강의명: 유닉스시스템

실습 번호: 4

실습 제목: 파일 및 디렉토리(Files and Directories)

학생 이름: 김준호

학번: 201710860

1 파일 타입

1.1

수행할 프로그램의 인자는 main함수에서 arrgv[1] – argv[argc-1] 까지로 주어진다.

주어진 모범 수행 화면에서 아래와 같이 수행한다.

$ ./mytype /home/unix/test-directory/\*

위 명령에서 \*은 pathname expansions으로 실제 인자로 들어가는 결과는 아래와 같다.



/home/unix/test-directory 아래 있는 모든 파일 앞에 /home/unix/test-directory/ 가 붙어있는 각각의 string들이다.

따라서 수행되는 명령어는 다음과 같이 생각할 수 있다.

$ ./mytype path0 path1 path2 … pathk

그러므로 for문을 이용하여 각 path의 type과 그에 대한 path를 출력하는 mytype함수를 사용하여 프로그램을 작성할 수 있다.

작성한 main문을 살펴보면 아래와 같다.

Text

Description automatically generated

이제 mytype을 작성해야한다. lstat system call은 path name과 statue의 address를 인자로 주면 해당 path 에 있는 파일의 status를 인자로 주어진 address에 접근하여 저장한다.

만약 오류가 발생하면 -1을 return한다.

path는 const char \* type, status는 struct stat type이다.

lstat, struct stat은 모두 sys/stat.h에 정의돼있으므로 이를 우선 include해야한다.



lstat을 이용하여 struct stat에 status 정보를 받아오는데 struct stat의 구조를 살펴보아야한다. 그 구조는 아래와 같다.

Text, letter

Description automatically generated

우리가 원하는 파일의 type 정보를 가지고 있는 변수는 mode\_t st\_mode이다.

따라서 pathname이 주어졌을 때 그 file에 오류가 없다면 그 file의 type이 무엇인지 구하는 코드는 아래와 같다.

Text

Description automatically generated

만일 lstat을 호출하였을 때 오류가 발생할 수 있으므로 이에 대한 처리를 위해

perror, exit 함수를 사용하였고 이를 위해 stdio.h, errno.h와 stdlib.h를 include 하였다.

Text

Description automatically generated

위 코드에서 구한 mode를 사람이 알아볼 수 있도록 바꾸어 출력해야한다.

이때 각 파일들의 type은 하나의 char로 표현될 수 있는데 다음과 같다.

1. Regular file : ‘-‘

2. Directory file : ‘d’

3. Character file: ‘c’

4. Block special file: ‘b’

5. FIFO named pipe: ‘p’

6. Socket: ‘s’

7. symbolic link: ‘l’

각 mode가 위 type중 어떤 것을 나타내는지 알아내기 위해 sys.stat.h에서 macro를 지원해주고 이를 사용할 수 있다. 각 macro들은 아래와 같다.

1. S\_ISREG(mode)

2. S\_ISDIR(mode)

3. S\_ISCHR(mode)

4. S\_ISBLK(mode)

5. S\_ISFIFO(mode)

6. S\_ISSOCK(mode)

7. S\_ISLNK(mode)

각각의 macro가 위에 file type을 설명할 때 사용했던 번호와 매치되어 각각의 type인지 아닌지를 1, 0으로 나타낸다.

따라서 mode는 이미 구했고 이를 이용하여 if, else 문으로 각각의 case에 대하여 어떤 type인지 정해주면 된다.

구현한 코드는 아래와 같다.

Text

Description automatically generated

그리고 이들을 “%c %s\n” format으로 출력하는데 출력은 터미널, 즉 stdout으로 출력하면 된다.

이를 위해 fprintf 함수를 사용할 수 있고 이는 stdio.h 에 정의돼있다.

출력할 file descriptor는 stdout이다.

구현한 코드는 아래와 같다.

Text

Description automatically generated

1.2

Text

Description automatically generated

1.3

Text

Description automatically generated

2 파일 모드

2.1

실습 1과 같이 for문으로 각 file들의 mode를 출력해주면 된다.

여기서 mode는 실습1에서 선언한 mode 변수와는 다른 의미이다. permission을 의미한다.

permission은 전통적으로 8진수로 표현됐으며 3자리로 이루어져있다.

permission 앞의 1자리는 uid, gid를 set하거나 file 수행 종료후에도 memory에 남게하는 sticky bit을 나타낸다.

이들은 따라서 00644와 같이 이루어져있고 가장 앞의 0은 8진수를 의미하고 뒤에 네 자리 중 가장 뒤에 있는 644는 각각 순서대로 owner, group, other가 가지는 권한을 의미한다.

4는 read, 2는 write, 1은 execute를 의미하는데 각 permission은 이들의 합으로 구성된다.

