
강원혁신플랫폼
리눅스프로그래밍
리눅스(Linux)의 기본 구조와 파일 시스템

퀴즈

C언어 공유 라이브러리를 사용할 때 공유 라이브러리 경로를 지정하는 환경변수는 무엇인가?

LD_LIBRARY_PATH

학습목표

학습 내용

1. 라이브러리(Library)
2. 리눅스(Linux) 저수준 파일 관련 함수
3. 표준 입출력 함수

학습 목표

- 라이브러리 만들기로 직접 실행해 볼 수 있다.
- 리눅스 파일 시스템 기본 함수에 대해 설명할 수 있다.
- 표준 입출력 함수의 종류와 역할에 대해 파악할 수 있다.

라이브러리(Library)

라이브러리(Library)

라이브러리(Library)의 특징

- 자주 사용되는 루틴(함수)들의 모음으로 목적 코드를 병합해서 생성
- 하나의 큰 블랙 박스로, 큰 프로젝트는 여러 모듈로 나눠서 관리
- 정적(static) 라이브러리와 공유(Shared) 라이브러리로 구분
- 유닉스의 라이브러리들은 모두 lib로 시작

라이브러리(Library)

라이브러리 생성을 위한 유틸리티

01 02 03 04

ar 명령어

- 목적 파일들을 병합하여 라이브러리를 만들 때 사용

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

라이브러리(Library)

- 라이브러리 생성을 위한 유틸리티

01 02 03 04

ranlib 명령어

- 라이브러리에 포함된 객체들의 아카이브(archive) 인덱스 작성

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

라이브러리(Library)

라이브러리 생성을 위한 유틸리티

01 02 03 04

nm 명령어

- 라이브러리에서 해당 객체의 심볼(Symbol)과 연관된 이름이나 타입 등의 정보를 출력

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

라이브러리(Library)

- 라이브러리 생성을 위한 유틸리티

01 02 03 04

strip 명령어

- 목적 파일이나 실행 파일에서 불필요한 심볼을 제거해서 파일의 용량을 줄임

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

따라하기 실행 파일 구조 확인

파일 각 영역에 대한 크기 확인

```
$ size test
   text    data     bss     dec     hex filename
   1799     616       8    2423     977 test
```

함수의 위치 확인

```
$ nm test
0000000000003db8 d _DYNAMIC
0000000000003fa8 d _GLOBAL_OFFSET_TABLE_
0000000000002000 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
```

따라하기 실행 파일 구조 확인

```
… 중간 생략
0000000000011f3 T input
0000000000001189 T main
                 U malloc@GLIBC_2.2.5
00000000000011c5 T print
                 U printf@GLIBC_2.2.5
0000000000001100 t register_tm_clones
$
```

따라하기 라이브러리 만들기

목적 파일 생성

```
$ gcc -c input.c print.c
```

정적 라이브러리 생성

```
$ ar rs libmine.a input.o print.o
ar: creating libmine.a
```

정적 라이브러리 확인

```
$ ar tv libmine.a
rw-r--r-- 0/0   1584 Jan  1 09:00
1970 input.o
rw-r--r-- 0/0   1504 Jan  1 09:00
1970 print.o
```

따라하기 라이브러리 만들기

정적 라이브러리, 함수 삭제
$ ar d libmine.a input.o

파일 속성 확인
$ file libmine.a
libmine.a: current ar archive

따라하기 라이브러리 만들기

정적 라이브러리 확인

```
$ ar tv libmine.a
rw-r--r-- 0/0   1504 Jan  1 09:00
1970 print.o
```

정적 라이브러리, 함수 추가

```
$ ar rs libmine.a input.o
$ ar tv libmine.a
rw-r--r-- 0/0   1504 Jan  1 09:00
1970 print.o
rw-r--r-- 0/0   1584 Jan  1 09:00
1970 input.o
```

따라하기 라이브러리 만들기

정적 라이브러리, 함수 내용 확인

```
$ nm libmine.a

print.o:
0000000000000000 T print
                 U printf

input.o:
                 U __isoc99_scanf
0000000000000000 T input
                 U malloc
```

따라하기 라이브러리 만들기

libmine.a 탐색 실패

```
$ gcc -o test main.c -lmine -L.
/usr/bin/ld: skipping
incompatible ./libmine.so when
searching for -lmine
```

라이브러리 이용 컴파일 성공

```
$ gcc -o test main.c libmine.a -L.
```

따라하기 공유 라이브러리 만들기

공유 라이브러리 생성 및 실행
PIC: Position Independent Code
$ gcc -fPIC -c print.c input.c
$ gcc -shared -o libmine.so print.o input.o

확인을 위해 libmine.a 삭제
$ rm libmine.a

공유 라이브러리 이용 파일 생성
$ gcc -o test main.c -lmine -L.

