10. File Processing

Programming in C

Teacher: Po-Wen Chi

neokent@gapps.ntnu.edu.tw

March 25, 2020

Department of Computer Science and Information Engineering,
National Taiwan Normal University

Objectives

- Understand the concept of files and streams.
- Create and read data as strings.
- Create, read and update data in binary mode.
- Some useful functions.

What is a File?

File

- C views each file simply a sequential stream of bytes.
- Each file ends with an end-of-file indicator, which is raised by your system instead of file.

Figure 1: It looks like there is an additional byte appended to the file. However, that additional byte does not exist. Instead, it is a signal raised by your system.

File Stream

- Streams provide a **higher-level** interface for you to access data.
  - fprintf
  - fscanf

Actually, You Have Already Used File Streams

- Three standard streams:
  1. standard input. **0**
  2. standard output. **1**
  3. standard error. **2**
- Really? Where?

scanf

The scanf() function reads input from the **standard input stream stdin**.

scanf implementation

```
int
scanf (const char *fmt, ...)
{
    int                 count;
    va_list ap;

    va_start (ap, fmt);
    count = vfscanf (stdin, fmt, ap);
    va_end (ap);
    return (count);
}
```

File Descriptor

- File descriptors provide a primitive, low-level interface to input and output operations.
- In linux, **everything can be treated as a file**.
  - Network socket.
  - Device driver.
  - Inter-process communication.
- What operations can be supported?
  - Open.
  - read.
  - write.
  - close.
- If you want to control some detail settings, you need to use file descriptor instead of file stream.

File Stream vs. File Descriptor

- Structure:
  - Stream: **FILE ***.
  - Descriptor: **int**.
- What you can do with file streams, you can absolutely do them with file descriptor. The reverse is not true.
- File streams have more convenient interfaces for you.

So ...

- In this class, I will show you how to access data with file stream related functions.
- You need to be familiar with these functions.
- I will also show you how to do the same thing with file descriptor.

I/O Redirection

Do You Know How to Write Data to a File?

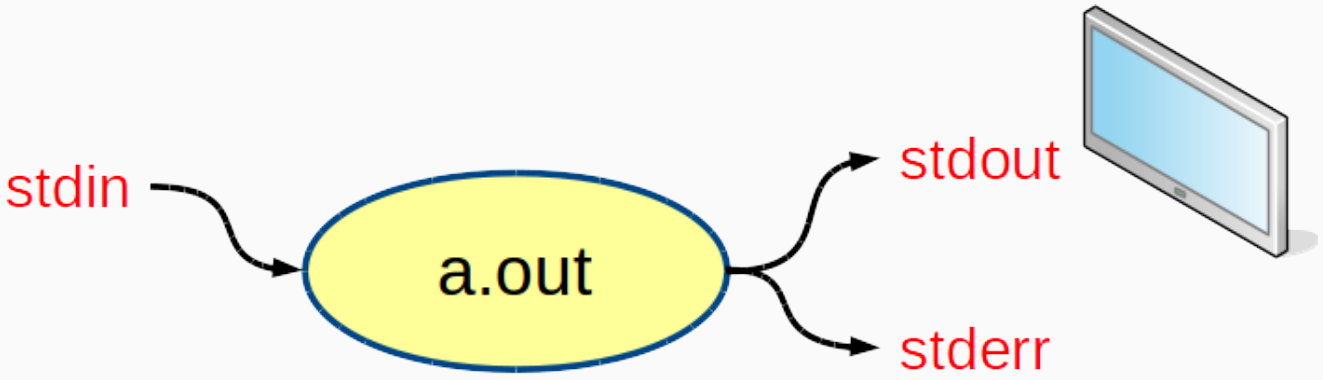
```
#include <stdio.h>

int main()
{
    printf( "Hello World!\n" );
    return 0;
}
```

The functions **printf()** writes output to stdout, the standard output stream.

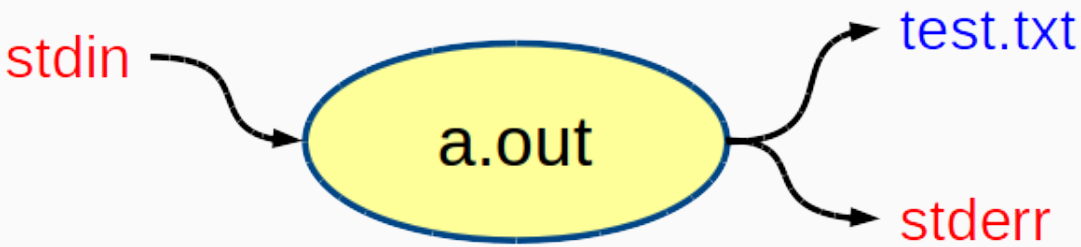
Can I write "Hello World!" to another file?

The Easiest Way: I/O Redirection

- Before redirection:



- **./a.out > test.txt**
- After redirection:



Let's try it.

I/O Redirection

- > filename
  - Redirect stdout to a file.
  - Creates the file if not present, otherwise **overwrites** it.
- >> filename
  - Redirect stdout to a file.
  - Creates the file if not present, otherwise **appends** to it.

Wait a Minute! Something is Wrong ...

Let's try **hello.v2**.

```
$ ./hello.v2
Hello World!
Hello Kitty!: Success
$ ./hello.v2 > 123.txt
Hello Kitty!: Success
$ cat 123.txt
Hello World!
```

Wait a Minute! Something is Wrong ...

Let's try **hello.v2**.

```
$ ./hello.v2
Hello World!
Hello Kitty!: Success
$ ./hello.v2 > 123.txt
Hello Kitty!: Success
$ cat 123.txt
Hello World!
```

What's wrong!

I/O Redirection

- > filename
- 1> filename
  - Redirect stdout to file "filename."
- >> filename
- 1>> filename
  - Redirect and append stdout to file "filename."
- 2> filename
  - Redirect stderr to file "filename."
- 2>> filename
  - Redirect and append stderr to file "filename."
- &> filename
  - Redirect stdout, stderr to file "filename."

How About STDIN?

