응용과제 8번

<소스코드>

```
#include<sys/stat.h>
#include<dirent.h>
#include<stdlib.h>
#include<stdio.h>
#include<pwd.h>
#include<grp.h>
#include<time.h>

#define MODE_SIZE         9
#define MODE_STR_SIZE    10

static short octarray[MODE_SIZE]={0400,0200,0100,0040,0020,0010,0004,0002,0001};
static char perms[MODE_STR_SIZE]="rwxrwxrwx";

void normal_print();
void detail_print();

int main(int argc, char*argv[])
{
        if(argc<2)
        {
                fprintf(stderr,"사용법 : %s (디렉토리명)₩n세부정보 %s -l (디렉토리명)₩n", argv[0], argv[0]);
                exit(1);
        }

        char *dirnam;
        int opt;
        while((opt = getopt(argc, argv, "l")) != -1)
        { //옵션 주어질경우 진행
                if(opt == 'l' && argc == 3)
                {
                        dirnam = argv[2];
                        detail_print(dirnam);
                        exit(0);
                }
                else
                {
                        fprintf(stderr,"잘못된 옵션 사용함₩n사용법 : %s (디렉토리명)₩n세부정보 %s -l (디렉토리명)₩n"
                                        ,argv[0], argv[0]);
                        exit(6);
                }
        }
        //옵션 주어지지 않을경우 진행
        dirnam = argv[1];
        normal_print(dirnam);
        return 0;
}

void normal_print(char *dirnam)
{
```

```
        DIR *dp;
        if((dp=opendir(dirnam)) == NULL)
        {
                perror("디렉토리 열기실패");
                exit(2);
        }

        struct dirent *dent;
        int count = 0;
        while((dent = readdir(dp)))
        {
                printf("%12s ", dent->d_name);
                count++;
                if(count%7 == 0)
                {
                        printf("₩n");
                }
        }
        closedir(dp);
        printf("₩n");
}

void detail_print(char *dirnam)
{
        DIR *dp;
        if((dp=opendir(dirnam)) == NULL)
        {
                perror("디렉토리 열기실패");
                exit(3);
        }

        printf("%s%s %s    %s   %11s    %10s  %5s  %s  %s₩n","종류및","권한","링크","inode값",
                        "소유자","소유그룹","크기","마지막수정일","파일명");

        struct stat buf;
        struct dirent *dent;
        while((dent = readdir(dp)))
        {
                if(lstat(dent->d_name, &buf) < 0)
                {
                        perror("lstat");
                        exit(4);
                }

                int i;
                char mode_str[MODE_STR_SIZE];
                for(i=0; i<MODE_SIZE; i++)
                {
                        if(buf.st_mode & octarray[i])
                        {
                                mode_str[i] = perms[i];
                        }
```

```
                else
                {
                        mode_str[i] = '-';
                }
        }
        mode_str[MODE_STR_SIZE -1] = '₩0';

        char file_type;
        if(S_ISFIFO(buf.st_mode)) file_type = 'p';
        if(S_ISDIR(buf.st_mode)) file_type = 'd';
        if(S_ISREG(buf.st_mode)) file_type = '-';
        if(S_ISLNK(buf.st_mode)) file_type = 'l';

        struct passwd *pw;
        struct group *grp;
        pw = getpwuid(buf.st_uid);
        grp = getgrgid(buf.st_gid);

        struct tm time;
        time = *localtime(&(buf.st_mtime));

        char time_buf[BUFSIZ];
        strftime(time_buf, sizeof(time_buf), "%m월 %d %H:%M", &time);

        printf("%c%s. %2o %12d %10s %10s %5d %s %s",
                        file_type, mode_str, (unsigned int)buf.st_nlink, (int)buf.st_ino, pw->pw_name,
                        grp->gr_name, (int)buf.st_size, time_buf, dent->d_name);

        if(S_ISLNK(buf.st_mode))
        {
                char lnk_src[BUFSIZ];
                int lnk;
                lnk = readlink(dent->d_name, lnk_src, BUFSIZ);
                if(lnk == -1)
                {
                        perror("readlink");
                        exit(5);
                }
                lnk_src[lnk] = '₩0';
                printf(" -> %s₩n", lnk_src);
        }
        else
        {
                printf("₩n");
        }
    }
    closedir(dp);
}
```

