Advanced System Programming

My\_command set

구현 계획서

Kookmin Univ.

Computer Science Dept

20113315 이형준

1. **개요**
   1. 개발환경

개발자 : 이형준

개발 환경 : Raspberry Pi 2, Model B

RASPBIAN WHEEZY 3.18

Using putty‘s SSH by window 8.1K

개발 도구 : nano 2.0.2 by LINUX

사용 컴파일러 : GNU Compiler Collection 4.9

* 1. 계획서 목차

구현 명령어 :

rm [-f, -i]

mkdir [ -p]

ln [ -s]

head/tail [ -n]

cat

touch

chmod

pwd

각 구현된 명령어 별로 다음과 같은 목차를 수반한다.

* + - * 메뉴얼 페이지
      * 테스트 내용
      * 사용된 시스템콜 및 황용
      * 소스코드설명
      * 주석처리된 소스코드

1. **구현명령어**
   1. **mkdir**
      1. **메뉴얼 페이지**

Name : **make directorie**

Synopsis : mkdir [OPTION]... DIRECTORY...

Description : Create the DIRECTORY, if they do not already exist.

option

**-p**

no error if existing, make parent directories as needed

디렉토리를 생성한다. 파라미터의 개수에 따라서 여러개의 디렉토리 생성이 가능하다. 파라미터로 디렉토리의 상대경로 / 절대경로를 포함시켜서, 해당 경로 안에 있는 디렉토리 생성이 가능하다. 단 해당 경로가 존재하지 않으면 에러메세지를 출력한다.

**-p옵션**

경로가 존재하지 않을때, 디렉토리 구조를 한번에 생성한다

옵션은 작성한 위치에 관계없이 적용된다.

* + 1. **테스트**
       1. 현재 디렉토리에서 디렉토리를 하나 생성한다.

[p@hjlee~] $ mkdir dir1

[p@hjlee~] $ ls

dir1 a b

**Result**



**Success**

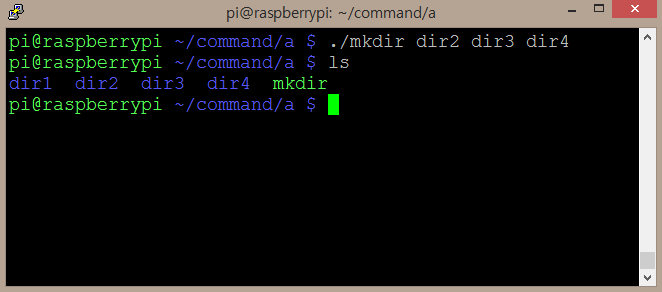
* + - 1. 여러개의 디렉토리 명을 입력한다.

[p@hjlee~] $ mkdir [dir2] [dir3] [dir4]

[p@hjlee~] $ ls

dir4 dir3 dir2 dir1 a b

**Result**



**Success**

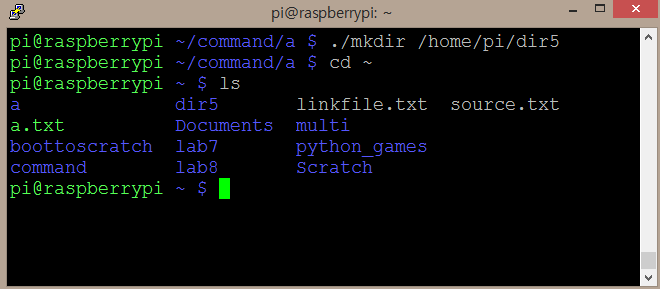
* + - 1. 파라미터에 상대경로 /절대경로를 포함시킬 수 있다.

[p@hjlee~] $ mkdir /home/pi/dir5

[p@hjlee~] $ ls

dir5 dir4 dir3 dir2 dir1 a b

**Result**



**Success**

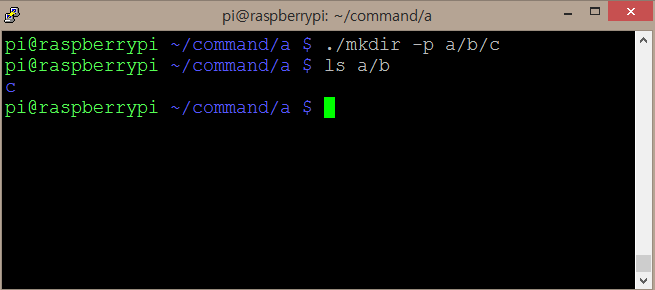
* + - 1. **–p** 옵션

[p@hjlee~] $ mkdir -p a/b/c/

[p@hjlee~] $ ls a/b

c

**Result**



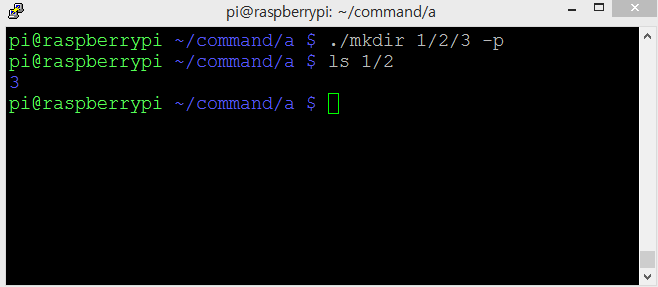
**Success**

* + - 1. -p 옵션은 작성한 위치에 관계없이 적용된다.

[p@hjlee~] $ mkdir a/b/c -p

결과는 위와 동일하다.

**Result**



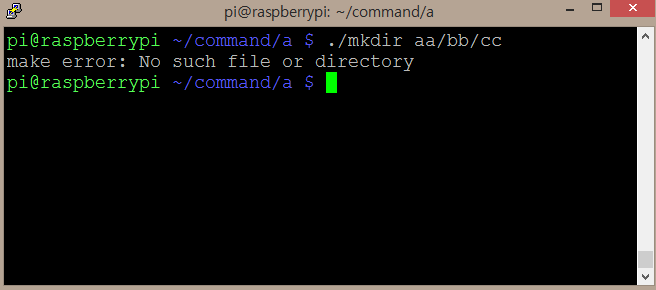
**Success**

* + - 1. 파라미터에 경로를 입력할때, 경로가 존재하지 않으면 에러처리한다.

[p@hjlee~] $ mkdir aa/bb/cc

[p@hjlee~] $ ls

**Result**



**Success**

* + 1. **사용된 시스템콜**

**mkdir()**

Name : make directory

Synopsis : int mkdir(const char \*pathname, mode\_t mode);

pathname : 생성할 디렉토리 명

mode : permission관련 / umask.

Description : pathname이름을 가지는 디렉토리를 만들려고 시도한다.

return : 성공시 0, 실패시 -1 반환한다.

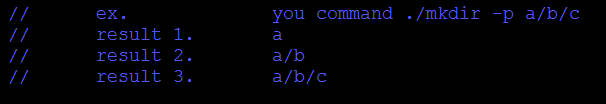
* + 1. **소스코드설명**
       1. option을 detect한다.

argv로 입력받은 파라미터 안에 옵션이 포함되어 있는지 검사한다.

* + - 1. option의 유/무 에 따라서 분기를 나눈다.
      2. 해당 option에 맞는 flag를 대입해서 함수를 호출한다.
      3. 호출된 함수는 다음과 같은 작업을 한다.
         1. P옵션을 가졌을때.

우선 / 을 기준으로 자른다. 예를들어 인자로 a/b/c가 들어왔다면 a, b, c 이렇게 세 부분으로 나눈다. 그리고 그 결과값을 배열에 넣어서 차례차례 front부터 디렉토리를 생성해주고, 그 생성된 디렉토리에 들어가서 다음 디렉토리를 생성한다. ./mkdir –p a/b/c 을 명령어에 삽입시켰다면 우선 a를 생성하고 그다음 a디렉토리에 들어가서 b를 생성한다. 그리고 a/b디렉토리에 들어가서 c를 생성한다.

다음은 코드의 주석부분이다. 이부분을 참고하면 이해하기 편할것이다.



