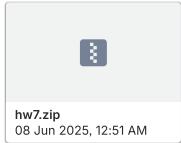


## (2025 OS)7. File system operations



### Overview

In this assignment, you will practice functions that are related to file system operations to deal with files and directories. You will learn how to create, read, write, navigate files, as well as how to retrieve properties of files within directories. These skills are essential for automating tasks and working efficiently with files in programming.

### Part I

#### Descriptions

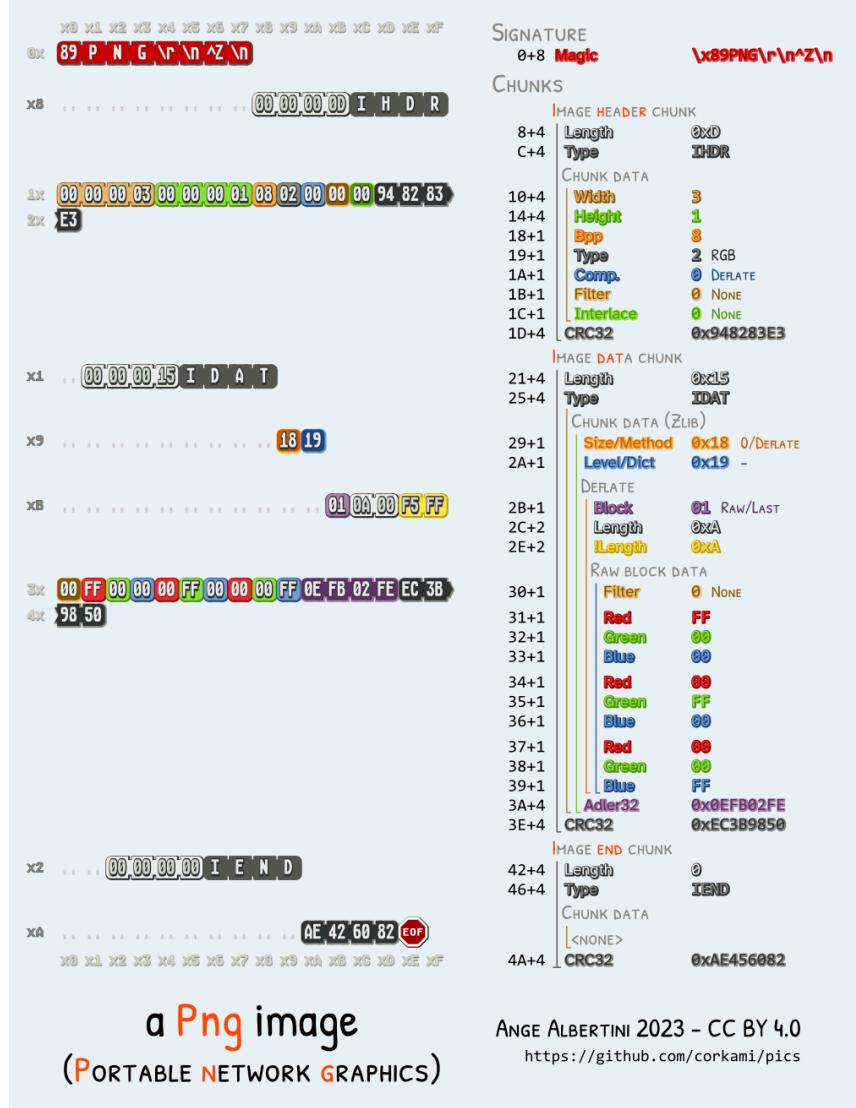
In Part I, you are required to use file operation APIs to implement a PNG exiftool. [ExifTool](#) is a platform-independent Perl library plus a command-line application for reading, writing and editing meta information in a wide variety of files. But, in this part, we are only focus on reading and parsing metadata only in PNG format.

