5장 과제

2018312567 조명하

실습문제 1.

1.1 선택한 옵션

① -l : 파일 나열에 있어, 파일 형태, 사용권한, 하드링크 번호, owner 이름, group 이름, 파일 크기, 시간(따로 지정하지 않으면 파일이 만들어진 날자다)을 자세하게 나열한다. 시간은 여섯달 이전 것이면, 시간이 생략되고, 파일의 연도가 포함된다.

② -G : 자세한 목록 나열에서 group 정보를 제외한다.

③ -m : 파일을 가로로 나열한다. 가로로 나열할 수 있는 만큼 최대한 나열한다.

1.2 소스 코드

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

char type(mode_t);
char *perm(mode_t);
void printStat(char*, struct stat*);
void printGStat(char*, struct stat*);
```

-G 옵션은 group 정보를 제외하므로 함수 printGstat를 새로 만들었다.

```
/* 디렉토리 내용을 자세히 리스트한다. */
int main(int argc, char **argv)
{
    DIR *dp;
    char *dir;
    struct stat st;
    struct dirent *d;
    char path[BUFSIZ+1];
    char **names; //디렉토리 이름 저장할 배열
    struct stat *states; //디렉토리 상태 구조체 저장할 배열
    char *option; //옵션 저장할 스트링
    int cnt = 2; //현재 디렉도리와 부모 디렉토리 제외하기 위해
    int num = 0;//파일 개수
    int max = 0;//파일 이름 길이
    int size = 0;//파일 크기

    int i = 0;
    int j;//인덱스용
    char *temp;//swap위해
```

현재 디렉토리와 부모 디렉토리 정보를 제외하고 출력하기 위해 변수 cnt를 만들어 디렉토리 내 파일을 읽을 때 처음 두 번은 건너뛰고 시작하도록 한다.

```c
/*명령 입력받기*/
if (argc == 1) {
    dir = ".";
    option = "default";
}
else if (argc == 2){
    dir = argv[1];
    option = "default";
}
else if (argc == 3){
    dir = argv[1];
    option = argv[2];
}
else {
    printf("사용법: command dir_name -option\n");
    exit(1);
}

if ((dp = opendir(dir)) == NULL)  // 디렉토리 열기
        perror(dir);

while ((d = readdir(dp)) != NULL){  // 디렉토리 내의 파일 개수 구하기
    num++;
    if (strlen(d->d_name) > max){
        max = strlen(d->d_name);     //가장 긴 파일 이름 길이 구하기
    }
}
```

입력받은 명령에 따라 대상 디렉토리와 옵션을 설정한다.

명령어 입력이 잘못되었거나 디렉토리를 열 수 없으면 에러 메시지를 출력하고 프로그램을 종료한다.

파일 이름을 배열에 저장할 때 크기를 알아야 하므로 가장 길이가 긴 이름의 길이를 구한다.

```c
rewinddir(dp);//디렉토리 맨 처음으로 돌아가기

/*디렉토리 내 파일들의 이름과 상태를 저장할 배열 동적 할당하기*/
names = (char**)malloc(sizeof(char*)*num);
states = (struct stat*)malloc(sizeof(struct stat)*num);
for (i = 0;i<num;i++){
    names[i] = (char*)malloc(sizeof(int)*max);
}

/*디렉토리 내 파일 이름 names에 저장*/
j = 0;
for (i = 0;i<num;i++){
    d = readdir(dp);
    if (i>=2){
        names[j++] = d->d_name;
    }
}
close(dp);//디렉토리 닫기

/*알파벳 순으로 정렬*/
for (i=0;i<num-3;i++){
    for (j=i+1;j<num-2;j++){
        if (strcmp(names[i], names[j]) > 0){
            temp=names[i];
            names[i] = names[j];
            names[j] = temp;
        }
    }
}
```

dp를 디렉토리의 맨 처음으로 설정한다.

디렉토리 내 파일들의 이름과 상태를 저장할 배열을 만들고 디렉토리 내 파일의 숫자만큼 동적 할당한다.

파일들의 이름을 저장할 배열은 이전에 구했던 가장 긴 파일의 이름의 길이만큼 다시 동적할당해 2차원 배열을 만든다.

만든 2차원 배열에 디렉토리 내 파일들의 이름을 읽어서 순서대로 저장한다. 이때 처음 두 번은 현재 디렉토리와 부모 디렉토리이기 때문에 건너뛰고 세 번째부터 저장한다.

이름을 저장한 후 디렉토리를 닫는다.

ls명령어는 기본적으로 알파벳 순으로 파일 이름을 출력하기 때문에 파일 이름을 알파벳 순으로 정렬한다.

