**강의명: 유닉스 시스템**

**실습 번호: 4**

**실습 제목: 파일 및 디렉토리(Files and Directories)**

**학생 이름: 황귀훈**

**학번: 201710885**

**1. 파일 타입**

**1.1**

**#include <errno.h>**

**#include <sys/stat.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int mytype (const char \*path) {**

**struct stat status;**

**mode\_t mode;**

**char ftype;**

**if (lstat(path, &status) == -1) {**

**perror("lstat");**

**exit(1);**

**}**

**mode = status.st\_mode;**

**if (S\_ISREG(mode)) {**

**ftype='-';**

**}**

**else if(S\_ISDIR(mode)) {**

**ftype='d';**

**}**

**else if(S\_ISCHR(mode)) {**

**ftype='c';**

**}**

**else if(S\_ISBLK(mode)) {**

**ftype='b';**

**}**

**else if(S\_ISFIFO(mode)) {**

**ftype='p';**

**}**

**else if(S\_ISSOCK(mode)) {**

**ftype='s';**

**}**

**else if(S\_ISLNK(mode)) {**

**ftype='l';**

**}**

**else {**

**exit(1);**

**}**

**fprintf(stdout, "%c %s\n", ftype, path);**

**return 0;**

**}**

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

**{**

**for(int i=1; i<argc; i++)**

**mytype(argv[i]);**

**return 0;**

**}**

위 프로그램은 실습 1번 파일 타입을 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 타입과 파일의 이름을 형식 "%c %s\n"으로 출력하는 프로그램이다. 파일의 타입은 일반 파일, 디렉토 리 파일, 문자 특별 파일, 블록 특별 파일, fifo 파이프, 소켓, 심볼 링크가 있는데 이들은 각 각 문자 '-', 'd', 'c', 'b', 'p', 's', 'l'로 표현된다.

먼저 헤더 파일들을 include 해줘야 하는데 struct stat과 mode\_t를 사용하기 위해 **sys/stat.h**를 include했고 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror()함수를 사용하는데 이를 위해 **errno.h**를 include했다. 또한 exit(), fprintf()함수를 사용 하기위해 stdlib.h, stdio.h를 include하였다.

이 프로그램에서 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 타입을 알아내고 타입과 이름을 출력하기 위해 mytype()이라는 함수를 작성했고 main()함수는 mytype()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 하나씩 넘겨주도록 작성했다. 인수의 개수는 argc에 저장된다.

mytype()함수를 보면 struct stat 형식의 status를 정의하는데 여기서 status는 파일의 상태정보를 갖는 구조체로 다수의 변수가 존재한다. mode\_t type의 mode변수를 선언했고, 파일의 타입은 char 변수 ftype에 저장된다. 이 함수에서는 주어진 파일의 상태정보(type)를 알아내기 위해 system call 함수 lstat()을 부른다. lstat()의 첫 번째 인수는 path로 mytype()에서 받아온 path를 받는다. 두 번째 인수는 그 경로의 파일의 상태 정보를 담을 구조체의 주소이다. lstat()의 첫 번째 인수는 path로 mytype()에서 받아온 path를 받는다. 두 번째 인수는 그 path의 상태를 담을 구조체의 주소이다. 이 함수는 넘어온 인수(파일 경로)에 해당하는 파일의 타입과 파일의 이름을 형식 "%c %s\n"으로 출력한다.

시스템 호출의 경우 return value를 꼭 체크하고 문제가 있는 경우 에러 메시지를 출력해야 하기 때문에 if문을 사용하여 lstat()의 결과 값이 -1 이라면 perror()함수를 사용해 오류 메시지를 출력하는데 perror()함수에는 그 system call 함수의 이름(“lstat”)을 준다. 오류 메시지를 출력한 뒤 exit(1) 함수를 사용해서 프로그램을 종료한다. exit(1)은 오류로 프로그램을 종료함을 의미한다.