따라하기 공유 라이브러리 만들기

공유 라이브러리를 위치 확인 오류

```
$ ./test
./test: error while loading shared libraries:
libmine.so: cannot open shared object file: No
such file or directory
```

공유 라이브러리를 위치 환경 변수 설정

```
$ export
LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
```

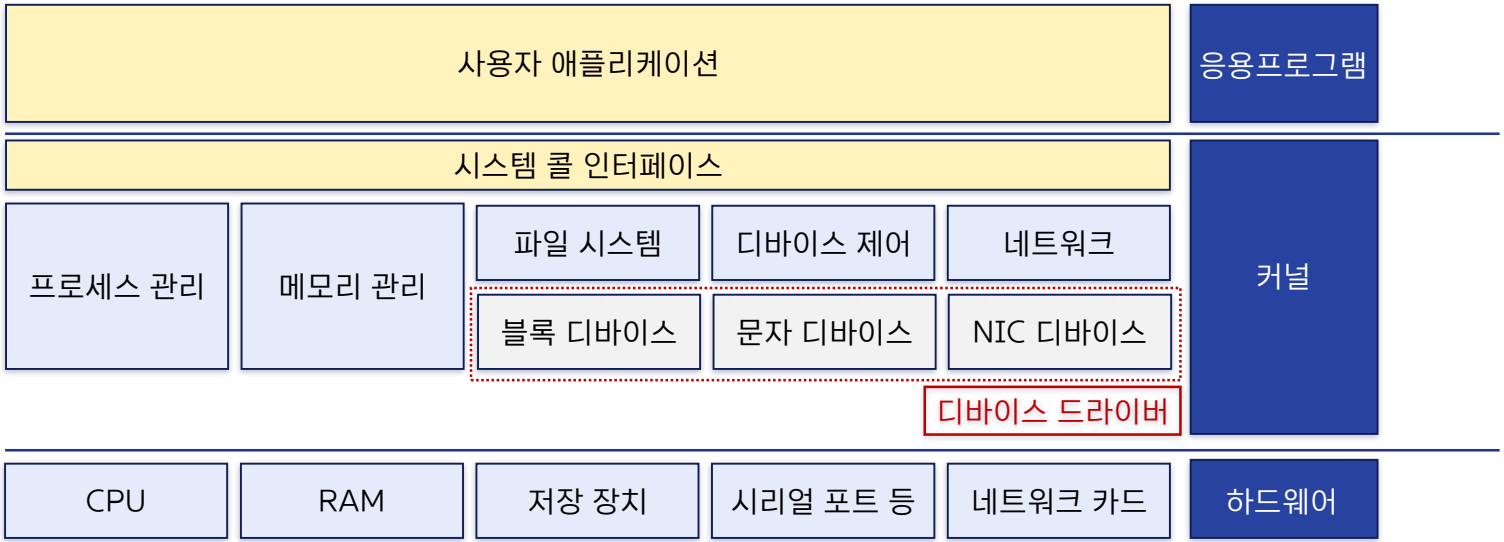
실행 확인

```
$ ./test
SharedLibTest
SharedLibTest
$
```

강원혁신플랫폼
리눅스프로그래밍

리눅스 저수준 파일 관련 함수

리눅스 커널의 블록도(Monolithic Kernel)



