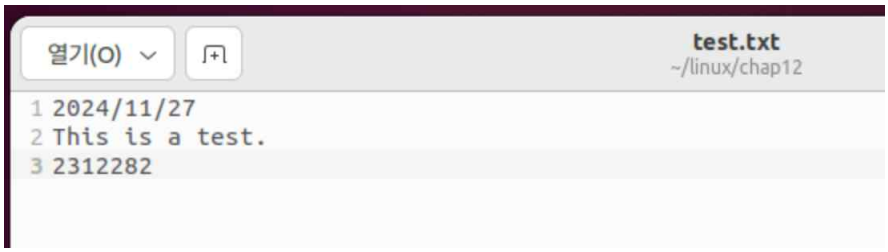
리눅스시스템 Lab12

분반: 001 학과: 컴퓨터과학전공 학번: 2312282 이름: 임다희

1. 문서 편집

- 3줄 이상 적힌 test.txt 파일 생성 뒤,

(1) test.txt 파일을 만들어 이 파일에 대해 $ ls -sl 명령을 사용하고 터미널 창을 캡쳐한 뒤 그 출력 결과를 설명한다.




```
u2312282@dahee-VirtualBox:~/linux/chap12$ gedit test.txt
u2312282@dahee-VirtualBox:~/linux/chap12$ ls -sl test.txt
4 -rw-rw-r-- 1 u2312282 u2312282 35 11월 27 13:18 test.txt
```

- 설명

ls -sl 명령어를 통해 test.txt 파일의 상태를 확인할 수 있다.

순서대로 블록 수(4블록), 파일 타입(일반 파일), 접근 권한(사용자: 읽기/쓰기, 그룹: 읽기/쓰기, 그 외: 읽기만 가능하다), 링크 수(1개), 사용자 아이디(u2312282), 그룹 아이디(u2312282), 최종 수정 시간(11월 27일 13:18), 파일 이름(test.txt) 가 출력된다.

(2) test.txt 파일에 대해 $ stat 명령어를 사용하고 터미널 창을 캡쳐한 뒤 그 출력 결과를 설명한다.

```
u2312282@dahee-VirtualBox:~/linux/chap12$ stat test.txt
  파일: test.txt
  크기: 35              블록: 8          입출력 블록: 4096   일반 파일
Device: 803h/2051d      Inode: 1314494     Links: 1
2접근: (0664/-rw-rw-r--)  UID: ( 1001/u2312282)   GID: ( 1001/u2312282)
접근: 2024-11-27 13:18:20.914587355 +0900
수정: 2024-11-27 13:18:20.924589131 +0900
변경: 2024-11-27 13:18:20.924589131 +0900
생성: 2024-11-27 13:18:20.914587355 +0900
```

- 설명

stat 명령어를 통해 파일의 자세한 상태 정보를 확인할 수 있다. 출력되는 내용은 파일의 이름(test.txt), 파일의 크기(35byte), 할당된 블럭 수(8개), 할당된 블럭의 크기(4096byte), 파일의 유형(일반 파일), 디바이스 번호(16진수의 값), 파일의 Inode 값, 파일의 링크 수, 파일의 접근 권한(사용자: 읽기/쓰기, 그룹: 읽기/쓰기, 그 외: 읽기만 가능하다), 사용자 ID, 그룹 ID, 파일의 최종 접근 시간, 최종 수정 시간, 속성 또는 내용의 최종 변경 시간, 파일을 생성한 시간 이다.

(3) test.txt 파일에 대해 $ ls -i 명령을 사용하고 터미널 창을 캡쳐한다.

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ls -i test.txt
1314494 test.txt
```

(4) test.txt 파일에 대해 $ touch 명령어를 사용하고 터미널 창을 캡쳐한 뒤 사용 전 후의 파일 속성을 비교하여 설명한다.

```
u2312282@dahee-VirtualBox:~/linux/chap12$ touch test.txt
u2312282@dahee-VirtualBox:~/linux/chap12$ ls -sl test.txt
4 -rw-rw-r-- 1 u2312282 u2312282 35 11월 27 13:22 test.txt
u2312282@dahee-VirtualBox:~/linux/chap12$ stat test.txt
  파일: test.txt
  크기: 35              블록: 8          입출력 블록: 4096   일반 파일
Device: 803h/2051d      Inode: 1314494     Links: 1
접근: (0664/-rw-rw-r--)  UID: ( 1001/u2312282)   GID: ( 1001/u2312282)
접근: 2024-11-27 13:22:52.039614630 +0900
수정: 2024-11-27 13:22:52.039614630 +0900
변경: 2024-11-27 13:22:52.039614630 +0900
생성: 2024-11-27 13:18:20.914587355 +0900
```

-설명

$ touch 명령어를 사용하면 파일의 최종 접근 시간, 수정 및 변경 시간을 명령어가 사용된 시점의 시간으로 변경할 수 있다. $ ls -sl 명령어, $ stat 명령어를 이용해 test.txt 의 상태 정보에 접근하면 두 명령어 모두에서 test.txt의 최종 수정 시간이 처음과 달라진 것을 확인할 수 있다.

2. 링크

(1) test.txt 파일에 대한 하드 링크와 심볼릭 링크를 만들고 터미널 창을 캡쳐한다.

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ln test.txt test_01.txt
u2312282@dahee-VirtualBox:~/linux/chap12$ ln -s test.txt test_02.txt
```