Sure. You can follow this command:

```
$ ./abs < number.txt
```

How About STDIN?

Sure. You can follow this command:

```
$ ./abs < number.txt
```

So now you know how most TAs test your code.

Text File Processing

Example

Please see **example.10.fprintf/main.c**.

fopen

FILE *fopen(const char *path, const char *mode);

- path: The file path.
- mode: Points to a **string** beginning with one of the following sequences:
  - "r": read
  - "w": write
  - "a": append
  - More options are in the manual.
- Return: File *
  - It it successes, a FILE pointer is returned.
  - If it fails, NULL is returned and **errno** is set to indicate the error.
  - Always remember to check the pointer return!
    - Can you write a file in /etc?

fprintf

- Almost the same with **printf**. Except that you need to give a FILE stream pointer.
- **Quiz**: What if you enter a NULL pointer??

End-of-File Indicator

- Linu/Mac OS X/UNIX: <Ctrl> d
- Windows: <Ctrl> z + enter

fclose

int fclose(FILE *stream);

- The fclose() function **flushes** the stream pointed to by stream (writing any buffered output data using fflush(3)) and **closes** the underlying file descriptor.
- That is, the buffered data will be written to the file when you close it.
  - **Quiz**: Without closing it, will data be written to the file?

IMPORTANT

After open a file, please remember to close it.

IMPORTANT

After open a file, please remember to close it.

Why? There is nothing wrong ...

IMPORTANT

After open a file, please remember to close it.

Why? There is nothing wrong ...

There is a maximum number of simultaneously opened files.

Example v2

- If I forget how to input EOF through the keyboard, how can I terminate the program?
  - Ctrl + C ??
- Let's see the example 2.

Example Append

Please see main.append.c

What is the difference??

How About Reading a Text File?

- Just like fprintf, we have fscanf.
- Of course, the file should be opened in **"r"** mode.
- Please see example.10.fscanf.
- **Quiz**: What will happen if the file does not exist?

Tips

- When you use **fprintf** and **fscanf** to handle text files, it implies that you know the file format clearly.
- In this case, the input check may not be necessary, right?
  - YES: the format is determined by you.
  - NO: someone may modify the file and make you read the file.

Quiz

Can you open a file twice without closing the first one first?

Quiz

Can you open a file twice without closing the first one first?

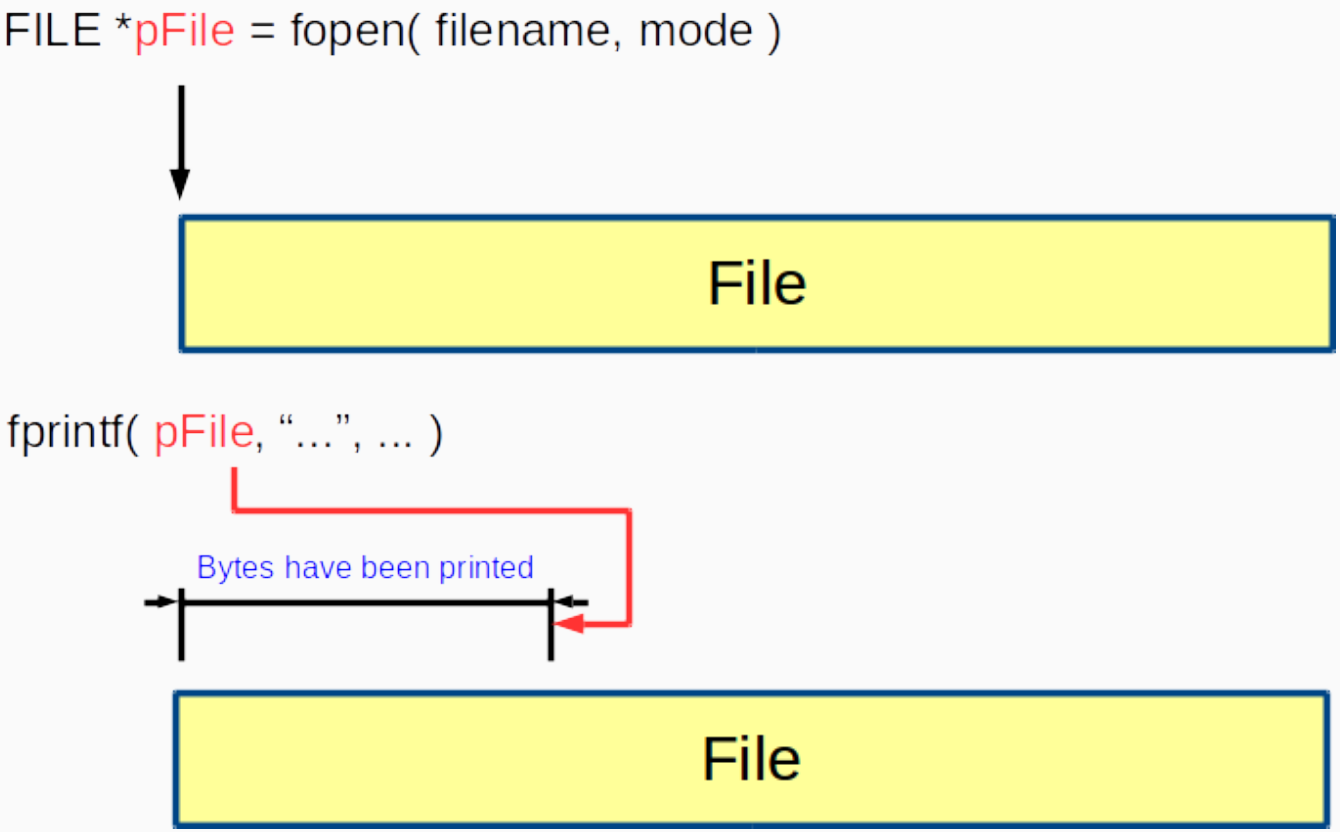
Can I open a file twice in the write mode without closing the first one first?

Practice: Score Sorting

Please write a program to sort score.txt according to students' scores. Save the result to another file.