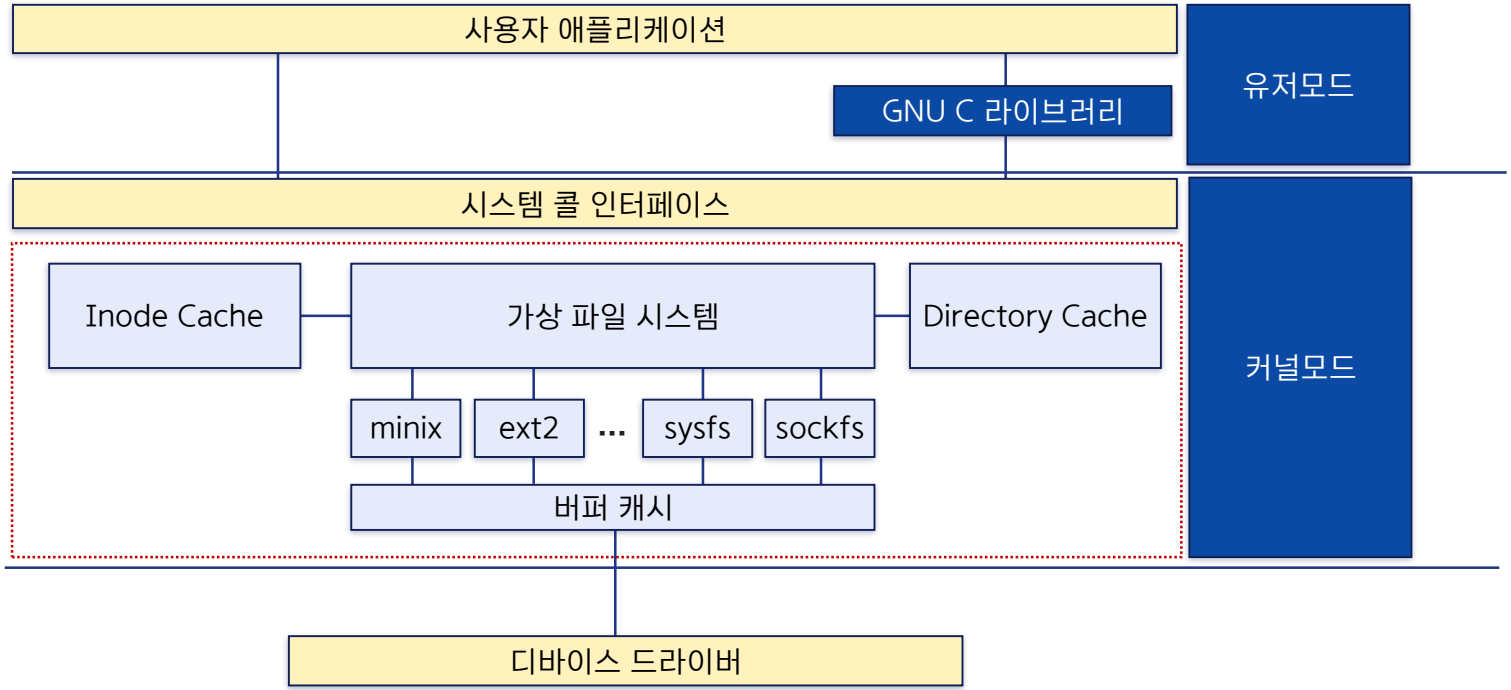
출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

유닉스의 파일 시스템 구성요소

파일	내용
루트 파일 시스템 (Root File System)	시스템 프로그램과 디렉터리들이 포함된 시스템 초기화 및 관리에 필요한 내용을 담고 있어 부팅에 꼭 필요한 파일 시스템이다. 하드디스크에 적어도 하나의 파일 시스템이 존재한다.
일반 파일 (Regular File)	수행 가능한 프로그램 파일이나 원시 프로그램, 파일 텍스트 파일, 데이터 파일 등 컴퓨터에 의해 처리될 수 있는 파일들이 저장된다.
디렉터리 파일 (Directory File)	다른 파일과 디렉터리에 관한 정보를 저장하는 논리적인 단위이며 계층 구조로 구성되는데, 파일명과 아이노드 번호를 연결한다.
특수 파일 (Special File)	디바이스들을 위한 파일이며 디바이스에 접근하기 위해 사용되는데, 명명된 파이프(named pipe) 파일, 심볼릭 링크 파일, 디바이스 파일 등이 있다.