헤더, 상수정의, 함수선언, main

```
#include<sys/stat.h>
#include<dirent.h>
#include<stdlib.h>
#include<stdio.h>
#include<pwd.h>
#include<grp.h>
#include<time.h>

#define MODE_SIZE       9
#define MODE_STR_SIZE   10

static short octarray[MODE_SIZE]={0400,0200,0100,0040,0020,0010,0004,0002,0001};
static char perms[MODE_STR_SIZE]="rwxrwxrwx";

void normal_print();
void detail_print();

int main(int argc, char*argv[])
{
        if(argc<2)
        {
                fprintf(stderr,"사용법 : %s (디렉토리명)\n세부정보 %s -l (디렉토리명)\n", argv[0], argv[0]);
                exit(1);
        }

        char *dirnam;
        int opt;
        while((opt = getopt(argc, argv, "l")) != -1)
        { //옵션 주어질경우 진행
                if(opt == 'l' && argc == 3)
                {
                        dirnam = argv[2];
                        detail_print(dirnam);
                        exit(0);
                }
                else
                {
                        fprintf(stderr,"잘못된 옵션 사용함\n사용법 : %s (디렉토리명)\n세부정보 %s -l (디렉토리명)\n"
                                        ,argv[0], argv[0]);
                        exit(6);
                }
        }
        //옵션 주어지지 않을경우 진행
        dirnam = argv[1];
        normal_print(dirnam);
        return 0;
}
```

파일권한을 위해 octarray,perms를 선언해주고 값을 할당해주었습니다. 일반적으로 동작할부분을 normal_print, -l 옵션을 사용해 세부내용을 출력하기위해 detail_print함수를 작성해주었습니다. main에서 argc가 2이하이면 사용법과 세부사용법을 출력후 종료해줍니다. argv에서 디렉토리명을 받을것이기 때문에 캐릭터포인터 dirnam을 선언해주고 getopt를 사용하기위해 opt를 선언해주었습니다.

while문을 사용하여 getopt를 루프시킵니다. 옵션으로 l을 받고 l은 단독적으로 쓰이지 않기 때문에 디렉토리명까지 포함하여 argc가 3인지 검사해줍니다. 조건을 만족하면 dirnam은 argv[2]로 할당됩니다. 그후 detail_print함수를 소출해주었습니다. 이때 매개변수로 argv[2]값인 dirnam이 들어갑니다.

그 외 옵션을 부여하지 않았을때는 getopt루프가 돌지 않습니다. 때문에 메인에 dirnam을 argv[1]로 할당해주었고 이때는 보통 ls 명령을 출력해야하므로 normal_print함수를 호출해주었습니다. 역시 매개변수는 실행파일과 디렉토리만 받기 때문에 argv[1]값인 dirnam이 들어갑니다.

normal_print 함수

```
void normal_print(char *dirnam)
{
        DIR *dp;
        if((dp=opendir(dirnam)) == NULL)
        {
                perror("디 렉 토 리  열 기 실 패 ");
                exit(2);
        }

        struct dirent *dent;
        int count = 0;
        while((dent = readdir(dp)))
        {
                printf("%12s ", dent->d_name);
                count++;
                if(count%7 == 0)
                {
                        printf("\n");
                }
        }
        closedir(dp);
        printf("\n");
}
```

디렉토리포인터 dp를 선언하고 응용과제 7번과같이 한행에 출력할 수 있는 개수를 제한하기위해 count도 선언해줍니다. opendir함수를 사용하여 메인에서 전달받은 dirnam을 열기합니다. 이때의 디렉토리포인터는 dp입니다. 이때 NULL을 리턴하면 오류메세지를 출력후 종료합니다.

디렉토리 항목을 출력하기위해 dirent.h 헤더에 정의된 구조체 *dent를 선언해줍니다. 반복문을 통해 readdir함수를 호출하고 dp가 가리키는 파일을 dent구조체에 저장합니다. 이는 readdir함수가 더 이상 읽을것이 없을 때 종료될 것입니다. 그후 print문을 사용하여 dent구조체로 d_name에 접근하여 출력합니다. 출력하면 count를 ++하여 나중에 카운트가 7이되고 이값을 7로나누면 몫이 0이되기에 이때 개행을 해줍니다.

