

오픈소스 소프트웨어 실습

Open-Source Software Lab

#4

페이지 단위로 파일 내용 보기 : more

\$ more 파일

파일(들)의 내용을 페이지 단위로 화면에 출력한다.

- * Enter를 누르면 출력물을 한 줄씩 진행
- * Spacebar를 누르면 출력물을 한 페이지씩 진행

```
> more input.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.

The Unix operating system was first developed in assembly language,
but by 1973 had been almost entirely recoded in C, greatly facilitating
its further development and porting to other hardware.
Today's Unix system evolution is split into various branches,
developed over time by AT&T as well as various commercial vendors,
universities (such as University of California, Berkeley's BSD),
and non-profit organizations.

The Open Group, an industry standards consortium, owns the UNIX trademark.
Only systems fully compliant with and certified according to the Single
UNIX Specification are qualified to use the trademark;
others might be called Unix system-like or Unix-like,
although the Open Group disapproves[1] of this term.
However, the term Unix is often used informally to denote any operating
system that closely resembles the trademarked system.

During the late 1970s and early 1980s, the influence of Unix in academic
circles led to large-scale adoption of Unix(particularly of the BSD variant,
--More-- (29%)
```

파일 앞부분 보기 : head

`$ head [-n] 파일`

파일(들)의 앞부분을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

```
❖ head input.txt
```

```
Unix is a multitasking, multi-user computer operating system originally  
developed in 1969 by a group of AT&T employees at Bell Labs, including  
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,  
and Joe Ossanna.
```

```
The Unix operating system was first developed in assembly language,  
but by 1973 had been almost entirely recoded in C, greatly facilitating  
its further development and porting to other hardware.
```

```
Today's Unix system evolution is split into various branches,  
developed over time by AT&T as well as various commercial vendors,
```

```
❖ head -5 input.txt
```

```
Unix is a multitasking, multi-user computer operating system originally  
developed in 1969 by a group of AT&T employees at Bell Labs, including  
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,  
and Joe Ossanna.
```

파일 뒷부분 보기 : tail

```
$ tail [-n] 파일
```

파일(들)의 뒷부분을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

```
❯ tail input.txt
```

```
Linux, which is used to power data centers, desktops, mobile phones,  
and embedded devices such as routers, set-top boxes or e-book readers.  
Today, in addition to certified Unix systems such as those already  
mentioned, Unix-like operating systems such as MINIX, Linux, Android,  
and BSD descendants (FreeBSD, NetBSD, OpenBSD, and DragonFly BSD) are  
commonly encountered.
```

```
The term traditional Unix may be used to describe a Unix or  
an operating system that has the characteristics of either Version 7  
Unix or UNIX System V. x2❯
```

```
❯
```

파일 복사 : cp_(copy)

```
$ cp [-i] 파일1 파일2
```

파일 1을 파일 2에 복사한다. -i는 대화형 옵션이다.

- * 복사 대상 파일과 이름이 같은 파일이 이미 존재하면 덮어쓰기(overwrite)
- * 보다 안전한 사용법 : 대화형 -i(interactive) 옵션을 사용

```
> cp input.txt input1.txt
> ls
2021012345.txt  cpfile  cpfile.c  input1.txt  input.txt  main.c
> more input1.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.

The Unix operating system was first developed in assembly language,
but by 1973 had been almost entirely recoded in C, greatly facilitating
its further development and porting to other hardware.
Today's Unix system evolution is split into various branches,
developed over time by AT&T as well as various commercial vendors,
universities (such as University of California, Berkeley's BSD),
and non-profit organizations.

The Open Group, an industry standards consortium, owns the UNIX trademark.
Only systems fully compliant with and certified according to the Single
UNIX Specification are qualified to use the trademark;
others might be called Unix system-like or Unix-like,
although the Open Group disapproves[1] of this term.
However, the term Unix is often used informally to denote any operating
system that closely resembles the trademarked system.


During the late 1970s and early 1980s, the influence of Unix in academic
circles led to large-scale adoption of Unix(particularly of the BSD variant,
--More-- (29%)
```

파일 이동 : mv_(move)

```
$ mv [-i] 파일1 파일2
```

파일 1의 이름을 파일 2로 변경한다. -i는 대화형 옵션이다

```
> ls
2021012345.txt  cpfile  cpfile.c  input1.txt  input.txt  main.c
> mv input1.txt input2.txt
> ls
2021012345.txt  cpfile  cpfile.c  input2.txt  input.txt  main.c
> |
```



파일 삭제 : `rm`(remove)

```
$ rm [-i] 파일
```

파일(들)을 삭제한다. -i는 대화형 옵션이다.