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

리눅스의 파일 시스템 기본 구조



출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

유닉스의 파일 시스템 구성요소

함수	내용	비고
open( )	파일을 읽거나 쓰기 위해 열거나 생성한다.	
creat( )	비어있는 파일을 생성한다.	open( ) 함수로 대체 가능
close( )	열려있는 파일을 닫는다.	
read( )	열려있는 파일로부터 데이터를 읽어온다.	
write( )	열려있는 파일에 데이터를 저장한다.	
lseek( )	파일 포인터를 특정 위치로 이동한다.	
unlink( )	파일을 삭제한다.	
remove( )	파일이나 디렉터리를 삭제한다.	
fcntl( )	파일과 관련되어 있는 속성을 설정하거나 조정한다.	ioctl( )
dup( )	파일 디스크립터를 복사한다.	dup2( )

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

open( ) 함수의 플래그

플래그	내용	비고
O_RDONLY	읽기 전용	0
O_WRONLY	쓰기 전용	1
O_RDWR	읽기/쓰기 모두 가능	2
O_APPEND	쓰기 작업 수행 시 파일의 끝에 새로운 내용을 추가한다	
O_CREAT	파일이 없을 경우 파일을 생성한다	3번째 인자 사용
O_EXCL	파일이 있는 경우에 에러를 발생시킨다	O_CREAT와 함께 사용
O_TRUNC	기존의 파일의 내용이 있으면 지운다	
O_NONBLOCK	넌블로킹(Non-blocking) 모드로 전환한다	
O_SYNC	쓰기 연산마다 버퍼를 이용하지 않고, 변경된 내용을 바로 디스크에 저장한다	

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

open( ) 함수의 파일의 접근 권한

플래그	내용	값
S_IRWXU	파일의 소유자에게 읽고, 쓰고, 실행하는 모든 권한을 부여한다.	00700
S_IRUSR	파일의 소유자에게 읽기 권한을 부여한다.	00400
S_IWUSR	파일의 소유자에게 쓰기 권한을 부여한다.	00200
S_IXUSR	파일의 소유자에게 실행 권한을 부여한다.	00100
S_IRWXG	소유자가 속한 그룹에 읽고, 쓰고, 실행하는 모든 권한을 부여한다.	00070
S_IRGRP	소유자가 속한 그룹에 읽기 권한을 부여한다.	00040
S_IWGRP	소유자가 속한 그룹에 쓰기 권한을 부여한다.	00020
S_IXGRP	소유자가 속한 그룹에 실행 권한을 부여한다.	00010
S_IRWXO	다른 사람에게 읽고, 쓰고, 실행하는 모든 권한을 부여한다.	00007
S_IROTH	다른 사람에게 읽기 권한을 부여한다.	00004
S_IWOTH	다른 사람에게 쓰기 권한을 부여한다.	00002
S_IXOTH	다른 사람에게 실행 권한을 부여한다.	00001

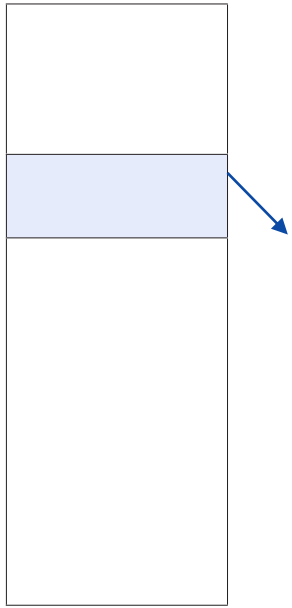
출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

PCB와 파일 디스크립터 테이블

프로세스 컨트롤 블록

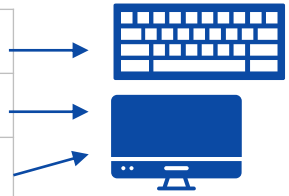
포인터	상태
프로세스 ID	
프로그램 카운터	
버퍼 포인터	
레지스터	
메모리 제한	
열린 파일 목록	
자식(Child)링크	
……	