반복문이 종료되면 오픈하였던 dp를 닫아줍니다.

detail_print 함수

```
void detail_print(char *dirnam)
{
        DIR *dp;
        if((dp=opendir(dirnam)) == NULL)
        {
                perror("디 렉 토 리  열 기 실 패 ");
                exit(3);
        }

        printf("%s%s %s    %s   %11s    %10s  %5s  %s  %s\n","종 류 및 ","권 한 ","링 크 ","inode값 ",
                        "소 유 자 ","소 유 그 룹 ","크 기 ","마 지 막 수 정 일 ","파 일 명 ");

        struct stat buf;
        struct dirent *dent;
        while((dent = readdir(dp)))
        {
                if(lstat(dent->d_name, &buf) < 0)
                {
                        perror("lstat");
                        exit(4);
                }

                int i;
                char mode_str[MODE_STR_SIZE];
                for(i=0; i<MODE_SIZE; i++)
                {
                        if(buf.st_mode & octarray[i])
                        {
                                mode_str[i] = perms[i];
                        }
                        else
                        {
                                mode_str[i] = '-';
                        }
                }
                mode_str[MODE_STR_SIZE -1] = '\0';

                char file_type;
                if(S_ISFIFO(buf.st_mode)) file_type = 'p';
                if(S_ISDIR(buf.st_mode)) file_type = 'd';
                if(S_ISREG(buf.st_mode)) file_type = '-';
                if(S_ISLNK(buf.st_mode)) file_type = 'l';

                struct passwd *pw;
                struct group *grp;
                pw = getpwuid(buf.st_uid);
                grp = getgrgid(buf.st_gid);

                struct tm time;
                time = *localtime(&(buf.st_mtime));

                char time_buf[BUFSIZ];
                strftime(time_buf, sizeof(time_buf), "%m월  %d %H:%M", &time);

                printf("%c%s. %2o %12d %10s %10s %5d %s %s",
                                file_type, mode_str, (unsigned int)buf.st_nlink, (int)buf.st_ino, pw->pw_name,
                                grp->gr_name, (int)buf.st_size, time_buf, dent->d_name);

                if(S_ISLNK(buf.st_mode))
                {
                        char lnk_src[BUFSIZ];
                        int lnk;
                        lnk = readlink(dent->d_name, lnk_src, BUFSIZ);
                        if(lnk == -1)
                        {
                                perror("readlink");
                                exit(5);
                        }
                        lnk_src[lnk] = '\0';
                        printf(" -> %s\n", lnk_src);
                }
                else
                {
                        printf("\n");
                }
        }
        closedir(dp);
}
```

<추가한것>

1. 파일의 종류 출력 (p,d,-,l)

반복문의 readdir함수로 dp의 파일을 읽어올때마다 해당 디렉토리의 항목(d_name)을 읽어와 lstat함수를 사용하여 buf에 정보를 저장해주었습니다. 처음에는 stat함수만 사용했다가 심볼릭 링크항목이 표시되지 않아서 lstat함수를 사용하였습니다.. 저장한 buf.st_mode를 매크로를 이용한 파일종류 검색을통해 피포파일인지, 디렉토리인지, 일반파일인지, 링크파일인지 구분하여 f_type에 p,d,-,l 문자를 저장해주었습니다.

2. 파일의 권한 출력 (rwxrwxrwx)

octarray배열을 선언하고, 0400,0200,0100,0040,0020,0010,0004,0002,0001을 저장합니다. 이는 유저,그룹,기타 사용자의 읽기,쓰기,실행권한입니다. perms배열을 선언하고 rwxrwxrwx를 저 장해줍니다. 권한을 문자열로 출력하기위해 mode_str를 선언합니다. 그후 반복문을통해 MODE_SIZE만큼 루프해줍니다. stat_buf.st_mode와 octarray[i] 를 and연산하여 참이면 mode_str[i]에 perms[i]를 저장합니다 perms는 rwxrwxrwx이므로 해당 하는 권한을 가진다면 해당하는 문자를 mode_str에 저장할것입니다. 만약 아니라면 mode_str 에 - 를 저장합니다. 배열의 마지막에 널문자를 추가후 출력시 %s로 mode_str를 출력해줍니다.