(2) 만들어진 하드 링크와 심볼릭 링크의 차이점을 설명한다.

하드 링크는 기존 파일에 대한 새로운 이름으로, 기존 파일을 대표하는 i-노드를 똑같이 가리키는 것으로 구현한다. test.txt와 그의 하드 링크 test_01.txt의 i-노드 값을 확인하면 두 값이 같음을 확인할 수 있다.

심볼릭 링크는 실제 파일의 경로명을 저장하고 있는 일종의 특수 파일이자 포인터의 역할을 하며, 기존 파일과 i-노드의 값이 다르다. test.txt와 그의 심볼릭 링크 test_02.txt의 i-노드 값을 확ㅇ니하면 두 값이 다름을 확인할 수 있다.

(3) $ ls -i 명령을 사용하고 터미널 창을 캡쳐한다.

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ls -i
1314494 test.txt  1314494 test_01.txt  1314508 test_02.txt
```

3. 프로그램 작성 : cat.c

(1) cat 기능 구현을 위해 cat.c를 어떻게 프로그래밍 하였는지 간단하게 설명한다. **단, 설명에는 fgets() 함수가 아닌 어떠한 기능을 사용하여 프로그래밍 하였는지를 반드시 포함하여야 한다.** (기능 구현이 100% 안 되었더라도 구현한 범위까지 설명을 작성해 주세요.)

```c
//001 컴퓨터과학전공 2312282 임다희

#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]){
        FILE *fp;
        int c;
        int show_line_numbers=0;
        int line_number=1;

        if(argc>1){
                if (strcmp(argv[1],"-n")==0){show_line_numbers=1;}
                else if(argv[1][0] == '-') { // $ other options
                        printf("Error: Invalid option '%s'. Please use the -n option\n",argv[1]);
                        return 1;
                }
        }


        if(show_line_numbers==0){ //no option
                if(argc < 2){ // $ cat
                        fp=stdin;
                        int is_first_char=1;
                        while((c=fgetc(fp))!=EOF){
                                if(is_first_char){
                                        printf("   ");
                                        is_first_char=0;}
                                if(c=='\n'){is_first_char=1;}
                                fputc(c,stdout);
                                }
                }

                else{ // $ cat file1, file2..
                for(int i=1; i<argc; i++){
                        fp=fopen(argv[i],"r");
                        if(fp==NULL){
                                printf("File Error: %s\n",argv[i]);
                                return 1;
                        }

                         c=fgetc(fp);
                         while(c!=EOF){
                                 fputc(c,stdout);
                                 c=fgetc(fp);
                                 }
                         fclose(fp);
                         }
                 }
        }

        else if(show_line_numbers==1){ // $ cat -n *
                 if(argc==2){ // $ cat -n
                         fp=stdin;
                         int is_first_char=1;
                         while((c=fgetc(fp))!=EOF){
                                 if(is_first_char){
                                         printf("\t%d  ",line_number);
                                         line_number++;
                                         is_first_char=0;}
                                 if(c=='\n'){is_first_char=1;}
                                 fputc(c,stdout);
                         }
                 }

                 else{ // $ cat -n file1, file2...
                 for(int i=2; i<argc; i++){
                         fp=fopen(argv[i],"r");
                         if(fp==NULL){
                                 printf("File Error: %s\n",argv[i]);
                                 return 1;
                                 }

                         printf("\t%d  ",line_number);
                         c=fgetc(fp);
                         while(c!=EOF){
                                 int next=fgetc(fp);
                                 fputc(c,stdout);
                                 if(c=='\n' && next!=EOF){
                                         line_number++;
```