```
/* 옵션별 함수 호출하기*/
if (strcmp(option, "default")==0 || strcmp(option, "-G")==0){
    for (i=0;i<num-2;i++){
        printf("%s  ", names[i]);    //파일 이름 나열하기
    }
    printf("\n");
    exit(0);
}
else if (strcmp(option, "-l")==0 || strcmp(option, "-ml")==0){
    for (i=0;i<num-2;i++){
        lstat(names[i], states+i);  // 파일 상태 states에 저장하기
        size += states[i].st_blocks; // 파일 크기 구하기
    }
    printf("합계 %d\n", size/2);
    for (i=0;i<num-2;i++){
        printStat(names[i], states+i);  // 상태 정보 출력
    }
    exit(0);
}
else if (strcmp(option,"-m")==0 || strcmp(option,"-lm")==0 || strcmp(option, "-Gm")==0 || strcmp(option,"-mG")==0 || strcmp(option, "-lmG")==0 || strcmp(option, "-Glm")==0){
    for (i=0;i<num-3;i++){
        printf("%s,  ", names[i]);  // 마지막 파일 제외하고 이름과 , 출력
    }
    i++;
    printf("%s\n", names[i]);       // 마지막 파일 이름 출력
    exit(0);
}
else if (strcmp(option, "-lG")==0 || strcmp(option,"-Gl")==0 || strcmp(option, "-mlG")==0 || strcmp(option,"-mGl")==0 || strcmp(option ,"-Gml")==0){
    for (i=0;i<num-2;i++){
        lstat(names[i], states+i);  // 파일 상태 states에 저장하기
        size += states[i].st_blocks; // 파일 크기 구하기
    }
    printf("합계 %d\n", size/2);
    for (i=0;i<num-2;i++){
        printGStat(names[i], states+i);  // 그룹을 제외한 파일들의 상태 정보 출력
    }
    exit(0);
}
else {
    printf("Support -l, -m, -G ONLY\n");   // 지원하지 않는 옵션 입력한 경우
    exit(1);
}
```

입력받은 옵션에 따라 적절한 함수를 호출한다.

1) 옵션이 없거나 –G를 선택한 경우, 파일 이름만 출력되도록 해야 한다. (실제 ls 명령어 예시)

```
[chomyungha@localhost 05]$ ls -G
a.out  du.c  list  list2.c
```

파일 이름을 저장한 배열에서 차례대로 값을 읽어 출력한다.

2) 옵션이 –l, -ml인 경우, 파일의 다양한 정보가 출력되도록 해야 한다. (실제 ls 명령어 예시)

```
[chomyungha@localhost 05]$ ls -l
합계 36
-rwxrwxr-x. 1 chomyungha chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$ ls -ml
합계 36
-rwxrwxr-x. 1 chomyungha chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$
```

총 디렉토리 사용량도 처음에 출력해야 하므로 저장한 파일 이름을 사용해 파일의 상태를 불러와서 사용한 블록 크기를 저장해 그 결과값/2를 출력한다.

printStat 함수로 파일 유형, 접근권한, 링크 수, 소유자, 그룹, 파일 크기, 접근 날짜, 파일 이름을 출력한다.

3) 옵션이 –m, -lm, -Gm, -mG, -lmG, -Glm인 경우, 파일의 이름을 나열해야 한다. (실제 ls 명령어 예시)

