유닉스 명령어 정리

명령어

cd [디렉토리]
“cd “ -> 자신의 홈 디렉토리로 이동
”cd f1“ -> 하위 디렉토리 f1으로 이동

ADT

cd만 준 경우(argc == 0)
	사용자 정보를 얻어와 홈 디렉토리를 명령어 인자로 설정

명령어 인자(argv[0])로 이동
실패 : 에러 출력 후 종료
성공 : 현재 작업디렉토리를 저장하는 문자열(cur_work_dir) 변경

라이브러리 함수

#include <pwd.h>
struct passwd * getpwuid(uid_t uid) - 주어진 UID에 해당하는 사용자의 정보

```
struct passwd{
	char pw_name		- 사용자 이름
	char pw_passwd  - 사용자 암호 x나 *로 표시됨
	uid_t pw_uid		- 사용자 id
	gid_t pw_gid		- 그룹 id
	char pw_gecos		- 사용자 설명
	char pw_dir		- 홈 디렉토리
	char pw_shell		- 로그인 쉘
}
```

#include <unistd.h>
uid_t getuid(void) - 현재 프로세스를 실행 중인 실제 사용자 ID (UID)

#include <unistd.h>
int chdir(const char * path) - 현재 작업 디렉토리를 path로 변경

#include <unistd.h>
char *getcwd(char *buf, size_t size) - 현재 작업디렉토리 얻기 -> buf에 size만큼 저장

코드 구현

```
cd(){
	if(argc == 0){
		struct passwd *pwp;
		pwp = getpwuid(gwtuid());
		argv[0] = pwp->pw_dir;
	}
	if(chdir(argv[0]<0)
		PRINT_ERR_RET();
	else
		getwd(cur_work_dir,SZ_STR_BOF);
}
```

명령어

chmod 8진수모드값 파일명
”chmod 777 f1“ -> f1파일의 모드를 777로 변경

ADT

sscanf를 통해 입력받은 모드값(argv[0])를 8진수로 변경
파일을 해당 모드로 변경
실패 : 에러문구 출력 후 종료
성공 : 정상적으로 파일 모드 변경됨

라이브러리 함수

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
int sscanf(const char *str, const char *format, 자료형 변수) - 문자열에서 데이터를 추출
-> str에서 format형식으로 변수에 저장

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
int chmod(const char *pathname, mode_t mode) - pathname에 있는 파일/디렉토리의 권한을 mode로 변경

코드 구현

```
changemod(){
        int mode;
        sscanf(argv[0],“%o”,&mode);
        if(chmod(arbv[1],mode) < 0)
                PRINT_ERR_RET();
}
```

명령어

date - 오늘 현재의 “년월일시분초”로 보여주는 명령어
“ date ”

ADT

time_t 변수에 현재 시간을 저장.
struct tm *ltm에 localtime()을 활용하여 현재 시간 저장
strftime()을 통해 ltm구조체에서 format형식에 맞는 데이터를 max크기 만큼 문자열에 저장
asctime(), ctime(), puts() 을 통해 출력

라이브러리 함수

```
#include <time.h>
```

struct tm *localtime(const time_t *timep); - time_t형식의 시간을 현지 시간(로컬 타임존 기준)으로 변환

```
struct tm{
    int tm_sec;     // 초 (0-60)
    int tm_min;     // 분 (0-59)
    int tm_hour;    // 시 (0-23)
    int tm_mday;    // 일 (1-31)
    int tm_mon;     // 월 (0-11)
    int tm_year;    // 년 (1900년 기준)
    int tm_wday;    // 요일 (0: 일요일 ~ 6: 토요일)
    int tm_yday;    // 1년 중 날짜 (0-365)
    int tm_isdst;   // 서머타임 여부
}
```

char *asctime(const struct tm *tm) - struct tm포맷을 사람이 읽을 수 있는 문자열

char *ctime(const time_t *timep) - time_t값을 직접 사람이 읽을 수 있는 문자열

size_t strftime(char *s, size_t max, const char *format, const struct tm *tm);
날짜와 시간을 원하는 형식으로 문자열로 포맷팅
tm의 정보 s에 max크기 만큼 “ format 형식 ”으로 저장

format 형식

%Y	4자리 연도	2025
%y	2자리 연도	25
%m	월 (01~12)	05
%d	일 (01~31)	12
%H	시 (00~23)	14
%M	분 (00~59)	30
%S	초 (00~60)	45
%A	요일 (전체)	Monday
%a	요일 (약어)	Mon
%B	월 이름 (전체)	May
%b	월 이름 (약어)	May
%c	지역 설정 전체 날짜/시간	Mon May 12 14:30:45 2025
%x	지역 날짜	05/12/25
%X	지역 시간	14:30:45

