고급 시스템 프로그래밍
컴퓨터공학부 20113316
임 병 준

Linux / Unix My_command set
설계 계획서

1. 개요

Linux / Unix 의 명령어들 중 주어진 '기본 명령어'의 60%이상을 시스템콜을 이용하여 같은 기능을 수행하도록하는 프로그램들을 직접 구현한다. 구현은 프로그래밍 언어는 C를 50%를 사용하도록 한다. 구현한 명령어들은 주어진 모든 옵션의 기능을 모두 포함해야한다.

선택한 명령어 및 옵션에 대한 사전 조사 내용, 구현하려는 명령어 및 옵션의 테스트 계획, 각 명령어 구현에 필요할 것으로 예상되는 주요 시스템콜 및 이의 활용에 대한 사전 조사를 기술한다.

* 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]

2. 명령어 목록

(1) ls [-l, -a, -R]

이 명령어는 타입이 디렉토리가 아닌 어떠한 파일을 매개변수로 받게되면 그 파일의 이름을 출력한다. 디렉토리가 매개변수로 입력되면 그 디렉토리 내부에 존재하는 숨겨지지 않은 모든 파일의 이름을 출력한다. 만약 매개변수가 전달되지 않으면 현재 디렉토리에 존재하는 숨겨지지 않은 모든 파일을 이름을 출력한다.

[옵션]

-l : 파일의 이름과 함께 접근권한, 소유자, 소유그룹, 생성된 날짜 및 시간 등의 정보를 추가적으로 출력한다. 만약 출력이 터미널로 연결되어있다면 파일들을 나열하기 이전에 나열될 모든 파일들의 크기를 모두 합친값을 출력한다.

-a : 디렉토리 내부의 모든 파일을 출력할때 숨겨진파일(파일명이 . 으로 시작하는)들도 모두 출력한다. 현재 디렉토리와 이전 디렉토리를 의미하는 . 과 .. 도 포함된다.

-R : 파일의 이름을 출력할때 해당 파일이 디렉토리라면 그 디렉토리 내부에 존재 파일들과 디렉토리의 내부 모든 파일들도 재귀적으로 출력한다.

[테스트 계획]

i) 옵션없이 명령어를 실행하였을 경우

```
imb:bin imb$ ls
test1 test2 test3
imb:bin imb$
```

ii) -l 옵션으로 명령어를 실행한 경우

```
imb:bin imb$ ls -l
total 0
-rw-r--r--  1 imb  staff  0 11 23 16:58 test1
-rw-r--r--  1 imb  staff  0 11 23 16:58 test2
-rw-r--r--  1 imb  staff  0 11 23 16:58 test3
imb:bin imb$
```

iii) -a 옵션으로 명령어를 실행한 경우

```
imb:bin imb$ ls -a
.      ..     test1 test2 test3
imb:bin imb$
```

iv) -l -a 옵션을 함께 사용한 경우

```
imb:bin imb$ ls -al
total 0
drwxr-xr-x   5 imb  staff   170 11 23 17:03 .
drwx------+ 55 imb  staff  1870 11 23 16:59 ..
-rw-r--r--   1 imb  staff     0 11 23 16:58 test1
-rw-r--r--   1 imb  staff     0 11 23 16:58 test2
-rw-r--r--   1 imb  staff     0 11 23 16:58 test3
imb:bin imb$
```

v) -l -a 옵션을 함께 사용하고 파일을 매개변수로 전달한 경우

```
imb:bin imb$ ls -al test1
-rw-r--r--  1 imb  staff  0 11 23 16:58 test1
imb:bin imb$
```

vi) -R 옵션으로 명령어를 실행한 경우

```
imb:bin imb$ ls -R
a     test1 test2 test3

./a:
b

./a/b:
c

./a/b/c:
imb:bin imb$
```

[주요 시스템콜]

opendir, readdir, closedir, write

(2) cat

이 명령어는 매개변수로 전달받은 파일의 내용을 읽어서 표준출력으로 출력한다. 만약 매개변수로 '-'문자가 전달되거나 파일명이 명시되지 않으면 표준입력에서 문자열을 읽어서 표준출력으로 출력한다. 모든 출력이 완료되면 줄바꿈 문자를 한번 출력한다.