* + - * 1. P옵션이 아니라면

P옵션일때와 함수의 코드는 똑같다. 단 for문이 한번만 실행되고 코드는 똑같다고할 수있다.

* + - 1. p옵션이 없는데, 경로가 존재하지 않을경우 에러처리를 해준다.
    1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다.**

**보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  |  |
|  | void p\_option\_cutAndmake(int n, int argc, char\* dir[]); |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | if(argc<2){ |
|  | printf("error\n"); |
|  | exit(1); |
|  | } |
|  |  |
|  | int option\_p=0; |
|  | int i,j; |
|  | //option detect |
|  | for(i=j=1; i<argc; i++) { |
|  | if(argv[i][0]=='-'){ |
|  | if(argv[i][1] == 'p') |
|  | option\_p=1; |
|  | } |
|  | } |
|  | //not -p option. |
|  | if(option\_p==0){ |
|  | for(i=1; i<argc; i++){ |
|  | if(mkdir(argv[i],0777)!=0) |
|  | perror("make error"); |
|  | } |
|  | } |
|  | // -p option adjust |
|  | if(option\_p==1){ |
|  | for(i=1; i<argc; i++){ |
|  | if(argv[i][0] == '-') |
|  | continue; |
|  | p\_option\_cutAndmake(i,argc,argv); |
|  | } |
|  | } |
|  | return 0; |
|  | } |
|  |  |
|  | void p\_option\_cutAndmake(int i, int argc, char\* dir[]) |
|  | { |
|  | int j,cnt=0; |
|  | char\* temp[50]; |
|  | char dirCopy[50]; |
|  | char temp2[50]; |
|  | char\* token=NULL; |
|  | //dummy char\* copy |
|  | strcpy(dirCopy,dir[i]); |
|  | //token is "/" |
|  | //cut the string as a "/" standard |
|  | token=strtok(dirCopy,"/"); |
|  | while(token != NULL){ |
|  | //save the cutting result to temp arr |
|  | temp[cnt++]=token; |
|  | token=strtok(NULL,"/"); |
|  | } |
|  |  |
|  | // ex. you command ./mkdir -p a/b/c |
|  | // result. a b c |
|  |  |
|  |  |
|  | for(j=0;j<cnt;j++){ |
|  | if(j==0){ |
|  | //make first dir ex. 'a' |
|  | if(mkdir(temp[0],0777)!=0) |
|  | perror("make error"); |
|  | strcpy(temp2,temp[0]); |
|  | } |
|  |  |
|  | // ex. you command ./mkdir -p a/b/c |
|  | // result 1. a |
|  | // result 2. a/b |
|  | // result 3. a/b/c |
|  |  |
|  |  |
|  | else{ |
|  | //attach '/' string |
|  | strcat(temp2,"/"); |
|  | strcat(temp2,temp[j]); |
|  | //make second,third... dir |
|  | // ex. a/b ex. a/b/c |
|  | if(mkdir(temp2,0777)!=0) |
|  | perror("make error"); |
|  | } |
|  |  |
|  |  |
|  | } |
|  | } |

* 1. **rm**
     1. **메뉴얼 페이지**

Name : **remove files or directories**

Synopsis : rm [OPTION]... FILE...

Description : This manual page documents the GNU version of rm. rm

removes each specified file. By default, it does not remove

directories.

**-f**

ignore nonexistent files, never prompt

**-i**

prompt before every removal

파일을 제거한다. 특정 파일 하나를 제거할 수 도있고, 여러 파일을 삭제할 수 도있다. 옵션을 추가해서 명령어를 입력하지 않았을경우, 삭제확인을 하지 않고 바로 삭제가 된다. 옵션에 따라서 삭제확인을 물어보고 삭제할 수 있다.

**-f**

삭제확인을 하지 않고 바로 삭제 가능하다.

**-i**

삭제확인을 하고 삭제할 수 있다.

* + 1. **테스트**
       1. 특정 파일하나를 삭제한다.

[p@hjlee ~] $ ls

a b c

[pi@hjlee ~] $ rm a

[p i@hjlee ~] $ ls

b c

**Result**



**Success**

* + - 1. 파일이 아닌 디렉토리는 삭제되지 않는다.

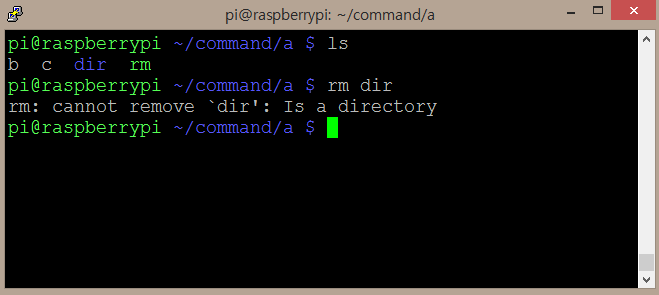
[p@hjlee ~] $ ls

b c dir

[pi@hjlee ~] $ rm dir

rm: cannot remove ‘dir’ : Is a directory

**Result**



**Success**

* + - 1. **–f** 옵션

삭제확인을 하지 않고 바로 삭제할 수 있다.

[p@hjlee ~] $ ls

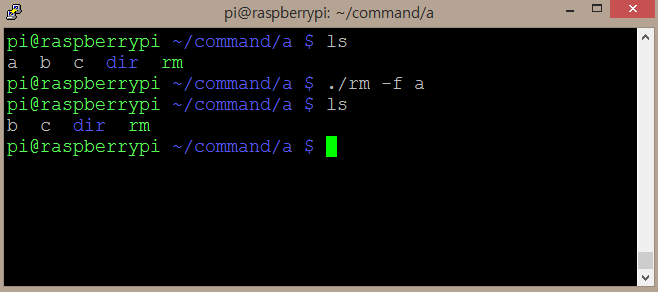
a b c

[pi@hjlee ~] $ rm –f a

[pi@hjlee ~] $ ls

b c

**Result**



**Success**

* + - 1. **–i** 옵션

삭제확인을 하고 삭제할 수 있다.

[p@hjlee ~] $ ls

a b c

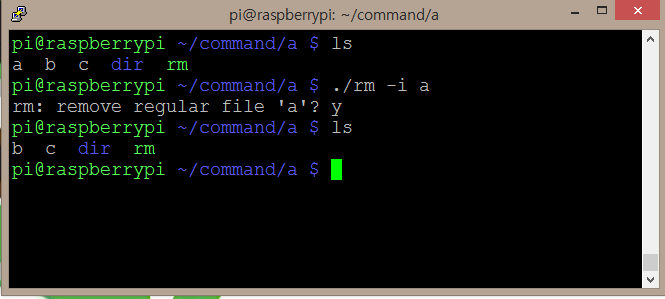
[pi@hjlee ~] $ rm –i a

rm: remove regular file ‘a’? y

[pi@hjlee ~] $ ls

b c

**Result**



**Success**

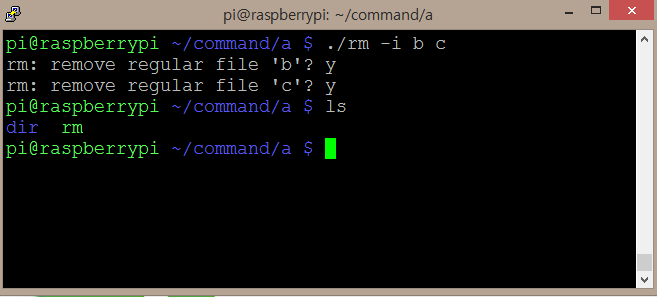
* + - 1. 여러개의 파일을 삭제할 수 있다.

**[pi@hjlee ~] $ rm -i b c**

**rm: remove regular file ‘b’? y**

**rm: remove regular file ‘c’? y**

**Result**



**Success**

* + 1. **사용된 시스템콜**

**open()**

Name : open

Synopsis : int open (const char \*FILENAME, int FLAGS[, mode\_t MODE]) FILENAME : 대상 파일 이름

Flags : 파일에 대한 옵션

Description : 파일을 사용하기 위해 연다.

retrun : 성공시 file descripter 값, 실패시 -1 반환.

mode :

O\_CREAT 해당파일이 없으면 생성.

