**PRACTICAL-2**

**Aim:**

Implement copy command using open, create, read, write, access and close system call. Be sure to include all necessary error checking including, ensuring the source file exists. Test your program with following specifications.a. File extensionwith .txt, .c, .zip, .exe, .tarb. Copy the whole directory.

**Program Code:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <libgen.h>

#include<dirent.h>

char\* get\_path(char\* path, char\* file\_name) { char\* new\_path = (char \*)malloc(strlen("") + 256); new\_path = strdup(""); strcat(new\_path, path); strcat(new\_path, "/");

strcat(new\_path, file\_name);

return new\_path;

}

void copy\_file(char\* source, char\* dest) { char buf; int source\_file, dest\_file, n;

source\_file = open(source, O\_RDONLY); if (source\_file == -1) { perror("\nSOURCE FILE ERROR"); exit(0);

} else {

dest\_file = open(dest, O\_WRONLY | O\_TRUNC | O\_CREAT, 0641); if (dest\_file == -1) {

perror("\nDESTINATION FILE ERROR"); exit(0);

} else {

while ((n = read(source\_file, &buf, 1)) != 0) {

write(dest\_file, &buf, 1);

}

close(source\_file);

close(dest\_file);

}

}

}

void copy\_dir(char\* source\_path, char\* dest\_path) { char\* dest\_dir\_name[256]; char\* source\_dir\_name[256];

char\* de;

char\* source = (char \*)malloc(strlen("") + 256); char\* dest = (char \*)malloc(strlen("") + 256); struct dirent\* dir; int i = 0, j = 0; for(i = 0; i < 256; i++) { dest\_dir\_name[i] = NULL;

source\_dir\_name[i] = NULL;

}

i=0;

DIR\* fs = opendir(source\_path); while((dir = readdir(fs)) != NULL) { de = dir->d\_name;

source = get\_path(source\_path, de); dest = get\_path(dest\_path, de);

if(dir->d\_type == DT\_DIR) {

if (mkdir(dest) == -1) {

perror("\nDIRECTORY CREATION ERROR"); exit(0);

} else {

dest\_dir\_name[i++] = strdup(dest); source\_dir\_name[j++] = strdup(dest);

}

} else if (dir->d\_type == DT\_REG) {

copy\_file(source, dest);

}

}

1. = 0;
2. = 0;

while(dest\_dir\_name[i] != NULL) { copy\_dir(source\_dir\_name[i++], dest\_dir\_name[j++]);

}

free(source); free(dest); free(de); free(fs); free(dir); for(i = 0; i < 256; i++) { free(dest\_dir\_name[i]); free(source\_dir\_name[i]);

}

}

void main(int argc, char\* argv[]) { if (argc == 3) {

struct stat stat\_data;

int status;

if (argv[1][strlen(argv[1]) - 1] == '/' ) { argv[1][strlen(argv[1]) - 1] = ' ';

}

if (argv[2][strlen(argv[2]) - 1] == '/' ) { argv[2][strlen(argv[2]) - 1] = ' ';

}

status = stat(argv[1], &stat\_data); if (S\_ISDIR(stat\_data.st\_mode)) { status = stat(argv[2], &stat\_data); if (S\_ISDIR(stat\_data.st\_mode)) {

copy\_dir(argv[1], argv[2]);

} else {

perror("\nINVALID INPUT");

exit(0);

} } else {

status = stat(argv[2], &stat\_data); if (S\_ISDIR(stat\_data.st\_mode)) { if (argv[2][strlen(argv[2]) - 1] == ' ' ) { argv[2][strlen(argv[2]) - 1] = '/';

} else { strcat(argv[2], "/");

}

strcat(argv[2], basename(argv[1]));

}

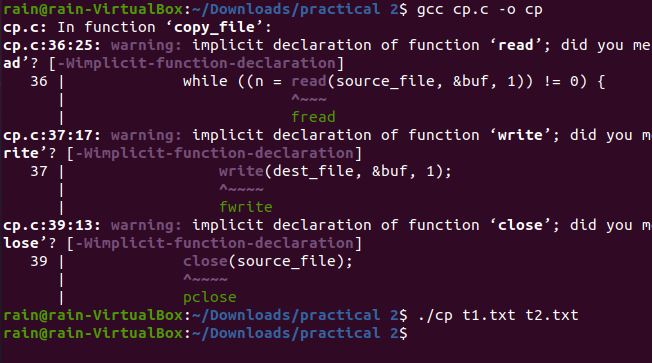
copy\_file(argv[1], argv[2]);

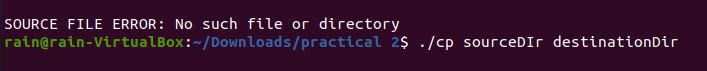
}

}

}

**Output:**



****

****