[옵션]

이번 프로젝트에서 이 명령어에 대한 옵션은 구현하지 않는다.

[테스트 계획]

i) 매개변수로 주어지는 모든 파일을 순서대로 읽어서 출력한다. 이때, 각각의 출력 후에는 줄바꿈 문자를 출력한다.

```
imb:bin imb$ cat test1 test2 test3
It's test file 1.
It's test file 2.
It's test file 3.
imb:bin imb$
```

ii) 존재하지 않는 파일을 전달받았을 경우에는 에러 메세지를 출력한다.

```
imb:bin imb$ cat none
cat: none: No such file or directory
imb:bin imb$
```

iii) 매개변수로 디렉토리를 입력받은 경우에는 에러 메세지를 출력한다.

```
imb:bin imb$ cat dir/
cat: dir/: Is a directory
imb:bin imb$
```

iv) 매개변수로 장치파일을 입력받은 경우에는 에러 메세지를 출력한다.

```
imb:bin imb$ sudo cat /dev/xcpm
cat: /dev/xcpm: Operation not supported by device
imb:bin imb$
```

v) 매개변수로 주어지는 파일들중 존재하지 않는 파일이 섞여있는경우 각각 에러 메세지와 정상출력을 순서대로 수행한다.

```
imb:bin imb$ cat none dir test
cat: none: No such file or directory
cat: dir: Is a directory
It's test file.
imb:bin imb$
```

vi) 매개변수로 FIFO파일이 입력되면 해당 FIFO에서 읽어서 출력한다.

```
imb:bin imb$ echo 1 > fifo &
[1] 19737
imb:bin imb$ cat < fifo
1
[1]+  Done                    echo 1 > fifo
imb:bin imb$
```

vii) 존재하지 않는 옵션을 입력한 경우 에러 메세지와 사용예를 출력한다.

```
imb:bin imb$ cat -a
cat: illegal option -- a
usage: cat [-benstuv] [file ...]
imb:bin imb$
```

[주요 시스템콜]

open, read, write, stat

(3) touch

이 명령어는 매개변수로 전달되는 모든 경로에 내용이 없는 파일을 생성한다. 만약 이미 존재하는 파일 또는 디렉토리가 매개변수로 전달된다면 해당 파일 또는 디렉토리의 마지막으로 사용된 시간을 현재 시간으로 갱신한다. 여기서 갱신되는 시간은 Access time과 Modification time이다.

[옵션]

이번 프로젝트에서 이 명령어에 대한 옵션은 구현하지 않는다.

[테스트 계획]

i) 매개변수로 존재하지 않는 디렉토리를 입력받은 경우 에러 메세지를 출력.

```
imb:bin imb$ touch none/
touch: none/: No such file or directory
imb:bin imb$
```

ii) 존재하지 않는 파일명을 매개변수로 입력할 경우 내용이 빈 해당 파일명을 갖는 일반 파일을 생성한다.

```
imb:bin imb$ touch test
imb:bin imb$ cat test
imb:bin imb$
```

iii) 존재하는 파일 또는 디렉토리를 입력받았을 경우 마지막 접근 시간과 수정 시간을 현재 시간으로 갱신한다.

```
imb:bin imb$ stat test
16777220 1056346 -rw-r--r-- 1 imb staff 0 0 "Nov 23 19:04:03 2015" "Nov 23 19:04
:03 2015" "Nov 23 19:04:03 2015" "Nov 23 19:01:30 2015" 4096 0 0 test
imb:bin imb$ touch test
imb:bin imb$ stat test
16777220 1056346 -rw-r--r-- 1 imb staff 0 0 "Nov 23 19:05:08 2015" "Nov 23 19:05
:08 2015" "Nov 23 19:05:08 2015" "Nov 23 19:01:30 2015" 4096 0 0 test
imb:bin imb$
```

iv) 다수의 파일명을 입력받아서 한번에 여러 파일에 대하여 touch 명령어를 수행한다.

```
imb:touch imb$ touch 1 2 3
imb:touch imb$ ls
1 2 3
imb:touch imb$
```