O\_EXCL O\_CREAT를 사용시에 이미 파일이 있다면, 파일의 보존.

O\_TRUNC 기존 파일의 내용 삭제.

**remove()**

Name : remove

Synopsis : int remove(const char \*pathname);

Description : 파일 또는 디렉토리를 삭제한다.

return : 성공시 0, 실패시 -1 반환한다.

* + 1. **소스코드설명**
       1. option을 detect한다.

argv로 입력받은 파라미터 안에 옵션이 포함되어 있는지 검사한다.

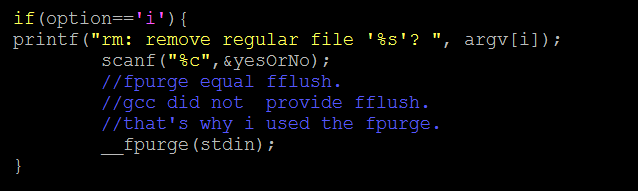
* + - 1. option의 유/무, -f / -i 분기를 나눈다.
      2. 해당 option에 맞는 flag를 대입해서 함수를 호출한다.
      3. 호출된 함수는 다음과 같은 작업을 한다.
         1. 우선, 옵션에 관계없이 제거하고자 하는 것이 파일인지 디렉토리인지 구별한다.

stat구조체를 사용하였으며, stat.st\_mode의 변수를 가져와서 구별하였다.

* + - * 1. i 옵션을 가졌을때.

삭제확인을 하는 문구를 띄운다. 물론 여러 파일을 한번에 삭제할 경우가 있기에, 반복문을 사용하였다. 사용자의 허락을 구하는 문구를 띄우고 scanf를 이용하여 y / n 를 사용자로부터 입력받는다. 한가지 특별한점은 scanf의 버퍼를 비우기 위해서 **\_\_fpurge 함수를** 사용하였다는 것이다.

gcc 에서는 fflush를 사용할 수 없기 때문에 대체해서 사용하였다. 이 함수는 **리눅스에서만 사용가능하다**. \_\_fpurge란 버퍼를 비우는 함수로서, #include <stdio\_ext.h>에 포함되어있다.



* + - * 1. f 옵션을 가졌을때.

사용자에게 삭제확인을 하지 않고 바로 제거를 하게 된다. 파라미터에 옵션이 포함되어있지 않아도 삭제확인을 하지 않는다.

* + 1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다.**

**보는데 참고하고 이해해 주길 바람.**

|  |
| --- |
| /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <unistd.h> |
|  | #include <fcntl.h> |
|  | #include <utime.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  | #include <errno.h> |
|  | #include <stdio\_ext.h> |
|  |  |
|  | #define BUFSIZE 512 |
|  |  |
|  | int rmProcess(int i, int argc, char \*argv[], int option); |
|  |  |
|  | typedef enum {false, true } bool; |
|  | bool isFile = true; |
|  | bool option\_f, option\_i =false; |
|  |  |
|  |  |
|  |  |
|  | int main(int argc, char \*argv[]) { |
|  | int i,j, fd; |
|  | struct stat stat\_buf; |
|  | char buffer[BUFSIZE]; |
|  |  |
|  | if (argc < 2) { |
|  | fprintf(stderr, "Usage: %s <files>...\n", argv[0]); |
|  | return 0; |
|  | } |
|  | //option detect |
|  | for(i=j=1; i<argc; i++) { |
|  | if(argv[i][0]=='-'){ |
|  | if(argv[i][1] == 'f') |
|  | option\_f=true; |
|  | else if(argv[i][1] == 'i') |
|  | option\_i=true; |
|  | } |
|  | } |
|  | // -i option |
|  | if(option\_i){ |
|  | for(i=1; i<argc; i++){ |
|  | if(argv[i][0] == '-') |
|  | continue; |
|  | rmProcess(i,argc,argv,'i'); |
|  | } |
|  | } |
|  |  |
|  | // "-f option" and "no option" |
|  | //because empty option equal -f option. |
|  | else if(option\_f){ |
|  | for(i=1; i<argc; i++){ |
|  | if(argv[i][0] == '-') |
|  | continue; |
|  | rmProcess(i,argc,argv,'f'); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | int rmProcess(int i, int argc, char \*argv[], int option ){ |
|  |  |
|  | struct stat stat\_buf; |
|  | char buffer[BUFSIZE]; |
|  | int yesOrNo='y'; |
|  |  |
|  | //distinguish file and directory |
|  | memset(&stat\_buf,0,sizeof(struct stat)); |
|  | stat(argv[i],&stat\_buf); |
|  | if(S\_ISDIR(stat\_buf.st\_mode)){ |
|  | isFile=false; |
|  | fprintf(stderr,"cannot remove '%s' : is a directory\n",argv[i]); |
|  | return 0; |
|  | } |
|  |  |
|  | if(option=='i'){ |
|  | printf("rm: remove regular file '%s'? ", argv[i]); |
|  | scanf("%c",&yesOrNo); |
|  | //fpurge equal fflush. |
|  | //gcc did not provide fflush. |
|  | //that's why i used the fpurge. |
|  | \_\_fpurge(stdin); |
|  | } |
|  |  |
|  | if(yesOrNo=='y' || yesOrNo == 'Y'){ |
|  | if(remove(argv[i])<0) |
|  | printf("file remove fail\n"); |
|  | } |
|  | else |
|  | printf("%s is not remove\n",argv[i]); |
|  | } |
|  |  |

* 1. **ln**
     1. **메뉴얼 페이지**

Name : **make links between files**

Synopsis : ln [OPTION]... [-T] TARGET LINK\_NAME

Description : In the 1st form, create a link to TARGET with the name LINK\_NAME. In the 2nd form, create a link to TARGET in the current directory. In the 3rd and 4th forms, create links to each TARGET in DIRECTORY. Create hard links by default, symbolic links with --symbolic. When creating hard links, each TARGET must exist. Symbolic links can hold arbitrary text; if later resolved, a relative link is interpreted in relation to its parent directory.option

**-s**

make symbolic links instead of hard links

파일의 링크를 만들어 냅니다. 디폴트옵션이었을때 하드링크를 만들수 있습니다.

-s옵션을 통해서 심볼릭 링크를 만들수 있습니다.

**하드링크**란 하드에 저장되어있는 어떤 자료에 이름을 여러개 부여 하는것이다.

ln test test\_hardlic 를 입력했을때, 원본파일(test)를 지워도 test\_hardlick는 남아있다.

i number를 알고 두곳의 디렉토리에서 이 넘버를 기록하는것이다.

**심볼릭링크**란 자신이link되어있는파일에 대한 경로를 수록, 포인터개념이다.

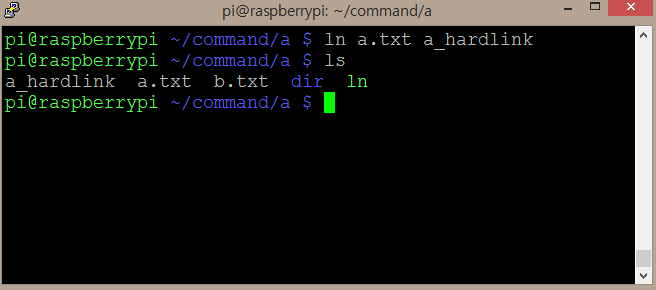
ln –s test test\_simbbolic 을 했을때, test\_simbolic 파일은 이름만 존재할 뿐 아무것도없고 단지 test파일을 가리키고 있을 뿐이다. test\_simbolic 파일을 호출하면 test파일로

연결을 해주는 역활을한다. 원본파일 (test)를 삭제시, test\_simbolic은 무용지물이 된다.

* + 1. **테스트**
       1. 하드링크 파일을 생성한다.

