Advanced System Programming

My_command set

설계 계획서

Kookmin Univ.
Computer Science Dept
20113315 이형준

1. 프로젝트 개요

A. 목표

명령어의 매뉴얼 페이지를 참고하여 자신이 구현할 명령어 및 그 옵션을 선택하여 정의 하시오.

기본 명령어의 60% 이상

주어진 옵션 모두 포함

매뉴얼 참조하여 옵션 추가 가능

구현 언어: C (50% 이상) 또는 기타 스크립트 언어 (shell, awk, perl, python 등)

홈디렉토리에 bin 디렉토리를 만들어 구현된 명령어의 실행 화일을 모은다. (PATH 에 ~/bin 을 추가하면 다른 명령처럼 실행 가능함)

B. 기본 명령어

ls [-l, -a, -R]

od [-a]

cat

touch

head/tail [-n]

chmod

cd

pwd

cp [-R, -f]

mv [-f]

rm [-f, -i]

ln [-s]

mkdir [-p]

<< Challenging >>

- who / w / finger
- find
- minishell - 자신의 쉘을 구현하여 그 쉘에서 위 명령어들이 실행되도록 한다.

C. 설계 레포트 내용

- 선택한 명령어 및 옵션에 대한 사전 조사 내용 (매뉴얼 페이지, 실행 예제 등)
- 구현하려는 명령어 및 옵션의 테스트 계획 (명령어를 옵션별로 실행시, 출력 예상 결과 등)
- 각 명령어 구현에 필요할 것으로 예상되는 시스템콜 및 이의 활용에 대한 사전 조사
- 기타

2. 명령어 및 옵션 사전 조사

A. ls

Name : **list directory contents**

Synopsis : ls [OPTION]... [FILE]...

Description : List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

option

-l

use a long listing format

-a

do not ignore entries starting with .

-R

list subdirectories recursively

B. cat

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

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

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

option

C. touch

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

D. head / tail

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 of each 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

E. chmod

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

F. cd

Name : **change the shell working directory**
Synopsis : cd [OPTION] [dir]
Description : Change the current directory to DIR. The default DIR is the value of the HOME shell variable.
option

G. pwd

Name : **print name of current/working directory**
Synopsis : pwd [OPTION]...
Description : Print the full filename of the current working directory.
option

H. rm

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

I. ln

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

J. mkdir

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

3. 명령어 및 옵션의 테스트

A. ls [-l, -a, -R]

i. ls [file]

$ ls

현재 디렉토리의 내용을 출력한다.

$ ls [directory]

상대경로나 절대경로로 작성된 모든 디렉토리에 대한 리스트를 출력해 준다
[directory] 로 들어가서 ls 명령어를 입력한 것과 결과가 같다.

$ ls [file]

파일의 절대경로와 상대경로를 입력할 수 있다.
file 이 존재한다면, [file] 에 작성한 글자 그대로 출력한다.

$ ls –a

. **..** cache Document lab7 python_game Scratch

$ ls –l

[permission] [user] [byte] [date] [Filename]
drwxr-xr-x 3 pi pi 4096 May 6 2015 Documents
drwxr-xr-x 2 pi pi 4096 May 7 2015 lab7
drwxr-xr-x 2 pi pi 4096 May 7 2015 lab8
drwxrwxr-x 2 pi pi 4096 Jan 27 2015 python_games
drwxr-xr-x 2 pi pi 4096 May 6 2015 Scratch

$ ls –R

하위 디렉토리가 있다면, 그 디렉토리의 ls 도 출력한다.
(Recursively)
./Documents:
Scratch Projects

./lab7:
a client.c gclient.c megq.h

./lab7/a:

./python_games:
4row_arrow.png gem4.png pentomino.py
4row_black.png gem5.png pinkgirl.png
4row_board.png gem6.png Plain_Block.png
4row_computerwinner.png gem7.pn

B. cat

i. stdout 으로 출력한다.

[pi@hjlee ~] $ cat a.txt

this is a.txt

ii. 파라미터에 상대경로와 절대경로를 포함시킬 수 있다.

[pi@hjlee ~] $ cat /home/pi/a.txt

this is a.txt

C. touch

i. [file]이 존재한다면 파일에 최종접근 시간을 갱신한다.

[pi@hjlee ~] $ touch [file]

[p@hjlee ~] $ ls –l

-rw-r—r—1 pi pi 0 Nov 19 20:39 file

ii. [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

iii. 파라미터에 상대경로와 절대경로를 포함시킬 수 있다.