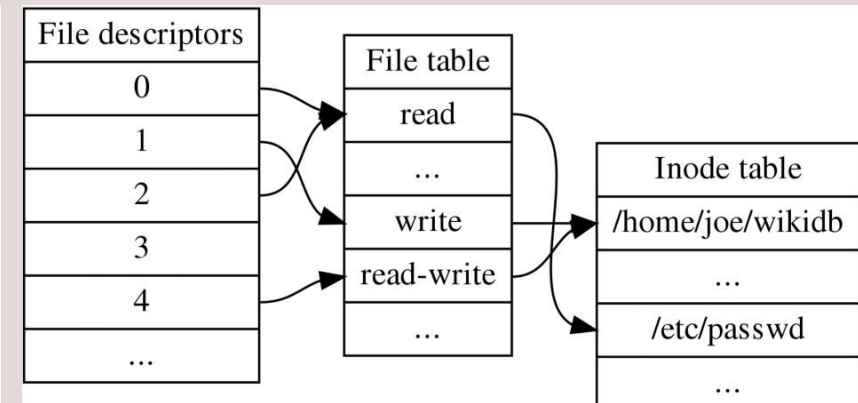
```
> ls
2021012345.txt  cpfile  cpfile.c  input2.txt  input.txt  main.c
> rm input2.txt
> ls
2021012345.txt  cpfile  cpfile.c  input.txt  main.c
> |
```

File descriptor

In Unix and related computer operating systems, a file descriptor (FD, less frequently fildes) is an abstract indicator (handle) used to access a file or other input/output resource, such as a pipe or network socket.

A file descriptor is a non-negative integer, generally represented in the C programming language as the type `int` (negative values being reserved to indicate "no value" or an error condition).

Integer value	Name	<code><unistd.h></code> symbolic constant ^[1]	<code><stdio.h></code> file stream ^[2]
0	Standard input	STDIN_FILENO	stdin
1	Standard output	STDOUT_FILENO	stdout
2	Standard error	STDERR_FILENO	stderr



System calls for file io

open(2) — Linux manual page

SYNOPSIS

```
#include <sys/stat.h>
#include <fcntl.h>

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

close(2)

SYNOPSIS

```
#include <unistd.h>

int close(int fd);
```

파일 디스크립터를 닫음. 즉, 파일 사용을 끝냈음을 시스템에게 알림

creat(3p)

SYNOPSIS

```
#include <sys/stat.h>
#include <fcntl.h>

int creat(const char *path, mode_t mode);
```

새 파일 생성시 사용

open 에서 O_CREAT|O_WRONLY|O_TRUNC flag와 같음

read(2)

SYNOPSIS

```
#include <unistd.h>

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

파일로 부터 임의의 byte를 버퍼로 복사하는데 사용

write(2)

SYNOPSIS

```
#include <unistd.h>

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

버퍼로 부터 임의의 byte를 파일에 쓰는데 사용

FLAG

O_RDONLY	읽기 전용
O_WRONLY	쓰기 전용
O_RDWR	읽기 쓰기
O_CREAT	파일이 존재 하지 않으면 생성
O_EXCL	O_CREAT와 함께 사용되며 파일이 존재 시 에러처리
O_TRUNC	파일이 존재 시 잘라버림
O_APPEND	파일의 뒷부분에 추가

MODE

미리 정의된 mode들이 존재 한다. (검색)
실습에서는 PERM 0644 -rw-r--r-- 직접 부여

Simple copy_file() 구현하기

```
oswTA@sseojinn:~$ cd ..  
oswTA@sseojinn:/home$ ls  
cpfile.c      osw2020004457  osw2021013563  osw2022020691  osw2022079870  
osw2018044420  osw2020012888  osw2021016680  osw2022032860  osw2022085532  
osw2018044857  osw2020013027  osw2021020091  osw2022035914  osw2022086353  
osw2018045650  osw2020016553  osw2021020691  osw2022045105  osw2022086462  
osw2018050855  osw2020021621  osw2021022979  osw2022046317  osw2022087910  
osw2019005714  osw2020024657  osw2021025650  osw2022049734  osw2022088031  
osw2019014066  osw2020056771  osw2021029789  osw2022050300  osw2022089798  
osw2019024702  osw2020069507  osw2021064911  osw2022052342  osw2022091685  
osw2019033054  osw2020071658  osw2021090419  osw2022053209  osw2022093309  
osw2019067501  osw2020072724  osw2021095978  osw2022055014  osw2022094075  
osw2019071221  osw2020076317  osw2021099370  osw2022055550  osw2022098822  
osw2019081067  osw2020079752  osw2022004775  osw2022059543  oswTA  
osw2019084502  osw2020084102  osw2022005269  osw2022059689  sseojinn  
osw2019091576  osw2020090064  osw2022008604  osw2022069598  ywc  
osw2019099961  osw2021006071  osw2022010746  osw2022077001
```