3. 파일의 링크수 출력

반복문에서 stat함수를통해 디렉토리의 파일항목이름을 반복하여 읽기 때문에 정보를 저장했던 buf구조체의 st_nlink항목만 출력하면됩니다. 출력시 unsined int형으로 buf.st_nlink항목을 출력해주었습니다.

4. 파일의 inode값 출력

반복문에서 stat함수를통해 디렉토리의 파일항목이름을 반복하여 읽기 때문에 정보를 저장했던 buf구조체의 st_ino 항목만 출력하면됩니다. 출력시 int형으로 buf.st_ino항목을 출력해주었습니다.

5. 파일의 소유자 출력

반복문에서 stat함수를 호출하여 buf에 저장하였고 이 buf.st_uid항목이 이 파일의 소유자 uid입니다. 이 uid를 이름형태로 나타내기위해 pwd.h헤더에 정의된 passwd구조체 *pw를 선언하고 getpwuid함수를 이용해 buf.st_uid에 해당하는 사용자를 찾아 정보를 검색합니다. 출력시 pw포인터를 pw_name에 접근하여 소유자명을 출력해주었습니다.

6. 파일의 소유그룹 출력

반복문에서 stat함수를 호출하여 buf에 저장하였고 이 buf.st_gid항목이 이 파일의 소유그룹 gid입니다. 이 gid를 그룹명 형태로 나타내기위해 grp.h헤더에 정의된 group구조체 *grp를 선언하고 getgrgid함수를 이용해 buf.st_gid에 해당하는 그룹를 찾아 정보를 검색합니다. 출력시 grp포인터를 gr_name에 접근하여 소유그룹명을 출력해주었습니다.

7. 파일의 크기 출력

반복문에서 stat함수를통해 디렉토리의 파일항목이름을 반복하여 읽기 때문에 정보를 저장했던 buf구조체의 st_size항목만 출력하면됩니다. 출력시 int형으로 buf.st_size항목을 출력해주었습니다.

8. 파일의 마지막수정일 출력

반복문에서 stat함수를 통해 디렉토리의 파일항목을 읽어 해당항목의 mtime을 사용합니다. time.h에 정의되어있는 tm구조체 time을 선언합니다. 그후 localtime함수를 사용하여 파일의 mtime을 time구조체에 저장합니다. lcocaltime함수는 unix timestamp를 년,월,일등의 값을 나타내는 멤버가 있는 time구조체에 대한 포인터를 반환하는 역할을 합니다. tm구조체는 다음과 같이 time.h에 선언되어있습니다.

```
struct tm {
  int tm_sec;
  int tm_min;
  int tm_hour;
  int tm_mday;
  int tm_mon;
  int tm_year;
  int tm_wday;
  int tm_yday;
  int tm_isdst;
};
```

이번에는 strftime함수를 사용하여 문자열을 저장하기위해 선언하였던 time_buf에 time_buf의 사이즈만큼 월, 일, 시간, 분을 저장하였던 time구조체에서 불러와 문자열로 저장합니다. 후에 출력시 %s로 time_buf를 출력해줍니다.

9. 링크파일인 경우 원본파일 출력

조건문을 사용하여 S_ISLNK(buf.st_mode)로 심볼릭 링크파일인지 먼저 확인합니다.
심볼릭 링크파일인 경우 readlink함수를 호출하여 디렉토리 파일항목을 lnk_src배열에, BUFSIZ만큼 저장합니다. 이때 readlink시 오류발생하면 메시지 출력후 3을 출력해줍니다. 문자열 마지막에 널문자를 추가해 문자열조건을 만족시킵니다. 이렇게하면 lnk배열에 심볼릭 링크파일의 원본파일명이 저장될 것입니다. 그후 ->표시와 함께 %s에 lnk_src를 출력해줍니다.

반복문이 종료되고 closedir함수를이용해 dp를 닫아줍니다.