[pi@hjlee ~] $ ls –l /home/pi/a

-rw-r—r—1 pi pi 0 Nov 19 13:32 a

[pi@hjlee ~] $ touch /home/pi/a

[pi@hjlee ~] $ ls –l /home/pi/a

-rw-r—r—1 pi pi 0 Nov 19 20:39 a

D. cd

i. 자신의 홈디렉토리(~)로 이동한다

[pi@hjlee etc] $ cd (or cd ~)

[pi@hjlee ~] $

ii. 루트 디렉토리(/)로 이동한다.

[pi@hjlee ~] $ cd /

[pi@hjlee /] $

iii. 절대경로(항상 root 부터 시작)를 지정해서 특정 장소로 이동한다.

[pi@hjlee /] $ cd /etc/menu

[pi@hjlee menu] $

iv. 상대경로를 지정하여 이동한다.

현재위치에서부터 해당 상 디렉토리로 이동한다.

[pi@hjlee etc] $ cd menu/submenu

[pi@hjlee submenu] $

현재위치에 있는 menu 디렉토리로 이동한다.

[pi@hjlee etc] $ cd ./menu

[pi@hjlee menu] $

현재위치에서 상위 디렉토리에 있는 opt 로 이동한다.

[pi@hjlee menu] $ cd ../opt

[pi@hjlee opt] $

E. chmod

i. 파일, 디렉토리의 권한을 변경한다.

[pi@hjlee ~] $ chomd 755 test.c

[pi@hjlee ~] $ ls -al

[permission] [user] [byte] [date] [Filename]

drwxr-xr-x 3 pi pi 4096 May 6 2015 Documents

-rwxr-xr-x 2 pi pi 4096 May 7 2015 test.c

ii. 여러개의 권한 변경도가능

F. pwd

i. 현재 작업 폴더의 위치를 보여준다.

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

/etc/menu

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

/

G. rm

i. 특정 파일하나를 삭제한다.

[p@hjlee ~] **$ ls**

a b c

[pi@hjlee ~] **$ rm a**

[pi@hjlee ~] **$ ls**

b c

ii. 파일이 아닌 디렉토리는 삭제되지 않는다.

[p@hjlee ~] **$ ls**

b c dir

[pi@hjlee ~] **$ rm dir**

rm: cannot remove 'dir' : Is a directory

iii. **–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

iv. **–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

v. 여러개의 파일을 삭제할 수 있다.

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

rm: remove regular file 'b'? y

rm: remove regular file 'c'? y

H. head

i. 파일의 앞부분에서 부터 10 행까지 보여준다.

[p@hjlee~] $ head [file]

file line 1

file line 2

.

.

file line 10

[p@hjlee~] $

ii. 여러개의 파일을 입력가능하다.

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

==> file1 <==

f1 line 1

f1 line 2

.

.

f1 line10

==> file2 <==

f2 line 1

f2 line 2

.

.

f2 line 10

[p@hjlee~] $

iii. 파라미터에 상대경로와 절대경로를 포함시킬 수 있습다.

[p@hjlee~] $ head /home/pi/[file]

f1 line 1

f1 line 2

.

.

f1 line 10

[p@hjlee~] $

iv. **–n** 옵션

파일의 앞부분에서 부터 지정된 행까지만 보여준다.

[p@hjlee~] $ head –n 3 [file]

file line 1

file line 2

file line 3

[p@hjlee~] $

cf) –n 옵션은 작성한 위치에 관계없이 적용된다.

[p@hjlee~] $ head [file] –n 3

결과는 위와 동일합니다

l. tail

i. 파일의 뒷부분에서 부터 10 행까지 보여준다.

[p@hjlee~] $ tail [file]

file line 11

file line 12

.

.

file line 20

[p@hjlee~] $

ii. 여러개의 파일을 입력가능하다.

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

==> file1 <==

f1 line 11

f1 line 12

.

.

f1 line 20

==> file2 <==

f2 line 11

f2 line 12

.

.

f2 line 20

[p@hjlee~] $

iii. 파라미터에 상대경로와 절대경로를 포함시킬 수 있다.

[p@hjlee~] $ tail /home/pi/[file]

f1 line 11

f1 line 12

.

.

f1 line 20

[p@hjlee~] $

iv. **–n** 옵션

파일의 앞부분에서 부터 지정된 행까지만 보여준다.

[p@hjlee~] $ tail –n 3 [file]

file line 18

file line 19

file line 20

[p@hjlee~] $

cf) –n 옵션은 작성한 위치에 관계없이 적용된다.

[p@hjlee~] $ tail [file] –n 3

결과는 위와 동일합니다

J. mkdir

i. 현재 디렉토리에서 디렉토리를 하나 생성한다.

[p@hjlee~] $ mkdir dir1

[p@hjlee~] $ ls

dir1 a b

ii. 여러개의 디렉토리 명을 입력한다.

