컴퓨터학부 20201841 박세연

<symlink>

1. 소스코드

**[ssu\_symlink.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

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

// 첫번째 인자로 실제 파일의 경로 이름, 두번째 인자로 심볼릭 링크의 이름을 받아야함

if (argc != 3){

fprintf(stderr, "usage: %s <actualname> <symname>\n", argv[0]);

exit(1);

}

// symlink 함수를 통해 첫번째 파일의 심볼릭 링크를 생성, 생성한 심볼릭 링크의 이름은 두번째 인자로 지정

if (symlink(argv[1], argv[2]) < 0){

fprintf(stderr, "symlink error\n");

exit(1);

}

else {

printf("symlink: %s -> %s\n", argv[2], argv[1]);

}

exit(0);

}

1. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<fopen>

1. 소스코드

**[ssu\_test.txt]**

Linux System Programming!

Unix System Programming!

Linux Mania

Unix Mania

**[ssu\_fopen.c]**

#include <stdio.h>

#include <stdlib.h>

int main(void) {

char \*fname = "ssu\_test.txt";

char \*mode = "r";

// fopen으로 fname으로 지정한 파일을 open함

if (fopen(fname, mode) == NULL) {

fprintf(stderr, "fopen error for %s\n", fname);

exit(1);

}else {

printf("Success!\nFilename: <%s>, mode <%s>\n", fname, mode);

}

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<freopen>

1. 소스코드

**[ssu\_freopen.c]**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

int main(void){

char \*fname = "ssu\_test.txt";

int fd;

printf("First printf : Hello, OSLAB!\n");

if ((fd = open(fname, O\_RDONLY)) < 0) {

fprintf(stderr, "open error for %s\n", fname);

exit(1);

}

// freopen으로 파일을 쓰기 모드로 다시 열고 stdout(표준출력)에 연결

if (freopen(fname, "w", stdout) != NULL)

printf("Second printf : Hello, OSLAB!!\n");

// 출력되는 내용은 콘솔이 아닌 ssu\_test.txt에 출력됨

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<utime>

1. 소스코드

**[ssu\_utime.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <utime.h>

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

{

struct utimbuf time\_buf;

struct stat statbuf;

int fd;

int i;

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

// 인자로 받는 파일들의 정보를 stat 구조체에 저장

if(stat(argv[i], &statbuf) < 0) {

fprintf(stderr, "stat error for %s\n", argv[i]);

continue;

}

// 파일을 읽기/쓰기 모드로 열고, O\_TRUNC 옵션으로 파일의 내용을 비움

if ((fd = open(argv[i], O\_RDWR | O\_TRUNC)) < 0) {

fprintf(stderr,"open error for %s\n", argv[i]);

continue;

}

close(fd); // 파일디스크립터 닫음

time\_buf.actime = statbuf.st\_atime;

// 파일의 마지막 접근시간을 utimbuf 구조체 변수에 저장

time\_buf.modtime = statbuf.st\_mtime;

// 파일의 마지막 수정시간을 utimbuf 구조체 변수에 저장

// utime으로 위에서 저장한 파일의 마지막 접근 시간과 마지막 수정 시간을 원래대로 복구

if (utime(argv[i], &time\_buf) < 0) {

fprintf(stderr, "utime error for %s\n", argv[i]);

continue;

}

}

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<dir1>

1. 소스코드

**[ssu\_oslab\_1.c]**

#include <stdio.h>

#include <stdlib.h>

int main(void){

printf("This is oslab file\n");

exit(0);

}

**[ssu\_directory\_1.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

#include <string.h>

#include <sys/stat.h>

#define DIRECTORY\_SIZE MAXNAMLEN

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

struct dirent \*dentry;

struct stat statbuf;

char filename[DIRECTORY\_SIZE + 1];

DIR \*dirp;

// 디렉터리 경로를 인자로 받음

if (argc <2) {

fprintf(stderr, "usage: %s <directory>\n", argv[0]);

exit(1);

}

// opendir로 사용자가 지정한 디렉터리를 연다

// chdir로 현재 작업 디렉터리를 사용자가 지정한 디렉터리로 변경

// 둘 중 하나라도 오류가 발생하면 오류 메세지 출력 후 강제 종료

if ((dirp = opendir(argv[1])) == NULL || chdir(argv[1]) == -1) {

fprintf(stderr, "opendir, chdir error for %s\n", argv[1]);

exit(1);

}

// readdir 함수를 반복문으로 호출하여 디렉터리 내의 모든 항목에 대한 정보를 가져옴

while((dentry = readdir(dirp)) != NULL) {

// d\_ino == 0이면 유효하지 않은 엔트리로 무시

if (dentry->d\_ino == 0)

continue;

memcpy(filename, dentry->d\_name, DIRECTORY\_SIZE);

// 디렉터리 엔드리에서 파일이름은 filename 배열로 복사

// stat 함수로 현재 파일의 정보를 statbuf에 저장

if (stat(filename, &statbuf) == -1) {

fprintf(stderr, "stat error for %s\n", filename);

break;

}

// 파일이 일반 파일인 경우에만 파일의 이름과 크기를 출력

if ((statbuf.st\_mode & S\_IFMT) == S\_IFREG)

printf("%-14s %ld\n", filename, statbuf.st\_size);

else

printf("%-14s\n", filename);

}

exit(0);

}

2. 실행결과

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<dir2>

1. 소스코드

**[ssu\_osdir/ssu\_dir1/ssu\_file1]**

Hi,

it's the Keyword

Bye.