[주요 시스템콜]

stat, open, close

(4) head/tail [-n]

head 명령어는 주어진 파일의 줄 수를 세어서 파일의 시작부분부터 지정된 줄 수 만큼 표준출력으로 출력하는 프로그램이다. 파일을 입력하지 않은경우 표준출력으로부터 입력을 받는다. 줄 수가 명시되지 않을 경우 기본설정으로 10줄을 출력한다.

tail 명령어는 파일의 가장 끝에서 부터 지정된 줄 수 만큼 거꾸로 거슬러 올라가면서 표준출력으로 출력한다. 나머지 설정은 head 명령어와 같다.

[옵션]

-n : 파일의 시작부분부터 몇줄을 출력할지 사용자로부터 명시받는다. -n 옵션 이후에 매개변수로 입력되는 숫자 만큼의 줄을 출력한다.

[테스트 계획]

i) 다수의 매개변수가 입력되면 입력받은 모든 파일의 내용을 지정된 줄 수 만큼 표준출력으로 출력한다. 이때, 마지막 출력을 제외한 각각의 출력사이에는 줄바꿈 문자가 추가적으로 출력된다.

```
imb:bin imb$ head 1 2 3
==> 1 <==
It's test file.

==> 2 <==
It's test file.

==> 3 <==
It's test file.
imb:bin imb$
```

ii) 존재하지 않는 파일에 대하여 경고문을 출력할 때는 각 출력사이에 줄바꿈 문자를 출력하지 않는다.

```
imb:bin imb$ head test none none
==> test <==
It's test file.
head: none: No such file or directory
head: none: No such file or directory
imb:bin imb$
```

iii) -n 옵션을 명시하지 않은 경우 기본값인 10줄을 출력한다.

```
imb:bin imb$ head test
line 0
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
imb:bin imb$
```

iv) -n 옵션을 명시하였을 경우 -n 옵션의 매개변수 만큼의 줄을 출력한다.

```
imb:bin imb$ head -n 5 test
line 0
line 1
line 2
line 3
line 4
imb:bin imb$
```

v) 옵션이 파일보다 뒤쪽에 입력되는 경우에 파일로 취급한다.

```
imb:bin imb$ head test -n 3
==> test <==
It's test file.
head: -n: No such file or directory
head: 3: No such file or directory
imb:bin imb$
```

vi) -n 옵션의 매개변수로 0보다 작은 정수 또는 정수가 아닌 값을 입력할 경우 에러 메세지를 출력한다.

```
imb:bin imb$ head -n 0 test
head: illegal line count -- 0
imb:bin imb$ head -n -1 test
head: illegal line count -- -1
imb:bin imb$ head -n 0.1 test
head: illegal line count -- 0.1
imb:bin imb$ head -n abc test
head: illegal line count -- abc
imb:bin imb$
```

vii) 정의되지 않은 옵션을 명시할 경우 에러메세지와 사용 예를 출력한다.

```
imb:bin imb$ head -m 10 test
head: illegal option -- m
usage: head [-n lines | -c bytes] [file ...]
imb:bin imb$
```

vii) 파일명이 입력되지 않으면 지정된 줄 수 만큼 입력받으면서 줄바꿈 문자을 입력받을때 출력한다.

```
imb:bin imb$ head -n 3
line1
line1
line2
line2
line3
line3
imb:bin imb$
```

[주요 시스템콜]

read, write

(5) chmod

이 명령어는 매개변수로 입력받은 파일의 접근권한을 변경한다.

