UNIX Programming

#01

UNIX명령어

컴퓨터과학과

201710957

이유진

**1) mycat [option][file]**

fp

newfp

FILE\* Function(FILE\* fp)

임시파일 파일포인터 newfp 생성

newfp에 정해진 형식으로 fp읽어서 출력후

fp cloase

FILE\* **squeez**(FILE\* fp)

FILE\* **changeTab** (FILE\* fp)

FILE\* **lineNum** (FILE\* fp)

FILE\* **lineDollar** (FILE\* fp)

원본파일을 읽어 임시파일에 저장하고, 임시파일을 정해진 형식에 맞춰 조작하며 다시 읽어들이기를 위의 형태의 함수를 반복한다.

void **print**(FILE\* fp) 🡪 최종 임시파일을 standard out 을 이용하여 출력한다

**<CODE>**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

void print(FILE\* fp){ **// 최종 결과를 stdout으로 출력하는 함수**

char buf[BUFSIZ];

while(fgets(buf, BUFSIZ , fp) != NULL )

printf("%s",buf);

rewind(fp);

}

FILE\* squeez(FILE\* fp){ **// -s 옵션을 처리하는 함수**

FILE \* tmp\_fp;

char\* empty = "\n";

char buf[BUFSIZ];

int squeez\_ = 0; **// squeez 여부를 나타내는 flag**

if((tmp\_fp = tmpfile() ) == NULL ) {

perror("tempfile()");

exit(1);

}

while(fgets(buf, BUFSIZ , fp) != NULL ){

if(squeez\_){ **//squeez 할 예정임**

if( strcmp(buf, empty)==0 ){ **// current line is empty**

continue; **// 건너뜀**

}

else{

squeez\_ = 0;

fputs(buf,tmp\_fp);  **// 출력**

}

}

else{ **//squeez 하지 않을예정**

if( strcmp(buf, empty)==0 ){ **// current line is empty line**

squeez\_ = 1; **//change flag to ready**

}

fputs(buf,tmp\_fp); **// 출력**

}

}

fclose(fp); **//읽은 파일포인터 close**

rewind(tmp\_fp); **// 다시 읽을 예정이므로 rewind**

return tmp\_fp;

}

FILE\* changeTab (FILE\* fp){

FILE \* tmp\_fp;

char buf[BUFSIZ];

if((tmp\_fp = tmpfile() ) == NULL ) {

perror("tempfile()");

exit(1);

}

while(fgets(buf, BUFSIZ , fp) != NULL ){

for(int i = 0 ; i < strlen(buf) ; i++){

if(buf[i] == '\t'){

**//한글자인 '\t'가 "^I"로 한글자 늘어나므로 한칸씩 뒤로 밈**

for(int j = strlen(buf) ; j > i ; j--){

buf[j+1] = buf[j];

}

buf[i] = '^';

buf[i+1] = 'I'; **//탭문자 대체**

i++;

buf[strlen(buf)+2] = '\0';

}

}

fputs(buf,tmp\_fp);

}

fclose(fp);

rewind(tmp\_fp);

return tmp\_fp;

}

int numlen(int a){ **// Integer 의 자릿수를 계산한다.**

int result = 0;

int ten = 1;

if( a== 0)

return 1;

while( (a/ten) != 0) {

ten \*= 10;

result ++;

}

return result;

}

FILE\* lineNum (FILE\* fp){

**// 출력 형식 : "\t"+ num + "\t" + str**

FILE \* tmp\_fp;

char buf[BUFSIZ];

int num = 1;

char\* empty = "\n"; **// 빈 줄을 비교하기 위한 empty 문자열**

if((tmp\_fp = tmpfile() ) == NULL ) {

perror("tempfile()");

exit(1);

}

while(fgets(buf, BUFSIZ , fp) != NULL ){

int length = numlen(num); **//Line number 의 자릿수**

int temp = num;

**// empty line은 뒤로 shift 하지않고 단순 덧붙여야함.**

if( strcmp(buf,empty) == 0) {

buf[0] = '\t';

**//10진수 숫자를 문자열로 바꿈**

**//linke number 길이 (length)만큼 작성**

for(int i = length ; i >=1 ; i--){

//숫자적어

buf[i] = '0'+(temp % 10);

temp/=10;

}

**//문자열 끝에 개행문자와 '\0' 세팅**

buf[length+1] = '\n';

buf[length+2] = '\0';

}

else{

**// line number 크기 만큼 뒤로 밀어냄**

for(int i = strlen(buf) ; i >= 0 ; i--){

buf[i+2+length] = buf[i];

}

**//문자열 끝에 개행문자와 '\0' 세팅**

buf[strlen(buf)+3+length] = '\0';

buf[0] = '\t';

**//10진수 line number 를 문자열로 바꿈**

for(int i = length ; i >=1 ; i--){

**//숫자작성**

buf[i] = '0'+(temp % 10);

temp/=10;

}

buf[length+1] = '\t';

}

fputs(buf,tmp\_fp);

num++;

}

fclose(fp);

rewind(tmp\_fp);

return tmp\_fp;

}

FILE\* lineDollar (FILE\* fp){ **// 끝에 $기호를 붙이는 함수**

char buf[BUFSIZ];

FILE \* tmp\_fp;

if((tmp\_fp = tmpfile() ) == NULL ) {

perror("tempfile()");

exit(1);

}

**// 문자열 끝에 $ 기호를 붙이고, 개행문자와 ‘\0’ 세팅**

while(fgets(buf, BUFSIZ , fp) != NULL ){

int len = strlen(buf);

buf[len-1] = '$';

buf[len] = '\n';

buf[len+1] = '\0';

fputs(buf,tmp\_fp);

}

fclose(fp);

rewind(tmp\_fp);

return tmp\_fp;

}

int main(int argc, char \* argv[]){

FILE \*rfp, \*tmp\_fp ;

char buf[BUFSIZ];

extern int optopt;

int n;

int flag[] = { 0, 0, 0, 0} ; **// 실행할 함수를 알기위한 flag**

**// 순서대로 squeez, tab, dollar, num**

if( argc < 2 ){ **// argc 가 최소 2개 있어야 하기 때문에**

printf("no File\n");

exit(1);