Hold
Waiting
Ready
Running
Blocked
Terminated
Suspend
Child Blocked



플래그 파일 포인터

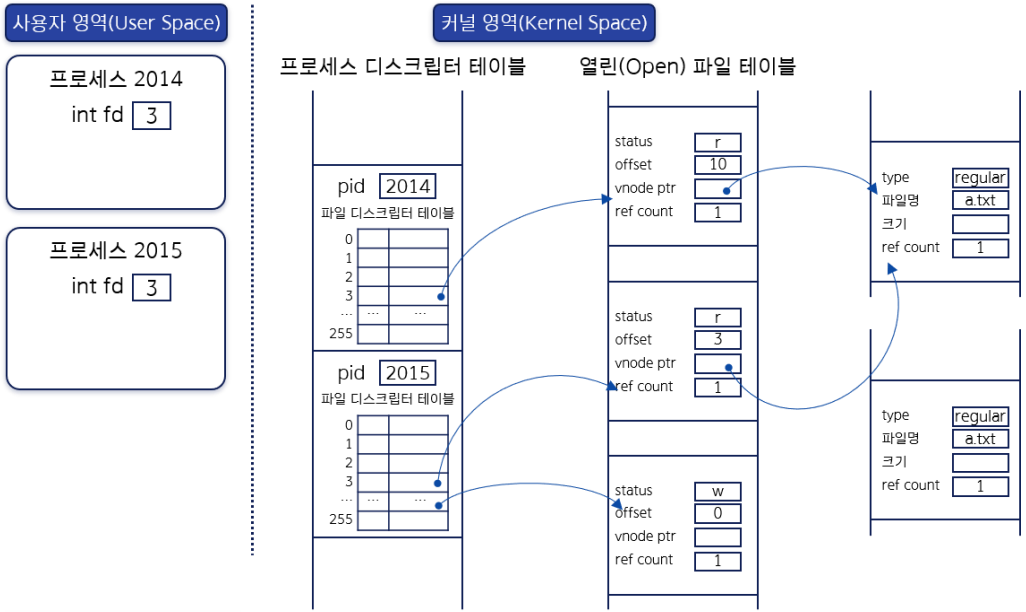
0	Stdin
1	Stdout
2	Stderr
3	
4	
5	
…	
255	

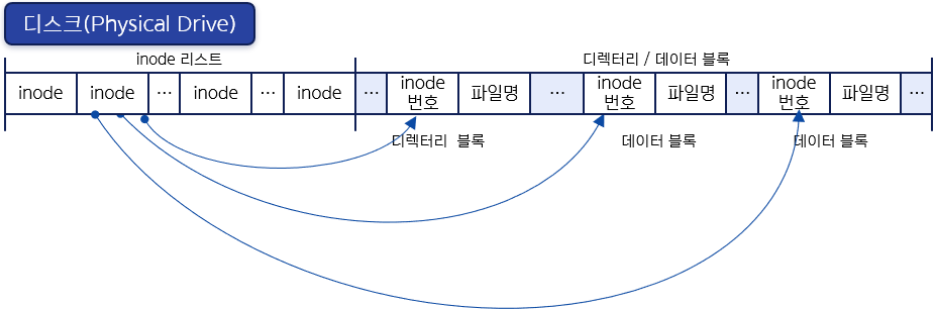


파일 디스크립터 테이블

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

프로세스와 파일


사용자 영역(User Space)
프로세스 2014
int fd 3
프로세스 2015
int fd 3
커널 영역(Kernel Space)
프로세스 디스크립터 테이블
열린(Open) 파일 테이블
pid 2014
파일 디스크립터 테이블
0
1
2
3
255
pid 2015
파일 디스크립터 테이블
status r
offset 10
vnode ptr
ref count 1
status r
offset 3
vnode ptr
ref count 1
status w
offset 0
vnode ptr
ref count 1
type regular
파일명 a.txt
크기
ref count 1
type regular
파일명 a.txt
크기
ref count 1


디스크(Physical Drive)
inode 리스트
디렉터리 / 데이터 블록
inode
inode
inode
inode
inode 번호
파일명
inode 번호
파일명
inode 번호
파일명
디렉터리 블록
데이터 블록
데이터 블록

lseek(), whence의 값

인자	내용	offset의 값
SEEK_SET	파일의 첫 부분에서부터의 상대적인 거리	양수 또는 0
SEEK_CUR	파일의 현재 오프셋으로부터의 상대적인 거리	양수 또는 음수
SEEK_END	파일의 마지막에서부터의 상대적인 거리	음수 또는 0