**[p@hjlee~] $ ln ./sourcefile ./hlinkfile**

**Result**

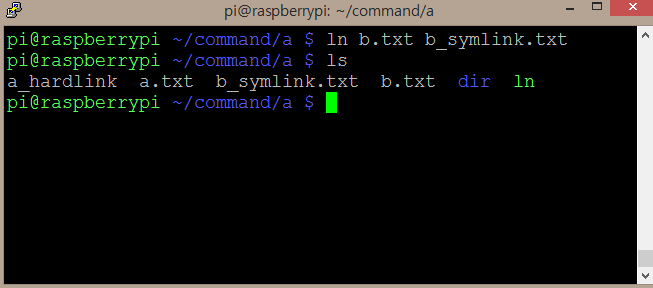


**Success**

* + - 1. 심볼릭 링크 파일을 생성한다.

**[p@hjlee~] $ ln -s ./sourcefile ./slinkfile**

**Result**



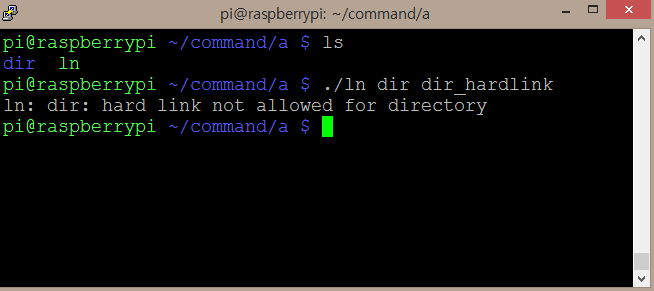
**Success**

* + - 1. 디렉토리는 하드링크가 불가능하다.

**[p@hjlee~] $ ln dir ./hlinkfile**

**ln: ‘dir’: hard link not allowed for directory**

**Result**



**Success**

* + 1. **사용된 시스템콜**

**symlink()**

Name : symlink

Synopsis : int symlink( const char \*oldpath, const char \*newpath)

Description : 심볼릭 링크를 생성한다.

return : 성공시 0, 실패시 -1 반환한다.

**link()**

Name : link

Synopsis : int link( const char \*oldpath, const char \*newpath)

Description : 하드 링크를 생성합니다. oldpath 와 newpath를 동일하게 사용 가능합니다. 즉, 하나의 파일에 여러 이름을 지정할 수 있다.

return : 성공시 0, 실패시 -1 반환한다.

* + 1. **소스코드설명**
       1. option을 detect한다.

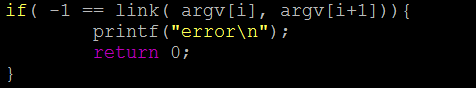
argv로 입력받은 파라미터 안에 옵션이 포함되어 있는지 검사한다.

* + - 1. option의 유/무 로서 분기를 나눈다.
      2. 해당 option에 맞는 flag를 대입해서 함수를 호출한다.
      3. 호출된 함수는 다음과 같은 작업을 한다.
         1. 디폴트 옵션이라면 hard link를 실행하게 된다.

실행하기 이전에 파라미터로 들어온 인자가 파일인지 디렉토리 인지 구별한다.

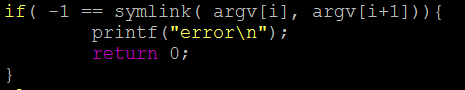
stat구조체를 사용하였으며 stat\_buf.mode를 이용해서 파일인지 디렉토리인지 구별한다. 그리고 **디렉토리는 하드링크를 할 수 없으므로 에러처리를 한다.**

hardlink 를 구현하기 위해서 다음에 나오는 **link함수를** 사용하였다.



* + - * 1. –s 옵션일 경우 심볼릭 링크를 진행한다.

symlink함수를 사용하였으며 다음과 같다.



* + 1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  | #include <errno.h> |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | struct stat stat\_buf; |
|  | int option\_s=0; |
|  | int i,j; |
|  | if(argc<2){ |
|  | printf("error\n"); |
|  | exit(1); |
|  | } |
|  | //option detect |
|  | for(i=j=1; i<argc; i++) { |
|  | if(argv[i][0]=='-'){ |
|  | if(argv[i][1] == 's') |
|  | option\_s=1; |
|  | } |
|  | } |
|  |  |
|  | //hard link |
|  | if(option\_s==0){ |
|  | for(i=1; i<argc; i++){ |
|  | //is directory ? is file? |
|  | //distinguish file and dir |
|  | stat(argv[i],&stat\_buf); |
|  | if(S\_ISDIR(stat\_buf.st\_mode)){ |
|  | fprintf(stderr,"ln: %s: hard link not allowed for directory\n",argv[i]); |
|  | return 0; |
|  | } |
|  | //make a hard link |
|  | if( -1 == link( argv[i], argv[i+1])){ |
|  | printf("error\n"); |
|  | return 0; |
|  | } |
|  | else |
|  | break; |
|  | } |
|  | } |
|  | //symlink |
|  | if(option\_s==1){ |
|  | for(i=1; i<argc; i++){ |
|  | if(argv[i][0] == '-') |
|  | continue; |
|  | //make a symlink |
|  | if( -1 == symlink( argv[i], argv[i+1])){ |
|  | printf("error\n"); |
|  | return 0; |
|  | } |
|  | else |
|  | break; |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |

* 1. **head/tail**
     1. **메뉴얼 페이지**

Name : **output the first part of files**

Synopsis : head [OPTION]... [FILE]...

Description : Print the first 10 lines of each FILE to standard output.

option

**-n**

print the first K lines instead of the first 10; with

the leading `-', print all but the last K lines ofeach file

Name : output the last part of files

Synopsis : tail [OPTION]... [FILE]...

Description : Print the last 10 lines of each FILE to standard output.

option

**-n**

output the last K lines, instead of the last 10; or

use -n +K to output lines starting with the Kth

**head**

텍스트로된 파일의 앞부분을 지정한 만큼 출력합니다. 디폴트 옵션을 주고 명령어를 입력한다면, 파일의 앞부분부터10줄까지 출력을 해 줍니다. –n 옵션을 통하여 정해진 행 까지 출력이 가능합니다.

**tail**

텍스트로된 파일의 뒷부분을 지정한 만큼 출력합니다. 디폴트 옵션을 주고 명령어를 입력한다면, 파일의 뒷부분부터10줄까지 출력을 해 줍니다. –n 옵션을 통하여 정해진 행 까지 출력이 가능합니다.

* + 1. **테스트 - head**
       1. 파일의 앞부분에서 부터 10행까지 보여준다.

[p@hjlee~] $ head [file]

file line 1

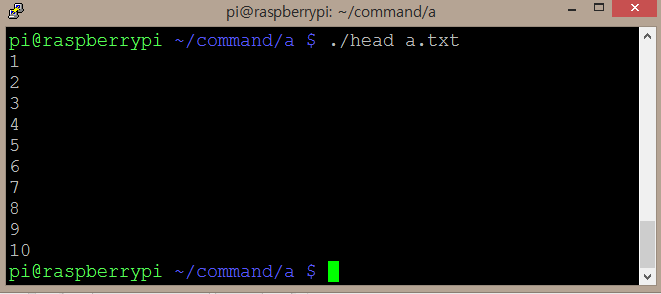
file line 2

.

.

file line 10

**Result**



**Success**

* + - 1. 여러개의 파일을 입력가능하다.

[p@hjlee~] $ head [file1] [file2]

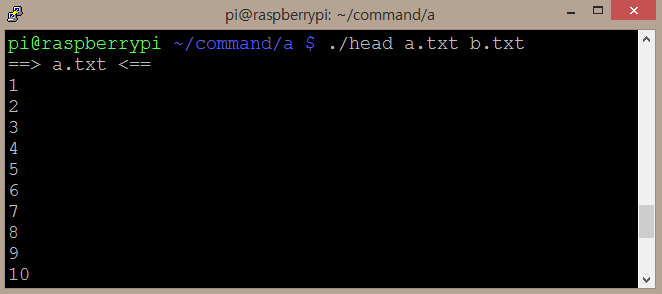
==> file1 <==

. . .