Here are some file operation APIs:

- [fopen](#)
- [fread](#)
- [fseek](#)
- [fwrite](#)
- [fclose](#)

#### Requirement & Testing

As described above, you need to create a program `hw7_1_yourStudentID.c` to parse metadata of a PNG file. A simple PNG file structure shows below:



credit: Ange Albertini from <https://github.com/corkami/pics>

To parsing it, you have to do the following steps

1. Use binary mode to open the file
2. Read its magic, check the file is actually a PNG. (You can just check 0x89 P N G as a simpler implementation)
3. After check the file is PNG, try to locate the IHDR chunk. You have to check the chunk type is IHDR. If it is, do the next step, otherwise bypass the chunk (seek to `chunk_length` after current file-position indicator) and read next chunk.
4. Reading IHDR chunk data. You have to read the image width (`width`), image height (`height`), bit depth (`bpp`), image type (`type`) and IHDR CRC (`crc32`)

To verified the parsing correctness, you can use [this tool](#) to check.

After parsing the PNG metadata, you have to use file writing APIs to create a report **automatically**. The report format shows below:

If the file is not PNG file ( `{file name}` is the field to fill):

```

1 # {{file name}}
2 ...
3 ...

```

```
4 FileType: Unknown
5 ``
6
7
```

If the file is PNG file ( `{}` is the field to fill):

```
1 # {{file name}}
2 ``
3
4 FileType: PNG
5 Size: {{image width}}x{{image height}}
6 Bit Depth: {{bit depth}}
7 Color Type: {{image type name}}
8 IHDR CRC: {{CRC}}
9 ``
10 
11
12
13
```

You need to use **append** mode to insert the new report under old report.

After finish the program, use `gcc -o hw7_1 hw7_1_yourStudentID.c` to compile. And use `./hw7_1 imagefile outputfile` to execute it.

```
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ gcc -o hw7_1 hw7_1_answer.c
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ ./hw7_1 ./hw7_1_imgs/1.png output.md
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ ./hw7_1 ./hw7_1_imgs/6.png output.md
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ cat output.md
# ./hw7_1_imgs/1.png
```
FileType: PNG
Size: 439x512
Bit Depth: 8
Color Type: RGB
IHDR CRC: 8ed2b5ec
```

# ./hw7_1_imgs/6.png
```
FileType: Unknown
```

ywc@ywc-labpc:~/桌面/os_ta/filesystem$
```

sample output

## Note

1. There is a `hw7_1_template.c` for you to start.
2. You can use images inside `hw7_1_imgs/` folder for testing.
3. You are only allow to use `fopen`, `fread`, `fwrite`, `fseek`, `fclose` to deal with file operations. C++ file stream, `open`, `read`, `write`, `close` are **not** allowed.
4. Format for CRC is `%02x%02x%02x%02x`.
5. Convert table from `image type` to `image type name` is showed in `hw7_1_template.c`.
6. Be aware of [endianness](#).
7. For the detail PNG format, you can read [Portable Network Graphics \(PNG\) Specification \(Third Edition\)](#) as for your interest. It is not necessary to read it in this assignment.

## Part II

### Descriptions

In Part II, you are required to use file and directory property APIs to implement a custom `ls` program. You will try to read files' metadata in a directory and output their basic information.

Here are some directory and file property related APIs and types:

- [opendir](#)
- [readdir](#)
- [closedir](#)
- [lstat](#)
- [stat type](#)

### Requirement & Testing

The requirement for your `ls`-like program `hw7_2_yourStudentID.c` must satisfy the following requirement:

- Read the target directory from `argv[1]`.
- For each file / directory in the target directory, output the following information:
  - File type
    - If the file is a regular file, file type field is `-`.
    - If the file is a directory, file type field is `d`.
    - If the file is a link, file type field is `l`.
    - If the file is a FIFO, file type field is `f`.
    - If the file is a socket, file type field is `s`.
    - If the file is a block device, file type field is `b`.
    - If the file is a character device, file type field is `c`.
    - If the file isn't belongs to any of the above types, file type field is `?`.
  - File permission
    - There are 3 subfield in file permission field - owner permission, group permission, and other user permission.
    - for each subfield, there are 3 permission - `r` for read permission, `w` for write permission, `x` for execute permission. If each subfield user has read / write / execute permission, set `r / w / x` correspondingly, otherwise set `-` if subfield user has no such permission.
    - If SUID (set-user-ID) bit is set, set `S` within the execute permission in the owner subfield if owner has execute permission. If SUID bit is set but owner has no execute permission, set `S`.
    - Like SUID, if SGID (set-group-ID) bit is set, set `S` within the execute permission in the group subfield if group user has execute permission. If SGID bit is set but group user has no execute permission, set `S`.
    - Like SUID and SGID, if SBIT (sticky bit) bit is set, set `t` within the execute permission in the other subfield if other user has execute permission. If SBIT bit is set but other user has no execute permission, set `T`.
  - UID