}

**// 읽어들일 파일 포인터 open**

if((rfp = fopen(argv[argc-1], "r")) == NULL ){

perror("open err");

exit(1);

}

**//임시파일 파일 포인터 open**

if((tmp\_fp = tmpfile() ) == NULL ) {

perror("tempfile()");

exit(1);

}

**//임시파일에 읽을 파일 파일 복사**

while(fgets(buf, BUFSIZ , rfp) != NULL )

fputs(buf,tmp\_fp);

fclose(rfp);

rewind(tmp\_fp);

while((n=getopt(argc, argv, "stTeEnbA")) != -1){

if(optopt != 0 )

return 0;

switch (n) {

case 's' : **// squeez**

flag[0] = 1;

break;

case 't' :

case 'T' : **// tab문자 ^I 로 바꾸**

flag[1] = 1;

break;

case 'e' : **// 맨마지막 끝에 $**

case 'E' :

flag[2] = 1;

break;

case 'n' : **//line number 표시**

case 'b' :

flag[3] = 1;

break;

case 'A' :

flag[1]=flag[2] = 1;  **// -t -e 와 동일동작**

break;

}

}

**// 정해진 순서대로 실행하지 않으면 정확하지 않은 결과가 나타날수 있음**

**// squeez -> changeTap -> lineDollar -> lineNum 으로 처리함**

if( flag[0])

tmp\_fp = squeez(tmp\_fp);

if( flag[1])

tmp\_fp = changeTab(tmp\_fp);

if(flag[2])

tmp\_fp = lineDollar(tmp\_fp);

if( flag[3] )

tmp\_fp = lineNum(tmp\_fp);

**//처리 후 최종본 stdout에 print**

print(tmp\_fp);

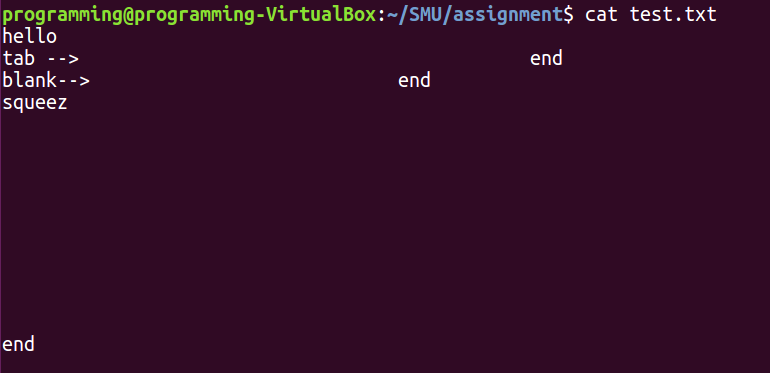
fclose(tmp\_fp);

return 0;

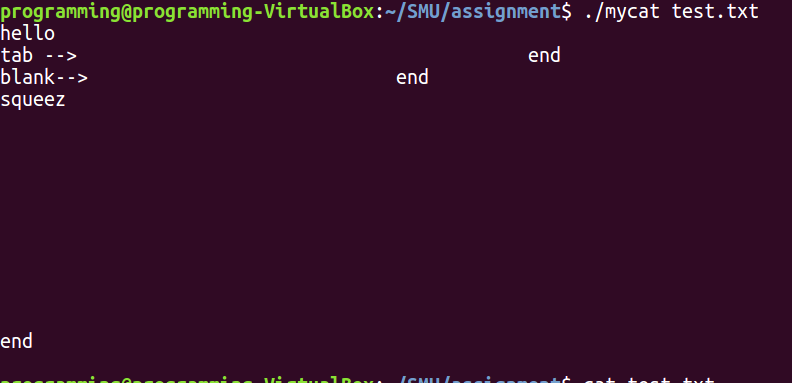
}

**<수행예제>**

수행 예제를 위해 test.txt 파일을 생성하였다.

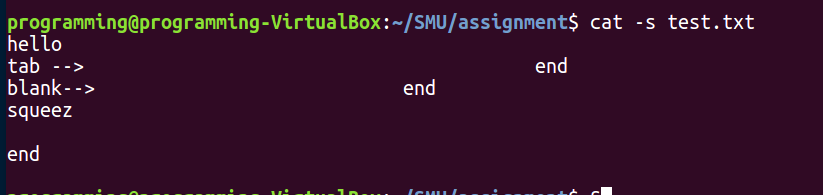


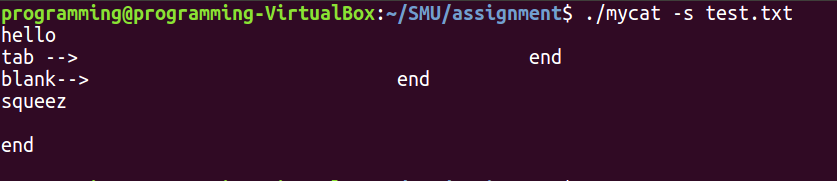
1 ) ./mycat test.txt



2 ) ./mycat -s test.txt

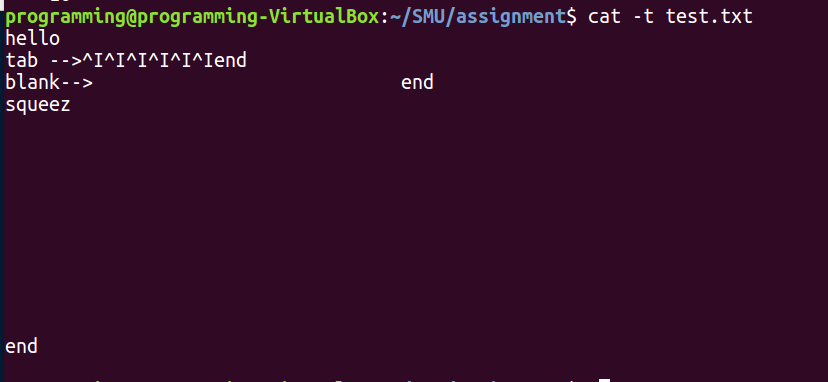
-s 옵션을 이용하면 빈 줄을 압축한다.