==> file2 <==

. . .

**Result**



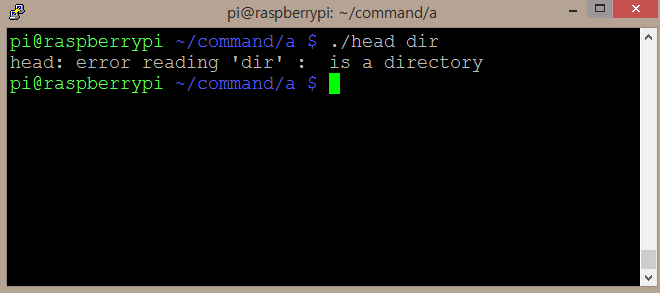
**Success**

* + - 1. 파라미터에 디렉토리를 대입할 수 없습니다.

[p@hjlee~] $ head dir

head: error reading ‘dir’ : IS a directory

[p@hjlee~] $



**success**

* + - 1. **–n** 옵션

파일의 앞부분에서 부터 지정된 행까지만 보여준다.

[p@hjlee~] $ head –n 2 [file]

file line 1

file line 2

[p@hjlee~] $



**success**

* + - 1. –n 옵션은 작성한 위치에 관계없이 적용된다.

[p@hjlee~] $ head [file] –n 3

* + 1. **테스트 – tail**
       1. 파일의 뒷부분에서 부터 10행까지 보여준다.

[p@hjlee~] $ tail [file]

file line 11

...

file line 20

**Result**



**Success**

* + - 1. 여러개의 파일을 입력가능하다.

[p@hjlee~] $ tail [file1] [file2]

==> file1 <==

. . .

==> file2 <==

. . .

**Result**



**Success**

* + - 1. 파라미터에 디렉토리를 대입할 수 없습니다.

[p@hjlee~] $ tail dir

tail: error reading ‘dir’ : IS a directory

[p@hjlee~] $



**success**

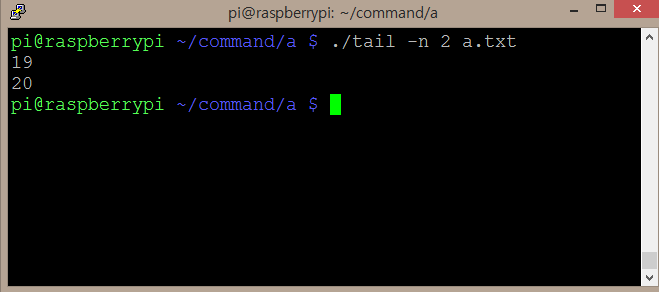
* + - 1. **–n** 옵션

파일의 앞부분에서 부터 지정된 행까지만 보여준다.

[p@hjlee~] $ tail –n 2 [file]

file line 19

file line 20



**success**

* + - 1. –n 옵션은 작성한 위치에 관계없이 적용된다.

[p@hjlee~] $ tail [file] –n 3 

* + 1. **사용된 시스템콜**

**open()**

Name : open

Synopsis : int open (const char \*FILENAME, int FLAGS[, mode\_t MODE]) FILENAME : 대상 파일 이름

Flags : 파일에 대한 옵션

MODE : O\_CREAT 옵션에 의해 파일이 생성될 때, 지정되는 접근권한.

Description : 파일을 사용하기 위해 열기 한다.

retrun : 성공시 file descripter 값, 실패시 -1 반환.

mode :

O\_CREAT 해당파일이 없으면 생성.

O\_EXCL O\_CREAT를 사용시에 이미 파일이 있다면, 파일의 보존.

O\_TRUNC 기존 파일의 내용 삭제.

* + 1. **소스코드설명**

head와 tail은 알고리즘 적으로 매우 유사하다. 그래서 필자는 head에 대해서 자세히 언급할 것이고, tail은 주요 부분만 다룰 예정이다.

**head**

* + - 1. option을 detect한다.

argv로 입력받은 파라미터 안에 옵션이 포함되어 있는지 검사한다.

* + - 1. option의 유/무 로서 분기를 나눈다.
      2. 해당 option에 맞는 flag를 대입해서 함수를 호출한다.
      3. 호출된 함수는 다음과 같은 작업을 한다.
         1. 실행하기 이전에 파라미터로 들어온 인자가 파일인지 디렉토리 인지 구별한다.

stat구조체를 사용하였으며 stat\_buf.mode를 이용해서 파일인지 디렉토리인지 구별한다. 그리고 **디렉토리는 head명령어를 사용할 수 없으므로, 에러처리를 해준다.**

* + - * 1. fopen을 통해서 해당 파일을 열고, 디폴트 옵션이라면 n을 10으로 설정한다. while roop을 진행하며너 차례차례 출력한다.
        2. tail의 경우 거꾸로 출력한다.
    1. **주석처리된 소스코드 – head**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  |  |
|  |  |
|  | #define buf 128 |
|  |  |
|  | void head(int n, int argc, char\* argv[]); |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | int option\_n=0; |
|  | int i,j,n; |
|  | //default nValue= 10 |
|  | int nValue=10; |
|  | char \*char\_nValue; |
|  | FILE \*fp; |
|  | char str[buf]; |
|  | if(argc<2){ |
|  | printf("error\n"); |
|  | exit(1); |
|  | } |
|  |  |
|  | //option detect |
|  | for(i=j=1; i<argc; i++) { |
|  | if(argv[i][0]=='-'){ |
|  | if(argv[i][1] == 'n'){ |
|  | if(argc<=3){ |
|  | printf("parameter error\n"); |
|  | exit(1); |
|  | } |
|  | option\_n=1; |
|  | //print the first nValue lines instead of the first 10 |
|  | char\_nValue=argv[i+1]; |
|  | nValue=atoi(char\_nValue); |
|  | } |
|  | else{ |
|  | printf("incorrect option\n"); |
|  | exit(1); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | head(nValue,argc,argv); |
|  | } |
|  | void head(int nValue, int argc, char\* argv[]){ |
|  | int i,n; |
|  | FILE\* fp; |
|  | char str[buf]; |
|  | struct stat stat\_buf; |
|  | n=nValue; |
|  | for(i=1; i<argc; i++){ |
|  |  |
|  | //dir detect |
|  | if(!(argv[i][0]=='-')){ |
|  | stat(argv[i],&stat\_buf); |
|  | if(S\_ISDIR(stat\_buf.st\_mode)){ |
|  | fprintf(stderr,"head: error reading '%s' : is a directory\n",argv[i]); |
|  | exit(1); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | //muti parameter print process |
|  | //no -n option |
|  | if(n==10){ |
|  | if(argc>2) |
|  | printf("==> %s <==\n", argv[i]); |
|  | } |
|  | // yes -n option |
|  | else{ |
|  | if(argv[i][0] == '-'){ |
|  | i++; |
|  | continue; |
|  | } |
|  | if(argc>4) |
|  | printf("==> %s <==\n", argv[i]); |
|  | } |
|  | //main algorithm |
|  | fp=fopen(argv[i],"r"); |
|  | if (fp == NULL) |
|  | fprintf(stderr, "Can't open file\n"); |
|  | while(n) |
|  | { |
|  | fgets(str,buf,fp); |
|  | printf("%s",str); |
|  | n--; |
|  | } |
|  | fclose(fp); |
|  | //recovery |
|  | n=nValue; |
|  | } |
|  | } |