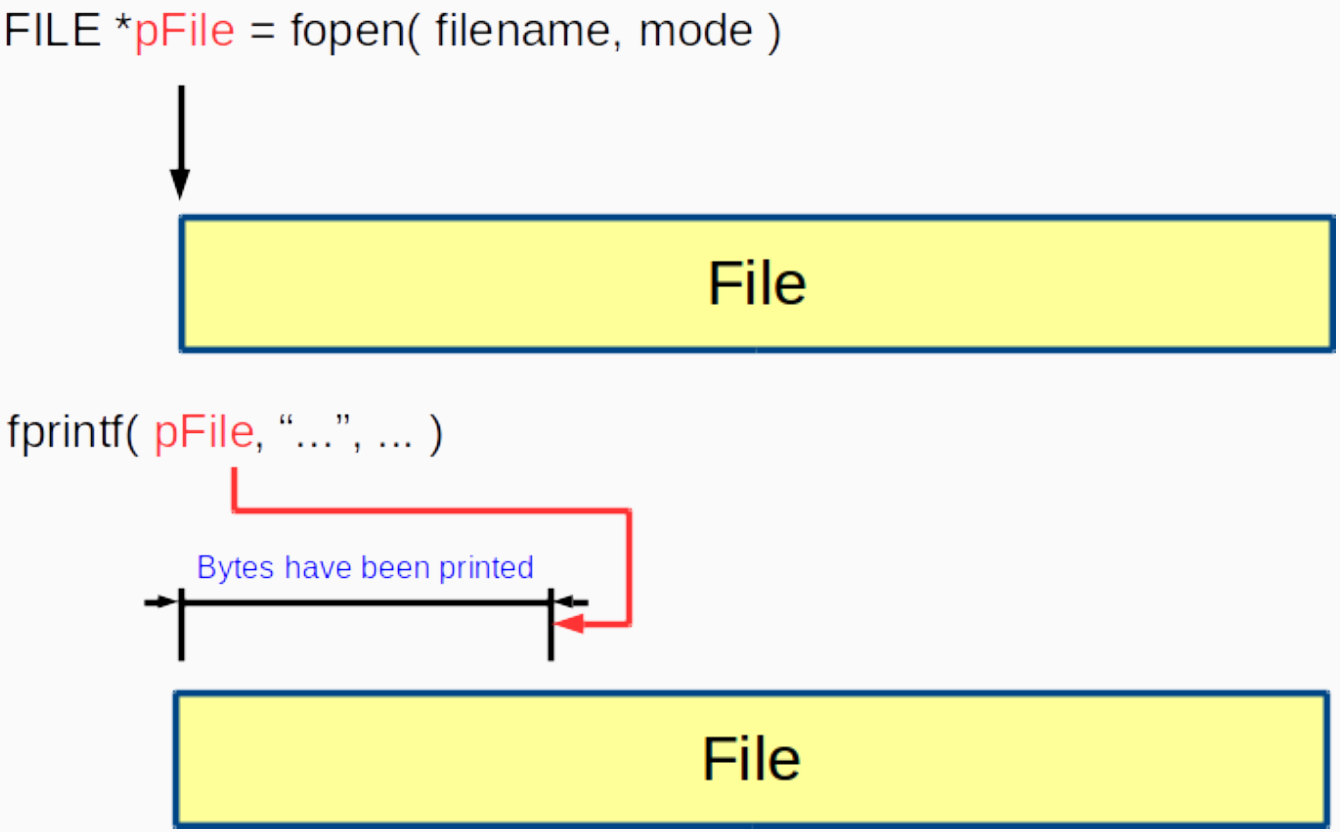
File Position Pointer

File Position Pointer



So how can we get back to the start point??