```
[chomyungha@localhost 05]$ ls -m
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -lm
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -Gm
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -mG
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -lmG
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -lGm
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$ ls -Glm
a.out, du.c, list, list2.c
[chomyungha@localhost 05]$
```

마지막 파일을 제외하고 파일 이름 뒤에 ,를 붙여서 출력한다.

4) 옵션이 –lG, -Gl, -mlG, -mGl, -Gml 인 경우, 파일의 다양한 정보를 출력하되, 그룹 정보만 제외하고 출력해야 한다. (실제 ls 명령어 예시)

```
[chomyungha@localhost 05]$ ls -lG
합계 36
-rwxrwxr-x. 1 chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$ ls -Gl
합계 36
-rwxrwxr-x. 1 chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$ ls -mlG
합계 36
-rwxrwxr-x. 1 chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$ ls -mGl
합계 36
-rwxrwxr-x. 1 chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$ ls -Gml
합계 36
-rwxrwxr-x. 1 chomyungha 13240  4월  8 11:46 a.out
-rw-r--r--. 1 chomyungha     0  4월  8 11:16 du.c
-rwxrwxr-x. 1 chomyungha 13240  4월  8 14:11 list
-rw-r--r--. 1 chomyungha  3499  4월  8 15:45 list2.c
[chomyungha@localhost 05]$
```

옵션 –l인 경우와 마찬가지로 디렉토리 사용량을 구해서 출력하고 printGstat를 호출해 그룹 정보를 빼고 파일의 상태를 출력한다.

5) 지원하지 않는 옵션을 입력한 경우, 지원하는 옵션의 종류를 출력하고 종료한다.

```
void printGStat(char *file, struct stat *st) {
    printf("%c%s.", type(st->st_mode), perm(st->st_mode));
    printf("%3d ", st->st_nlink);
    printf("%s ", getpwuid(st->st_uid)->pw_name);
    printf("%6d ", st->st_size);
    printf("%.12s ", ctime(&st->st_mtime)+4);
    printf("%s\n", file);
}

void printStat(char *file, struct stat *st) {
    printf("%c%s.", type(st->st_mode), perm(st->st_mode));
    printf("%3d ", st->st_nlink);
    printf("%s %s", getpwuid(st->st_uid)->pw_name, getgrgid(st->st_gid)->gr_name);
    printf("%6d ", st->st_size);
    printf("%.12s ", ctime(&st->st_mtime)+4);
    printf("%s\n", file);
}

/* 파일 타입을 리턴 */
char type(mode_t mode) {
    if (S_ISREG(mode))
        return('-');
    if (S_ISDIR(mode))
        return('d');
    if (S_ISCHR(mode))
        return('c');
    if (S_ISBLK(mode))
        return('b');
    if (S_ISLNK(mode))
        return('l');
    if (S_ISFIFO(mode))
        return('p');
    if (S_ISSOCK(mode))
        return('s');
}


/* 파일 허가권을 리턴 */
char* perm(mode_t mode) {
    int i;
    static char perms[10];

    strcpy(perms, "---------");

    for (i=0; i < 3; i++) {
        if (mode & (S_IREAD >> i*3))
            perms[i*3] = 'r';
        if (mode & (S_IWRITE >> i*3))
            perms[i*3+1] = 'w';
        if (mode & (S_IEXEC >> i*3))
            perms[i*3+2] = 'x';
    }
    return(perms);
}
```

원본 list2 프로그램과 동일하고, printStat에서 그룹 정보를 제외하고 출력하는 printGStat만 추가했다.

```c
void printGStat(char *file, struct stat *st) {
    //printf("%5d ", st->st_blocks);
    printf("%c%s", type(st->st_mode), perm(st->st_mode));
    printf("%3d ", st->st_nlink);
    printf("%s ", getpwuid(st->st_uid)->pw_name);
    printf("%6d ", st->st_size);
    printf("%.12s ", ctime(&st->st_mtime)+4);
    printf("%s\n", file);
}