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

[p@hjlee~] $ ls

dir4 dir3 dir2 dir1 a b

iii. 파라미터에 상대경로 /절대경로를 포함시킬 수 있다.

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

[p@hjlee~] $ ls

dir5 dir4 dir3 dir2 dir1 a b

(. .. 사용가능하다)

mkdir 은 경로를 지정해 줄 경우에 [directory]가 미리 존재하여야 한다.

iv. **–p** 옵션

경로가 존재하지 않을때, 디렉토리 구조를 한번에 생성한다.

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

[p@hjlee~] $ ls a/b

c

cf) –p 옵션은 작성한 위치에 관계없이 적용된다.

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

결과는 위와 동일하다.

K. ln

i. 하드링크 파일을 생성한다.

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

ii. 심볼릭 링크 파일을 생성한다.

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

iii. 디렉토리는 하드링크가 불가능하다.

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

ln: 'dir': hard link not allowed for directory

4. 주요 시스템콜 및 이의 활용

A. ls [-l, -a, -R]

각 명령어 구현에 필요할 것으로 예상되는 주요 시스템콜

opendir()

Name	: opendir
Synopsis	: DIR *opendir(const char *name);
Description	: 지정한 디렉토리를 열기를 합니다.
retirn	: 성공하면 디렉토리 구조체인 DIR 포인터를, 실패시 NULL 을 반환.

readdir()

Name	: readdir
Synopsis	: **struct** dirent *readdir(DIR *dir);
Description	: opendir()에 해당하는, 모든 파일과 디렉토리 정보를 가져온다.
return	: 파일이나 디렉토리 정보를 반환하고, 실패하면 NULL 을 반환한다.

closedir()

Name	: closedir
Synopsis	: **int** closedir(DIR *dir);
Description	: opendir()에 해당하는 디렉토리를 닫는다.
return	: 성공시 0, 실패하면 1 을 반환한다.

closedir()

Name	: closedir
Synopsis	: **int** closedir(DIR *dir);
Description	: opendir()에 해당하는 디렉토리를 닫는다.
return	: 성공시 0, 실패하면 1 을 반환한다.

fork()

Name	: closedir
Synopsis	: pid_t fork(void);
Description	: 현재 실행되는 프로세스에 대해 복사본 프로세스를 생성한다.
return	: 부모 = 자식 프로세스 PID, 자식 = 0 이 반환한다.

B. cat

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_CREAT 를 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 -1 반환.

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 -1 반환 한다.

C. head & tail

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_CREAT 를 사용시에 이미 파일이 있다면, 파일의 보존.

O_TRUNC 기존 파일의 내용 삭제.

D. chmod

chmod()

Name : change mode

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

Description : 파일의 접근권한을 변경한다.

return : 성공시 0 , 실패시 -1 반환.

E. touch

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_CREAT 를 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 buf 에 채워넣는다. 성공시 stat 구조체에 파일의 정보를 복사한다.

return : 성공시 0 을, 실패시 -1 반환한다.

utime()

Name : utime

Synopsis : int utime(const char *filename, struct utimbuf *buf);

Description : utime()는 filename 으로 지정된 inode 의 접근 시간과 수정시간을 buf 의 actime 과 modtime 값으로 각각 변경한다.

return : 성공시 0 을, 실패시 -1 반환.

F. cd

chdir()

Name : chdir

Synopsis : int chdir(const char *path);

path : 변경할 디렉토리의 경로

Description : 워킹 디렉토리를 바꾸는 작업을 한다.

return : 성공시 0, 실패시 -1 반환한다.

G. pwd

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 -1 반환한다.

H. rm

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.
mode :
O_CREAT 해당파일이 없으면 생성.
O_EXCL O_CREAT 를 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.

I. mkdir

mkdir()

Name : make directory
Synopsis : int mkdir(const char *pathname, mode_t mode);
pathname : 생성할 디렉토리 명
mode : permission 관련 / umask.
Description : pathname 이름을 가지는 디렉토리를 만들려고 시도한다.
return : 성공시 0, 실패시 -1 반환한다.

J. ln

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 newpath 를 동일하게 사용 가능합니다. 즉, 하나의 파일에 여러 이름을 지정할 수 있다.
return : 성공시 0, 실패시 -1 반환한다.

5. 기타

cd

문제점 : cd 는 쉘 내부에 구현되어 있다. 쉘에서 프로그램을 실행시키면 새 process(child)가 생기게 되는데, chdir() 함수를 통해서 바뀐 작업 디렉토리를 parent 에 반영이 불가능 하다고 한다. 좀 더 자료를 찾아봐서 해결책을 마련해 보고자 한다.