출력

실행파일만 실행

```
[khm970514@lily ch3]$ gcc -o myls2 myls2.c
[khm970514@lily ch3]$ ./myls2
사 용 법  : ./myls2 (디 렉 토 리 명 )
세 부 정 보  ./myls2 -l (디 렉 토 리 명 )
```

옵션없이 진행

```
[khm970514@lily ch3]$ ./myls2 .
          .           ..    ex3_1.exe    ex3_2.exe      ex3_3.c   ex3_10.exe     ex3_11.c
    ex3_16.c         myls     mystat.c        myls2       bak3.c      ex3_2.c    ex3_3.exe
     ex3_4.c    ex3_4.exe      ex3_5.c    ex3_5.exe      ex3_6.c    ex3_6.exe      ex3_7.c
   ex3_7.exe     unix.txt    ex3_8.exe     unix.lnk      ex3_9.c    ex3_9.exe     unix.sym
    ex3_10.c   ex3_11.exe   mystat.exe         test       myls.c      ex3_8.c     ex3_12.c
  ex3_12.exe     ex3_13.c   ex3_13.exe          td1     ex3_14.c   ex3_14.exe       testd1
    ex3_15.c   ex3_15.exe   ex3_16.exe     ex3_17.c      myls3.c      myls2.c      ex3_1.c
      mystat       bak2.c
```

–l만 사용한 경우

```
[khm970514@lily ch3]$ ./myls2 -l
잘 못 된  옵 션  사 용 함
사 용 법  : ./myls2 (디 렉 토 리 명 )
세 부 정 보  ./myls2 -l (디 렉 토 리 명 )
```

없는옵션 사용

```
[khm970514@lily ch3]$ ./myls2 -o
./myls2: invalid option -- 'o'
잘 못 된  옵 션  사 용 함
사 용 법  : ./myls2 (디 렉 토 리 명 )
세 부 정 보  ./myls2 -l (디 렉 토 리 명 )
```