* + 1. **주석처리된 소스코드 – tail**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  |  |
|  | #define buf 128 |
|  |  |
|  | void head(int n, int argc, char\* argv[]); |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | int option\_n=0; |
|  | int i,j,n; |
|  | //default nValue= 10 |
|  | int nValue=10; |
|  | char \*char\_nValue; |
|  | FILE \*fp; |
|  | char str[buf]; |
|  |  |
|  | if(argc<2){ |
|  | printf("error\n"); |
|  | exit(1); |
|  | } |
|  |  |
|  | //option detect |
|  | for(i=j=1; i<argc; i++) { |
|  | if(argv[i][0]=='-'){ |
|  | if(argv[i][1] == 'n'){ |
|  | if(argc<=3){ |
|  | printf("parameter error\n"); |
|  | exit(1); |
|  | } |
|  | option\_n=1; |
|  | //print the first nValue lines instead of the first 10 |
|  | char\_nValue=argv[i+1]; |
|  | nValue=atoi(char\_nValue); |
|  | } |
|  | else{ |
|  | printf("incorrect option\n"); |
|  | exit(1); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | head(nValue,argc,argv); |
|  | } |
|  | void head(int nValue, int argc, char\* argv[]){ |
|  | int i,n; |
|  | int lineCnt=0; |
|  | struct stat stat\_buf; |
|  | FILE\* fp; |
|  | char str[buf]; |
|  | n=nValue; |
|  | for(i=1; i<argc; i++){ |
|  | //dir detect |
|  | if(!(argv[i][0]=='-')){ |
|  | stat(argv[i],&stat\_buf); |
|  | if(S\_ISDIR(stat\_buf.st\_mode)){ |
|  | fprintf(stderr,"head: error reading '%s' : is a directory\n",argv[i]); |
|  | exit(1); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | //muti parameter print process |
|  | //no -n option |
|  | if(n==10){ |
|  | if(argc>2) |
|  | printf("==> %s <==\n", argv[i]); |
|  | } |
|  | // yes -n option |
|  | else{ |
|  | if(argv[i][0] == '-'){ |
|  | i++; |
|  | continue; |
|  | } |
|  | if(argc>4) |
|  | printf("==> %s <==\n", argv[i]); |
|  | } |
|  | //main algorithm |
|  | fp=fopen(argv[i],"r"); |
|  | if (fp == NULL) |
|  | fprintf(stderr, "Can't open file\n"); |
|  | while(fgets(str,buf,fp)){ |
|  | lineCnt++; |
|  | //printf("%s",str); |
|  | } |
|  | fclose(fp); |
|  | fp=fopen(argv[i],"r"); |
|  | while(fgets(str,buf,fp)){ |
|  | if(lineCnt--<=n) |
|  | printf("%s", str); |
|  | } |
|  |  |
|  | fclose(fp); |
|  | //recovery |
|  | n=nValue; |
|  | } |
|  | } |

* 1. **cat**
     1. **메뉴얼 페이지**

Name : concatenate files and print on **the standard output**

Synopsis : cat [OPTION]... [FILE]...

Description : Concatenate FILE(s), or standard input, to standard output.

option

cat은 파일의 내용을 보기 위한 명령어이다. 파일을 stdin으로 부터 받아서 stdout으로 출력해 준다. 커널이 제공하는 리다이렉션 기능 ( > < >> << ) 과 연동해서 많이 사용한다. 파라미터로 디렉토리는 허용하지 않는다.

* + 1. **테스트** 
       1. stdout 으로 출력한다.

[pi@hjlee ~] $ cat a.txt

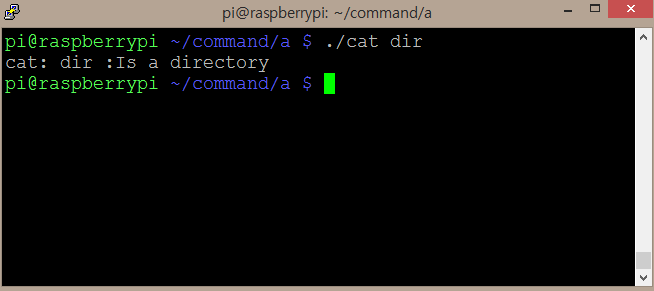


**Success**

* + - 1. 파라미터에 디렉토리를 넣을수 없다.

[pi@hjlee ~] $ cat dir

cat: dir: Is a directory



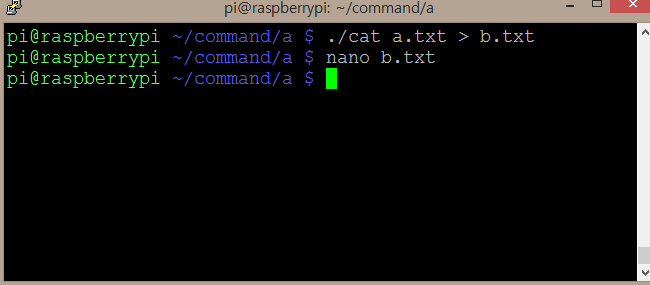
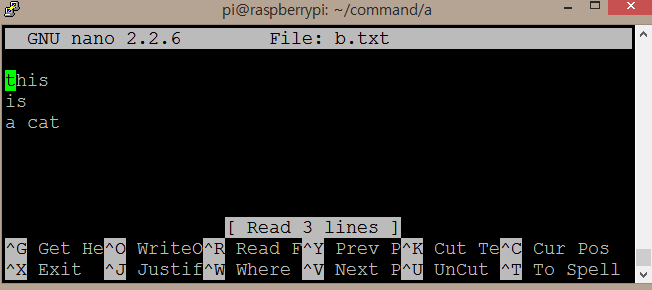
**success**

* + - 1. 리다이렉션이 가능하다.

[pi@hjlee ~] $ cat dir

cat: dir: Is a directory

**result**



**success**

* + 1. **사용된 시스템콜**

**open()**

Name : open

Synopsis : int open (const char \*FILENAME, int FLAGS[, mode\_t MODE]) ` FILENAME : 대상 파일 이름

Flags : 파일에 대한 옵션

MODE : O\_CREAT 옵션에 의해 파일이 생성될 때, 지정되는 접근권한.

Description : 파일을 사용하기 위해 연다.

retrun : 성공시 file descripter 값, 실패시 -1 반환 한다.

mode :

O\_CREAT 해당파일이 없으면 생성.

O\_EXCL O\_CREAT를 사용시에 이미 파일이 있다면, 파일의 보존.

O\_TRUNC 기존 파일의 내용 삭제.

**read()**

Name : read

Synopsis : ssize\_t read (int fd, void \*buf, size\_t nbytes)

Description : open() 함수로 열기를 한 파일의 내용을 읽는다.

return : 정상적으로 실행하였을 경우 읽어들인 바이트 수를 , 실패시 -1반환.

**write()**

Name : write

Synopsis : ssize\_t write (**int** fd, const **void** \*buf, size\_t n)

Description : open() 함수로 열기를 한 파일에 쓰기를 한다.

return : 정상적으로 쓰기를 했다면 쓰여진 바이트 수를 , 실패시 -1반환 한다.

* + 1. **소스코드설명**
       1. argv로 입력받은 스트링이 잘못 되어있다면 에러처리를 한다.
       2. 입력받은 파일명이 디렉토리일 경우 에러처리를 한다.
       3. open 시스템콜을 이용하여 파일을 열고, read 로 읽어서 write 한다.
    2. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <fcntl.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  | #include <errno.h> |
|  |  |
|  | #define BUFSIZE 512 |
|  |  |
|  | int main(int argc, char \*argv[]) { |
|  | int i, fd; |
|  | struct stat stat\_buf; |
|  | char buffer[BUFSIZE]; |
|  | size\_t nread; |
|  |  |
|  | if (argc < 2) { |
|  | fprintf(stderr, "Usage: %s <files>...\n", argv[0]); |
|  | return 0; |
|  | } |
|  |  |
|  | for (i=1; i<argc; i++) { |
|  | //dir detect |
|  | if(!(argv[i][0]=='-')){ |
|  | stat(argv[i],&stat\_buf); |
|  | if(S\_ISDIR(stat\_buf.st\_mode)){ |
|  | fprintf(stderr,"cat: %s :Is a directory\n",argv[i]); |
|  | exit(1); |
|  | } |
|  | } |
|  |  |
|  | if ((fd = open(argv[i], O\_RDWR ,0644)) < 0) { |
|  | fprintf(stderr, "%s: open error: %s\n", |
|  | argv[i], strerror(errno)); |
|  | continue; |
|  | } |
|  | } |
|  |  |
|  | while((nread=read(fd,buffer,BUFSIZE))>0){ |
|  | if(write(1,buffer,nread)<nread){ |
|  | close(fd); |
|  | return (-3); |
|  | } |
|  | } |
|  | } |
|  |  |

* 1. **touch**
     1. **메뉴얼 페이지**

Name : **change file timestamps** (access time , modification time)

Synopsis : touch [OPTION]... [FILE]...