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

fcntl( ) 함수의 옵션

옵션	내용	비고
F_DUPFD	파일 디스크립터를 복사할 때 사용되며, 세 번째 인수보다 크거나 같은 값 중에 가장 작은 미사용 값을 반환한다.	
F_GETFD	파일 디스크립터의 플래그를 반환한다.	FD_CLOEXEC
F_SETFD	파일 디스크립터의 플래그를 설정한다.	
F_GETFL	파일 테이블에 저장되어 있는 파일 상태 플래그를 반환한다.	
F_SETFL	파일 상태 플래그(O_APPEND, O_NONBLOCK, O_SYNC 등)를 설정한다.	
F_GETOWN	SIGIO, SIGURG 시그널을 받는 프로세스 ID와 프로세스 그룹 ID를 반환한다.	
F_SETOWN	SIGIO, SIGURG 시그널을 받는 프로세스 ID와 프로세스 그룹 ID를 설정한다.	

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

fcntl( ) 함수의 잠금 인자

인자	내용
F_SETLK	flock 구조체의 내용에 따라 잠금을 설정하거나 해제한다.
F_SETLKW	잠금을 할 수 없으면 -1을 즉시 반환한다.
F_GETLK	잠금 설정을 확인하는데 S_SETLK와 같으나 잠금을 만들 수 없으면 대기한다. 잠금이 설정되어 있으면 flock 구조체의 l_type을 F_UNLOCK으로, l_whence를 SEEK_SET으로 설정하고 나머지 구조체 값은 변경하지 않고 전달한다.
F_RDLCK	읽기 잠금 설정
F_WRLCK	쓰기 잠금 설정
F_UNLCK	설정된 잠금 해제

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

따라하기 copy.c

```
#include <unistd.h>     /* 유닉스 표준(UNIX Standard) 시스템 콜을 위한 헤더 파일 */
#include <fcntl.h>
#include <stdio.h>      /* perror() 함수 */
#include <sys/stat.h>
#include <sys/types.h>

int main(int argc, char **argv) {
    int n, in, out;
    char buf[1024];
```

따라하기 copy.c

```
/* 명령행 인수로 복사할 파일명이 없는 경우에 에러를 출력하고 종료한다. */
  if (argc < 3) {
      write(2, "Usage : copy file1 file2\n", 25);
      return -1;
  }

  /* 복사의 원본이 되는 파일을 읽기 모드로 연다. */
  if ((in = open(argv[1], O_RDONLY)) < 0) {
      perror(argv[1]);
      return -1;
  }
```

따라하기 copy.c

```
 /* 복사할 결과 파일을 쓰기 모드(새로운 파일 생성 | 기존에 파일 내용 지움)로 연다. */
    if ((out = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC, S_IRUSR|S_IWUSR))
< 0) {
        perror(argv[2]);
        return -1;
    }

 /* 원본 파일의 내용을 읽어서 복사할 파일에 쓴다. */
    while ((n = read(in, buf, sizeof(buf))) > 0)
        write(out, buf, n);
```

따라하기 copy.c

```
    /* 열린 파일들을 닫는다. */
    close(out);
    close(in);

    /* 프로그램의 정상 종료 시 0을 반환한다. */
    return 0;
}
```

실행 확인

```
$ gcc -o copy copy.c
$ ./copy copy.c copy1.c
$ ls -al copy*c
-rw-r--r-- 1 freetime freetime 1209 Jul  8 11:26 copy.c
-rw------- 1 freetime freetime 1209 Jul  8 14:40 copy1.c
$
```

