**로그인 / 로그아웃**

**telnet** : 유닉스 시스템에 접속

**[option]**

**[ex]**

* telnet hanb.co.kr

**logout / exit** : 유닉스 시스템에서 접속 해제

**프로세스**

**ps** : 현재 실행 중인 프로세스의 정보를 출력

**[option]**

-ef : 모든 프로세스에 대한 상세 정보 출력

**kill** : 프로세스 강제 종료

**[option]**

-9 : 강제 종료

**[ex]**

- kill 5000

- kill -9 5001

**파일 / 디렉토리 조작 명령**

**pwd** : 현재 디렉토리 경로 출력

**ls** : 디렉토리 내용 출력

**[option]**

- a : 숨김 파일 출력

- l : 파일 상세정보 출력

**cd** : 현재 디렉토리 변경

**cp** : 파일 / 디렉토리 복사

**[option]**

- r : 디렉토리 복사

**mv** : 파일 / 디렉토리 이름변경과 이동

**[ex]**

- mv a.txt b.txt // a.txt를 b.txt로 변경

- mv a.txt dir1 // a.txt를 dir1으로 이동

- mv dir1 dir2 // dir1을 dir2로 이동

**rm** : 파일 / 디렉토리 삭제

**[option]**

- r : 디렉토리 삭제

**[ex]**

- rm a.txt // a.txt 삭제

- rm -r dir1 // dir1 삭제

**mkdir** : 디렉토리 생성

- mkdir [디렉토리 이름]

**rmdir** : 빈 디렉토리 삭제

- rmdir [디렉토리 이름]

**cat** : 파일 내용 출력

- cat [파일 이름]

**more** : 파일 내용을 쪽단위로 출력

- more [파일 이름]

**chmod** : 파일 접근권한 변경

- chmod 755 a.exe

- chmod go+x a.exe

**grep** : 패턴 검색

- grep abcd a.txt // abcd 패턴이 있는지 확인?

**vi 편집기 내부 명령**

**입력 모드 전환**

- i, a, o, O

**커서 이동**

- j, k, h, l, 방향키

**한 글자 수정**

- r

**단어 수정**

- cw

**검색하여 수정**

- :%s/aaa/bbb/g -> aaa를 bbb로 모두(g) 바꾼다

**붙이기**

- p

**글자 삭제**

- x

**저장하고 종료**

- :wq!

- ZZ (대문자)

**행 붙이기**

- J (대문자)

**행 번호 보이기**

- :set nu

**명령 모드 전환**

- [Esc]

**행 이동**

- :행 번호

- #[행번호]G

**여러 글자 수정**

- #[글자수]s

**명령 취소**

- u

- U

**복사**

- #[복사할 줄 수]yy

**커서 이후 삭제**

- D (대문자)

**행 삭제**

- dd

**저장하지 않고 종료**

- :q!

**화면 다시 표시**

- ctrl + l

**행 번호 없애기**

- :set nonu

**기타 명령**

**su** : 사용자 계정 변경

**[ex]**

- su root

- su – han02

**tar** : 파일 / 디렉토리 묶기

**[option]**

- cvf : tar 파일 생성

- tvf : tar 파일 내용 보기

- xvf : tar 파일 풀기

**[ex]**

- tar cvf a.tar \* // 현재 디렉토리 안의

모든 파일을 a.tar로 묶는다.

- tar tvf a.tar // a.tar의 파일 목록을 본다.

- tar xvf a.tar // a.tar파일을 푼다.

**whereis** : 파일 위치 검색

- whereis ls

**which** : 파일 위치 검색

- which telnet

**Makefile과 make 사용하기**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ex1\_3/ex1\_3\_main.c\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<stdio.h>

extern int addnum(int a, int b);

int main(void){

    int sum;

    sum = addnum(1, 5);

    printf("Sum 1~5 = %d\n", sum);

    return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ex1\_3/ex1\_3\_addnum.c\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int addnum(int a, int b){

    int sum = 0;

    for(; a<=b; a++)

        sum += a;

    return sum;

}

**Makefile, make를 사용하지 않을 때의 실행방법**

$ gcc -o ex1\_3\_addnum.o ex1\_3\_addnum.c

$ gcc -o ex1\_3\_main.o ex1\_3\_main.c

$ gcc -o add ex1\_3\_addnum.o ex1\_3\_main.o

$ ./add

**[요약]** 하나하나 전부 오브젝트(.o)를 만들어 실행시켜야 한다. -> 번거로움

**Makefile과 make를 사용하여 오브젝트 만들기**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ex1\_3/Makefile\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Makefile