예로 6은 read (=4) + write(=2), 4는 read 이다.

따라서 00644는 해당 파일을 소유하고 있는 user는 read, write를 할 수 있으며 owner는 아니지만 group에 속한 user는 read만 할 수 있으며, 그 외의 다른 other user도 그러하다.

오른쪽에서 두번째에 있는 0은 uid, gid를 set하거나 sticky bit을 의미한다.

lstat을 이용하여 실습 1과 같이 struct stat에 정보를 담아오게 되는데 이때도

mode\_t st\_mode

위 변수에 담겨오게 된다.

실습1과는 다르게 file들의 mode(permission)을 알려주는 macro가 존재하지 않는다. 따라서 user가 직접 그 값들을 추출해내야한다. 메뉴얼을 살펴보면 아래와 같은 masks가 존재한다.

Table

Description automatically generated

각각의 masks가 어떻게 정의돼있는지 sys/stat.h를 살펴보면 아래와 같다.

#define S\_IRUSR \_\_S\_IREAD

#define S\_IRGRP (S\_IRUSR >> 3)

#define S\_IROTH (S\_IRGRP >> 3)

#define S\_IWUSR \_\_S\_IWRITE

#define S\_IWGRP (S\_IWUSR >> 3)

#define S\_IWOTH. (S\_IWGRP >> 3)

#define S\_IXUSR \_\_S\_IEXEC

#define S\_IXGRP (S\_IXUSR >> 3)

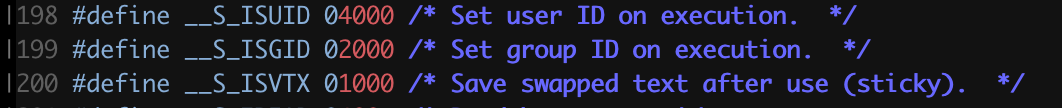
#define S\_IXOTH (S\_IXGRP >> 3)

\_\_S\_IREAD, \_\_S\_IWRITE, \_\_S\_IEXEC은 bits.stat.h에 정의돼 있고 아래와 같다.

Text

Description automatically generated with medium confidence

위 정보들을 종합해보면 st\_mode의 LSB로 부터 시작하는 9개의 bits가 permission을 나타내는 bits임을 알 수 있으며, uid, gid, sticky bit을 설정하는 bit들오 아래와 같이 정의돼있다.



정리해보면 앞의 3bit은 uid, gid, sticky bit, 뒤 9bit은 permission을 의미한다.

이들의 정보들만을 가져오려면 LSB로부터 시작하는 12개의 bit만을 읽어야하므로

아래와 같은 bitwise and operation으로 읽을 수 있다.



7은 3개의 bit이 모두 이진수로 1 1 1 로 set 돼있는 것을 의미하기 때문이다.

그리고 mode를 “%05o %s\n” format으로 출력하기만 하면된다.

이때도 역시 fprintf를 사용하기 위해 stdio.h 를 include하고, 출력한다.



Text

Description automatically generated with medium confidence

실습 1과 똑같은 논리로 main을 작성하였고 코드는 아래와 같다.

Text

Description automatically generated

2.2

Text

Description automatically generated

2.3

Graphical user interface, text, application

Description automatically generated

3 파일 uid/gid

3.1

파일의 uid, gid를 출력하기 위해서는 다시 struct stat의 변수들을 살펴보면 되는데 아래를 살펴보자

Text

Description automatically generated with medium confidence

stat의 일부이다. 여기에 st\_uid, st\_gid가 있는 것을 알 수 있고

mode = status.st\_mode

대신에

uid = status.st\_uid;

gid = status.st\_gid;

를 사용하여 format대로 출력만하면 실습 1, 2에서 사용한 논리를 그대로 사용할 수 있을 것 같다.

먼저 fprintf, perror를 위해 stdio.h 를 include,

perror를 위해 errno.h include

exit을 위해 stdlib.h

lstat, struct stat, mode\_t 를 위해 sys/stat.h 을 include한다.

Text

Description automatically generated

마찬가지로 main함수를 아래와 같이 작성한다.

Text

Description automatically generated

struct stat의 st\_uid, st\_gid의 return type이 각각 uid\_t, gid\_t이므로 이와 같이 선언을 한다.

Text

Description automatically generated with medium confidence

lstat syscall을 호출하여 status에 stat 정보를 받아온다. 오류가 발생하면 출력하고 종료한다.

Text

Description automatically generated

uid, gid 값을 각각 받아오고 이를 fprintf로 출력한다.

Text

Description automatically generated

3.2

Text

Description automatically generated

3.3

Graphical user interface, text

Description automatically generated with medium confidence

4 파일 크기

4.1

실습 1, 2, 3에서 한 바와 같이 struct stat을 살펴보면

Text, letter

Description automatically generated

off\_t st\_size가 file의 size를 bytes로 표현한 것을 나타내는 것을 알 수 있다.