status 구조체에서 우리가 필요한 부분은 mode 부분이기 때문에 변수 mode에 status안에 있는 변수들 중 st\_mode 값을 저장했다. mode 에는 type 과 mode 두가지가 있는데 우리가 필요한 것은 type부분으로 이것은 매크로를 사용해서 얻을 수 있다. 여기서 파일의 타입은 Regular file(‘-’), Directory file(‘d’), Character special file(‘c’), Block special file(‘b’), FIFO named pipe(‘p’), Socket(‘s’), Symbolic link(‘l’)가 있는데 이들은 각각 매크로 S\_ISREG(), S\_ISDIR(), S\_ISCHR(), S\_ISBLK(), S\_ISFIFO(), S\_ISSOCK(), S\_ISLNK() 로 타입을 알아낼 수 있다. 각각의 매크로에 mode를 인수로 주었을 때 결과 값으로 true가 나오면 그 매크로에 해당하는 파일임을 의미한다.

따라서 if문을 사용하여 lstat()함수로 넘어온 파일이 어떤 타입인지 매크로와 mode를 사용해서 비교한 뒤 fprintf()함수를 사용하여 그 파일의 타입과 이름을 출력하도록 했다. 성공적인 return으로 return 0을 사용했다.

main()함수는 단순하게 mytype()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 for문을 사용하여 하나씩 넘겨주도록 작성했다. 성공적인 return으로 return 0을 사용했다.

**1.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**1.3**

****

**2. 파일 모드**

**2.1**

**#include <errno.h>**

**#include <sys/stat.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int mymode(const char \*path) {**

**struct stat status;**

**mode\_t mode;**

**if (lstat(path, &status) == -1) {**

**perror("lstat");**

**exit(1);**

**}**

**mode = status.st\_mode & 07777;**

**fprintf(stdout, "%05o %s\n", mode, path);**

**return 0;**

**}**

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

**{**

**int i;**

**for (i=1; i<argc; i++)**

**mymode(argv[i]);**

**return 0;**

**}**

위 프로그램은 실습 2번 파일 모드를 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 모드와 파일의 이름을 형식 "%05o %s\n"으로 출력하는 프로그램이다.

먼저 헤더 파일들을 include 해줘야 하는데 struct stat과 mode\_t를 사용하기 위해 **sys/stat.h**를 include했고 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror()함수를 사용하는데 이를 위해 **errno.h**를 include했다. 또한 exit(), fprintf()함수를 사용 하기위해 stdlib.h, stdio.h를 include하였다.

이 프로그램에서 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 모드를 알아내고 파일의 모드와 이름을 출력하기 위해 mymode()라는 함수를 작성했고 main()함수는 mymode()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 하나씩 넘겨주도록 작성했다. 인수의 개수는 argc에 저장된다.

mymode()함수를 보면 struct stat 형식의 status를 정의하는데 여기서 status는 파일의 상태정보를 갖는 구조체로 다수의 변수가 존재한다. mode\_t type의 mode변수를 선언했다. 이때 mode는 permission을 의미한다. 이 함수에서는 주어진 파일의 상태정보(mode)를 알아내기 위해 system call 함수 lstat()을 부른다. lstat()의 첫 번째 인수는 path로 mymode()에서 받아온 path를 받는다. 두 번째 인수는 그 경로의 파일의 상태 정보를 담을 구조체의 주소이다. 이 함수는 넘어온 인수(파일 경로)에 해당하는 파일의 모드와 파일의 이름을 형식 "%05o %s\n"으로 출력한다. 이때 앞의05는 다섯자리로 출력하는데 빈자리는 0으로 채우는 것을 의미하고 o는 8진수의 숫자를 출력함을 의미한다.

시스템 호출의 경우 return value를 꼭 체크하고 문제가 있는 경우 에러 메시지를 출력해야 하기 때문에 if문을 사용하여 lstat()의 결과 값이 -1 이라면 perror()함수를 사용해 오류 메시지를 출력하는데 perror()함수에는 그 system call 함수의 이름(“lstat”)을 준다. 오류 메시지를 출력한 뒤 exit(1) 함수를 사용해서 프로그램을 종료한다. exit(1)은 오류로 프로그램을 종료함을 의미한다.