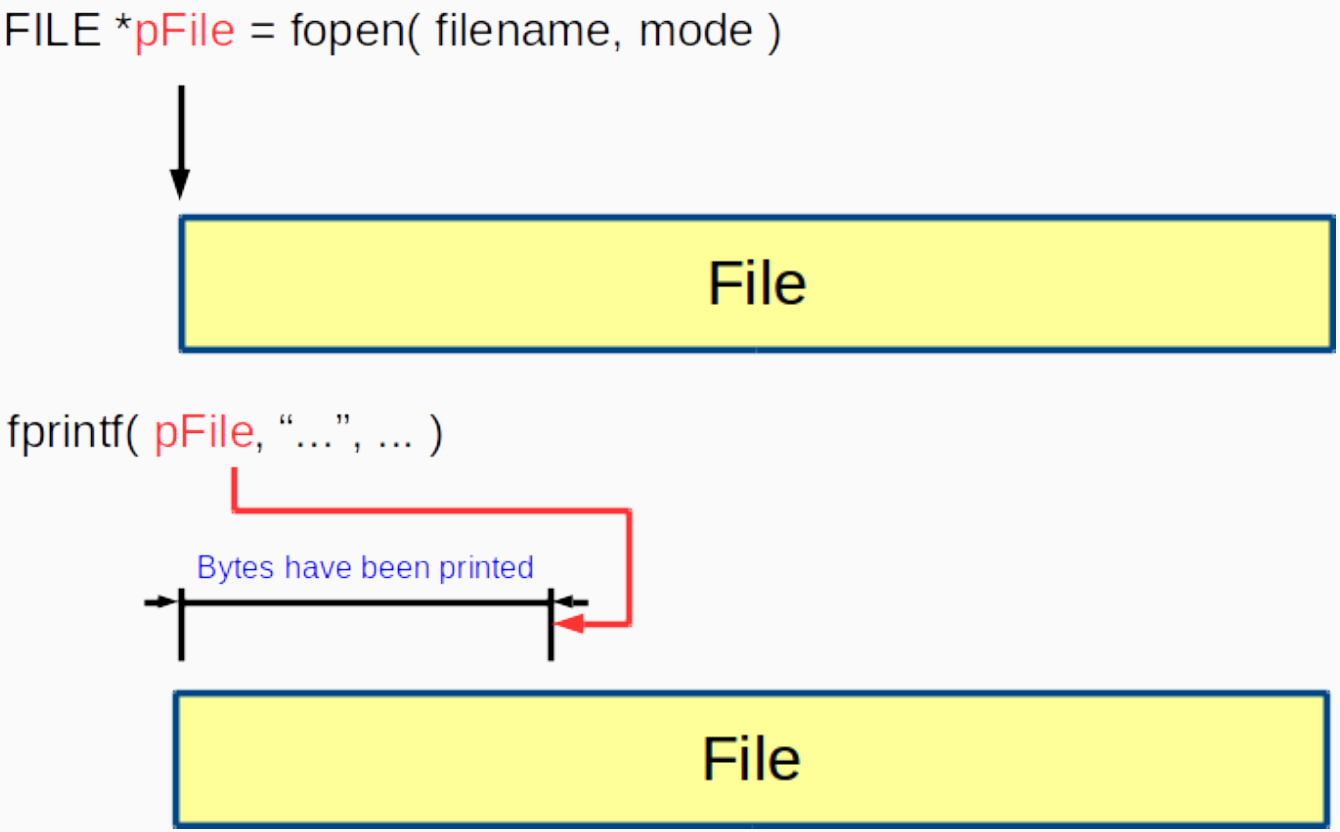
File Position Pointer



So how can we get back to the start point??

Why do we need this?

File Position Pointer



So how can we get back to the start point??

Why do we need this? Video Playback.

Resetting the File Position Pointer

- Of course, you can close the file and reopen it.
- Actually, we have another function that can help us.

void rewind(FILE *stream);

The rewind() function sets the file position indicator for the stream pointed to by stream to the beginning of the file.

Please see example.10.rewind

Can I Change the File Pointer to Anywhere?

Yes you can!

int fseek(FILE *stream, long offset, int whence);

The fseek() function sets the file position indicator for the stream pointed to by stream. The new position, measured in bytes, is obtained by adding offset bytes to the position specified by whence.

- **SEEK_SET**: the start of the file.
- **SEEK_CUR**: the current position indicator.
- **SEEK_END**: end-of-file.

.

Can I Change the File Pointer to Anywhere?

Yes you can!

int fseek(FILE *stream, long offset, int whence);

The fseek() function sets the file position indicator for the stream pointed to by stream. The new position, measured in bytes, is obtained by adding offset bytes to the position specified by whence.

- **SEEK_SET**: the start of the file.
- **SEEK_CUR**: the current position indicator.
- **SEEK_END**: end-of-file.

So rewind() is equal to fseek(stream, 0L, SEEK_SET).

Another Function

long ftell(FILE *stream);

The ftell() function obtains the current value of the file position indicator for the stream pointed to by stream.

Let's see example.10.fseek

Practice

Please open lyrics.txt and print each line in the reverse order.

Binary File Processing

Binary Files

- The text file stores the data in the form of characters.
- However, not all files are text files. For example, can you use your text editor to open a picture file?
- Besides, text file is not storage efficient.
  - How much space do you need to store the number **123**?
  - Text File:
  - Binary File:

Binary Files

- The text file stores the data in the form of characters.
- However, not all files are text files. For example, can you use your text editor to open a picture file?
- Besides, text file is not storage efficient.
  - How much space do you need to store the number **123**?
  - Text File: 3 bytes.
  - Binary File:

Binary Files

- The text file stores the data in the form of characters.
- However, not all files are text files. For example, can you use your text editor to open a picture file?
- Besides, text file is not storage efficient.
  - How much space do you need to store the number **123**?
  - Text File: 3 bytes.
  - Binary File: 1 byte.

fread, fwrite

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

The function fread() reads nmemb items of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.

On success, fread() returns **the number of items** read. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).

fread, fwrite

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

The function fread() reads nmemb items of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.

On success, fread() returns **the number of items** read. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).

Is there any difference between the following two code?

1. fread(ptr, 1, 100, pFile);
2. fread(ptr, 100, 1, pFile);

fwrite

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);

The function fwrite() writes nmemb items of data, each size bytes long, to the stream pointed to by stream, obtaining them from the location given by ptr.

On success, fwrite() returns **the number of items** written. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).

fwrite

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);

The function fwrite() writes nmemb items of data, each size bytes long, to the stream pointed to by stream, obtaining them from the location given by ptr.

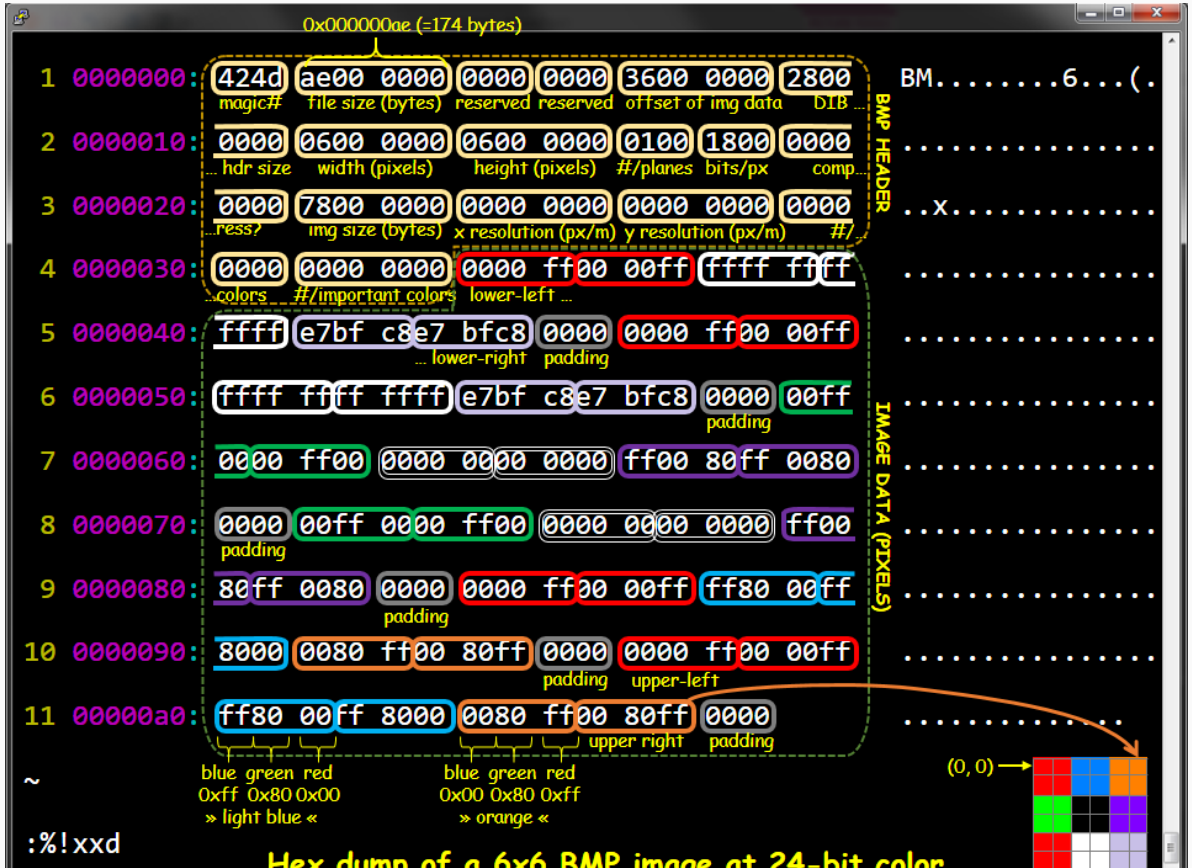
On success, fwrite() returns **the number of items** written. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero).

