# HW #1

## Copy Files and Directories

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

# Requirement

- Implement a program to copy files and directories
  - It is similar to ‘cp’ in Linux but simpler

- Required programming skills
  - Command line argument
  - Get options from command line argument
  - Linux system calls
    - File handling: open, read, write, file status (permission)
    - Directory handling

# Requirements

- The program should provide following functionalities depending on options
  - `-f`: Copy source file to target file

    ```
    $ ./copyfile -f SourceFile TargetFile
    ```

    option

  - `-m`: Copy multiple source files to target directory

    ```
    $ ./copyfile -m SourceFile1 SourceFile2 ... TargetDirectory
    ```

  - `-d`: Copy all files and directories from source directory to target directory

    ```
    $ ./copyfile -d SourceDirectory TargetDirectory
    ```

* Program name is '`copyfile`'

# Requirements

- The program should provide following functionalities depending on options
  - `-v`: verbose mode → print out what is being done
    - This option should be used with other options

```
$ ./copyfile -fv SourceFile TargetFile
$ ./copyfile -mv SourceFile1 SourceFile2 ... TargetDirectory
$ ./copyfile -dv SourceDirectory TargetDirectory
```

    - See p.9 for the results of this option

# Requirements

- In this homework, do not use the standard C library for file I/O
  - File I/O functions such fopen(), fread(), fwrite(), etc are not allowed.
  - However, non-file I/O functions such as printf() and sprintf(), etc are allowed.

- You must use system calls for file I/O
  - You may use open(), read(), write()

- The program should recognize command line arguments and options
  - Use argc and argv for command line arguments → See p.11~14
  - Use getopt() for parsing options from the entered arguments → See p.15

# Requirements

- Target files and directory must have same permission with source files and directory
  - Linux file system can grant permissions of files and directories that control the ability of users to read, write, and execute the contents of file
  - For more detailed about the file permission, you can refer to following link
    - https://help.ubuntu.com/community/FilePermissions
  - When you create the file, you can grant file permission
    - Please refer to third parameter (i.e., `mode`) of `open()`
      - See p.11 in Chapter 2 Lecture Slide
  - To get the file permission of source file, you can use `stat()`
    - `stat()` provides file information including the device ID, file type(directory or file…), file mode (permission), owner, etc.
    - https://man7.org/linux/man-pages/man2/lstat.2.html

# Requirements

- You may need to use following functions to handle directory
  - opendir(): open a directory
    - https://man7.org/linux/man-pages/man3/opendir.3.html
  - readdir(): read a directory
    - https://man7.org/linux/man-pages/man3/readdir.3.html
  - mkdir(): create a directory
    - https://man7.org/linux/man-pages/man2/mkdir.2.html

  - You can find some example in the Internet. You must figure out by yourself!

# Requirements

- Expected results

```
yunmin@mcnl-server:~/workspace/os/hw1$ ls -l
total 80
-rwxrwxr-x 1 yunmin yunmin 16784 Mar 15 00:14 arg
-rw-rw-r-- 1 yunmin yunmin   412 Mar 15 00:14 arg.c
-rwxrwxr-x 1 yunmin yunmin 17640 Mar 15 00:12 copyfile
-rw-rw-r-- 1 yunmin yunmin  5193 Mar 15 00:12 copyfile.c
-rwxrwxr-x 1 yunmin yunmin 16832 Mar 15 00:13 getopt
-rw-rw-r-- 1 yunmin yunmin   722 Mar 15 00:12 getopt.c
drwxrwxr-x 4 yunmin yunmin  4096 Mar 15 00:17 temp
yunmin@mcnl-server:~/workspace/os/hw1$ tree
.
├── arg
├── arg.c
├── copyfile
├── copyfile.c
├── getopt
├── getopt.c
└── temp
    ├── dir1
    │   └── copyfile.c
    └── dir2
        └── getopt.c

3 directories, 8 files
```

```
yunmin@mcnl-server:~/workspace/os/hw1$ ./copyfile -f arg.c arg2.c
yunmin@mcnl-server:~/workspace/os/hw1$ ./copyfile -f getopt getopt2
yunmin@mcnl-server:~/workspace/os/hw1$ ls -l
total 104
-rwxrwxr-x 1 yunmin yunmin 16784 Mar 15 00:14 arg
-rw-rw-r-- 1 yunmin yunmin   412 Mar 15 00:19 arg2.c
-rw-rw-r-- 1 yunmin yunmin   412 Mar 15 00:14 arg.c
-rwxrwxr-x 1 yunmin yunmin 17640 Mar 15 00:12 copyfile
-rw-rw-r-- 1 yunmin yunmin  5193 Mar 15 00:12 copyfile.c
-rwxrwxr-x 1 yunmin yunmin 16832 Mar 15 00:13 getopt
-rwxrwxr-x 1 yunmin yunmin 16832 Mar 15 00:19 getopt2
-rw-rw-r-- 1 yunmin yunmin   722 Mar 15 00:12 getopt.c
drwxrwxr-x 4 yunmin yunmin  4096 Mar 15 00:17 temp
```

Copied files

‘`tree`’ shows list contents of directories in a tree-like format.
(To install tree, type ‘sudo apt-get install tree in the terminal)

# Requirements

- Expected results

```
yunmin@mcnl-server:~/workspace/os/hw1$ ./copyfile -mv arg2.c getopt2 copyfile temp
Copy File: arg2.c -> temp/arg2.c
Copy File: getopt2 -> temp/getopt2
Copy File: copyfile -> temp/copyfile
yunmin@mcnl-server:~/workspace/os/hw1$ ./copyfile -dv temp temp2
Copy File: temp/getopt2 -> temp2/getopt2
Copy File: temp/dir1/copyfile.c -> temp2/dir1/copyfile.c
Copy File: temp/dir2/getopt.c -> temp2/dir2/getopt.c
Copy File: temp/arg2.c -> temp2/arg2.c
Copy File: temp/copyfile -> temp2/copyfile
```