[옵션]

이번 프로젝트에서 이 명령어에 대한 옵션은 구현하지 않는다.

[테스트 계획]

i) chmod 명령어를 적용 한 뒤 파일의 접근권한이 적절하게 변경된다.

```
imb:bin imb$ ls -l test
-rw-r--r--  1 imb  staff  0 11 23 19:05 test
imb:bin imb$ chmod 0755 test
imb:bin imb$ ls -l test
-rwxr-xr-x  1 imb  staff  0 11 23 19:05 test
imb:bin imb$
```

ii) chmod 명령어에 다수의 매개변수를 입력해서 접근권한을 한번에 변경할 수 있다.

```
imb:chmod imb$ ls -l
total 0
-rw-r--r--  1 imb  staff  0 11 23 22:03 1
-rw-r--r--  1 imb  staff  0 11 23 22:03 2
imb:chmod imb$ chmod 0400 1 2
imb:chmod imb$ ls -l
total 0
-r--------  1 imb  staff  0 11 23 22:03 1
-r--------  1 imb  staff  0 11 23 22:03 2
imb:chmod imb$
```

[주요 시스템콜]

chmod

(6) pwd

이 명령어는 현재 실행되고 있는 프로세스의 Current Working Directory (CWD)의 이름을 출력한다.

[옵션]

이번 프로젝트에서 이 명령어에 대한 옵션은 구현하지 않는다.

[테스트 계획]

i) 실행하였을때 현재 디렉토리를 출력한다.

```
imb:bin imb$ pwd
/Users/imb/Desktop/bin
imb:bin imb$
```

[주요 시스템콜]

getcwd, write

(7) rm [-f,-i]

이 명령어는 매개변수로 입력받은 디렉토리가 아닌 파일을 제거한다. 만약 파일에 쓰기권한이 허락되지 않고, 표준입력장치가 터미널일때 사용자에게 삭제 여부를 명령프롬프트를 통해 확인받는다.

[옵션]

-f : 만약 삭제할 파일이 접근권한(쓰기권한)이 허락되지 않아도 사용자에게 확인받지 않고 삭제작업을 수행한다. -f 옵션 이전에 명시된 -i 옵션은 무시된다.

-i : 파일의 접근권한에 관계없이 각각의 파일에 대한 삭제 여부를 사용자에게 명령프롬프트를 통해 확인받는다.

[테스트 케이스]

i) 연속적으로 삭제할 파일명을 입력하여 입력한 모든 파일을 지운다.

```
imb:rm imb$ ls
1 2 3
imb:rm imb$ rm 1 2 3
imb:rm imb$
```

ii) 쓰기권한이 없는 파일에 대하여 삭제여부를 확인 받는다.

```
imb:rm imb$ ls -l
total 0
-r--r--r--  1 imb  staff  0 11 23 21:52 1
-r--r--r--  1 imb  staff  0 11 23 21:52 2
imb:rm imb$ rm 1 2
override r--r--r--  imb/staff for 1? y
override r--r--r--  imb/staff for 2? y
imb:rm imb$
```

iii) -f 옵션을 사용할 경우 권한이 없는 파일에 대하여 삭제여부를 확인받지 않음.

```
imb:rm imb$ ls -l test
-r--r--r--  1 imb  staff  0 11 23 22:10 test
imb:rm imb$ rm -f test
imb:rm imb$
```

iv) -i 옵션을 사용할 경우 권한이 허용되는 파일에 대하여 삭제여부를 확인받음.

```
imb:rm imb$ ls -l test
-rw-r--r--  1 imb  staff  0 11 23 22:11 test
imb:rm imb$ rm -i test
remove test? y
```

[주요 시스템콜]

remove

(8) ln [-s]

이 명령어는 하드링크 또는 심볼릭 링크를 생성한다.