**[ssu\_osdir/ssu\_dir2/ssu\_file2]**

Hi,

it's not the keyword.

Bye.

**[ssu\_directory\_2.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dirent.h>

#include <limits.h>

#include <string.h>

#include <sys/stat.h>

#ifdef PATH\_MAX

static int pathmax = PATH\_MAX;

#else

static int pathmax = 0;

#endif

#define MAX\_PATH\_GUESSED 1024

#ifndef LINE\_MAX

#define LINE\_MAX 2048

#endif

char \*pathname;

char command[LINE\_MAX], grep\_cmd[LINE\_MAX];

int ssu\_do\_grep(void){

struct dirent \*dirp;

struct stat statbuf;

char \*ptr;

DIR \*dp;

// lstat으로 파일을 검사하고 파일의 정보를 저장함

if (lstat(pathname, &statbuf) < 0){

fprintf(stderr,"lstat error for %s\n", pathname);

return 0;

}

// 디렉터리가 아닌 경우 실행됨 (일반 파일인 경우 실행)

if (S\_ISDIR(statbuf.st\_mode) == 0) {

sprintf(command, "%s %s", grep\_cmd, pathname);

printf("%s : \n", pathname);

system(command);

return 0;

}

// 경로가 디렉터리인 경우

ptr = pathname + strlen(pathname); // ptr이 pathname의 마지막을 가리키게 됨

\*ptr++ = '/'; // '/'추가

\*ptr = '\0'; // 문자열의 끝을 나타내는 널문자 추가

if ((dp = opendir(pathname)) == NULL) { // 해당 디렉터리를 open

fprintf(stderr, "opendir error for %s\n", pathname);

return 0;

}

// 반복문으로 readdir 함수를 통해 디렉터리 내의 파일들을 읽음

while ((dirp = readdir(dp)) != NULL)

if (strcmp(dirp->d\_name, ".") && strcmp(dirp->d\_name, "..")) {

// .이나 ..이 아닌경우 실행

strcpy(ptr, dirp->d\_name); // 파일이름을 경로에 붙임

if (ssu\_do\_grep() < 0) // 재귀호출

break;

}

ptr[-1] = 0; // 슬래시 제거

closedir(dp);

return 0;

}

void ssu\_make\_grep(int argc, char \*argv[]){

int i;

strcpy(grep\_cmd, " grep");

// 사용자로부터 받은 grep 옵션들을 grep\_cmd 문자열에 추가함

for (i = 1; i < argc-1; i++){

strcat(grep\_cmd, " ");

strcat(grep\_cmd, argv[i]);

}

}

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

{

if (argc < 2) {

fprintf(stderr, "usage: %s <-CVbchilnsvwx> <-num> <-A num> <-B num> <-f file> \n"" <-e> expr <directory>\n", argv[0]);

exit(1);

}

// 파일의 최대 길이 설정

if (pathmax == 0){

if ((pathmax = pathconf("/", \_PC\_PATH\_MAX)) < 0)

pathmax = MAX\_PATH\_GUESSED;

else

pathmax++;

}

// 파일의 이름을 받을 변수를 동적할당

if ((pathname = (char \*)malloc(pathmax+1)) == NULL) {

fprintf(stderr, "malloc error\n");

exit(1);

}

strcpy(pathname, argv[argc-1]); // 인자로 받은 파일 경로를 복사하여 저장함

ssu\_make\_grep(argc, argv);

ssu\_do\_grep();

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<chdir>

1. 소스코드

**[ssu\_chdir.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void){

// chdir로 현재 작업 디렉터리를 /etc로 변경

if(chdir("/etc") < 0){

fprintf(stderr, "chdir error\n");

exit(1);

}

printf("chdir to /etc succeeded.\n");

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<getcwd>

1. 소스코드

**[ssu\_getcwd.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define PATH\_MAX 1024

int main(void){

char \*pathname;

// chdir로 현재 작업 디렉터리를 /home/seyeon으로 변경

if (chdir("/home/seyeon") < 0) {

fprintf(stderr, "chdir error\n");

exit(1);

}

pathname = malloc(PATH\_MAX);

// getcwd 함수로 현재 작업디렉터리를 pathname에 저장

if (getcwd(pathname, PATH\_MAX) == NULL){

fprintf(stderr, "getcwd error\n");

exit(1);

}

printf("current directory = %s\n", pathname); // 현재 작업 디렉터리 확인

exit(0);

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명