Verbose mode

Copied files

Copied directory

```
yunmin@mcnl-server:~/workspace/os/hw1$ tree
.
├── arg
├── arg2.c
├── arg.c
├── copyfile
├── copyfile.c
├── getopt
├── getopt2
├── getopt.c
├── temp
│   ├── arg2.c
│   ├── copyfile
│   ├── dir1
│   │   └── copyfile.c
│   ├── dir2
│   │   └── getopt.c
│   └── getopt2
└── temp2
    ├── arg2.c
    ├── copyfile
    ├── dir1
    │   └── copyfile.c
    ├── dir2
    │   └── getopt.c
    └── getopt2

6 directories, 18 files
```

# Requirements

- Use ‘`vi`’ to implement your program
  - There will be some problems about ‘vi’ in the midterm exam.
    - See p.16 ~ 17
- Write clean source code
  - Add proper comment in your source code
  - Consider code indentation for enhancing readability
- Upload ZIP file on LMS by compressing all your source codes
  - File name: hw01_student id.zip  (ex: hw01_20400022.zip)

- Due date: 11:59pm, 4/2 (Tue)

# COMMAND LINE ARGUMENT & GET OPTIONS & VI

# Command Line Argument

- When executing a program in either C or C++, there is a way to pass command line arguments
- Command line arguments can be options or some information provided to the program
- Each argument is separated by a space
- `main()` gets two parameters for arguments

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

  - `argc`: number of arguments
  - `argv`: argument list

# Command Line Argument

- Conventional rules:
  - Arguments are always passed to main().
  - There must be two
    - first is an integer → `int argc`
    - second char pointer to an array → `char *argv[]`
  - First argument (`argv[0]`) will always be the name of the calling program.
  - `argc` will always be at least 1
  - The first argument is always `argv[0]`
  - The last argument is always `argv[argc-1]`
  - `argv[argc]` will always be a null pointer
  - Arguments are always passed as character strings.
    - Numbers must be converted from characters to integers, floats, doubles, etc.

# Command Line Argument

- Example (arg.c)

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i;
    printf("Command Line Arguments!\n");
    printf("argc = %d\n", argc);
    for (i = 0; i < argc; i++)
    {
        // Print arguments
        // atoi: convert string to integer type value if the string is integer
        printf("argv[%d] = %s (%d) \n", i, argv[i], atoi(argv[i]));
    }
    return 0;
}
```

```
PS C:\ds\hw01> .\arg.exe handong global university
Command Line Arguments!
argc = 4
argv[0] = C:\ds\hw01\arg.exe (0)
argv[1] = handong (0)
argv[2] = global (0)
argv[3] = university (0)
PS C:\ds\hw01> .\arg.exe handong 1 2 data structures
Command Line Arguments!
argc = 6
argv[0] = C:\ds\hw01\arg.exe (0)
argv[1] = handong (0)
argv[2] = 1 (1)
argv[3] = 2 (2)
argv[4] = data (0)
argv[5] = structures (0)
```

# getopt()

- getopt() parses command line options
  - https://man7.org/linux/man-pages/man3/getopt.3.html
- Example (getopt.c)

In this homework,
we use short option format only.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    int opt;
    int flag_a = 0, flag_b = 0, flag_c = 0, flag_d = 0;

    while((opt = getopt(argc, argv, "abcd")) != -1) {
        switch (opt) {
            case 'a': flag_a = 1; break;
            case 'b': flag_b = 1; break;
            case 'c': flag_c = 1; break;
            case 'd': flag_d = 1; break;
            case '?': printf("Unknown flag : %c \n", optopt);
        }
    }

    if (flag_a)
        printf("Flag a is enable\n");
    if (flag_b)
        printf("Flag b is enable\n");
    if (flag_c)
        printf("Flag c is enable\n");
    if (flag_d)
        printf("Flag d is enable\n");

    return 0;
}
```

```
yunmin@mcnl-server:~/workspace/os/hw1$ ./getopt -a
Flag a is enable
yunmin@mcnl-server:~/workspace/os/hw1$ ./getopt -ab -c
Flag a is enable
Flag b is enable
Flag c is enable
yunmin@mcnl-server:~/workspace/os/hw1$ ./getopt -cba
Flag a is enable
Flag b is enable
Flag c is enable
yunmin@mcnl-server:~/workspace/os/hw1$ ./getopt -a 111 -b 111 -cd
Flag a is enable
Flag b is enable
Flag c is enable
Flag d is enable
yunmin@mcnl-server:~/workspace/os/hw1$ ./getopt -a 111 -f
./getopt: invalid option -- 'f'
Unknown flag : f
Flag a is enable
```

# Linux Command

- Learn -> ls, pwd, cd, rm, mv, mkdir, cp, cat, etc.

- The 40 Most-Used Linux Commands You Should Know (https://kinsta.com/blog/linux-commands/)

- 자주사용하는리눅스기본명령어모음 (한글) (https://shanepark.tistory.com/196)

# Vi Usage

- Basic vi Commands (https://www.cs.colostate.edu/helpdocs/vi.html)
- Vi Editor with Commands in Linux/Unix Tutorial (https://www.guru99.com/the-vi-editor.html)
- Vi 에디터사용법 (https://blog.lael.be/post/7321)