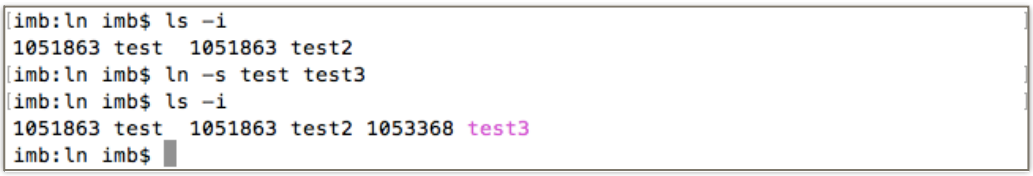
[옵션]

-s : 이 옵션은 ln명령어가 심볼릭 링크를 생성하도록 한다.

[테스트 케이스]

i) 하드링크를 생성한 경우 i-node number가 같고 내용도 같은 파일이 생성된다.

ii) -s 옵션으로심볼릭 링크를 생성한 경우 i-node number가 다르지만 같은 내용의 파일이 생성된다.



[주요 시스템콜]

link, symlink

(9) mkdir [-p]

이 명령어는 매개변수로 입력받은 경로에 해당 디렉토리를 생성한다.

[옵션]

-p : 입력받은 경로 상에 경로에 명시된 디렉토리가 존재하지 않으면 중간의 경로 상의 모든 디렉토리를 생성한다.

[테스트 케이스]

i) 기본 옵션으로 명령어를 실행할 경우 매개변수로 전달되는 모든 경로에 대하여 디렉토리를 생성한다.

```
imb:bin imb$ mkdir a b c
imb:bin imb$ ls
a       b       c
imb:bin imb$
```

ii) 이미 존재하는 파일 또는 디렉토리의 이름으로 디렉토리를 생성하려할 경우.

```
imb:bin imb$ mkdir test
imb:bin imb$ mkdir test
mkdir: test: File exists
imb:bin imb$
```

iii) -p 옵션을 명시하고 이미 존재하는 디렉토리를 생성하려할 경우에는 에러메세지를 출력하지 않음. (디렉토리 내부의 파일은 그대로 존재)

```
imb:bin imb$ mkdir test
imb:bin imb$ mkdir -p test
imb:bin imb$
```

iv) -p 옵션으로 디렉토리를 생성 한 경우.

```
imb:mkdir imb$ mkdir -p a/b/c
imb:mkdir imb$ tree
.
└── a
    └── b
        └── c

3 directories, 0 files
imb:mkdir imb$
```

v) 존재하지 않는 옵션을 사용한 경우 오류 메세지를 출력하면서 사용 예를 출력한다.

```
imb:bin imb$ mkdir -opt test
mkdir: illegal option -- o
usage: mkdir [-pv] [-m mode] directory ...
imb:bin imb$
```

[주요 시스템콜]

mkdir

3. 주요 시스템콜

1. open

```
#include <unistd.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
     returns file descriptor on Success, -1 on Error
```

flag : O_RDONLY | O_WRONLY | O_RDWR 가 대표적인 옵션으로 각각 읽기전용, 쓰기전용, 읽기 및 쓰기 옵션이다. 이 외에도 O_EXCL, O_TRUNC, O_CREAT 등의 옵션이 있다. O_CREAT 옵션을 사용할 때에는 접근권한 mode를 함께 명시하여야한다. 파일을 여는데 성공할 시 file descriptor를 리턴하고 파일을 여는데 실패할 시 -1값을 리턴한다.