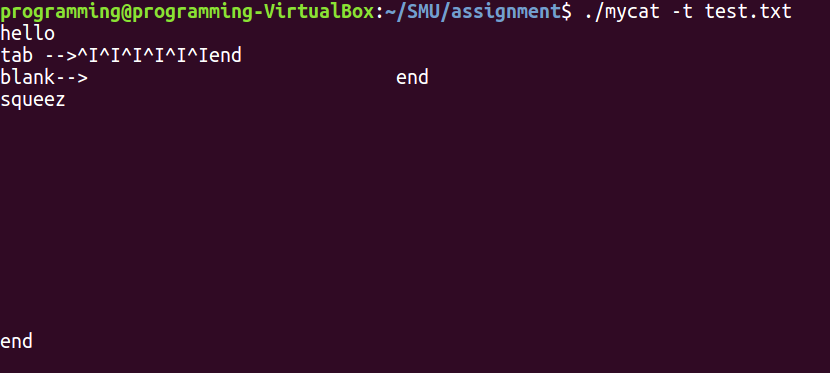




3 ) ./mycat -t test.txt

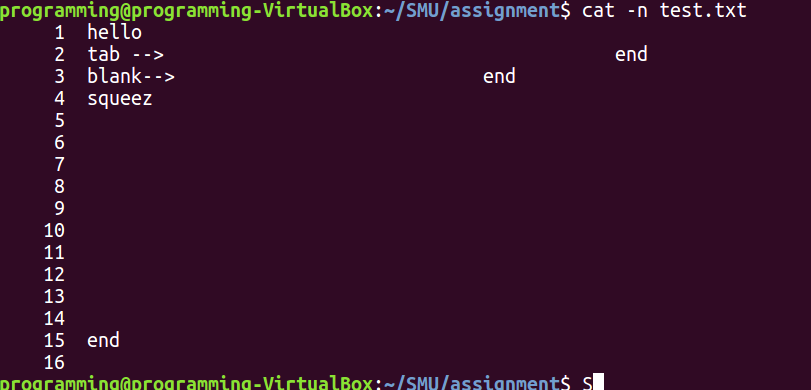
-t 옵션을 이용하면 탭문자를 ^I 로 바꾸어 출력한다.

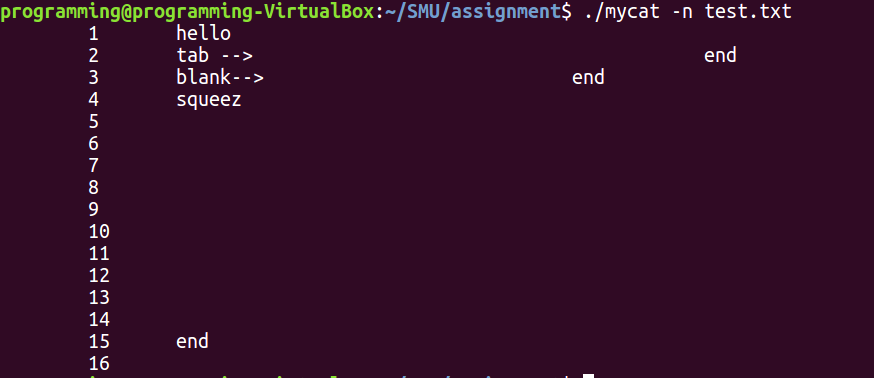




4 ) ./mycat -n test.txt

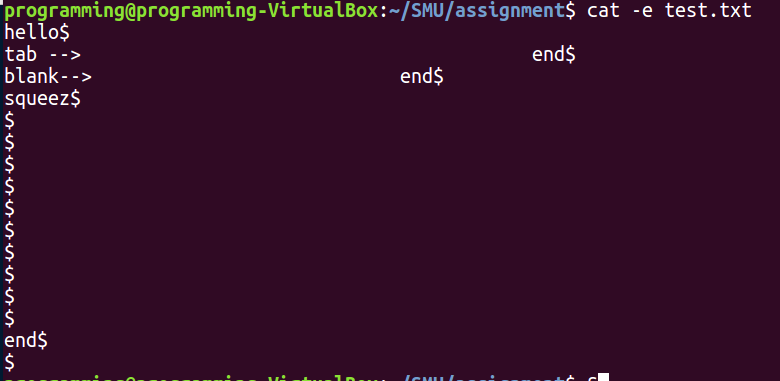
-n 옵션을 이용하면, 각 줄의 번호를 같이 출력한다.

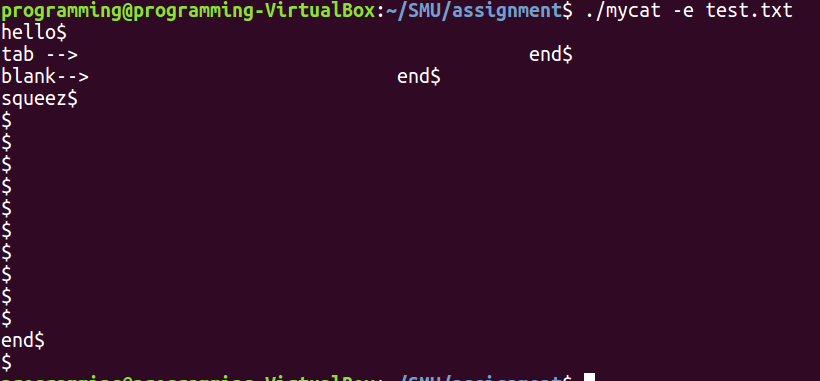




5 ) ./mycat -e test.txt

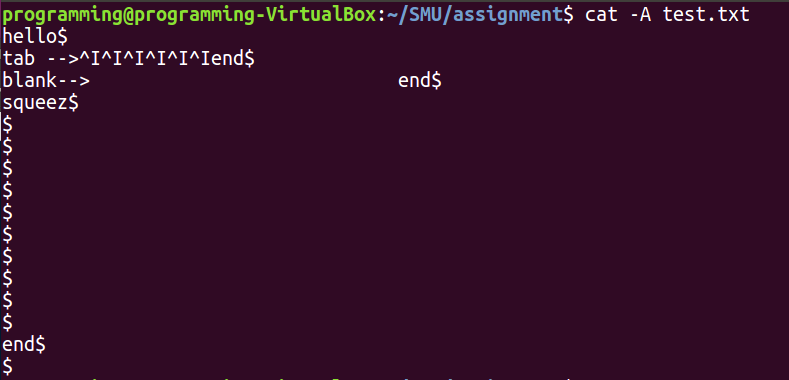
-e 옵션을 이용하면 문장의 끝에 $ 기호를 붙여 출력한다.