/* Option == -l */
void printStat(char *file, struct stat *st) {
    //printf("%5d ", st->st_blocks);
    printf("%c%s", type(st->st_mode), perm(st->st_mode));
    printf("%3d ", st->st_nlink);
    printf("%s %s ", getpwuid(st->st_uid)->pw_name, getgrgid(st->st_gid)->gr_name);
    printf("%6d ", st->st_size);
    printf("%.12s ", ctime(&st->st_mtime)+4);
    printf("%s\n", file);
}

/* 파일 타입을 리턴 */
char type(mode_t mode) {
    if (S_ISREG(mode))
        return('-');
    if (S_ISDIR(mode))
        return('d');
    if (S_ISCHR(mode))
        return('c');
    if (S_ISBLK(mode))
        return('b');
    if (S_ISLNK(mode))
        return('l');
    if (S_ISFIFO(mode))
        return('p');
    if (S_ISSOCK(mode))
        return('s');
}


/* 파일 허가권을 리턴 */
char* perm(mode_t mode) {
    int i;
    static char perms[10];

    strcpy(perms, "---------");

    for (i=0; i < 3; i++) {
        if (mode & (S_IREAD >> i*3))
            perms[i*3] = 'r';
        if (mode & (S_IWRITE >> i*3))
            perms[i*3+1] = 'w';
        if (mode & (S_IEXEC >> i*3))
            perms[i*3+2] = 'x';
    }
    return(perms);
}
```

기존 printStat 함수에서 그룹 정보를 출력하는 부분만 제외한 printGstat 함수를 새로 만들었다.

1.3 프로그램 실행 결과

```
[chomyungha@localhost 05]$ gcc -o list list2.c
[chomyungha@localhost 05]$ ./list
du.c list2.c a.out list
[chomyungha@localhost 05]$ ./list .
du.c list2.c a.out list
[chomyungha@localhost 05]$ ./list . -l
-rw-r--r--  1 chomyungha chomyungha     0 Apr  8 11:16 du.c
-rw-r--r--  1 chomyungha chomyungha  4010 Apr  8 17:24 list2.c
-rwxrwxr-x  1 chomyungha chomyungha 13240 Apr  8 11:46 a.out
-rwxrwxr-x  1 chomyungha chomyungha 13376 Apr  8 17:24 list
[chomyungha@localhost 05]$ ./list . -Gl
-rw-r--r--  1 chomyungha     0 Apr  8 11:16 du.c
-rw-r--r--  1 chomyungha  4010 Apr  8 17:24 list2.c
-rwxrwxr-x  1 chomyungha 13240 Apr  8 11:46 a.out
-rwxrwxr-x  1 chomyungha 13376 Apr  8 17:24 list
[chomyungha@localhost 05]$ ./list . m
Support -l, -m, -G ONLY
[chomyungha@localhost 05]$ ./list . -m
du.c, list2.c, a.out, list,
[chomyungha@localhost 05]$
```

대상 디렉토리를 지정하지 않은 경우, 현재 디렉토리를 대상으로 한다.

옵션을 지정하지 않은 경우, 현재 디렉토리 내의 파일 이름을 보여준다.

옵션이 –l인 경우, 디렉토리 내 파일의 상태 정보를 보여준다.

옵션이 –Gl인 경우, 디렉토리 내 파일의 상태 정보를 그룹 정보를 빼고 보여준다.

옵션이 m등 지원하지 않는 형식인 경우, 옵션이 잘못되었다는 메시지를 출력한다.

옵션이 –m인 경우, 디렉토리 내 파일의 이름을 ,로 나열한다.

실습문제 2.

1.1 소스 코드

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

void dirSize(char*);
DIR *dp;
char *file;
struct stat st;
struct dirent *d;
char path[BUFSIZ+1];
int size = 0;
```