따라서 아래와 같이 필요한 라이브러리를 먼저 include한다.

먼저 fprintf, perror를 위해 stdio.h 를 include,

perror를 위해 errno.h include

exit을 위해 stdlib.h

lstat, struct stat, mode\_t 를 위해 sys/stat.h 을 include한다.

Text

Description automatically generated

main은 실습 1, 2, 3과 같은 논리를 사용하여 아래와 같이 코드를 작성하였다.

Text

Description automatically generated

위 struct stat에서 살펴보았던 st\_size를 저장할 변수 size를 선언한다.

lstat으로 status를 받아오고 오류가 있으면 이를 출력하고 종료한다.

Text

Description automatically generated

size를 status로 부터 읽어 저장하고 이를 “%10ld %s\n” format으로 터미널에 출력한다.

코드는 아래와 같다.

Text

Description automatically generated with low confidence

4.2

Text

Description automatically generated

4.3

Text

Description automatically generated

5 파일 제거

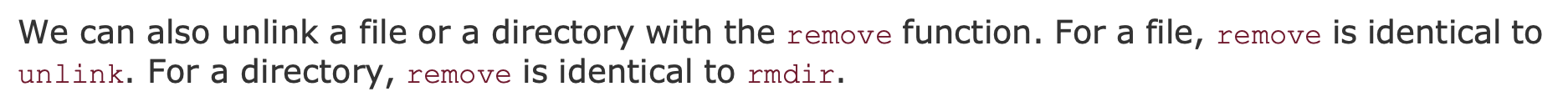
5.1

해당 실습은 실습 1, 2, 3, 4와는 조금 다른 유형의 실습인것 같다.

위에서는 lstat으로 원하는 값을 출력하는 것이면, 이번에는 lstat으로 해결하기 어려워보인다. 따라서 파일을 제거하는 syscall을 생각해보아야한다.

파일을 제거하는 syscall 함수는 unlink syscall이 있고 remove 라이브러리 함수가 있다.

메뉴얼에서는 파일을 제거할 때는 unlink, remove가 동일하다고 말한다.



해당 실습에서는 unlink syscall을 사용하도록 하겠다.

unlink syscall은 unistd.h에 정의돼있다.

unlink syscall은 성공 시 0, 실패 시 -1을 return한다. 따라서 실패했을 때 에러를 출력할 perror와 exit역시 사용해야하므로

unistd.h, stdio.h, errno.h, stdlib.h를 include한다.

Text

Description automatically generated

그리고 argv[1] 부터 argv[argc-1]까지 주어지는 파일들을 제거하는 것이므로 main문은 아래와 같이 작성하였다.

Text

Description automatically generated

myrm은 아래와 같이 작성하였다.

Text

Description automatically generated

여기서도 unlink syscall에 path를 인자로 넘겨주고 파일 삭제에 성공했는지, 실패했는지 결과를 return받아 오류가 있다면 출력하고 종료한다.

5.2

Text

Description automatically generated

5.3

Graphical user interface, text, application

Description automatically generated with medium confidence

6 파일 이름 변경

6.1

파일의 이름을 바꾸는 함수는 라이브러리 함수로 rename이 있다.

이 함수를 사용하면 쉽게 해결할 수 있을 것 같다.

rename함수의 prototype은 아래와 같다.

Text

Description automatically generated

oldpath에 있는 파일을 newpath로 이름을 바꾼다.

이 과정에서 path가 바뀌는 것이므로 실습 제목에서도 알 수 있듯이 리눅스 명령어 mv와 같은 기능을 한다.

에러가 발생했을 시 -1을 return하므로 이를 이용하여 에러가 나면 출력 후, 종료할 것이다.

rename은 stdio.h 에 정의돼있다. 에러 출력을 위해 perror, exit함수를 사용해야하므로

stdio.h, errno.h, stdlib.h를 include 한다.

Graphical user interface, text

Description automatically generated

실습 1, 2, 3, 4, 5와는 다르게 인자로 기존 파일, 이후에 이름이 바뀔 파일 2개 이므로

만약에 argc 가 3이 아니라면 인자가 부족하거나 너무 많은 것이므로 에러를 출력하고

인자는 실행 프로그램명인 argv[0]를 제외하고 2개가 되어야한다는 것을 stderr로 출력할 수 있다. 이를 코드로 작성하면 아래와 같다.

Chart, text

Description automatically generated

그리고 rename함수를 호출하고, 만약 에러가 발생하면 출력 후, 종료한다.

그렇지 않으면 그대로 종료한다.

Text

Description automatically generated

6.2

Text

Description automatically generated

6.3

Graphical user interface, text, application

Description automatically generated

끝.