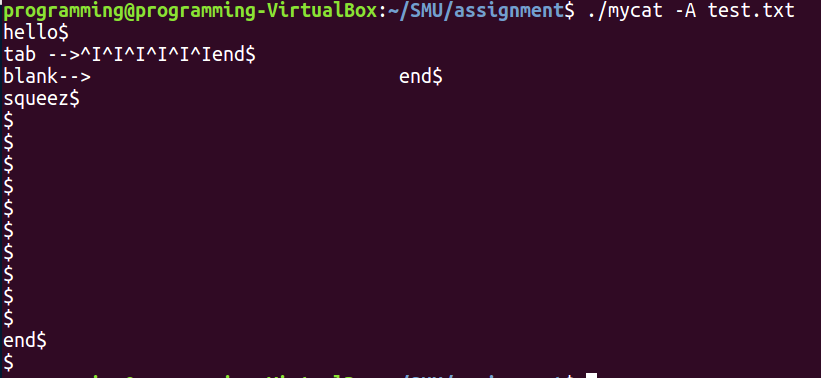




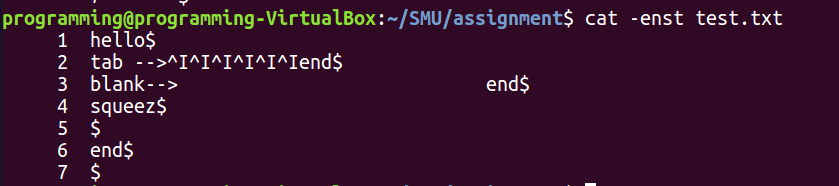
6 ) ./mycat -A test.txt

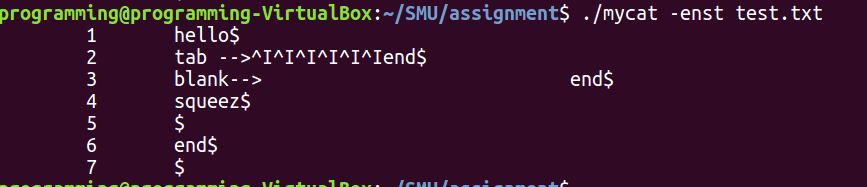
-A 옵션을 이용하면, 탭문자를 ^I로 치환하고 문자열 끝에 $을 덧붙여 출력한다.





7 ) 여러가지 옵션을 한번에 적용한 경우





**2) mycp source dest 또는 source … directory**

void copy(char \* source, char \* dest)

🡪 soure를 dest에 복사한다

void go\_copy(int i, int n, int s, int u, int is\_dir , char \*source, char\* cp)

🡪 option 에 맞는 flag들을 받아 copy를 진행할 것인지 결정한다.

**<CODE>**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

void copy(char \* source , char \* copy ){ **//파일을 복사 하는 함수**

int rfd = open(source, O\_RDONLY); **//읽어 들이기 위한 파일기술자 open**

int wfd = open(copy, O\_CREAT|O\_WRONLY|O\_TRUNC , S\_IRUSR|S\_IWUSR);

**// 복사하기위한 파일 기술자 open**

**// 파일이 존재해도 삭제하고 다시 write하기위해 Option 이용**

int n;

char buf[10];

if( rfd == -1){

perror("open source err");

exit(1);

}

if( wfd == -1){

perror("open copy err");

exit(1);

}

while( (n = read(rfd, buf, 4)) > 0 ){ **//4바이트씩 읽어 복사한다.**

if( write(wfd, buf, n) != n ) **// 읽은 크기와, 쓴 크기가 다르다면 err**

perror("Write");

if( n == -1 ) **// 읽은 크기가 -1 이라면 err**

perror("Read");

}

close(rfd);

close(wfd); **// 열었던 파일 기술자들 close**

}

void go\_copy(int i, int n, int s, int u, int is\_dir , char \*source, char\* cp){

**// 사용할 option들과 flag를 전달받아 복사 유무를 체크하는 함수**

int check;

int exist = 0; **// copy의 존재 유무를 확인할 flag**

int go = 0; **// copy 진행 유무를 나타낼 flag**

char aws[10];

int update = 0; **// copy 파일이 update 가 필요한지 나타낼 flag**

char path[50]; **//복사할 파일의 정확한 위치 저장**

**//cp source dest 형태 의 명령**

**// dest 의 정확한 위치를 지정하기위한 코드**

if(is\_dir){ **//dest가 dir**

strcpy(path, cp);

strcat(path,"/");

strcat(path, source); **// path 🡪 디렉토리/source이름**

}

else{ **// cp가 복사본 이름**

strcpy(path, cp);

}

**//dest 파일 존재 확인**

if( (check = access(path, 0) ) == 0 ) **// 존재**

exist = 1;

**// 존재함**

if( exist ==1 ){

if( s== 1) **// 같은파일이름파일존재하면 symlink 불가**

printf("cp: cannot create symbolic link 'test3.txt' to 'test.txt': File exists\n");

if(u){ **// -u flag 있음**

**// buf1, buf2 에 파일 정보 저장**

struct stat buf1, buf2;

stat(source, &buf1);

stat(path,&buf2);

**// 각 파일의 수정시간을 저장**

time\_t t1 = buf1.st\_ctime;

time\_t t2 = buf2.st\_ctime;

if( t1 > t2 ){ **// source수정시간 > dest 수정시간**

**// update 진행해야함**

**// 복사 시작**

if( n == 1 ) **// -n 은 존재할경우 무조건 복사 금지**

go = 0;

else if( i == 1) { **// -n 옵션 없고, -I option 있음**

**// overwrite 여부를 물어봄**

printf("cp : overwrite '%s' ?", "name");

scanf("%s", aws);

if( (aws[0] == 'y' || aws[0] == 'Y')){

go = 1;