강원혁신플랫폼
리눅스프로그래밍

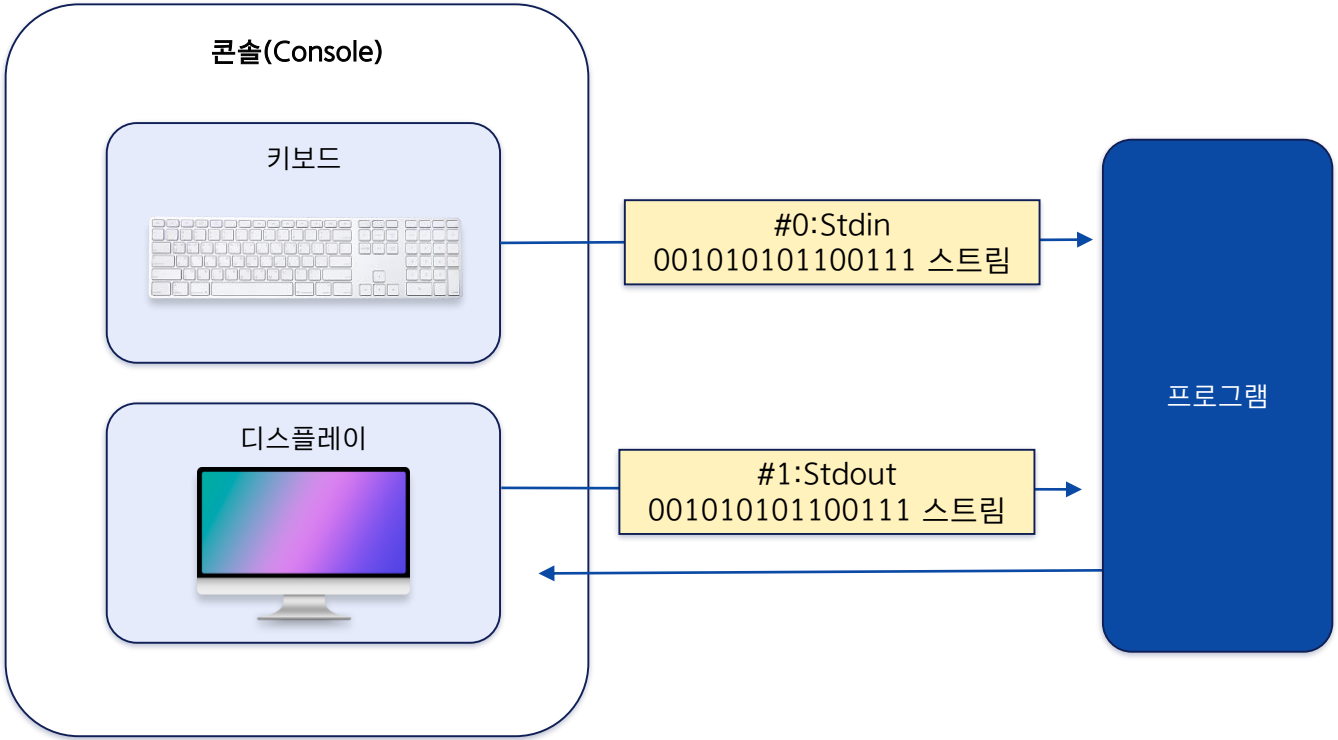
표준 입출력 함수

표준 입출력 함수

구분	입력	출력
문자 단위 입출력 함수	getc( ), fgetc( ) 등	putc( ), fputc( ) 등
줄 단위 입출력 함수	gets( ), fgets( ) 등	puts( ), fputs( ) 등
버퍼 기반의 입출력 함수	fread( ) 등	fwrite( ) 등
형식화된 입출력 함수	scanf( ), fscanf( ) 등	printf( ), fprintf( ) 등

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

스트림 의미와 표준 입출력 스트림



출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

fopen( ) 함수의 형태

타입	읽기	쓰기	파일생성	open( ) 함수와 비교	비고
r	O	X	X	O_RDONLY	rb
r+	O	O	X	O_RDWR	r + b / rb+
w	X	O	O	O_WRONLY|O_CREAT|O_TRUNC	wb
w+	O	O	O	O_RDWR|O_CREAT|O_TRUNC	w + b / wb+
a	X	O	O	O_WRONLY|O_CREAT|O_APPEND	ab
a+	O	O	O	O_RDWR|O_CREAT|O_APPEND	a + b / ab+