status 구조체에서 우리가 필요한 부분은 mode 부분이기 때문에 변수 mode에 status안에 있는 변수들 중 st\_mode 값을 저장했다. mode 에는 type 과 mode 두가지가 있는데 우리가 필요한 것은 mode부분으로 struct stat의 st\_mode의 LSB와 왼쪽으로 세 개의 8진수가 있는데 이 네 8진수 값은 permission을 의미한다. 즉 우리는 LSB부터 시작해서 12bit가 필요하기 때문에 bitwise and 연산을 통해 얻어 내야한다. 3개의 bit가 모두1일 때 7이기 때문에 12bit의 값을 얻기 위해 &07777을 해주면 된다. 그리고 이 mode를 형식에 맞게 출력 해주기만 하면 된다. 출력을 하기 위해 fprintf()함수를 사용했고 형식은 "%05o %s\n"으로 mode와 파일 이름을 출력하는데 mode출력 부분에서 앞의05는 다섯자리로 출력하는데 빈자리는 0으로 채우는 것을 의미하고 o는 8진수의 숫자를 출력함을 의미한다. 성공적인 return으로 return 0을 사용했다.

main()함수는 단순하게 mymode()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 for문을 사용하여 하나씩 넘겨주도록 작성했다. 성공적인 return으로 return 0을 사용했다.

다음은 st\_mode의 mask와 설명 및 값을 나타낸 표이다.

테이블이(가) 표시된 사진

자동 생성된 설명

**2.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**2.3**

****

**3. 파일 uid/gid**

**3.1**

**#include <errno.h>**

**#include <sys/stat.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int myugid(const char \*path) {**

**struct stat status;**

**uid\_t uid;**

**gid\_t gid;**

**if (lstat(path, &status) == -1) {**

**perror("lstat");**

**exit(1);**

**}**

**uid = status.st\_uid;**

**gid = status.st\_gid;**

**fprintf(stdout, "%5d %5d %s\n", uid, gid, path);**

**return 0;**

**}**

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

**{**

**int i;**

**for (i=1; i<argc; i++)**

**myugid(argv[i]);**

**return 0;**

**}**

위 프로그램은 실습 3번 파일 uid/gid를 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 uid, gid 및 파일의 이름을 형식 "%5d %5d %s\n"으로 출력하는 프로그램이다.

먼저 헤더 파일들을 include 해줘야 하는데 struct stat과 mode\_t를 사용하기 위해 **sys/stat.h**를 include했고 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror()함수를 사용하는데 이를 위해 **errno.h**를 include했다. 또한 exit(), fprintf()함수를 사용 하기위해 stdlib.h, stdio.h를 include하였다.

이 프로그램에서 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 uid/gid를 알아내고 이름과 같이 출력하기 위해 myugid()라는 함수를 작성했고 main()함수는 myugid()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 하나씩 넘겨주도록 작성했다. 인수의 개수는 argc에 저장된다.

myugid()함수를 보면 struct stat 형식의 status를 정의하는데 여기서 status는 파일의 상태정보를 갖는 구조체로 다수의 변수가 존재한다. struct stat의 st\_uid, st\_gid의 return type인 uid\_t, gid\_t에 맞게 변수 uid, gid를 선언했다. uid, gid는 각각 status의 st\_uid, st\_gid에 저장되어 있기 때문에 위에서 선언한 변수 uid, gid에 status.st\_uid, status.st\_gid를 저장했다.

앞의 1, 2번과 마찬가지로 파일의 상태 정보를 알기 위해 system call 함수로 lstat()함수를 불렀는데 lstat()의 첫 번째 인수는 path로 myugid()에서 받아온 path를 받는다. 두 번째 인수는 그 경로의 파일의 상태 정보를 담을 구조체의 주소이다.

시스템 호출의 경우 return value를 꼭 체크하고 문제가 있는 경우 에러 메시지를 출력해야 하기 때문에 if문을 사용하여 lstat()의 결과 값이 -1 이라면 perror()함수를 사용해 오류 메시지를 출력하는데 perror()함수에는 그 system call 함수의 이름(“lstat”)을 준다. 오류 메시지를 출력한 뒤 exit(1) 함수를 사용해서 프로그램을 종료한다. exit(1)은 오류로 프로그램을 종료함을 의미한다.

status 구조체에서 우리가 필요한 부분은 uid, gid 부분으로 각각 status 구조체의 field인 st\_uid, st\_gid에 담겨있고 이것을 위에서 변수 uid, gid에 저장했다. myugid()함수에서 파일의 uid, gid 및 파일의 이름을 형식 "%5d %5d %s\n"으로 출력해야 하기 때문에 fprintf()함수를 사용했다. 이때 5d는 다섯자리의 10진수를 의미한다. 성공적인 return으로 return 0을 사용했다.

main()함수는 단순하게 myugid()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 for문을 사용하여 하나씩 넘겨주도록 작성했다. 성공적인 return으로 return 0을 사용했다.

