Linux System Programming

by ProgCoach4U

파일 다루기 - Low Level

파일 열기

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

파라미터
- pathname: 파일 경로
- flags: 접근 모드를 포함한 여러 가지 플래그
- mode: 파일 생성 시 설정할 권한

반환값
- 성공 시 열린 파일 디스크립터(fd)
- 실패 시 -1

파일 열기 - 플래그

플래그	의미
O_RDONLY	읽기 전용
O_WRONLY	쓰기 전용
O_RDWR	읽기 + 쓰기
O_CREAT	파일이 존재하지 않으면 regular file 생성
O_NOFOLLOW	파일이 Symbolic link인 경우 파일 열기 실패
O_TRUNC	Regular file이 존재하는 경우 파일 사이즈를 0으로 만듬
O_APPEND	append mode로 열기(파일 오프셋이 파일 끝에 위치)

파일 열기 - 권한

Symbol	Value	Perm	Symbol	Value	Perm	Symbol	Value	Perm
S_IRWXU	0700	User RWX	S_IRWXG	0070	Grp RWX	S_IRWXO	0007	Other RWX
S_IRUSR	0400	User R--	S_IRGRP	0040	Grp R--	S_IROTH	0004	Other R--
S_IWUSR	0200	User -W-	S_IWGRP	0020	Grp -W-	S_IWOTH	0002	Other -W-
S_IXUSR	0100	User --X	S_IXGRP	0010	Grp --X	S_IXOTH	0001	Other --X

```
progcoach4u@ubuntu-vm:file_basic$ ls -al
total 64
drwxr-xr-x 2 progcoach4u progcoach4u  4096 11월 18 23:51 .
drwxr-xr-x 5 progcoach4u progcoach4u  4096 11월 18 23:53 ..
-rwxr-xr-x 1 progcoach4u progcoach4u  8696 11월 11 02:37 file_open
-rw-r--r-- 1 progcoach4u progcoach4u   667 11월 11 02:37 file_open.c
```

파일 닫기

```
int close(int fd);
```

파라미터
- fd: 파일 디스크립터

반환값
- 성공 시 0
- 실패 시 -1

파일 포지션 - 오프셋 설정

```
off_t lseek(int fd, off_t offset, int whence);
```

파라미터
- fd: 파일 디스크립터
- offset: 오프셋 값(양수/음수 모두 가능)
- whence: 오프셋의 기준
  - SEEK_SET: 파일의 시작 기준
  - SEEK_END: 파일의 끝 기준
  - SEEK_CUR: 현재 파일 포지션의 기준

반환값
- 성공시 완료 후 파일 포지션(오프셋값)
- 실패시 -1

파일 포지션 - 현재 오프셋 가져오기

ftell()과 같은 API가 없다!!

다음과 같이 현재 오프셋을 가져올 수 있다.

```
cur_offset = lseek(fd, 0, SEEK_CUR);
```

파일에 쓰기 - formatted

```
int dprintf(int fd, const char *format, ...);
```

파라미터
- fd: 파일 디스크립터
- format: 출력 포맷
- …: 가변 arguments

반환값
- 성공시 쓰여진 바이트값
- 실패시 음수

파일에 쓰기 - byte stream

```
ssize_t write(int fd, const void *buf, size_t count);
ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset);
```

파라미터
- fd: 파일 디스크립터
- buf: 버퍼 포인터
- count: 저장할 사이즈(바이트)
- offset: 저장할 위치(오프셋, 파일 시작 기준)

반환값
- 성공 시 실제로 파일에 저장한 사이즈(바이트).
  - 단, count보다 작을 수 있음
- 실패 시 -1

파일에서 읽기 - byte stream

```
ssize_t read(int fd, void *buf, size_t count);
ssize_t pread(int fd, void *buf, size_t count, off_t offset);
```

파라미터
- fd: 파일 디스크립터
- buf: 버퍼 포인터
- count: 저장할 사이즈(바이트)
- offset: 저장할 위치(오프셋, 파일 시작 기준)