Description : Update the access and modification times of each FILE to the current time.

option

touch는 파일의 날짜시간정보를 변경하는 명령어이다. 아무런 옵션없이 사용한다면 파일의 접근시간과 수정시간을 현재 시간으로 변경한다. 해당 파라미터의 파일이 존재하지 않는다면 파일의 크기가 0인 빈 파일을 생성한다. 디렉토리의 시간도 변경 가능하다.

* + 1. **테스트**
       1. 파일, 디렉토리의 최종접근 시간, 수정된시간 을 갱신한다.

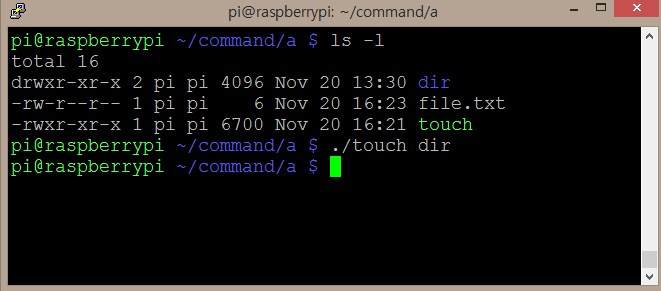
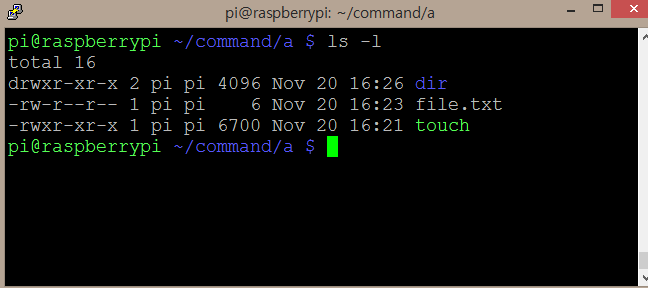
**[p@hjlee ~] $ ls –l**

-rw-r—r—1 pi pi 0 Nov 12 10:12 file

**[pi@hjlee ~] $ touch [file]**

**[p@hjlee ~] $ ls –l**

-rw-r—r—1 pi pi 0 Nov 19 20:39 file



**success**

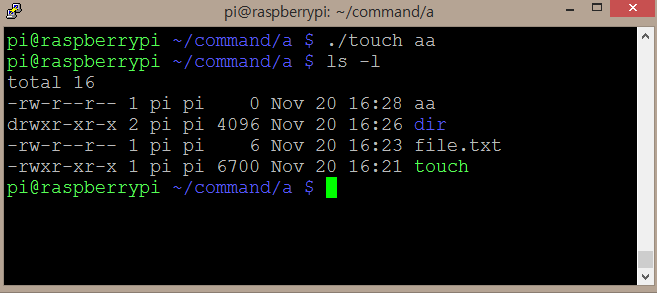
* + - 1. [file]이 존재하지 않는다면 , 현재 시간을 가진 파일을 생성한다.

**[pi@hjlee ~] $ touch [file]**

**[p@hjlee ~] $ ls –l**

-rw-r—r—1 pi pi 0 Nov 19 20:39 file

**result**



**success**

* + - 1. 여러개의 파일 및 디렉토리를 입력가능하다.

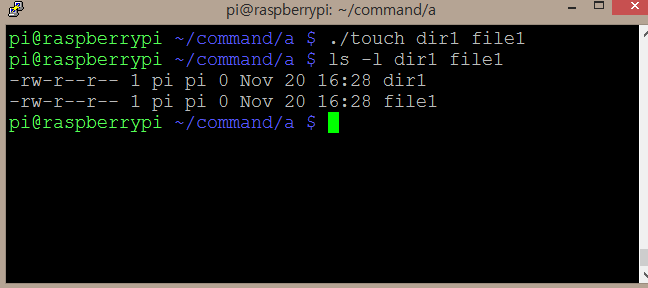
**[pi@hjlee ~] $ touch dir1 file1**

**[pi@hjlee ~] $ ls –l**

-rw-r—r—1 pi pi 0 Nov 19 20:39 file1

-rw-r—r—1 pi pi 0 Nov 19 20:39 dir1

**result**



**success**

* + 1. **사용된 시스템콜**

**open()**

Name : open

Synopsis : int open (const char \*FILENAME, int FLAGS[, mode\_t MODE]) FILENAME : 대상 파일 이름

Flags : 파일에 대한 옵션

MODE : O\_CREAT 옵션에 의해 파일이 생성될 때, 지정되는 접근권한.

Description : 파일을 사용하기 위해 열기 한다.

retrun : 성공시 file descripter 값, 실패시 -1 반환.

mode :

O\_CREAT 해당파일이 없으면 생성.

O\_EXCL O\_CREAT를 사용시에 이미 파일이 있다면, 파일의 보존.

O\_TRUNC 기존 파일의 내용 삭제.

**stat()**

Name : stat

Synopsis : int stat(const char \*file\_name, struct stat \* buf);

Description : 파일의 상태를 알아올수 있다. 첫번째 인자의 상태를 얻어와서 buf에채워넣는다. 성공시 stat 구조체에 파일의 정보를 복사한다.

return : 성공시0을, 실패시 -1반환한다.

**utime()**

Name : utime

Synopsis : int [utime](http://www.joinc.co.kr/modules/moniwiki/wiki.php/manSearch?google=none&name=utime)(const char \*filename, struct utimbuf \*buf);

Description : utime()는 filename으로 지정된 inode 의 접근 시간과 수정시간을buf의 actime 과 modtime 값으로 각각 변경한다.

return : 성공시0을, 실패시 -1반환.

* + 1. **소스코드설명**
       1. argv로 입력받은 스트링이 잘못 되어있다면 에러처리를 한다.
       2. 입력받은 파라미터의 종류(파일 / 디렉토리) 에 따라서 분기를 나눈다.
          1. 실행하기 이전에 파라미터로 들어온 인자가 파일인지 디렉토리 인지 구별한다.

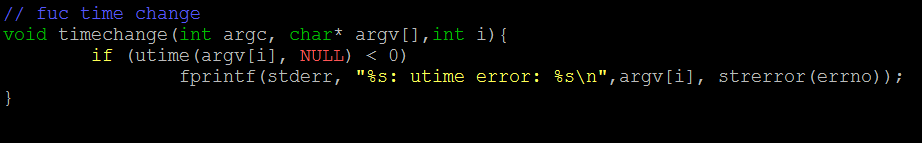
stat구조체를 사용하였으며 stat\_buf.mode를 이용해서 파일인지 디렉토리인지 구별한다.

* + - * 1. 만약 입력받은 파라미터가 디렉토리라면 timechange함수를 통해서 시간만 변경한다.
        2. 입력받은 파라미터가 파일이라면 파일의 존재 유무를 검사한다.

이때, **open함수의 O\_CREAT옵션을** 사용한다. 이 옵션을 사용해서, 파일이 존재하지 않았을때 길이가 0인 파일을 새로 생성시킨다.

그 후 timechange함수를 통해서 시간을 변경해준다.

* + - 1. 다음은 timechange함수이다. **utime 함수를** 통해서 변경해준다.