}

else

go = 0;

}

else{

go = 1;

}

}

else {

go = 0; **//복사필요 x**

}

}

else { **//update flag 없음**

if( n == 1 ) **// 존재하니까 복사x**

go = 0;

else if( i == 1) { // n == 0

printf("cp : overwrite '%s' ?", "name");

scanf("%s", aws);

if( (aws[0] == 'y' || aws[0] == 'Y')){

go = 1;

}

else

go = 0;

}

else{

go = 1;

}

}

}

else  **// flag 없음**

go = 1;

**// 본격 copy start**

if(go){

if(s) **//복사대신 심볼릭 링크**

symlink (source,path);

else

copy(source, path);

}

}

int main(int argc, char \* argv[]){

extern int optopt , optind;

int nn;

int i,n,s,u;

i = n = s = u =0;

char \* source;

char \* copy;

char \* copypath;

if( argc < 3 ){

printf("err\n");

exit(1);

}

while((nn=getopt(argc, argv, "insu")) != -1){

if(optopt != 0 )

return 0;

switch (nn) {

case 'i' : **// overwrtie 하기전에 물어보기**

i = 1;

break;

case 'n' : **// 물어보지말고 걍 overwirte 금지**

n = 1;

break;

case 's' : **// symbolic link**

s = 1;

break;

case 'u' :

u = 1;

break;

}

}

if( argc < (optind +2 ) ){ **// 인자 부족**

printf("missing\n");

return 0;

}

else if( argc == (optind+ 2) ){ **// cp source dest**

struct stat buf;

stat(argv[argc-1], &buf);

int kind = buf.st\_mode & S\_IFMT;

int is\_dir;

if( kind == S\_IFDIR) **// cp 파일1 디렉토리**

is\_dir =1;

else **// cp 파일 파일**

is\_dir = 0;

go\_copy(i,n,s,u,is\_dir, argv[argc-2] , argv[argc-1]);

}

else{

**// cp 파일1 파일2 파일3 ... 디렉토리**

struct stat buf;

for(int idx = optind; idx < argc-1 ; idx++){

stat(argv[idx], &buf);

int kind = buf.st\_mode & S\_IFMT;

if( kind == S\_IFDIR){

printf("err");

return 0;

}

go\_copy(i,n,s,u,1, argv[idx] , argv[argc-1]);

}

}

return 0;

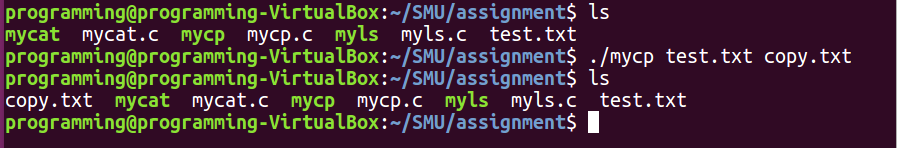
}

**<수행예제>**

1 ) ./mycp test.txt copy.txt

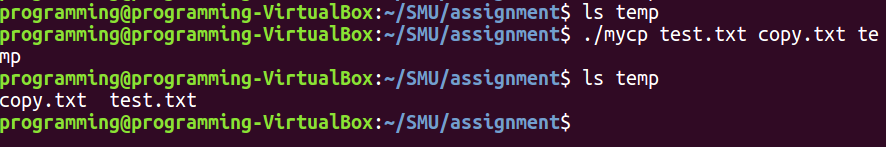
현재 파일 경로에 test.txt 파일을 copy.txt 이름으로 복사한다.

명령 실행 후 ls 를 실행하니, copy.txt가 생성된 것을 알수 있다.

****

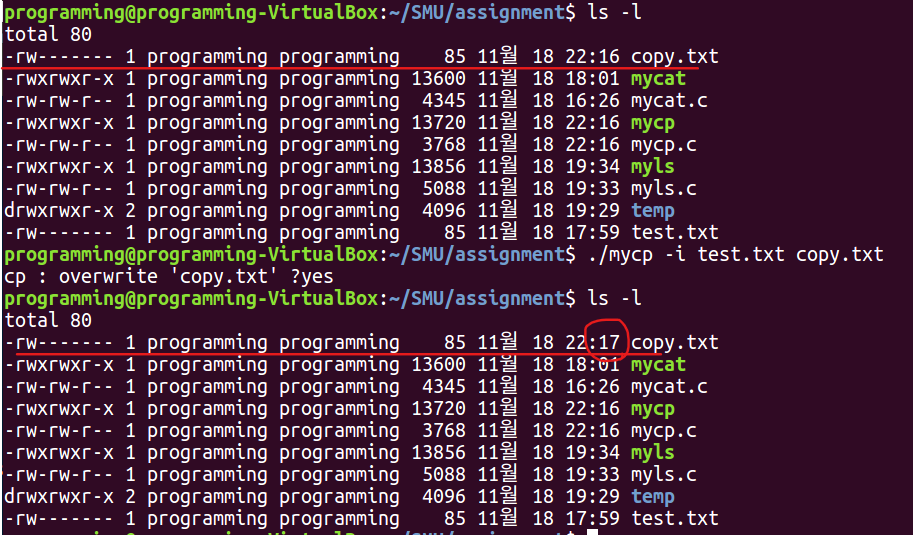
2 ) ./mycp test.txt copy.txt temp

test.txt파일과 copy.txt 파일을 temp 폴더에 복사한다.