Is there any difference between the following two code?

1. fwrite(ptr, 1, 100, pFile);
2. fwrite(ptr, 100, 1, pFile);

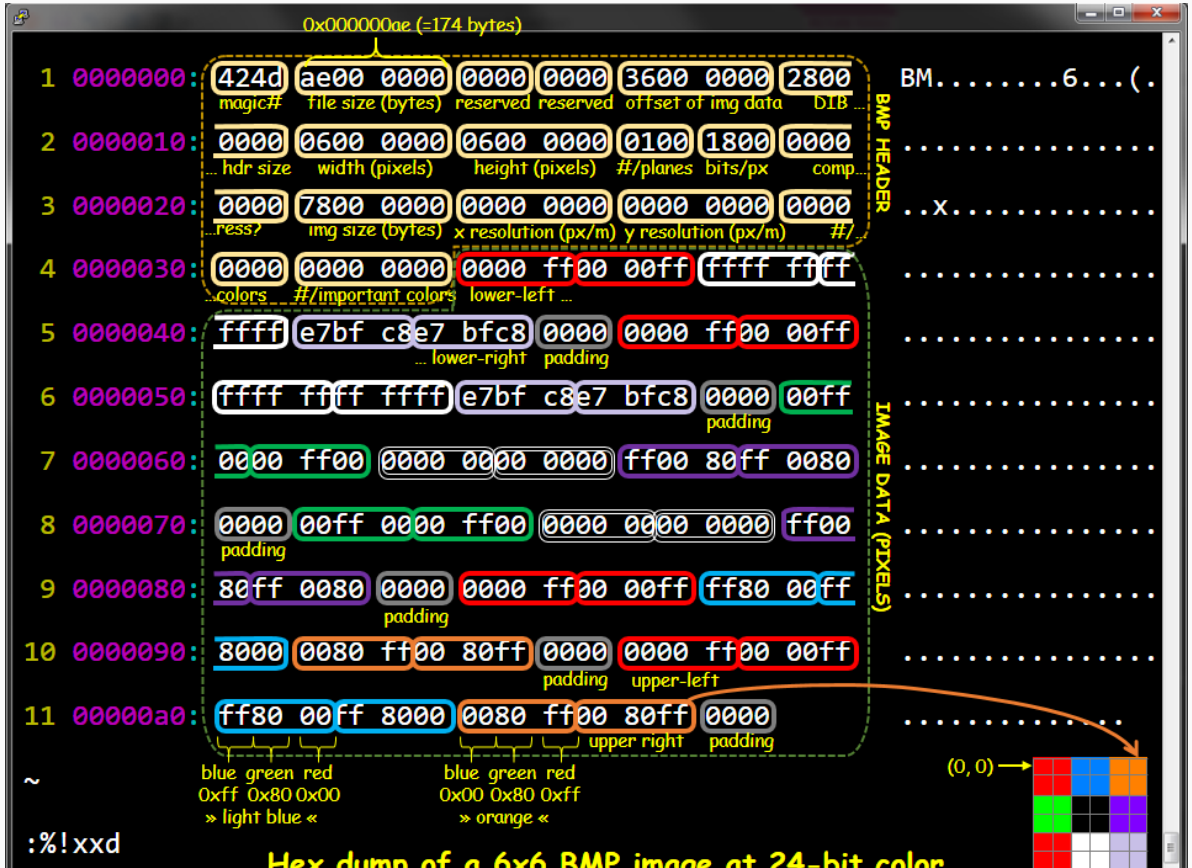
BMP File Format

- Here we will use BMP file as an example.
  - Why BMP, not JPG?