**3.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**3.3**

****

**4. 파일 크기**

**4.1**

**#include <errno.h>**

**#include <sys/stat.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int mysize(const char \*path) {**

**struct stat status;**

**off\_t size;**

**if (lstat(path, &status) == -1) {**

**perror("lstat");**

**exit(1);**

**}**

**size = status.st\_size;**

**fprintf(stdout, "%10ld %s\n", size, path);**

**return 0;**

**}**

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

**{**

**int i;**

**for(i=1; i<argc; i++)**

**mysize(argv[i]);**

**return 0;**

**}**

위 프로그램은 실습 4번 파일 크기를 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 크기 및 파일의 이름을 형식 "%10ld %s\n"으로 출력하는 프로그램이다.

먼저 헤더 파일들을 include 해줘야 하는데 struct stat과 mode\_t를 사용하기 위해 **sys/stat.h**를 include했고 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror()함수를 사용하는데 이를 위해 **errno.h**를 include했다. 또한 exit(), fprintf()함수를 사용 하기위해 stdlib.h, stdio.h를 include하였다.

이 프로그램에서 argv[1] 부터 argv[argc-1]에 주어지는 파일들에 대하여 파일의 크기를 알아내고 이름과 같이 출력하기 위해 mysize()라는 함수를 작성했고 main()함수는 mysize()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 하나씩 넘겨주도록 작성했다. 인수의 개수는 argc에 저장된다.

mysize()함수를 보면 struct stat 형식의 status를 정의하는데 여기서 status는 파일의 상태정보를 갖는 구조체로 다수의 변수가 존재한다. size를 저장할 변수로 off\_t 타입의 size를 선언했는데 off\_t는 long 타입과 같다.

파일의 상태 정보를 알기 위해 system call 함수로 lstat()함수를 불렀는데 lstat()의 첫 번째 인수는 path로 mysize()에서 받아온 path를 받는다. 두 번째 인수는 그 경로의 파일의 상태 정보를 담을 구조체의 주소이다.

시스템 호출의 경우 return value를 꼭 체크하고 문제가 있는 경우 에러 메시지를 출력해야 하기 때문에 if문을 사용하여 lstat()의 결과 값이 -1 이라면 perror()함수를 사용해 오류 메시지를 출력하는데 perror()함수에는 그 system call 함수의 이름(“lstat”)을 준다. 오류 메시지를 출력한 뒤 exit(1) 함수를 사용해서 프로그램을 종료한다. exit(1)은 오류로 프로그램을 종료함을 의미한다.

status 구조체에서 우리가 필요한 부분은 size부분으로 status 구조체의 field인 st\_size에 담겨있고 이것을 변수 size에 저장했다. mysize()함수에서 파일의 크기 및 파일의 이름을 형식"%10ld %s\n"으로 출력해야 하기 때문에 fprintf()함수를 사용했다. 이때 10은 열자리를 ld는 long타입의 출력을 의미한다. 성공적인 return으로 return 0을 사용했다.

main()함수는 단순하게 mysize()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 for문을 사용하여 하나씩 넘겨주도록 작성했다. 성공적인 return으로 return 0을 사용했다.

**4.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**4.3**

****

**5. 파일 제거**

**5.1**

**#include <errno.h>**

**#include <unistd.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int myrm(const char \*path) {**

**if (unlink(path) == -1) {**

**perror("unlink");**

**exit(1);**

**}**

**return 0;**

**}**

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

**{**

**int i;**

**for (i=1; i<argc; i++)**

**myrm(argv[i]);**

**return 0;**

**}**

위 프로그램은 실습 5번 파일 제거를 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1] 부터 argv[argc-1]에 주어지는 모든 파일을 제거하는 프로그램이다.