CC=gcc

CFLAGS=

OBJS=ex1\_3\_main.o ex1\_3\_addnum.o

LIBS=

all:    add

add:    $(OBJS)

    $(CC) $(CFLAGS) -o add $(OBJS) $(LIBS)

ex1\_3\_main.o: ex1\_3\_main.c

    $(CC) $(CFLAGS) -c ex1\_3\_main.c

ex1\_3\_addnum.o: ex1\_3\_addnum.c

    $(CC) $(CFLAGS) -c ex1\_3\_addnum.c

clean:

    rm -f$(OBJS) add core

**[주의 사항]** $가처음에 붙는 경우 스페이스가 아닌 탭으로 구분지어야 한다.

$ make 또는 $make -f Makefile

$ ./add

**2주차**

**저 수준 파일 입출력**

- 파일 지시자는 파일기술자 (file descriptor)

- 유닉스 커널의 시스템 호출을 사용하여 파일 입출력 실행

- 파일에 빠르게 접근 가능

- 바이트 단위로 파일의 내용을 다룬다.

- 특수 파일도 읽고 쓸 수 있다.

- 바이트 단위를 적당한 형태의 데이터로 변환하는 함수 등이 필요하다.

- open, close, read, write, lseek, dup, dup2, fcntl, fsync

**고 수준 파일 입출력**

- 파일 지시자는 파일 포인터 (file pointer)

- C언어의 표준 함수로 제공되며, 여러가지 형태의 데이터 형식을 지원함

- 버퍼 단위로 읽고 쓰기가 가능함

- fopen, fclose, fread, fwrite, fputs, fgets, fprintf, fscanf, fseek

★ 저 수준은 버퍼가 없어서 빈번하게 일어나는 작은 데이터 처리에 있어서 오버헤드가 발생할 수 있다.

★ 고 수준은 버퍼가 있어서 작은 단위들을 한 번에 처리할 수 있어서 오버헤드 발생을 줄여준다.

**파일 기술자**

- 현재 열려있는 파일을 구분하는 정수값

- 저수준 파일 입력에서 열린 파일을 참조하는데 사용

**값**

- 0 : 표준 입력,

- 1 : 표준 출력,

- 2 : 표준 오류

**파일 열기 : open**

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int open(const char \*path, int oflag [, mode\_t mode]);

**매개변수**

- **path** : 경로와 이름

- **oflag** : 파일 옵션

- O\_RDONLY : 파일을 읽기 전용으로 연다.

- O\_WRONLY : 파일을 쓰기 전용으로 연다.

- O\_RDWR | O\_CREAT : 파일을 읽기와 쓰기가 가능하게 연다.

- O\_CREAT : 파일이 없으면 파일을 생성한다

- O\_EXCL | O\_CREAT : O\_CREAT 옵션과 함께 사용할 경우 기존에 없는 파일이면 파일을 생성하지만, 파일이 이미 있으면 파일을 생성하지 않고 -1을 반환한다..

- O\_APPEND : 파일의 맨 끝에 내용을 추가한다.

- O\_TRUNC | O\_CREAT : 파일을 생성할 때 이미 있는 파일이고 쓰기 옵션으로 열었으면 내용을 모두 지우고 파일의 길이를 0으로 변경한다.

- O\_NONBLOCK/O\_NDELAY : 비블로킹(Non-blocking) 입출력

- O\_SYNC/O\_DSYNC : 저장장치에 쓰기가 끝나야 쓰기 동작을 완료

- **mode** : 파일 접근 권한 지정, 0644같이 숫자나 플래그 값으로 지정 가능

- S\_IRWXU 0700 소유자 읽기/쓰기/실행 권한

- S\_IRUSR 0400 소유자 읽기 권한

- S\_IWUSR 0200 소유자 쓰기 권한

- S\_IXUSR 0100 소유자 실행 권한

- S\_IRWXG 0070 그룹 읽기/쓰기/실행 권한

- S\_IRGRP 0040 그룹 읽기 권한

- S\_IWGRP 0020 그룹 쓰기 권한

- S\_IXGRP 0010 그룹 실행 권한

- S\_IRWXO 0007 기타 사용자 읽기/쓰기/실행 권한

- S\_IROTH 0004 기타 사용자 읽기 권한

- S\_IWOTH 0002 기타 사용자 쓰기 권한

- S\_IXOTH 0001 기타 사용자 실행 권한

**파일 닫기 : close**

#include <unistd.h>

int close(int fildes);

- 프로세스에서 열 수 있는 파일의 개수가 제한되어 있으므로 파일의 사용이 끝나면 닫아야 한다.