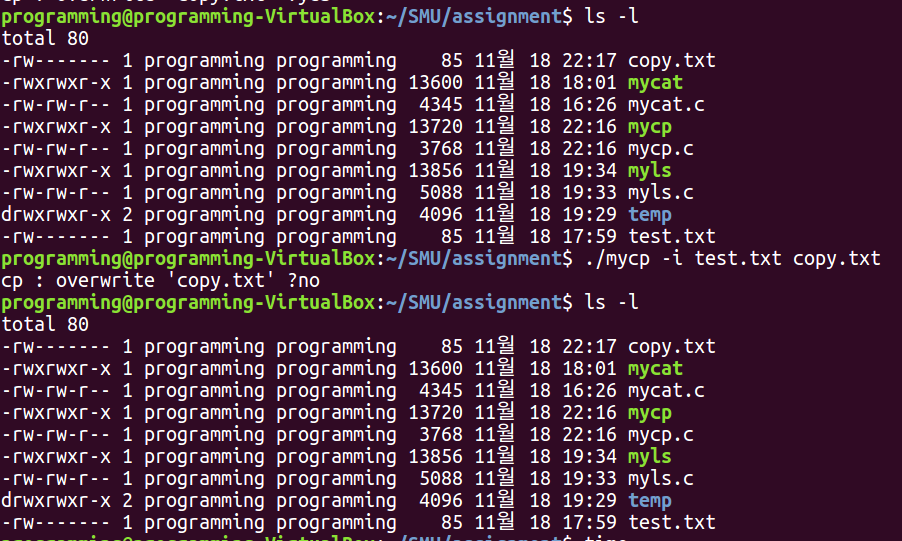
****

3 ) ./mycp -i test.txt copy.txt

이미 copy.txt 가 존재하기 때문에, overwrite 유무를 물어보고, Y혹은 y로 시작하는 대답이면 ovewrite한다.

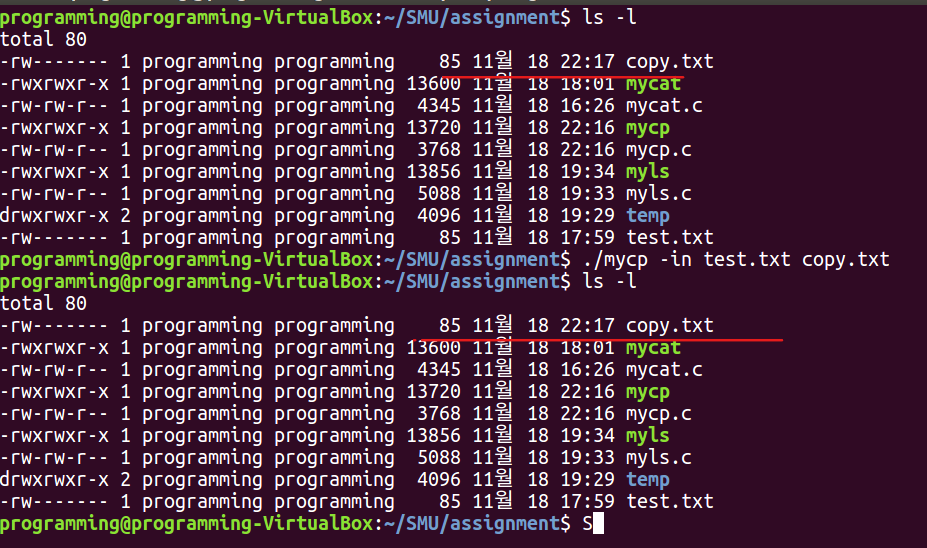


no를 입력하면 변화 없음.

****

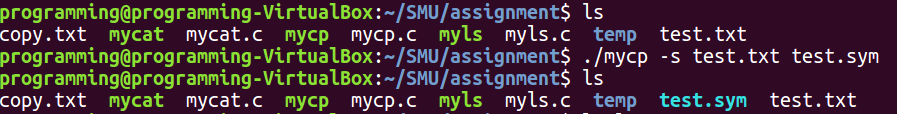
4 ) ./mycp -in test.txt copy.txt

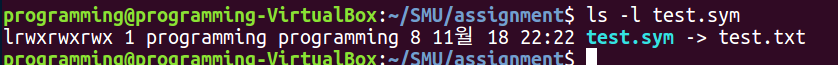
-n 옵션 으로 인해 중복된 파일은 복사하지않는다.



5 ) ./mycp -s test.txt test.sym

심볼릭 링크를 생성한다.





**3) myls [option] [file]**

void **print\_mode**(int mode)

🡪 S\_IFMT와 S\_IREAD , S\_IWRITE , S\_IEXEC 마스크를 이용하여 파일의 종류와 정수형 mode 를 캐릭터 타입으로 출력한다.

void **printFILE**(struct stat buf,char \* name , int l, int i ,int A, int a, int G , int Q,int m,int F)

🡪stat을 이용하여 파일들의 정보를 출력한다.

void **printDIR**(char \* dir , int l , int i, int A, int a, int G ,int Q ,int m,int F)

🡪디렉토리의 이름을 이용하여 디렉토리를 오픈하고, printFILE을 호출하여 각 item들을 출력한다.

**<CODE>**

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <pwd.h>

#include <grp.h>

#include <time.h>

#include <dirent.h>

#include <stdlib.h>

#include <string.h>

void print\_mode(int mode) {

int kind = mode & S\_IFMT;

if(kind == S\_IFDIR) **//파일의 종류에맞게 출력**

printf("d");

else if( kind == S\_IFBLK)

printf("b");

else if( kind == S\_IFCHR)

printf("c");

else if(kind == S\_IFLNK)

printf("l");

else if( kind == S\_IFSOCK)

printf("s");

else if( kind == S\_IFIFO)

printf("p");

else

printf("-");

int read = S\_IREAD;

int write= S\_IWRITE;

int exec = S\_IEXEC;

for(int i = 0 ; i < 3 ; i++ ){

if(( mode & read ) != 0 )

printf("r");

else

printf("-");

if( (mode & write ) != 0 )

printf("w");

else

printf("-");

if( (mode & exec ) !=0 )

printf("x");

else

printf("-");

read = read >> 3;

write = write >> 3;

exec = exec >> 3; **//3비트 shift 하면서 user->group->other**

}

printf(" ");

}

void printFILE(struct stat buf,char \* name , int l, int i ,int A, int a, int G , int Q,int m,int F){

**//inode i == 1일때**

if(i)

printf("%d ", (int)buf.st\_ino);

if(l){  **// -l 옵션**

print\_mode((int)buf.st\_mode);

printf("%d ", (int) buf.st\_nlink);

struct passwd \*pw = getpwuid(buf.st\_uid);

printf("%s ", pw->pw\_name );

if(!G){ **// group id 출력**

gid\_t group\_id = buf.st\_gid;

struct group\* group\_entry;

group\_entry = getgrgid(group\_id);

printf("%s ", group\_entry->gr\_name);