```
81                                      printf("\t%d  ",line_number);
82                                      }
83                                  c=next;
84                                  }
85                              line_number++;
86                              fclose(fp);
87                      }
88              }}
89
90 return 0;
91 }
```

- 설명

파일을 가리키는 변수 fp, 파일에서 읽어온 문자 하나를 나타내는 변수 c, 명령어가 -n 옵션으로 실행되었는지를 확인하는 변수 show_line_numbers, 줄번호를 나타내는 line_number 변수를 정의한다.

argc가 1 이상인 경우, cat 명령어 바로 다음에 오는 argv의 1번째 원소가 -n 옵션인지를 strcmp의 문자열 비교를 통해 확인한다. -n 옵션임으로 확인된 경우 show_line_numbers 변수의 값을 1로 변경한다. argc가 1 이상이면서 argv의 1번째 원소가 -n 외의 다른 옵션, 즉 "-"로 시작하는 다른 문자열이면서 "-n"이 아닌 경우에는 -n 옵션을 사용할 것을 요구하는 에러 메세지를 출력하고 프로그램을 종료한다.

show_line_numbers 가 0인 경우, 즉 어떤 옵션도 사용하지 않은 경우에서 argc<1($ cat, 파일명 입력 없이 명령어만 사용됨) 이라면 사용자 입력을 파일의 내용을 대신하여 받는다. 사용자가 입력한 내용과 그를 그대로 출력한 다음 줄의 내용을 구분하기 위해 출력되는 내용 각 줄의 첫 글자 앞에 printf를 통한 공백을 만든다. 사용자의 입력이 EOF가 아닌 동안 fgetc(fp)를 통해 변수 c에 입력 내용의 글자 하나씩을 차례대로 읽어들이고, 그 내용을 fputc(c,stdout)을 통해 화면의 바로 다음 줄에 출력한다.

show_line_numbers 가 0이면서 argc>=2인 경우($ cat 뒤에 파일 1개 이상의 이름이 오는 경우) for 문을 이용해 argv의 모든 원소(파일)에 차례로 접근하며 내용을 읽는다. 만일 접근한 파일명이 실제로 존재하지 않는 파일의 이름이라면 에러 메세지를 출력하고 프로그램을 종료한다. 파일이 존재하여 정상적으로 열릴 경우 읽어들인 문자가 EOF가 아닌 동안 계속해서 fgetc(fp)를 통해 변수 c에 파일 내용의 글자 하나씩을 차례대로 읽어들이고, 그 내용을 fputc(c,stdout)을 통해 화면에 출력한다. 파일 출력이 완료되면 fclose(fp)를 통해 해당 파일을 닫아준다.

show_line_numbers 가 1이면서 (-n 옵션이 사용되어 줄번호를 출력해야 하는 경우) argc==2인 경우는 명령어가 $ cat -n 과 같이 사용되어 사용자의 입력을 줄번호와 함께 바로 출력해야 한다. $ cat 의 형태로 사용되었을 때와 같이 작성하고, printf를 통해 출력되는 내용 각 줄의 첫 글자 앞에 line_number를 함께 출력한다. line_number는 한 번 출력된 후 1씩 증가한다.

show_line_numbers 가 1이면서 argc>2 인 경우는 $ cat -n 명령어 뒤에 1개 이상의 파일명이 함께 작성되어 해당 파일들의 내용을 전부 화면에 출력하는 기능을 수행한다. $ cat 뒤에 1개 이상의 파일명이 작성되었을 때와 같이 코드를 작성하고, 바로 직전에 화면에 출력된 문자가 '\n'(줄바꿈 기호) 이면서 다음으로 출력될 문자가 EOF가 아니라면 현재 위치는 줄의 맨 앞이므로 줄 번호

line_number를 출력한다. 한 파일에 대한 while문이 끝나면 line_number를 1 증가시켜 다음 파일의 줄번호 출력에 대비하고 fclose(fp)를 통해 파일을 닫는다.

전체적인 과정에서 fgets() 함수를 통해 문자열 전체를 읽어오는 대신 EOF를 만날 때까지(문자열이 끝날 때까지) 입력받은 문자열의 각 문자 하나씩을 fgetc()를 통해 읽어오는 방식을 사용하였다.

(2) 다음의 예시를 실행한 터미널 창을 캡쳐한다.

[cat 명령어]

[1] 파일 1개에 대해 정상적으로 실행

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat file.txt
Linux System!
This is Lab12.
```

[2] 파일 2개 이상에 대해 정상적으로 실행

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat file.txt test.txt
Linux System!
This is Lab12.
2024/11/27
This is a test.
2312282
```

[3] 파일이 제대로 열렸는지 검사

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ls
cat  cat.c  file.txt  test.txt  test_01.txt  test_02.txt  wc  wc.c
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat file1.txt
File Error: file1.txt
```

[4] 명령줄 인수가 없을 경우, 표준입력으로부터 입력받은 내용을 표준출력에 출력

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat
hi
  hi
hello
  hello
goodbye!
  goodbye!
```

[cat -n 명령]

[1] 정상적으로 실행

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat -n file.txt
     1  Linux System!
     2  This is Lab12.
```