**파일 읽기 : read**

#include <unistd.h>

ssize\_t read(int fildes, void \*buf, size\_t nbytes);

- 파일에서 nbytes로 지정한 크기만큼 바이트를 읽어서 buf에 저장

- 실제로 읽어온 바이트 개수 리턴

- 리턴값이 0이면 파일의 끝에 도달했음을 의미

- fildes를 읽어 buf에 nbytes만큼 저장, 읽어온 바이트 수 반환

**매개변수**

- fildes : 읽을 파일

- \*buf : 읽은 내용을 저장할 장소

- nbytes : 몇 바이트씩 읽을 것인지

**반환값**

- 읽은 바이트 수

**파일 쓰기 : write**

#include <unistd.h>

ssize\_t read(int fildes, void \*buf, size\_t nbytes);

- buf가 가리키는 메모리에서 nbytes로 지정한 크기만큼 파일에 기록

- 실제로 쓰기를 수행한 바이트 수를 리턴

- fildes에 nbytes만큼 buf에 저장, 기록한 바이트 수 반환

**매개변수**

- fildes : 쓸 파일

- \*buf : 읽을 파일

- nbytes : 몇 바이트씩 쓸 것인지

**반환값**

- 쓴 바이트 수

**파일 오프셋 위치 지정 : lssek**

#include <sys/types.h>

#include <unistd.h>

off\_t lseek(int fildes, off\_t offset, int whence);

- offset으로 지정한 크기만큼 오프셋을 이동시킨다.

- fildes의 whence를 기준으로 offset만큼 이동시킨다.

**매개변수**

- fildes : 오프셋을 지정할 파일

- offset : 이동할 오프셋 수

- whence : 이동시킬 기준

\* SEEK\_SET : 파일의 시작 기준

\* SEEK\_CUR : 현재 위치 기준

\* SEEK\_END : 파일의 끝 기준

**반환값**

- 현재 위치

**파일 기술자 복사 : dup**

#include <unistd.h>

int dup(int fildes);

- 기존 파일 기술자를 인자로 받아 새로운 파일 기술자 리턴

- 새로운 파일 기술자는 현재 할당할 수 있는 파일 기술자 중 가장 작은 값으로 자동할당

**파일 기술자 복사 : dup2**

#include <unistd.h>

int dup2(int fildes);

- 새로운 파일 기술자를 지정

**파일 기술자 제어 : fcntl**

- 파일 기술자가 가리키는 파일에 cmd로 지정한 명령을 수행

- fcntl 시스템호출은 이미 열려있는 파일의 특성 제어를 위해서 사용된다.

- cmd의 종류에 따라 인자(args)를 지정할 수 있음

#include <sys/types.h>

#include <unistd.h>

#include <fcntl.h>

int fcntl(int fd, int cmd, /\* arg \*/ ...);

**CMD 종류**

- F\_GETFD : fd대한 flag값을 반환한다. (FD\_CLOEXEC 정보만 반환)

- F\_SETFD : fd의 FD\_CLOEXEC의 값을 지정된 비트값으로 설정한다.

- F\_GETFL : fd에 대한 파일 상태 속성들을 반환,

- O\_RDONLY, O\_WRONLY, O\_RDWR, O\_EXEC, O\_SEARCH는 개별적으로 판정할 수 없고

- **ACCMODE**라는 마스크를 이용하여 알아낼 수 있다.

- F\_SETFL : 파일 상태 속성들을 설정

- **O\_APPEND, O\_NONBLOCK, O\_SYNC, O\_DSYNC, O\_RSYNC, O\_FSYNC, O\_ASYNC** 속성만 변경가능

- F\_GETOWN : 이것은 비동기 입출력과 관련되어서 사용되며, SIGIO와 SIGURG 신호를 받는 프로세스

아이디를 얻기 위해서 사용된다.