}

printf("%6d ", (int)buf.st\_size);

struct tm \* tm;

char temp[257];

time\_t t = buf.st\_ctime;

tm = localtime(&t);

strftime(temp, sizeof(temp),"%m월 %m %H:%M" , tm);

printf("%s ", temp);

}

if(Q){ **// 파일명 따옴표로 묶기**

printf("\"%s",name);

if(F){ **// 이름 뒤에 파일종류기호 덧붙임**

int kind = (int)buf.st\_mode & S\_IFMT;

if(kind == S\_IFDIR) **//파일의 종류에맞게 출력**

printf("/");

else if(kind == S\_IFLNK)

printf("@");

else if( kind == S\_IFSOCK)

printf("=");

else if( kind == S\_IFIFO)

printf("|");

}

printf("\""); **// 따옴표 출력**

}

else{ **//단순 이름출력**

printf("%s", name);

if(F){

int kind = (int)buf.st\_mode & S\_IFMT;

if(kind == S\_IFDIR) **//파일의 종류에맞게 출력**

printf("/");

else if(kind == S\_IFLNK)

printf("@");

else if( kind == S\_IFSOCK)

printf("=");

else if( kind == S\_IFIFO)

printf("|");

}

}

}

void printDIR(char \* dir , int l , int i, int A, int a, int G ,int Q ,int m,int F){

**//하나씩 읽어서 printFILE 하기.**

DIR \*dp;

struct dirent \*dent;

if( (dp = opendir(dir)) == NULL ){

perror("opendir");

exit(1);

}

int item = 0; **// 파일의 항목 수**

while( (dent = readdir(dp) )) { **// 파일의 항목 수를 count 한다**

item ++;

}

rewinddir(dp); **// 디렉토리 포인터 앞으로당김**

ㅡ

while( (dent = readdir(dp) )) {

item --;

struct stat sbuf;

char path[BUFSIZ];

char name[BUFSIZ]; **// 파일의 정확한 위치를 저장하기위한 문자열**

strcpy(path , dir);

strcat(path,"/");

sprintf(name, "%s", dent->d\_name);

strcat(path,name);

lstat(path,&sbuf);

if( strcmp( name, ".") ==0 ){ **// -a 일때만 출력**

if(a == 1){

**// 파일의 내용을 옵션에 맞춰 출력한다**

printFILE(sbuf, dent->d\_name, l,i,A,a,G,Q,m,F);

if(!l){ **//**

if(m && item)

printf(", ");

else

printf(" ");

}

else

printf("\n");

}

}

else if( strcmp( name, "..") ==0 ){ **// -a 일때만 출력**

if(a == 1){

printFILE(sbuf, dent->d\_name, l,i,A,a,G,Q,m,F);

if(!l){

if(m && item)

printf(", ");

else

printf(" ");

}

else

printf("\n");

}

}

else if( name[0] == '.' ){ **// 숨김파일**

if( a == 1 || A == 1 ){

printFILE(sbuf, dent->d\_name, l,i,A,a,G,Q,m,F);

if(!l){

if(m && item)  **//-l 형태가아니고 -m 옵션이있을때**

printf(", ");

else **// 마지막 파일은 따옴표 붙이지 않음**

printf(" ");

}

else

printf("\n");

}

}

else{

printFILE(sbuf, name , l,i,A,a,G,Q,m,F);

if(!l){

if(m && item) )  **//-l 형태가아니고 -m 옵션이있을때**

printf(", ");

else **// 마지막 파일은 따옴표 붙이지 않음**

printf(" ");

}

else **// 다출력하고 개행**

printf("\n");

}

}

if(!l) **// 출력하고 개행**

printf("\n");

}

int main(int argc, char \*argv[]){

int n;

struct stat buf;

int kind;

int l, i, A, a, G , Q,m,F; **//옵션을 표시하기위한 flag**

l = i = A = G =a = Q=m=F= 0; **//flag 초기화**

extern int optopt, optind;

while((n=getopt(argc, argv, "liAaGoQmF")) != -1){

if(optopt != 0)

return 0;

switch (n) {

case 'l' :

l = 1;

m = 0;

break;

case 'i' :

i = 1;

break;

case 'A' :

A = 1;

break;

case 'a' :

a = 1;

break;

case 'o' :

l = 1;

G = 1;

break;

case 'G' :

G = 1;

break;

case 'Q' :

Q = 1;

break;

case 'm' :

m = 1;

l = 0;

break;

case 'F' :

F=1;

break;

}

}

if( argc == optind ){ **//option 없는 cmd**

char\* cwd = getcwd(NULL, BUFSIZ);

printDIR(cwd, l,i,A,a,G,Q,m,F);

}

else{

lstat(argv[argc-1], &buf);

kind = buf.st\_mode & S\_IFMT;

if( kind == S\_IFDIR){ **// 디렉토리를 읽을 때**

printDIR(argv[argc-1], l,i,A,a,G,Q,m,F);

}

else{ **// 다른 파일들**

printFILE(buf,argv[argc-1], l,i,A,a,G,Q,m,F);

printf("\n");

}

}

return 0;

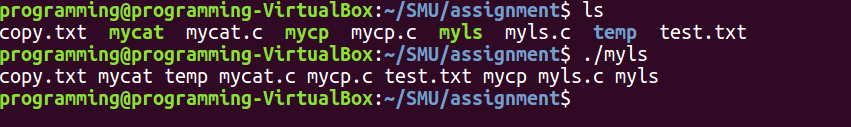
}

**<수행예제>**

1 ) ./myls

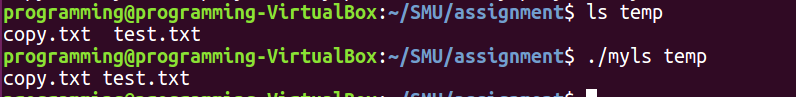
현재 디렉토리의 파일들을 출력한다.

파일의 순서는 정렬하지 않았기 때문에 동일하지 않지만, 출력된 파일은 동일하다.



