
25067540 a.out
25067542 debug
25067543 hello
25067539 hello.c
25067541 perr.c
25067546 winscp437.zip
```

Exercise - Extra

Ex.

- 특정 디렉토리 내의 모든 파일 이름과 inode 번호를 출력하는 프로그램을 작성
- 하위에 디렉토리가 있을 경우 recursive로 출력

```
./a.out /tmp/dirtest
25067538 .
25067521 ..
25067540 a.out
25067542 debug
25067543 dir1
25067539 dir1/hello.c
25067541 dir1/perr.c
25067546 winscp437.zip
```