- F\_SETOWN : 비동기 입출력과 관련되어서 사용되며, SIGIO, SIGURG 시그널을 수신하는 프로세스 아이디

(혹은 그룹)을 설정하기 위해서 사용된다.

**FD\_CLOEXEC란?**

**자세한 내용**

<https://www.ibm.com/docs/en/aix/7.2?topic=files-fcntlh-file>

**파일 삭제 : unlink**

// unlink

#include <unistd.h>

int unlink(const char \*path);

▪ path에 지정한 파일의 inode에서 링크 수를 감소시킨다.

▪ 링크 수가 0이 되면 path에 지정한 파일이 삭제된다.

▪ 파일 뿐만 아니라 디렉토리(빈 디렉토리 아니어도 됨)도 삭제된다.

**파일 삭제 : remove**

// remove

#include <stdio.h>

int remove(const char \*path);

▪ path에 지정한 파일이나 디렉토리를 삭제한다.

▪ 디렉토리인 경우 빈 디렉토리만 삭제한다.

**fsync**

int fsync(int filedes);

– 메모리에 위치하고 있는 파일의 내용을 디스크로 보내 메모리와 디스크의 내용을 동기화한다.

- 메모리의 내용이 디스크로 모두 기록되기 전에는 리턴하지 않는다

**고수준 파일 입출력 : 표준 입출력 라이브러리**

**파일 열기 : fopen**

#include <stdio.h>

FILE \*fopen(const char \*filename, const char \*mode);

- filename으로 지정한 파일을 mode로 지정한 모드에 따라 열고 파일 포인터를 리턴

**mode 값**

- r : 읽기 전용

- w : 새로 쓰기용, 기존 내용은 삭제

- a : 추가용

- rb : 읽기 전용, 바이너리

- wb : 새로 쓰기용, 바이너리, 기존 내용 삭제

- ab : 추가용, 바이너리

- r+ : 읽기와 쓰기용

- w+ : 쓰기와 읽기용

- a+ : 추가와 읽기용

- rb+ : 읽기와 쓰기용, 바이너리

- wb+ : 쓰기와 읽기용, 바이너리

- ab+ : 추가와 읽기용, 바이너리

**파일 닫기 : fclose**

#include <stdio.h>

int fclose(FILE \*stream);

**문자기반 입출력**

**문자 기반 입력 함수 : fgetc, getc, getchar, getw**

#include <stdio.h>

int fgetc(FILE \*stream); // 문자 한 개를 unsigned char 형태로 읽어온다.

int getc(FILE \*stream); // 매크로

int getchar(void); // 매크로

int getw(FILE \*stream); // 워드 단위로 읽어온다.

**문자 기반 출력 함수 : fputc, putc, putchar, putw**

#include <stdio.h>

int fputc(int c, \*stream);

int putc(int c, \*stream);

int putchar(int c);

int putw(int w, FILE \*stream);

**문자열기반 입출력**

**문자열 기반 입력 함수 : gets, fgets**

#include <stdio.h>

char \*gets(char \*s); // 표준 입력에서 문자열을 읽어들인다.

char \*fgets(char \*s, int n, FILE \*stream); // 파일(stream)에서 n보다 하나 적게 문자열을 읽어 s에 저장

**문자열 기반 출력 함수 : puts, fputs**

#include <stdio.h>

char \*puts(const char \*s);

char \*fputs(const char \*s, FILE \*stream);

**버퍼 기반 입출력**

**버퍼 기반 입력함수 : fread**

#include <stdio.h>

size\_t fread(void \*ptr, size\_t size, size\_t nitems, FILE \*stream);

- 항목의 크기가 size인 데이터를 nitems에 지정한 개수만큼 읽어 ptr에 저장

- 성공하면 읽어온 항목 수를 리턴

- 읽을 항목이 없으면 0을 리턴

**버퍼 기반 출력함수 : fwrite**

#include <stdio.h>

size\_t fwrite(const void \*ptr, size\_t size, size\_t nitems, FILE \*stream);

- 항목의 크기가 size인 데이터를 nitems에서 지정한 개수만큼 ptr에서 읽어서 stream으로 지정한 파일에 출력

- 성공하면 출력한 항목의 수를 리턴

- 오류가 발생하면 EOF를 리턴