코드 구현

```
date(){
        char stm[100];
        time_t ttm = time(NULL);
        struct tm *ltm = localtime(&ttm);
        strftime(stm,128,“stm: %a %b %H:%M:%S %Y”,ltm);
        printf(“atm: %s”,asctime(ltm));
        printf(“ctm: %s”,ctime(&ttm));
        puts(stm);
}
```

명령어

echo [출력할 문자열]

ADT

토큰 수(argc)만큼 문자열을 모두 출력하고 줄바꿈을 해준다.

라이브러리 함수

X

코드 구현

```
echo(){
        for(int i = 0; i<argc; i++){
                printf(“%s ”,argv[i];
        printf(“\n”);
}
```

명령어

hostname

“hostname” 현 컴퓨터의 이름을 출력

ADT

gethostname()을 불러와서 출력

라이브러리 함수

#include <unistd.h>

int gethostname(char *name, size_t len) - 시스템(호스트)의 호스트 이름(컴퓨터 이름)

코드 구현

```
hostname(){
        char hostname[SZ_STR_BOF];
        gethostname(hostname,SZ_STR_BOF);
        printf(“%s\n”,hostname);
}
```

명령어

id [계정이름]

“id a223100” -> uid id (계정이름) gid id (계정이름]

ADT

명령어 인자가 있다면 getpwnam() 아니면 getpwuid()

pwp = NULL이거나 getgrid()가 null이라면

        에러문구 출력

아니면

        성공문구 출력

라이브러리 함수

#include <sys/types.h>

#include <pwd.h>

struct passwd *getpwnam(const char *name) - 사용자 이름(name)에 해당하는 사용자 정보

struct passwd -> getpwuid와 동일

#include <grp.h>

struct group *getgrgid(gid_t gid) - 그룹 ID(gid)에 해당하는 그룹 정보

```
struct group {
    char   *gr_name;   // 그룹 이름
    char   *gr_passwd; // 그룹 암호 (일반적으로 x)
    gid_t   gr_gid;    // 그룹 ID
    char  **gr_mem;    // 그룹 구성원 목록 (NULL로 끝나는 문자열 배열)
};
```

코드 구현

```
id(){
        struct passwd *pwp;
        struct group *grp;
        pwp = (argc ? getpwnam(argv[0]) : getpwuid(getuid()));
        if(pwp == NULL || (grp = getgrgid(pwp->pw_uid))==NULL)
                printf(“%s: 잘못된 사용자 이름: %s”,cmd,argv[0]);
        else
                printf(“uid%d(%s) gid%d(%s)”,
                                        pwp->pw_gid, pwp->pw_name, grp->gr_gid, grp->gr_name);
}
```

명령어

ln 원본파일 링크파일 -> 원본 파일을 링크파일로 만들어줌
ln -s 원본파일 링크파일 -> 바로가기 파일을 만들어줌

ADT

만약 옵션 인자가 있다면 바로가기 파일 생성 아니면 하드링크 생성

라이브러리 함수

#include <unistd.h>
int symlink(const char *target, const char *linkpath) - 바로가기 파일 생성

int link(const char *oldpath, const char *newpath); - 하드링크 생성

코드 구현

```
ln(){
        if(optc ? symlink(argv[0],argv[1]) : link(argv[0],argv[1]))
                PRINT_ERR_RET();
}
```

명령어

mkdir 파일이름

ADT

mkdir의 함수가 있기 때문에 makedir라고 함수명 설정
mkdir()함수가 -1이라면 오류 출력 후 종료

라이브러리 함수

#include <sys/stat.h>
#include <sys/types.h>
int mkdir(const char *pathname, mode_t mode) - 디렉토리의 권한을 지정하고 생성

코드 구현

```
makedir(){
        if(mkdir(argv[0],0755)<0)
                PRINT_ERR_RET();
}
```