```
oswTA@sseojinn:/home$ vim cpfile.c
```

Simple copy_file() 구현하기

```
oswTA@sseojinn: /home
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define BUFSIZE 2048
#define PERM 0644

int main(){
    int fd1, fd2, n;
    char buf[BUFSIZE];
    //input check
    if(argc != 3){
        perror("arg error");
        exit(1);
    }
    //open file
    if((fd1=open()) < 0){
        perror("fd1 open err");
        exit(1);
    }
    //creat file but, use open() not creat()
    //check your file mode
"cpfile.c" 37L, 568C
```

1. main 함수에 인자전달하기 (ex ./simplecp (from) (to))
2. 인자로 전달받은 (from)을 open()으로 열기
3. 인자로 전달받은 (to)를 새로운 파일로 생성하기
4. (from)에서 read()한 내용을 (to)로 붙여넣기

Simple copy_file() 구현하기

```
oswTA@sseojinn:~$ gcc cpfile.c -o simplecp
oswTA@sseojinn:~$ ls
a.out  cpfile.c  examples.desktop  simplecp  test.c
oswTA@sseojinn:~$ history > hlist.txt
oswTA@sseojinn:~$ ./simplecp hlist.txt hlist2.txt
oswTA@sseojinn:~$ ls
a.out  cpfile.c  examples.desktop  hlist2.txt  hlist.txt  simplecp  test.c
```

```
oswTA@sseojinn:~$ cat hlist.txt
1  history
2  ls
3  ./a.out
4  history
5  exit
6  su
7  ls
8  cd ..
9  ls
10 pwd
11 cd ..
12 ls
13 cd home
14 ls
15 cd osw2018044420
16 ls
17 cat hlist.txt
18 cd ..
19 ls
20 exit
21 ls
22 cp cpfile.c /oswTA/test.c
```

```
oswTA@sseojinn:~$ cat hlist2.txt
1  history
2  ls
3  ./a.out
4  history
5  exit
6  su
7  ls
8  cd ..
9  ls
10 pwd
11 cd ..
12 ls
13 cd home
14 ls
15 cd osw2018044420
16 ls
17 cat hlist.txt
18 cd ..
19 ls
20 exit
21 ls
22 cp cpfile.c /oswTA/test.c
```

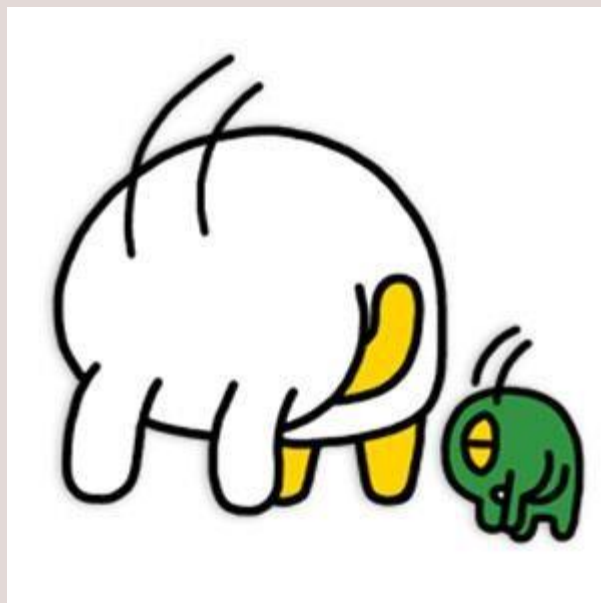
실습 과제

- 뼈대코드인 cpfile.c의 내용을 기반으로 한 파일의 내용을 다른 파일로 복사하는 simplecp.c를 구현
-지난 시간 만들었던 hlist.txt를 복사할 것.
- 본인의 홈 디렉토리(/home/osw학번)의 상위 디렉토리에 있는(/home) cpfile.c를 **복사**하여 자신의 홈 디렉토리에 저장
- 원본 파일을 RDONLY로 열기
- 대상 파일을 생성 (create 대신 open 사용)
- 원본 파일의 끝에 도달할 때까지 파일을 읽어 대상 파일에 기록
- 두 파일 모두 닫음
- 컴파일한 실행파일이 simplecp일 경우
./simplecp input.txt result.txt
명령이 수행되어야 함

simplecp.c 작성 화면과 실행 결과화면(총2개)을 캡처하여 양식 맞춰 메일

실습 담당 조교 연락처 및 과제 제출 양식

- ◆ 실습 조교 : 정만성
- ◆ 연구실 : 학연산클러스터 604호
- ◆ 메일제목 : 오픈소스SW_화(or수)_2021012345_정만성_*주차실습
- ◆ 이메일 : wjdakstjd@hanyang.ac.kr
- ◆ 주의 : 메일 및 양식을 복사/붙여넣기 활용하여 올바르게 보낼 것
(양식에 맞지 않거나, 잘못된 메일주소로 보낼 시 결석 처리)



끝