* + - 1. 나머지 에러처리를 해준다.
    1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <unistd.h> |
|  | #include <fcntl.h> |
|  | #include <utime.h> |
|  | #include <sys/stat.h> |
|  | #include <sys/types.h> |
|  | #include <errno.h> |
|  |  |
|  | void timechange(int argc, char \*argv[], int i); |
|  |  |
|  | int main(int argc, char \*argv[]) { |
|  | int i, fd; |
|  | struct stat stat\_buf; |
|  | struct utimbuf timebuf; |
|  |  |
|  | if (argc < 2) { |
|  | fprintf(stderr, "Usage: %s <files>...\n", argv[0]); |
|  | return 0; |
|  | } |
|  |  |
|  |  |
|  | for (i=1; i<argc; i++) { |
|  | //distingush directory and file |
|  | stat(argv[i],&stat\_buf); |
|  |  |
|  | //directory |
|  | if(S\_ISDIR(stat\_buf.st\_mode)) |
|  | timechange(argc,argv,i); |
|  |  |
|  | //file |
|  | else{ |
|  | //I used to "O\_CREAT" , so if is not there file, make it ! |
|  | if ((fd = open(argv[i], O\_RDWR | O\_CREAT ,0644)) < 0) { |
|  | fprintf(stderr, "%s: open error: %s\n",argv[i], strerror(errno)); |
|  | continue; |
|  | } |
|  | timechange(argc,argv,i); |
|  | } |
|  | } |
|  | return 0; |
|  | } |
|  |  |
|  | // fuc time change |
|  | void timechange(int argc, char\* argv[],int i){ |
|  | if (utime(argv[i], NULL) < 0) |
|  | fprintf(stderr, "%s: utime error: %s\n",argv[i], strerror(errno)); |
|  | } |

* 1. **chmod**
     1. **메뉴얼 페이지**

Name : **change file mode** bits

Synopsis : chmod [OPTION]... MODE[,MODE]... FILE...

Description : chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

option

리눅스 시스템의 파일 및 디렉토리에 대한 권한을 관리합니다.

r: 읽기권한(4)

w:쓰기권한(2)

x:실행권한(1)

-10자리의 의미를 파악하기 위해서 4부분으로 나누어서 확인합니다.

-[디렉토리 1자리][user권한 3자리][group권한 3자리][others권한 3자리]

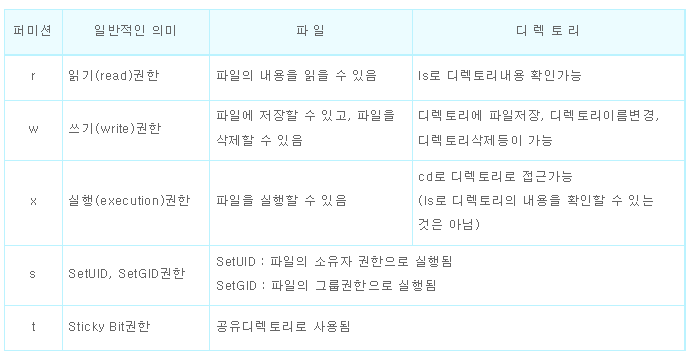
**추가설명**

**a. 퍼미션의 예**

-rwxrwxrwx    1    root    root    345   1월  22  13:36   FILE

drwxrwxrwx    3    root    root    4096   1월  23  13:25   DIR

**b. 파일과 디렉토리의 퍼미션의 차이점**



* + 1. **테스트**
       1. 파일, 디렉토리의 권한을 변경한다.

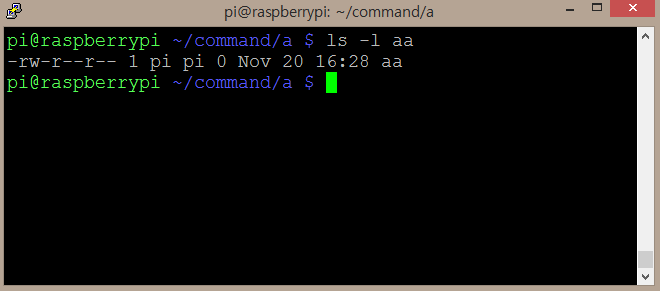
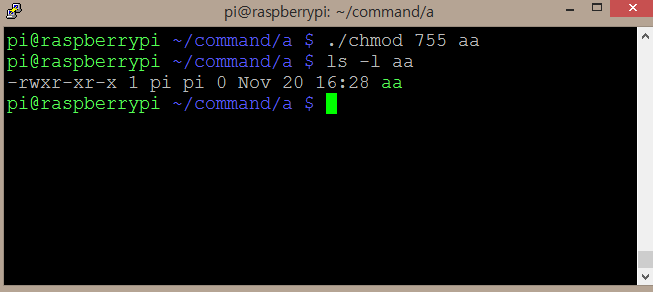
**[pi@hjlee ~] $ ls -l aa**

-rw-r--r-- 2 pi pi 4096 May 7 2015 aa

**[pi@hjlee ~] $ chomd 755 aa**

**[pi@hjlee ~] $ ls -l aa**

-rwxr-xr-x 2 pi pi 4096 May 7 2015 aa



**success**

* + 1. **사용된 시스템콜**

**chmod()**

Name : change mode

Synopsis : int chmod (**const** char \*file, mode\_t mode)

Description : 파일의 접근권한을 변경한다.

return : 성공시 0 , 실패시 -1반환.

* + 1. **소스코드설명**
       1. argv로 입력받은 스트링이 잘못 되어있다면 에러처리를 한다.
       2. 권한을 바꾸어줄 숫자를 입력받는데, 8진수로 변환해서 chmod함수에 넣어준다.



* + 1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <sys/stat.h> |
|  |  |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | int i,perm; |
|  | char \*ptr; |
|  | if(argc<3){ |
|  | printf("error\n"); |
|  | exit(1); |
|  | } |
|  |  |
|  | for(i=2; i<argc; i++){ |
|  | //change string to int. |
|  | //perm is octal number. |
|  | perm = strtol(argv[1],&ptr,8); |
|  | if ( -1 == chmod( argv[i], perm)) |
|  | printf("error\n"); |
|  | } |
|  |  |
|  | } |

* 1. **pwd**
     1. **메뉴얼 페이지**

Name : **print name of current/working directory**

Synopsis : pwd [OPTION]...

Description : Print the full filename of the current working directory.

option

현재 작업중인 폴더의 위치를 확인한다.

* + 1. **테스트**
       1. 현재 작업 폴더의 위치를 보여준다.

**[pi@hjlee menu] $ pwd**

**/etc/menu**

**result**



**success**

* + 1. **사용된 시스템콜**

**getcwd()**

Name : get current working directory

Synopsis : char \*getcwd(char \*buf, size\_t size);

buf : 현재 디렉토리의 경로가 저장 될 배열

size : 현재 디렉토리의 경로 문자열 크기

Description : 워킹 디렉토리를 바꾸는 작업을 합니다.

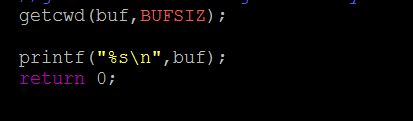
retirn : 성공시 현재 작업 디렉토리를 반환, 실패시 -1반환한다.

* + 1. **소스코드설명**
       - 1. getcwd함수를 사용한다.

getcwd란?

현재 작업 디렉토리의 이름을 구합니다.

성공하면 현재 작업디렉토리를 반환합니다.



* + - * 1. 에러처리를 해준다.
    1. **주석처리된 소스코드**

**copy&paste를 하였더니 들여쓰기 및 띄어쓰기가 제거가 된다. 보는데 참고하고 이해해 주길 바람.**

|  |  |
| --- | --- |
|  | /\* |
|  | Advanced System Programming |
|  | Linux / Unix |
|  | My\_command set |
|  |  |
|  | Kookmin UNIV. seoul, South Korea. |
|  | 20113315 이형준, hyungjun lee |
|  | hjlee1765@gmail.com |
|  | \*/ |
|  |  |
|  |  |
|  | #include <stdio.h> |
|  | #include <unistd.h> |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | char buf[BUFSIZ]; |
|  | //get current working directory |
|  | getcwd(buf,BUFSIZ); |
|  |  |
|  | printf("%s\n",buf); |
|  | return 0; |
|  |  |
|  | } |