-l 옵션 사용

```
[khm970514@lily ch3]$ ./myls2 -l .
종 류 및 권 한  링 크     inode값      소 유 자     소 유 그 룹   크 기   마 지 막 수 정 일   파 일 명
drwxrwxr-x.  5   -801811420  khm970514  khm970514  4096 11월  01 01:52 .
drwxrwxr-x. 10   -399274908  khm970514  khm970514    66 10월  21 13:54 ..
-rwxrwxr-x.  1   -801811419  khm970514  khm970514  8728 10월  06 14:26 ex3_1.exe
-rwxrwxr-x.  1   -801811397  khm970514  khm970514  8832 10월  06 14:29 ex3_2.exe
-rw-rw-r--.  1   -801810391  khm970514  khm970514   688 10월  06 14:31 ex3_3.c
-rwxrwxr-x.  1   -801810413  khm970514  khm970514  8840 10월  09 18:20 ex3_10.exe
-rw-rw-r--.  1   -801810386  khm970514  khm970514   437 10월  09 18:22 ex3_11.c
-rw-rw-r--.  1   -801810395  khm970514  khm970514   851 10월  09 18:38 ex3_16.c
-rwxrwxr-x.  1   -801809405  khm970514  khm970514  8800 10월  12 22:12 myls
-rw-rw-r--.  1   -801809400  khm970514  khm970514  1246 10월  12 22:25 mystat.c
-rwxrwxr-x.  1   -801808374  khm970514  khm970514 13456 11월  01 01:48 myls2
-rw-rw-r--.  1   -801808369  khm970514  khm970514  2688 11월  01 00:36 bak3.c
-rw-rw-r--.  1   -801810424  khm970514  khm970514   820 10월  06 14:29 ex3_2.c
-rwxrwxr-x.  1   -801811417  khm970514  khm970514  8680 10월  06 14:31 ex3_3.exe
-rw-rw-r--.  1   -801810430  khm970514  khm970514   569 10월  06 14:33 ex3_4.c
-rwxrwxr-x.  1   -801811416  khm970514  khm970514  8680 10월  06 14:33 ex3_4.exe
-rw-rw-r--.  1   -801810421  khm970514  khm970514   666 10월  06 14:34 ex3_5.c
-rwxrwxr-x.  1   -801811418  khm970514  khm970514  8680 10월  06 14:34 ex3_5.exe
-rw-rw-r--.  1   -801810427  khm970514  khm970514   607 10월  06 14:36 ex3_6.c
-rwxrwxr-x.  1   -801810432  khm970514  khm970514  8688 10월  06 14:36 ex3_6.exe
-rwxrwx---.  1   -801810422  khm970514  khm970514   676 10월  06 14:39 ex3_7.c
-rwxrwxr-x.  1   -801810423  khm970514  khm970514  8784 10월  06 14:39 ex3_7.exe
-rw-rw-r--.  2   -801810385  khm970514  khm970514    24 10월  09 18:05 unix.txt
-rwxrwxr-x.  1   -801810418  khm970514  khm970514  8784 10월  09 18:06 ex3_8.exe
-rw-rw-r--.  2   -801810385  khm970514  khm970514    24 10월  09 18:05 unix.lnk
-rw-rw-r--.  1   -801810384  khm970514  khm970514   385 10월  09 18:15 ex3_9.c
-rwxrwxr-x.  1   -801810414  khm970514  khm970514  8640 10월  09 18:16 ex3_9.exe
lrwxrwxrwx.  1   -801810412  khm970514  khm970514     8 10월  09 18:16 unix.sym -> unix.txt
-rw-rw-r--.  1   -801810383  khm970514  khm970514  1069 10월  09 18:20 ex3_10.c
-rwxrwxr-x.  1   -801810409  khm970514  khm970514  8688 10월  09 18:22 ex3_11.exe
-rwxrwxr-x.  1   -801810392  khm970514  khm970514  8840 10월  10 09:29 mystat.exe
drwxrwxr-x.  2  -1039409144  khm970514  khm970514     6 10월  11 17:55 test
-rw-rw-r--.  1   -801809401  khm970514  khm970514   522 10월  12 22:18 myls.c
-rw-rw-r--.  1   -801810410  khm970514  khm970514   648 10월  02 10:15 ex3_8.c
-rw-rw-r--.  1   -801810407  khm970514  khm970514   397 10월  09 18:24 ex3_12.c
-rwxrwxr-x.  1   -801810408  khm970514  khm970514  8688 10월  09 18:24 ex3_12.exe
-rw-rw-r--.  1   -801810401  khm970514  khm970514   461 10월  09 18:27 ex3_13.c
-rwxrwxr-x.  1   -801810406  khm970514  khm970514  8744 10월  09 18:27 ex3_13.exe
drwxrwxr-x.  2  -1472555575  khm970514  khm970514     6 10월  09 18:28 td1
-rw-rw-r--.  1   -801810398  khm970514  khm970514   550 10월  09 18:29 ex3_14.c
-rwxrwxr-x.  1   -801810404  khm970514  khm970514  8776 10월  09 18:29 ex3_14.exe
drwxrwxr-x.  2  -1039409106  khm970514  khm970514    23 10월  09 18:47 testd1
-rw-rw-r--.  1   -801810399  khm970514  khm970514   456 10월  09 18:33 ex3_15.c
-rwxrwxr-x.  1   -801810400  khm970514  khm970514  8784 10월  09 18:34 ex3_15.exe
-rwxrwxr-x.  1   -801810397  khm970514  khm970514  8944 10월  09 18:38 ex3_16.exe
-rw-rw-r--.  1   -801810387  khm970514  khm970514   870 10월  09 18:45 ex3_17.c
-rw-rw-r--.  1   -801809406  khm970514  khm970514  2665 10월  12 19:35 myls3.c
-rw-rw-r--.  1   -801808367  khm970514  khm970514  2946 11월  01 01:47 myls2.c
-rw-rw-r--.  1   -801810394  khm970514  khm970514   707 10월  06 14:26 ex3_1.c
-rwxrwxr-x.  1   -801809404  khm970514  khm970514  8808 10월  12 22:26 mystat
-rw-rw-r--.  1   -801808370  khm970514  khm970514  3035 11월  01 00:12 bak2.c
[khm970514@lily ch3]$
```