명령어

uname -> 운영체제 이름 및 버전 등 시스템 정보를 출력
uname -a -> 상세정보 출력

ADT

uname이라는 함수가 있음으로 함수 명을 unixname으로 설정
uname으로 함수를 호출하여 운영체제 이름을 출력
만약 옵션이 있다면 추가적으로 utsname 구조체 을 출력 후 줄바꿈 실시

```
struct utsname {
    char sysname[];     // OS 이름 (예: "Linux")
    char nodename[];    // 네트워크 노드 이름 (호스트 이름)
    char release[];     // OS 릴리스 버전 (예: "5.15.0")
    char version[];     // OS 버전 (예: "#54-Ubuntu SMP ...")
    char machine[];     // 하드웨어 아키텍처 (예: "x86_64")
};
```

라이브러리 함수

#include <sys/utsname.h>
int uname(struct utsname *buf); - buf에 운영체제 및 커널 관련 정보

코드 구현

```
unixname(){
        struct utsname un;
        uname(&un);
        printf("%s ",un.sysname);
        if(optc==1)
                printf("%s %s %s %s",un.nodename,un.release,un.version,un.machine);
        printf("\n");
}
```

명령어

rmdir 디렉토리이름

ADT

rmdir 함수가 있음으로 함수 명을 removedir 로 설정
rmdir()함수가 -1이면 에러 문구 출력 후 종료

라이브러리 함수

#include <unistd.h>
int rmdir(const char *pathname) - 디렉토리를 삭제

코드 구현

```
removedir(){
        if(rmdir(argv[0])<0)
                PRINT_ERR_RET();
}
```

명령어

pwd -> 현재 작업 디렉토리 출력

ADT

현재 작업 디렉토리 출력

라이브러리 함수

X

코드 구현

```
pwd(){
        printf("%s \n",cur_work_dir);
}
```

명령어

rm 파일이름

ADT

파일이 있는 경로를 가져온 후
디렉토리인지 바로가기 파일인지 비교 후 삭제

라이브러리 함수

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
int lstat(const char *pathname, struct stat *buf) - pathname 위치의 경로를 buf에 저장

```
struct stat {
    mode_t st_mode;     // 파일 종류 및 권한
    off_t  st_size;     // 파일 크기 (바이트)
    time_t st_mtime;    // 마지막 수정 시간
    nlink_t st_nlink;   // 하드 링크 수
    uid_t st_uid;       // 소유자 ID
    gid_t st_gid;       // 그룹 ID
};
```

S_ISDIR(m) - m이 디렉토리인지 판단

int unlink(const char *pathname);

코드 구현

```
rm(){
        struct stat buf;
        if(lstat(argv[0],&buf) < 0 || (S_ISDIR(buf.st_mode)? rmdir(argv[0]) : unlink(argv[0]) < 0)
                PRINT_ERR_RET():
}
```

명령어

mv 원본파일 바뀐파일

ADT

바뀐파일을 기존 파일의 하드링크로 만들고
기존파일을 삭제

라이브러리 함수

link(cons char *pathname);
unlink(cons char *pathname);

코드 구현

```
mv(){
        if(link(argv[0],argv[1]) < 0 || unlink(argv[0]) < 0)
                PRINT_ERR_RET();
}
```

명령어

quit 프로그램을 종료하는 명령어

ADT

x

라이브러리 함수

x

코드 구현

```
quit(){
        exit(0);
}
```

명령어

whoami - 사용자의 이름, uid, gid, 홈디렉토리 출력

ADT

1. 사용자의 이름 출력
2. 사용자의 name, uid, gid, homedir 출력

라이브러리 함수

#include <unistd.h>

char *getlogin(void) - 현재 터미널에 로그인한 사용자의 로그인 이름(사용자 ID가 아닌 이름)을 문자열로 반환

코드 구현

1. getlogin()

```
whoami(){
        char * username;
        username = getlogin();
        if(username! =NULL)
                printf("%s \n",username);
        else
                printf("터미널 장치가 아니라서 사용자 계정 정보를 구할 수 업습니다.\n");
}
```

2. getpwuid()

```
whoami(){
        struct passwd *pwp
        pwp = gwtpwuid(getuid());
        printf("%s %d %d \" %s \"",pwp->pw_name,pwp->pw_uid,pwp->pw_gid,pwp->pw_dir);
}
```