2. read

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
     return number of bytes read on Success, -1 on Error
```

주어진 file descriptor로 부터 count bytes 만큼을 읽고 buf에 저장한다. 성공할시 실제로 read함수가 읽어들인 bytes를 리턴하고, 실패할시 -1을 리턴한다.

3. write

```
#include <unistd.h>

ssize_t write(int fd, void *buf, size_t count);
     return number of bytes written on Success, -1 on Error
```

주어진 file descriptor에 buf에 있는 정보를 count bytes 만큼 쓴다. 성공할시 실제로 write 함수가 쓴 bytes를 리턴하고, 실패할시 -1을 리턴한다.

4. close

```
#include <unistd.h>

int close(int fd);
     return 0 on Succes, -1 on Error
```

주어진 file descriptor를 사용하지 못하도록 닫는다. 성공할 시 0을 리턴, 실패할 시 -1을 리턴한다.

5. mkdir

```
#include <sys/stat.h>
#include <sys/types.h>

int mkdir(const char *pathname, mode_t mode);
     return 0 on Success, -1 on Error
```

주어진 경로 pathname에 디렉토리를 생성한다. 성공할시 0을 리턴하고, 실패할시 -1을 리턴한다.

5. stat

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *pathname, struct stat *buf);
     return 0 on Success, -1 on Error
```

주어진 경로 pathname에 해당하는 파일 또는 디렉토리의 정보를 stat구조체 포인터 buf에 저장한다.

다음의 매크로들을 이용해 파일의 타입을 알아낼 수 있다.

S_ISREG(m) : Regular file 인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISDIR(m) : 디렉토리인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISCHR(m) : Character device인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISBLK(m) : Block device 인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISFIFO(m) : FIFO 인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISLNK(m) : Symbolic link 인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

S_ISSOCK(m) : Socket 인 경우 0이 아닌 정수를 리턴, 아닌경우 0을 리턴한다.

6. opendir

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir(const char *name);
   return directory stream pointer on Success, NULL on Error
```

주어진 이름의 디렉토리를 오픈하여 성공할시 디렉토리 스트림 포인터를 리턴, 실패할시 NULL을 리턴한다.

7. readdir

```
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
     return struct dirent pointer on Success, NULL on Error

struct dirent {
     ino_t      d_ino;          /* inode number */
     off_t      d_off;          /* current pos in dir stream */
     unsigend short  d_reclen;  /* length of this record */
     unsigend char   d_type;    /* type of file */
     char       d_name[256];    /* file name */
}
```

성공할시 주어진 디렉토리 스트림으로 부터 읽어온 dirent 구조체를 리턴, 실패할시 NULL을 리턴한다.

8. closedir

```
#include <sys/types.h>
#include <dirent.h>

int closedir(DIR *dirp);
     return 0 on Success, -1 on Error
```

디렉토리 스트림을 더 이상 사용하지 못하도록 닫음. 성공할시 0을 리턴, 실패할시 -1을 리턴한다.

9. link / symlink

```
#include <unistd.h>

int link(const char *oldpath, const char *newpath);
     return 0 on Success, -1 on Error

int symlink(const char *oldpath, const char *newpath);
     return 0 on Success, -1 on Error
```

oldpath의 파일의 하드링크 / 심볼릭링크를 newpath에 생성한다. 성공할시 0을 리턴하고, 실패할시 -1을 리턴한다.

10. unlink

```
#include <unistd.h>

int unlink(const char *pathname);
     return 0 on Success, -1 on Error
```

파라미터로 전달받은 경로에 있는 파일을 삭제한다. 파일이 어떠한 프로세서에서 열려 있는 상태라면 해당 프로세서가 파일을 닫을때까지 파일은 삭제되지 않는다. 성공할시 0을 리턴하고 실패할시 -1을 리턴한다.

11. rmdir

```
#include <unistd.h>

int rmdir(const char *pathname);
     return 0 on Success, -1 on Error
```

파라미터로 전달받은 경로에 있는 디렉토리를 삭제한다. 이때, 디렉토리는 반드시 비어있는 상태여야 한다. 성공할시 0을 리턴, 실패할시 -1을 리턴한다.

12. remove

```
#include <stdio.h>

int remove(const char *pathname);
     return 0 on Success, -1 on Error
```

파일을 삭제하거나 디렉토리를 삭제한다. 성공할시 0을 리턴하고, 실패할시 -1을 리턴한다. file 에서의 unlink와 directory에서의 rmdir을 합친것이라 할 수 있다.