BMP File Format

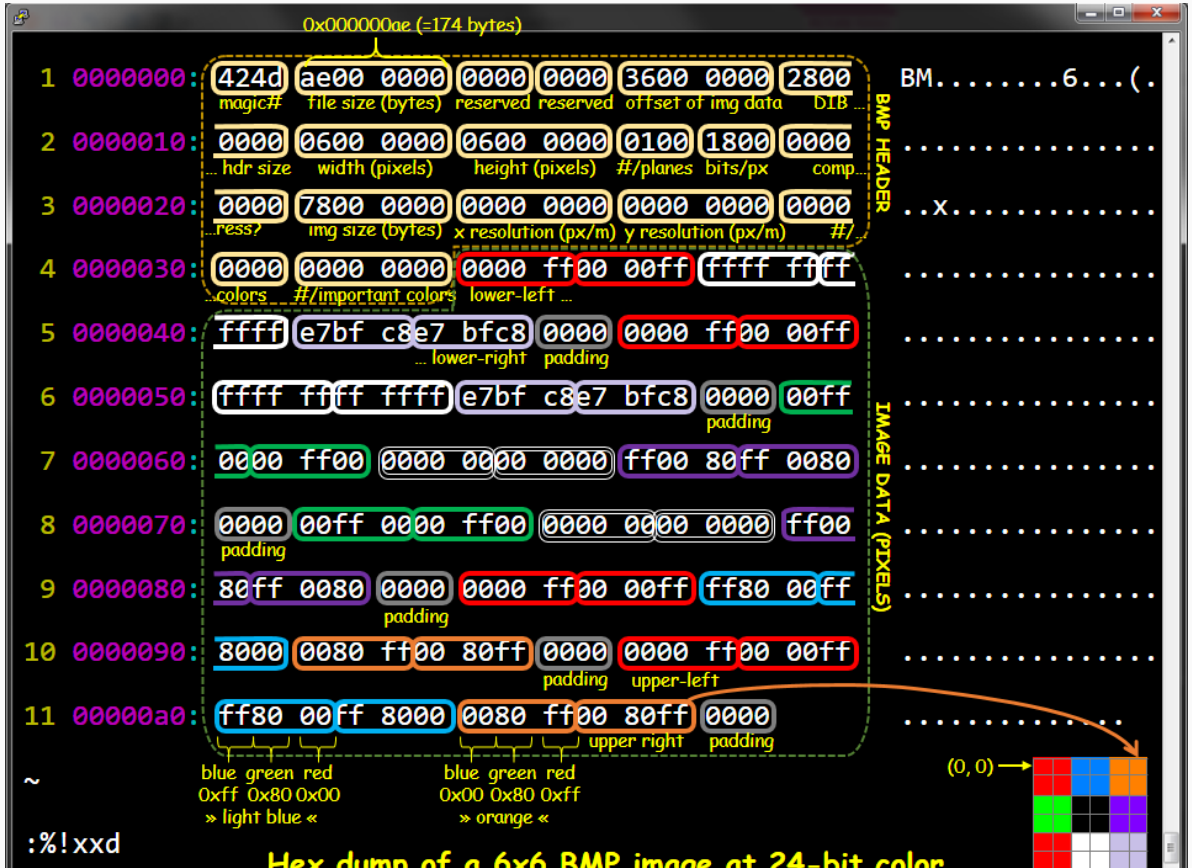
- Here we will use BMP file as an example.
  - Why BMP, not JPG? Because BMP is an uncompressed picture format.



BMP File Format

- Here we will use BMP file as an example.
  - Why BMP, not JPG? Because BMP is an uncompressed picture format.

`https://engineering.purdue.edu/ece264/17au/hw/HW15`



Example

Please see example.10.binary/bmp_read_v1.c

This program will parse the BMP file header and print information

- Actually, we should not write code like this.
- We often define a **structure** first. Then fread data of the structure size directly.
- There is an example code bmp_read_v2_error.c.

Quiz

Please build bmp_read_v2_error.c and run it. There are something wrong ... Why?

Quiz

Please build bmp_read_v2_error.c and run it. There are something wrong ... Why?

Because you forget __attribute__((packed))

Please see bmp_read_v2.c

Invert Colors

- Now we will invert colors of a BMP file.
- What does "invert colors" mean?
  - Invert the colors of image files, white becomes black, black becomes white, orange becomes blue and so on.
  - Technically speaking, for each 8-bit color, 255 - color.
- Please Please see bmp_write.c

Practice

Please write a program to transform a colorful BMP picture into a gray picture. Don't know how to do color transformation? I will tell you.

$$V = 0.299 \times R + 0.587 \times G + 0.114 \times B.$$

Then write V three times for one pixel. It means R, G, B are the same value.

How to Display a BMP file?

Before We Start

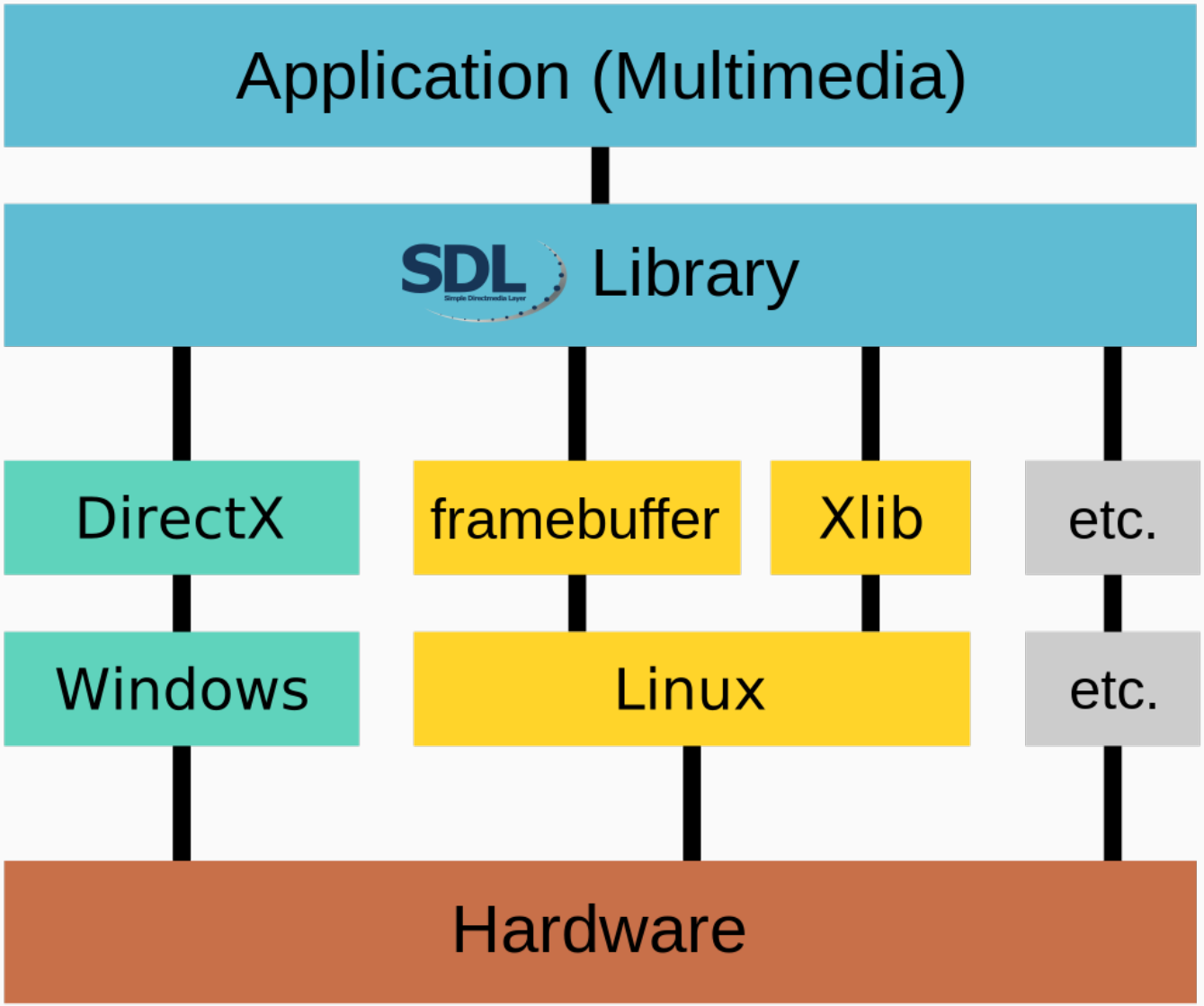
- There are lots of libraries that can help you do this. In fact, you should use them instead of developing from scratch.
- How to display an image on your screen?
  - You must learn at least one GUI library, like Gtk, Qt, and so on.
    - My suggestion: **Qt**.
  - Know how to display the image based on the function provided by the GUI platform.
- Unfortunately, I want you to display the image pixel by pixel.
  - I am a bad guy!!