2 ) ./myls temp

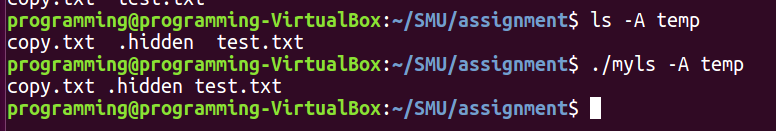
temp 디렉토리의 파일들을 출력한다.



3 ) ./myls -A temp

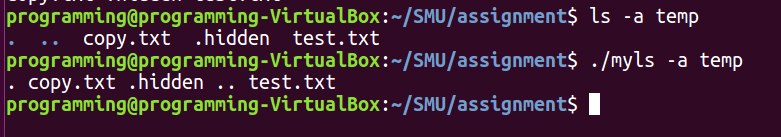
temp 디렉토리에는 hidden 이라는 이름을 가진 숨김파일이 있다.

-A 옵션을 이용하여 출력하면 1,2번 예시에는 보이지 않았던 hidden 파일이 출력된다.



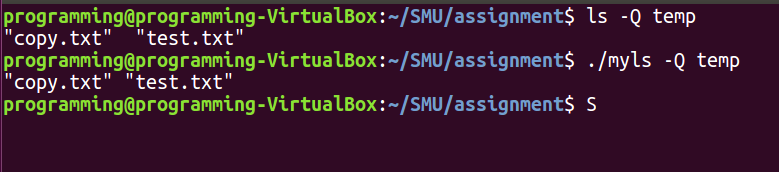
4 ) ./myls -a temp

-a 옵션을 이용하면, 숨김파일과, “.” , “..” 파일 또한 출력된다

-

5 ) ./myls – Q temp

-Q 옵션을 이용하여 파일의 이름을 따옴표를 이용하여 출력한다.

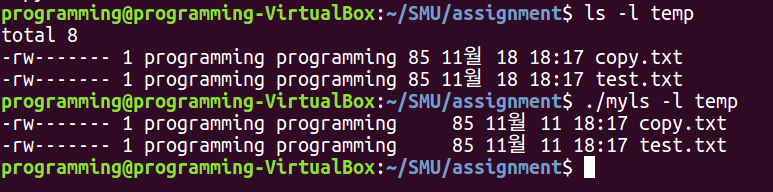


6 ) ./myls -l temp

temp 디렉토리에 있는 파일들을 -l 옵션을 이용하여 출력하게되면

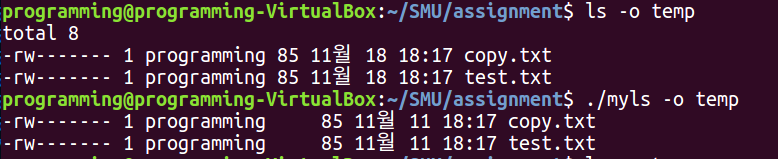
파일권한, 링크수, 소유사용자명, 소유그룹명, 수정시간, 파일이름 순서대로 출력된다.

또 모든 하위 파일들의 개수인 total이 출력되지만, 출력하지 못하였다.



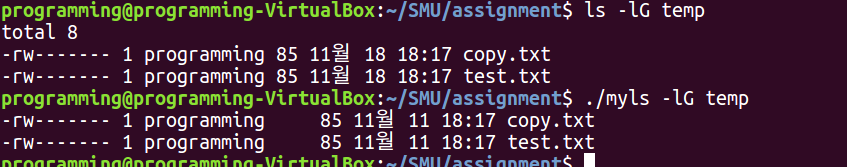
7 ) ./myls -o temp

-o 옵션을 이용하면, -l 옵션에서 소유 그룹명을 제외하여 출력한것과 같다.



8 ) ./myls -lG temp

-G 옵션을 이용하여 소유그룹명을 출력하지 않았다. 예시 7번과 동일한 결과가 나온다.

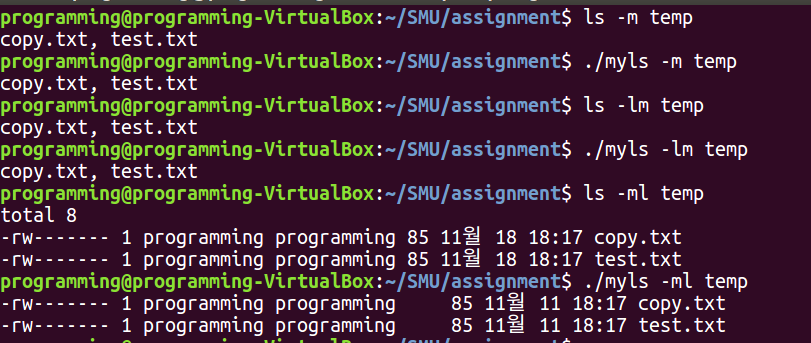


9 ) ./myls -m temp

-m 옵션을 이용하면 파일을 쉼표로 구분하여 출력한다.

-l 옵션과 동시에 실행될수 없으므로, 제일 마지막에 사용된 옵션을 기준으로 출력하게된다.

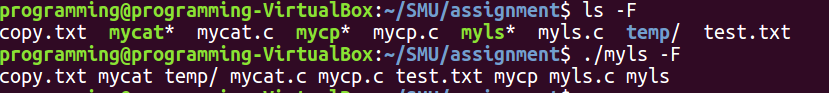
-lm 옵션에서는 -m 이 실행되고, -ml 옵션에서는 -l 이 출력된다.



10 ) ./myls -F

-F옵션을 이용하면 파일이 실행파일 일때는 \* , 디렉토리는 / , 심볼릭링크파일은 @ 소켓파일은 = , FIFO 파일은 | 으로 나타낸다.

실행파일은 일반파일의 한 종류이기 때문에 구분하지못하여 출력하지 않았고, 심볼릭링크만 @기호를 이용하여 출력하여야하는데, 링크파일을 @으로 출력하였기 때문에 하드링크또한 @문자가 붙어 출력하는 오류가 있다.



11 ) ./myls -li temp

temp 의 파일들을 inode를 포함하여 출력한다