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

따라하기 copy.c

```c
#include <stdio.h>      /* 표준 입출력(Standard I/O) 함수를 위한 헤더 파일 */

int main(int argc, char **argv)
{
    int n;
    FILE *in, *out;
    char buf[BUFSIZ];

    /* 명령행 인수로 복사할 파일명이 없는 경우에 에러를 출력하고 종료한다. */
    if (argc != 3) {
        fprintf(stderr, "Usage: fcopy file1 file2\n");
        return -1;
    }
```

따라하기 copy.c

```
 /* 복사의 원본이 되는 파일을 읽기 모드로 연다. */
    if ((in = fopen(argv[1], "r")) == NULL) {
        perror(argv[1]);
        return -1;
    }

/* 복사할 결과 파일을 쓰기 모드(새로운 파일 생성 | 기존에 파일 내용 지움)로 연다. */
    if ((out = fopen(argv[2], "w")) == NULL) {
        perror(argv[2]);
        return -1;
    }

    /* 원본 파일에서 파일 내용을 읽어서 복사할 파일에 쓴다. */
    while ((n = fread(buf, sizeof(char), BUFSIZ, in)) > 0)
        fwrite(buf, sizeof(char), n, out);
```

따라하기 copy.c

```
 /* 열린 파일들을 닫는다. */
    fclose(out);
    fclose(in);

    return 0;
}
```

실행 확인

```
$ gcc -o fcopy fcopy.c
$ ./fcopy fcopy.c fcopy1.c
$ ls -al fcopy*c
-rw-r--r-- 1 freetime freetime 1052 Jul  8 11:26 fcopy.c
-rw-r--r-- 1 freetime freetime 1052 Jul  8 14:46 fcopy1.c
```

표준 라이브러리의 기타 함수

구분	함수명	내용	헤더 파일	비고
출력	printf( )	형식 지정 문자열을 콘솔(stdout)로 출력한다.	<stdio.h>	fprintf( )
	putc( )	스트림(표준입력장치 및 파일)으로 문자를 출력한다.	<stdio.h>	fputc( )
	putchar( )	콘솔(stdout)로 문자를 출력한다.	<stdio.h>	putc( )
	putch( )	콘솔(stdout)로 문자를 출력하지만, LF(Line Feed: 새 줄 문자)를 CR(Carriage Return) + LF로 변환할 수 없다.	<conio.h>	
	ungetc( )	스트림에 문자를 다시 입력으로 사용한다.	<conio.h>	
	puts( )	콘솔(stdout)로 문자열을 출력한다.	<stdio.h>	fputs( )
	sprintf( )	문자열 버퍼에 지정된 형식의 문자열을 출력한다.	<stdio.h>	

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

표준 라이브러리의 기타 함수

구분	함수명	내용	헤더 파일	비고
입력	scanf( )	콘솔(stdin)에서 형식에 맞게 데이터를 읽는다.	<stdio.h>	fscanf( )
	getc( )	스트림에서 문자를 입력받는다.	<stdio.h>	fgetc( )
	getchar( )	문자를 버퍼에 입력받고 화면에 출력한다.	<stdio.h>	getche( )
	getch( )	문자를 입력받고 화면에 출력하지 않는다.	<conio.h>	
	gets( )	콘솔(stdin)로부터 문자열을 입력받는다.	<conio.h>	fgets( )
	sscanf( )	문자열 버퍼에서 형식화된 데이터를 읽어온다.	<stdio.h>	

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

표준 라이브러리의 기타 함수

구분	함수명	내용	헤더 파일	비고
기타	freopen( )	파일 포인터가 가르키는 파일을 닫고 새로운 모드로 다시 연다.	<stdio.h>	
	fdopen( )	이미 열려있는 파일을 스트림과 연결시킨다.	<stdio.h>	
	setbuf( )	버퍼의 관리 방법을 변경한다.	<stdio.h>	setvbuf( )
	fileno( )	스트림의 파일 번호를 반환한다.	<stdio.h>	
	ferror( )	스트림의 에러 상태를 검사한다.	<stdio.h>	
	feof( )	스트림이 끝까지 도달했는지 검사한다.	<stdio.h>	
	clearerr( )	설정된 에러 정보를 지운다.	<stdio.h>	

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

요약정리

- 리눅스의 기본 구조와 파일 시스템
  - 리눅스 저수준 파일 관련 함수
  - 표준 입출력 함수