Simple DirectMedia Layer

- SDL is a cross-platform software development library designed to provide a **hardware abstraction layer** for computer multimedia hardware components.
  - video
  - audio
  - input devices
  - CD-ROM
  - threads
  - shared object loading
  - networking
  - timers
  - 3D graphics

Software developers can use it to write high-performance computer games and other multimedia applications

SDL Architecture



SDL Example

SDL Object

- **SDL_Window**: the struct that holds all info about the Window itself: size, position, full screen, borders etc.
- **SDL_Renderer**: the struct that handles all rendering. It is tied to a SDL_Window so it can only render within that SDL_Window.
- **SDL_Textures** and **SDL_Surface**: The SDL_Renderer renders SDL_Texture, which stores the pixel information of one element. It's the new version of SDL_Surface which is much the same.
  - The important difference is that SDL_Surface uses software rendering (via CPU) while SDL_Texture uses hardware rendering (via GPU).

How to install??

sudo apt-get install libsdl2-dev libsdl2-image-dev

Note: I will not teach you all its functions. I only focus on the basic image processing part. However, if you are interested in it, you can study this yourself.

A Simple Example

Please see example.10.sdl.example.

I will not explain this code but I think you can read it and modify it yourself.

So I Know How to Display an Image

- You are right. Unfortunately, this is not allowed in this class. Why?
  - Again, I am a bad guy.
  - I want you to draw an image pixel by pixel.
- Please see example.10.sdl.bmp.example.
- What's wrong??
  1. Color issue.
  2. Flip.

File Descriptor

POSIX

- **Portable Operating System Interface**.
- A family of **standards** specified by theIEEE Computer Society for maintaining compatibility between operating systems.
  - Not by ISO/IEC.
    `http://www.open-std.org/jtc1/sc22/wg14/`
  - **Unix** was selected as the basis for a standard system interface.
  - Microsoft claims that it partially supports POSIX ...

POSIX

- **Portable Operating System Interface**.
- A family of **standards** specified by theIEEE Computer Society for maintaining compatibility between operating systems.
  - Not by ISO/IEC.
    `http://www.open-std.org/jtc1/sc22/wg14/`
  - **Unix** was selected as the basis for a standard system interface.
  - Microsoft claims that it partially supports POSIX ...