[2] 파일이 제대로 열렸는지 검사

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat -n file1.txt
File Error: file1.txt
```

[3] n이 아닌 다른 옵션을 준 경우, 안내 메시지 출력

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat -o
Error: Invalid option '-o'. Please use the -n option
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat -o file.txt
```

[4] 명령줄 인수가 없을 경우, 표준입력으로부터 입력받은 내용을 표준출력에 출력

```
u2312282@dahee-VirtualBox:~/linux/chap12$ ./cat -n
hi
     1  hi
hello
     2  hello
goodbye!
     3  goodbye!
```

4. 프로그램 작성 : wc.c

(1) wc 기능 구현을 위해 wc.c를 어떻게 프로그래밍 하였는지 간단하게 설명한다.

```c
//001 컴퓨터과학전공 2312282 임다희

#include <stdio.h>
#include <ctype.h>

int main(int argc, char *argv[]){
        FILE *fp;
        int c;
        int line_count=0;
        int word_count=0;
        int char_count=0;
        int word_end=0;

        if(argc<2){ // $ wc

                fp=stdin;
                while((c=fgetc(fp))!=EOF){
                        char_count++;

                        if(c=='\n'){
                                line_count++;}

                        if(isspace(c)){
                                if(word_end){
                                        word_count++;
                                        word_end=0;}
                                }
                        else{word_end=1;}
                }
                if(word_end){word_count++;}
        }

        else{ // $ wc file1
                fp=fopen(argv[1],"r");

                if(fp==NULL){
                        printf("File Error: %s\n",argv[1]);
                        return 1;}

                while((c=fgetc(fp))!=EOF){
                        char_count++;

                        if(c=='\n'){
                                line_count++;}

                        if(isspace(c)){
                                if(word_end){
                                        word_count++;
                                        word_end=0;}
                                }
                        else{word_end=1;}
                }
                if(word_end){word_count++;}
        }

        printf("\t%d\t",line_count);
        printf("%d\t",word_count);
        printf("%d\t",char_count);
        if(argv[1]){printf("%s",argv[1]);}
        fputc('\n',stdout);
        return 0;

}
```

-설명

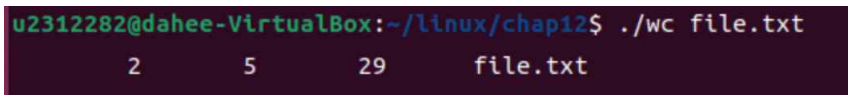
파일을 가리키는 fp, 줄 개수를 세는 line_count, 단어 개수를 세는 word_count, 문자 개수를 세는 char_count, 현재 나온 공백이 한 단어가 끝난 이후에 나온 공백인지 판단하는 word_end 변수들을 선언한다.

argc<2($ wc 형태로 사용된 경우)일 경우 사용자의 입력에 대해 줄, 단어, 문자 수를 세는 기능을 수행한다.(fp=stdin) fgetc(fp)를 통해 입력 내용의 문자를 하나씩 읽을 때마다 char_count를 1씩 증가시킨다. 또한 읽어들인 문자가 줄바꿈 기호 '\n'일 때마다 line_count를 1씩 증가시킨다. 읽어들인 문자가 공백문자이면서 한 단어가 끝난 직후에 나온 공백이라면(word_end==1) word_count를 1씩 증가시키고 word_end의 값을 0으로 초기화한다. 읽어들인 문자가 일반적인 문자일 경우 다음에

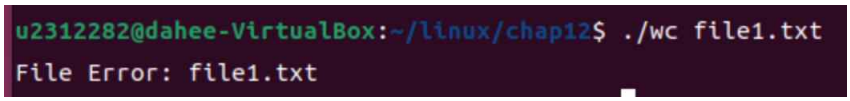
공백이 나온다면 단어가 끝난 것이다. 따라서 word_end 값을 1로 바꾼다.
argc>=2일 경우($ wc 파일명 형태로 사용될 경우)에도 앞의 코드와 동일하게 작성하되, fp는 파일명을 통해 받은 파일을 읽기 모드로 연 것에 해당한다. 만일 파일명이 실제로 존재하지 않는 파일에 해당한다면 경고 메세지를 출력하고 프로그램을 종료한다.

(2) 다음의 예시를 실행한 터미널 창을 캡쳐한다.

[1] 정상적으로 실행



[2] 파일이 제대로 열렸는지 검사



[3] 명령줄 인수가 없을 경우, 표준입력으로부터 입력받은 내용을 표준출력에 출력