Linux System Programming

by ProgCoach4U

파일 다루기 - basic

파일 열기

```
FILE *fopen(const char *pathname, const char *mode);
```

파라미터
- pathname: 파일 경로
- mode: 파일 열기 모드

반환값
- 성공 시 열린 파일 포인터(stream)
- 실패 시 NULL

파일 열기 모드

모드	읽기	쓰기	파일 포지션	파일 존재시	파일 부재시
“r”	O	X	파일 시작	성공	실패
“r+”	O	O	파일 시작	성공	실패
“w”	X	O	파일 시작	기존 파일 제거 후 생성	생성
“w+”	O	O	파일 시작	기존 파일 제거 후 생성	생성
“a”	X	O	파일 끝	성공	생성
“a+”	O	O	읽기 - 파일 시작 쓰기 - 파일 끝	성공	생성

파일 닫기

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

파라미터
- stream: 열린 파일 포인터

반환값
- 성공시 0
- 실패시 EOF

파일 포지션 - 현재 오프셋 가져오기

```
long ftell(FILE *stream);
```

파라미터
- stream: 열린 파일 포인터

반환값
- 성공시 파일 포지션(오프셋값)
- 실패시 -1

파일 포지션 - 오프셋 설정

```
int fseek(FILE *stream, long offset, int whence);
```

파라미터
- stream: 열린 파일 포인터
- 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 파일 포지션(오프셋값)
- 실패시 -1

파일 내용 읽고 쓰기

Text mode

- 문자를 저장할 때 사용되는 방식
- “10” 저장시 파일에는 0x31, 0x30이 저장됨
- fputc/fputs/fprintf
- fgetc/fgets/fscanf

Binary mode

- 데이터를 저장할 때 사용되는 방식
- 10 저장시 파일에 0x0a가 저장됨
- fwrite/fread

파일에 쓰기 - formatted

```
int fprintf(FILE *stream, const char *format, ...);
```

파라미터
- stream: 열린 파일 포인터
- format: 출력 포맷
- …: 가변 arguments

반환값
- 성공시 쓰여진 바이트값
- 실패시 음수

파일에서 읽기 - formatted

```
int fscanf(FILE *stream, const char *format, ...);
```

파라미터
- stream: 열린 파일 포인터
- format: 입력 포맷
- …: 가변 arguments

반환값
- 성공시 입력받은 아이템 개수
- 실패시 EOF

포맷 변환

변환자	
d	signed integer
u	unsigned integer, 10진수
o	unsigned integer, 8진수
x	unsigned integer, 16진수, (abcde)
X	unsigned integer, 16진수, (ABCDE)
s	NULL-terminated string
f	floating-point, [-]ddd.ddd
e	floating-point, [-]d.ddde±dd

integer 길이	
hh	sizeof(char)
h	sizeof(short)
(생략)	sizeof(int)
l (엘)	sizeof(long)
ll (엘엘)	sizeof(long long)

포멧 변환

데이터 타입	변환자
char	%hhd, %hhx
unsigned char	%hhu, %hhx
short	%hd, %hx
unsigned short	%hu, %hx
int	%d, %x
unsigned int	%u, %x
long	%ld(엘디), %lx(엘엑스)
unsigned long	%lu(엘유), %lx(엘엑스)
long long	%lld(엘엘디), %llx(엘엘에스)
unsigned long long	%llu(엘엘유), %llx(엘엘엑스)

파일에 쓰기 - character/string

```
int fputc(int c, FILE *stream);
```

파라미터
- c: 출력할 character
- stream: 열린 파일 포인터

반환값
- 성공시 쓰여진 character
- 실패시 EOF

```
int fputs(const char *s, FILE *stream);
```

파라미터
- s: 출력할 string(NULL-terminated)
- stream: 열린 파일 포인터

반환값
- 성공시 0보다 크거나 같은 수
- 실패시 EOF

파일에서 읽기 - character/string

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

파라미터
- stream: 열린 파일 포인터

반환값
- 성공시 읽은 character 값
- 실패시 EOF

```
char *fgets(char *s, int size,
            FILE *stream);
```

파라미터
- s: 스트링 버퍼
- size: 버퍼 사이즈
- stream: 열린 파일 포인터

반환값
- 성공시 캐릭터 포인터(스트링)
- 실패시 NULL

파일에 쓰기 - byte stream

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

파라미터
- ptr: 출력할 byte stream 포인터
- size: 출력 아이템 사이즈
- nmemb: 출력 아이템 개수
- stream: 열린 파일 포인터

반환값
- 실제로 출력된 아이템 개수
- 완전하게 쓰여졌는지 확인 필요

파일에서 읽기 - byte stream

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

파라미터
- ptr: 입력받을 버퍼 포인터
- size: 입력 아이템 사이즈
- nmemb: 입력 아이템 개수
- stream: 열린 파일 포인터

반환값
- 실제로 입력된 아이템 개수
- 완전하게 읽혀졌는지 확인 필요

감사합니다.