먼저 헤더 파일들을 include 해줘야 하는데 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror(), exit()함수를 사용하는데 이를 위해 **errno.h**, stdlib.h를 include했다. 또한 fprintf()함수를 사용 하기위해 stdio.h를 include하였다. 앞의 1~4번과는 다르게 unlink()함수를 호출하기 위해 unistd.h도 include했다.

이 프로그램에서 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 제거하기 위해 myrm()이라는 함수를 작성했고 main()함수는 myrm()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 하나씩 넘겨주도록 작성했다. 인수의 개수는 argc에 저장된다.

myrm()함수는 system call 함수로 unlink()를 호출하는데 myrm()의 인수(파일 경로)를 unlink()로 넘겨준다. unlink()함수는 인수로 전달받은 파일의 link를 삭제한다. 즉 그 파일을 삭제하는 것이다. 성공적인 return으로 return 0을 사용했다.

시스템 호출의 경우 return value를 꼭 체크하고 문제가 있는 경우 에러 메시지를 출력해야 하기 때문에 if문을 사용하여 unlink()의 결과 값이 -1 이라면 perror()함수를 사용해 오류 메시지를 출력하는데 perror()함수에는 그 system call 함수의 이름(“unlink”)을 준다. 오류 메시지를 출력한 뒤 exit(1) 함수를 사용해서 프로그램을 종료한다. exit(1)은 오류로 프로그램을 종료함을 의미한다.

main()함수는 단순하게 myrm()함수에 인수로 argv[1] 부터 argv[argc-1]에 주어지는 파일들을 for문을 사용하여 하나씩 넘겨주도록 작성했다. 성공적인 return으로 return 0을 사용했다.

**5.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**5.3**

****

**6. 파일 이름 변경**

**6.1**

**#include <errno.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

**if (argc != 3) {**

**fprintf(stderr, "usage: %s file1 file2\n", argv[0]);**

**exit(1);**

**}**

**if (rename(argv[1], argv[2]) == -1) {**

**perror("rename");**

**exit(1);**

**}**

**return 0;**

**}**

먼저 헤더 파일들을 include 해줘야 하는데 system call 함수를 호출하였을 때 오류가 발생하면 오류메시지를 출력하고 프로그램을 종료하기 위해 perror(), exit()함수를 사용하는데 이를 위해 **errno.h**, stdlib.h를 include했다. 또한 fprintf()함수를 사용 하기위해 stdio.h를 include하였다. 앞의 1~5번과는 다르게 라이브러리 함수 rename()을 호출했는데 이 함수는 stdio.h에 저장되어 있다.

위 프로그램은 실습 6번 파일 이름 변경를 해결하기 위해 작성한 프로그램이다. 위 프로그램은 argv[1]에 주어지는 파일을 argv[2]에 주어지는 파일로 이름을 변경하는 프로그램이다.

파일의 이름을 변경하는 라이브러리 함수로 rename()을 사용했다. rename()의 첫 번째 인수는 기존 이름이고 두 번째 인수는 새로 변경할 파일의 이름이다. 여러 파일에서 하는 것이 아니기 때문에 main문안에서 해결했다. 먼저 명령어 수행 시 $ ./mymv file1 file2 에서 argv[0] 은 ./mymv이고 argv[1]은 file1 argv[2]는 file2이다. 이때 argc는 3이기 때문에 만약 argc가 3이 아니라면 오류이기 때문에 if문을 사용하여 argc가 3이 아닌 경우 printf()함수를 사용하여 오류 메시지를 출력하고 exit(1)함수를 사용하여 오류로 프로그램을 종료하도록 했다. 오류 메시지는 stderr로 출력할 수 있다. 제대로 명령어가 입력될 경우 (argc=3인경우) 만약 rename()함수의 결과가 -1 이라면 permission문제거나 파일이 존재하지 않아 오류인 경우이므로 perror()함수를 사용하여 오류메시지를 출력하고(perror(“rename”)) exit(1)함수를 사용하여 오류로 프로그램을 종료하도록 했다. 문제없이 파일의 이름이 변경되면 return 0로 성공적인 프로그램 종료를 했다.

**6.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**6.3**

****

끝.