- Must be 4 bytes fixed length, right-aligned. (i.e. `%4d` )
- GID
  - Must be 4 bytes fixed length, right-aligned. (i.e. `%4d` )
- Size of file (in byte)
  - Must be 6 bytes fixed length, right-aligned. (i.e. `%6ld` )
- Time of last modification of file data
  - Show in human-readable string.
  - You can use [ctime](#) API to convert.
- Filename
- The output field format:  `{{file type}} {{file permission}} {{UID}} {{GID}} {{size}} {{filename}}` .

After finish your program, use `gcc -o hw7_2 hw7_2_yourStudentID.c` to compile. And use `./hw7_2 <dir>` to execute it. You can verify the result with `ls -al <dir>`.

```
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ gcc -o hw7_2 hw7_2_answer.c
ywc@ywc-labpc:~/桌面/os_ta/filesystem$ ./hw7_2 testdir/
-rw-rw-r-- 1000 1000 8 Sat Nov 16 23:55:11 2024 03.txt
-rw-rw-r-- 1000 1000 8 Sat Nov 16 23:54:09 2024 02.txt
-rw-rw-r-- 1000 1000 8 Sat Nov 16 23:54:17 2024 01.txt
drwxrwxr-x 1000 1000 4096 Sat Nov 16 23:58:04 2024 ..
drwxrwxr-x 1000 1000 4096 Sat Nov 16 23:57:25 2024 .
```

sample output

#### Note:

1. There is a `hw7_2_template.c` for you to start.
2. You are only allow to use `readdir` to read directory in your implementation.
3. Be aware of `\n` in `ctime` returned.

## Report

You need to write a report answering the following questions :

- Part I (File operation)
  - A screenshot of your test results.
  - Briefly explain your code.
- Part II (Directory and file property)
  - A screenshot of your test results.
  - Briefly explain your code.
  - Does `readdir` returns in some kind of orders? If so, how does it ordered? If not, how can the program be modified to make it an ordered output?
    - Note: If your answer is `doesn't`, please submit the **unordered** version (original version) when uploading.
  - Choose one directory you are interested. Observe it with your program and briefly describe your observation.
- Any difficulties you encountered during this homework?

## Submission

Please submit a **zip** file to E3 which contains your source code and report.

- Make sure your code can be compiled and run on Ubuntu 22.04 LTS.
- Make sure your testing output is correct.
- Your report should be submitted in PDF format.
- The structure of the zip file should be as the following:

`<student_id>.zip`

```
| - <student_id>/  
|   | - hw7_1_<student_id>.c  
|   | - hw7_2_<student_id>.c  
|   | - hw7_<student_id>.pdf
```

## Scoring Policy

- Part I
  - Source code (25%)
    - a. A screenshot of your test results (5%)
    - b. Briefly explain your code (10%)
- Part II
  - Source code (25%)
    - a. A screenshot of your test results (5%)
    - b. Briefly explain your code (10%)
    - c. Does `readdir` returns in some kind of orders? If so, how does it ordered? If not, how can the program be modified to make it an ordered output? (10%)
    - d. Choose one directory you are interested. Observe it with your program and briefly describe your observation (10%)
  - Any difficulties you encountered during this homework? (Optional)

For questions, please contact TA Jay Hsiao via E3 platform, or email <[shjay629@gmail.com](mailto:shjay629@gmail.com)>