반환값
- 성공 시 실제로 파일에서 읽은 사이즈(바이트).
  - 단, count보다 작을 수 있음
- 실패 시 -1

파일 디스크립터 → 파일 포인터 변환

```
FILE *fdopen(int fd, const char *mode);
```

파라미터
- fd: 파일 디스크립터
- mode: 권한

반환값
- 성공 시 파일 포인터(stream)
- 실패 시 NULL

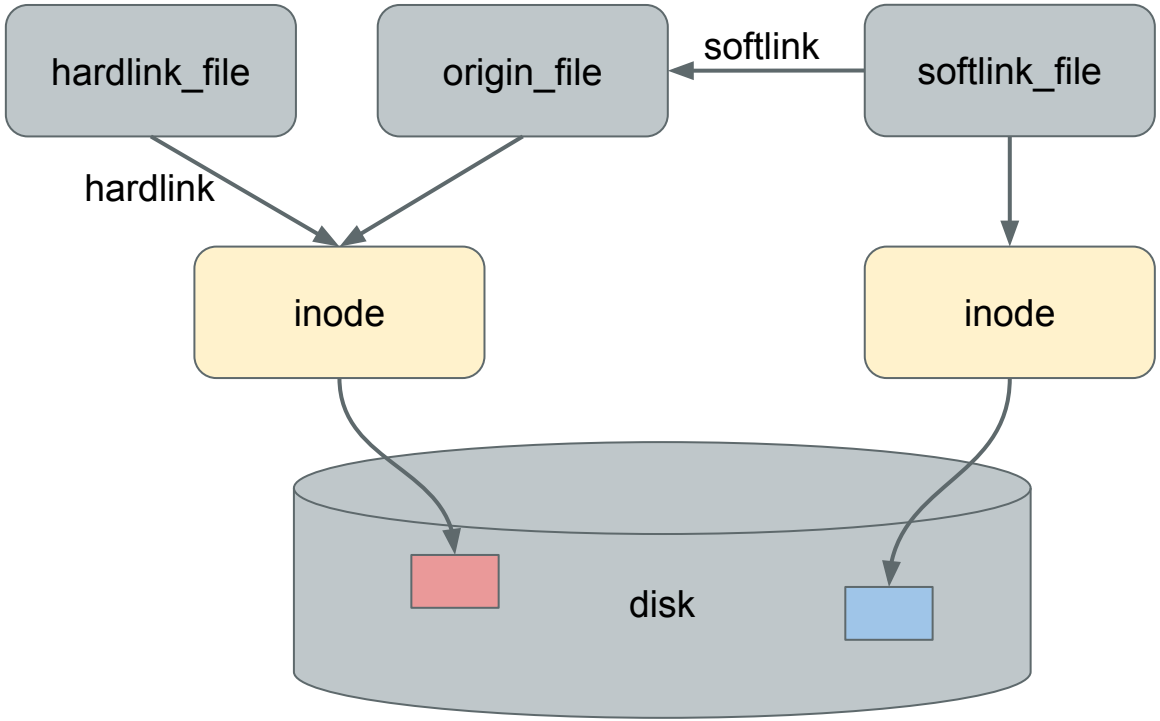
파일 포인터 → 파일 디스크립터 변환

```
int fileno(FILE *stream);
```

파라미터
- stream: 파일 포인터

반환값
- 성공 시 파일 디스크립터
- 실패 시 -1

Hardlink vs. Softlink



Hardlink vs. Softlink

hard link는…

- softlink에 비해 처리 속도가 빠름
- 다른 파일 시스템의 파일에 대해 생성 불가능
- 다른 파티션의 파일에 대해 생성 불가능
- 디렉터리에 대해 생성 불가능
- 인지하기 어려움

Hardlink 생성

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

파라미터
- oldpath: 원본 파일 경로
- newpath: 생성할 하드링크 경로

반환값
- 성공 시 0
- 실패 시 -1

Softlink 생성

```
int symlink(const char *target, const char *linkpath);
```

파라미터
- target: 원본 파일 경로
- linkpath: 생성할 소프트링크 경로

반환값
- 성공 시 0
- 실패 시 -1

Link 삭제

```
int unlink(const char *pathname);
```

파라미터
- pathname: 삭제할 파일 경로

반환값
- 성공 시 0
- 실패 시 -1

파일 속성 조회

```
int stat(const char *pathname, struct stat *statbuf);
int fstat(int fd, struct stat *statbuf);
int lstat(const char *pathname, struct stat *statbuf);
```

파라미터
- pathname: 파일 경로
- fd: 열린 파일 디스크립터
- statbuf: 파일 속성 버퍼

반환값
- 성공 시 0
- 실패 시 -1

파일 속성 조회

```
struct stat {
    dev_t     st_dev;         /* ID of device containing file */
    ino_t     st_ino;         /* Inode number */
    mode_t    st_mode;        /* File type and mode */
    nlink_t   st_nlink;       /* Number of hard links */
    uid_t     st_uid;         /* User ID of owner */
    gid_t     st_gid;         /* Group ID of owner */
    dev_t     st_rdev;        /* Device ID (if special file) */
    off_t     st_size;        /* Total size, in bytes */
    blksize_t st_blksize;     /* Block size for filesystem I/O */
    blkcnt_t  st_blocks;      /* Number of 512B blocks allocated */
    struct timespec st_atim;  /* Time of last access */
    struct timespec st_mtim;  /* Time of last modification */
    struct timespec st_ctim;  /* Time of last status change */
#define st_atime st_atim.tv_sec      /* Backward compatibility */
#define st_mtime st_mtim.tv_sec
#define st_ctime st_ctim.tv_sec
};
```

파일 속성 조회

Macro	Meaning
S_ISREG(statbuf.st_mode)	regular file?
S_ISDIR(statbuf.st_mode)	directory?
S_ISCHR(statbuf.st_mode)	character device?
S_ISBLK(statbuf.st_mode)	block device?
S_ISFIFO(statbuf.st_mode)	FIFO(named pipe)?
S_ISLNK(statbuf.st_mode)	symbolic link?
S_ISSOCK(statbuf.st_mode)	socket?

```
/* example */

stat(pathname, &sb);
if (S_ISREG(sb.st_mode)) {
     /* Handle regular file */
}
```

감사합니다.