함수 dirSize의 원형을 선언하고, 전역 변수들을 만든다.

```
/* 디렉토리 혹은 파일 사용량 출력하기 */
int main(int argc, char **argv)
{
    if (argc == 1) // 파일명 입력 안 하면 현재 디렉토리
        file = ".";
    else if (argc == 2)
        file = argv[1];
    else {
        printf("사용법: ./du filename\n");
        exit(1);   // 사용법 안 지키면 에러
    }

    if (lstat(file, &st) == -1){ // 파일 상태 읽기
        perror(file);
    }
    if (S_ISREG(st.st_mode)){ // 일반 파일이면
        size = st.st_blocks / 2; // 사이즈 읽어서 바로 출력하기
        printf("%d\t%s\n", size, file);
        exit(0);
    }
    else if (S_ISDIR(st.st_mode)){  // 디렉토리 파일이면
        dirSize(file);    // 재귀함수 호출해서 디렉토리 밑의 전체 파일 크기 더하기
        printf("%d\t%s\n", size / 2, file);
        exit(0);
    }
    else {
        printf("사용법: ./du filename\n"); //일반 파일이나 디렉토리 파일이 아니면 에러
        exit(1);
    }
}
```

명령어만 입력하면 현재 디렉토리를 대상으로 하고 그렇지 않으면 대상 파일을 설정한다.

명령어가 사용법대로 입력되지 않았으면 올바른 사용법을 출력하고 프로그램을 종료한다.

파일 상태를 읽어서 읽을 수 없으면 에러 메시지를 출력하고 프로그램을 종료한다.

정상적으로 읽은 경우, 일반 파일이면 그 파일의 사용량을 읽어서 바로 출력하고 프로그램을 종료한다.

디렉토리 파일이면 재귀함수인 dirSize를 호출하고 그 결과로 얻은 크기값을 출력한다.

둘 다 아닌 경우 지원하지 않는 파일이므로 파일이름을 입력하라는 메시지를 출력하고 프로그램을 종료한다.

```
/*디렉토리 사용량 계산 재귀함수*/
void dirSize(char *dir) {
    struct stat temp; // 현재 파일의 상태 저장할 임시 구조체
    int cnt = 2;  // 현재 디렉토리와 부모 디렉토리는 건너뛰기 위해
    if (lstat(dir, &temp) < 0 ) //  파일 상태 읽기
        perror(dir);

    if (S_ISREG(temp.st_mode)){  // 일반 파일이면 전역 변수 size에 자기 자신의 크기 더하고 종료
        size += temp.st_blocks;
        return;
    }
    else { //디렉토리 파일이면
        if ((dp = opendir(dir)) == NULL)  // 디렉토리 열기
            perror(dir);

        while ((d = readdir(dp)) != NULL) {
            if (cnt > 0) {
                cnt--;
                continue;  // 맨 앞 두 파일은 부모 디렉토리와 자기 자신이므로 건너뛰고
            }

            else {
                sprintf(path, "%s/%s", dir, d->d_name); // 파일 경로명 만들기
                dirSize(path);    //하위 디렉토리 혹은 파일의 경로명을 대상으로 다시 사용량 계산하기
            }

        }
    }
```

인수로 받은 파일 이름으로 상태를 읽어서 에러가 나면 에러메시지를 출력하고 프로그램을 종료한다.

일반 파일이면 자기 자신의 크기를 전역변수 size에 더하고 return한다.

디렉토리 파일이면 디렉토리를 열어서 에러가 나면 에러메시지를 출력하고 프로그램을 종료한다.

정상적으로 읽으면, 파일을 처음 두 개는 자기 자신과 부모 디렉토리이므로 세 번째부터 차례대로 읽어서 파일 경로명을 만들어 dirSize()에 인수로 파일 경로명을 주고 호출한다.

1.2 실행 결과

```
[chomyungha@localhost 05]$ gcc -o du du.c
[chomyungha@localhost 05]$ ./du
40      .
[chomyungha@localhost 05]$ ./du .
40      .
[chomyungha@localhost 05]$ ./du du.c
4       du.c
[chomyungha@localhost 05]$ du
40      .
[chomyungha@localhost 05]$
```

소스 코드를 컴파일한 후, 실행한다.

명령어만 입력하거나 .을 입력하면 현재 디렉토리를 대상으로 사용량을 보여 준다.

파일 이름을 입력하면 그 파일의 크기를 보여 준다.

실제 du명령어와 결과값이 같은 것을 알 수 있다.