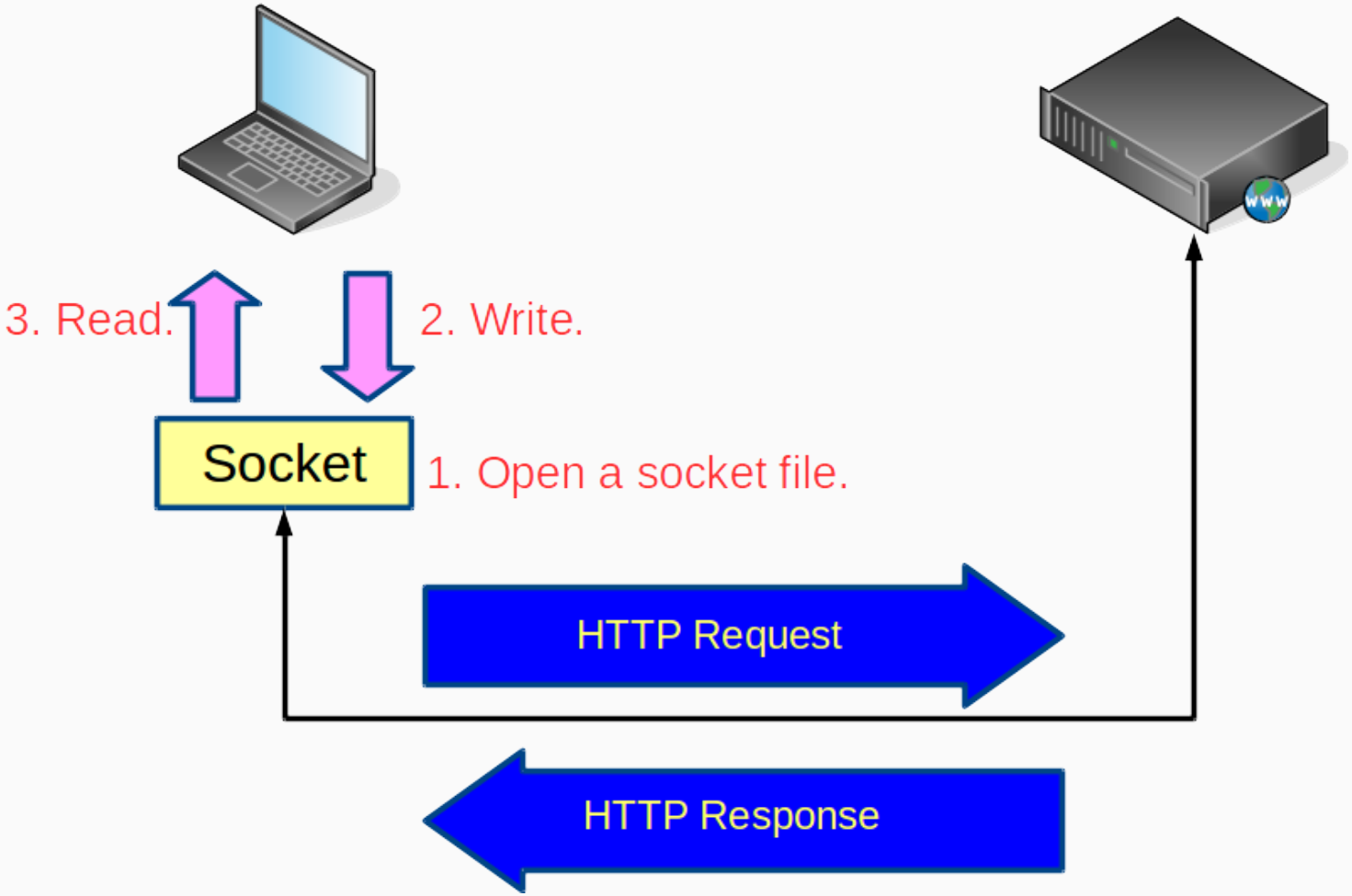
Why do I tell you these things?

Because from now on, what I teach you is not C standard but can only be applied in POSIX system.

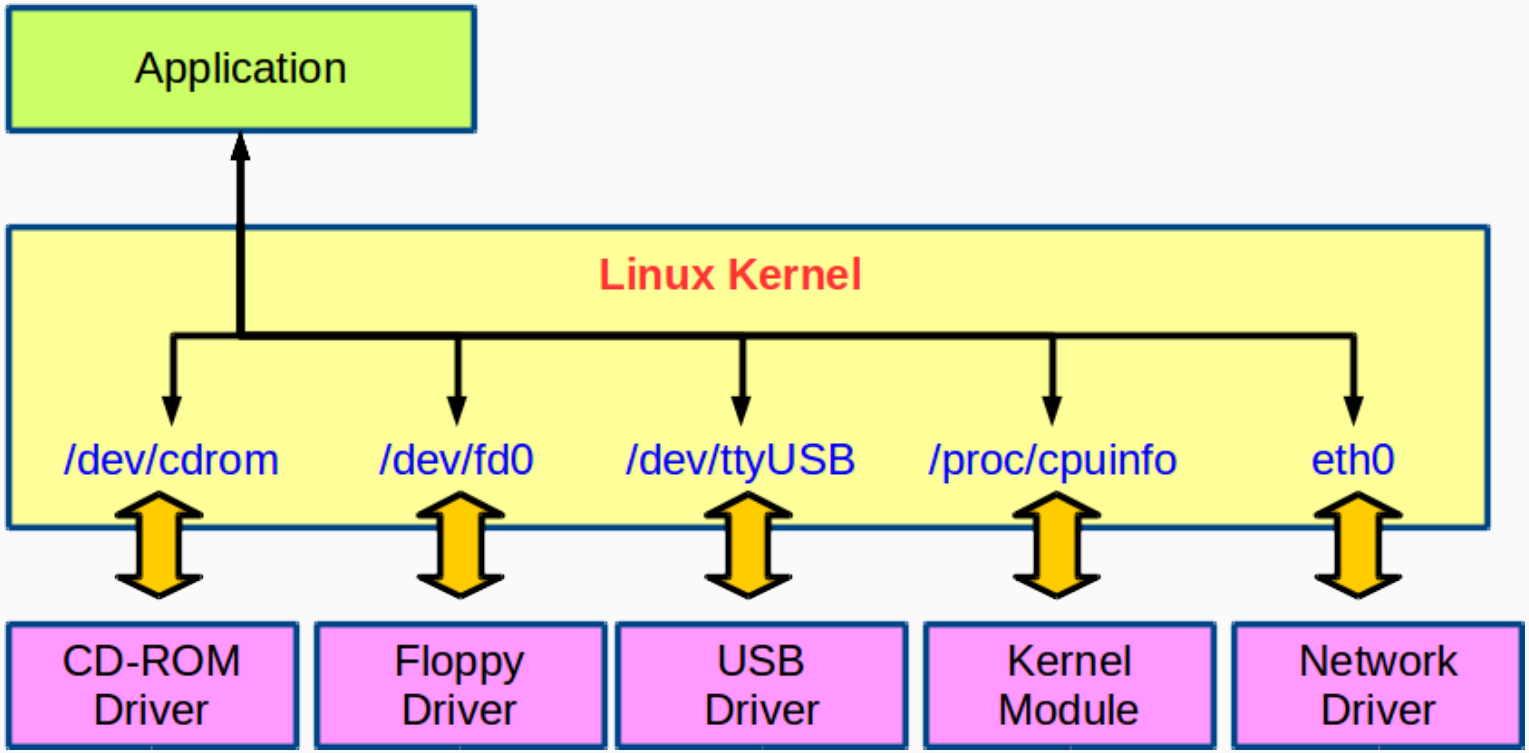
POSIX C Implementation

- C standard defines syntaxes and standard libraries without regulating implementation.
- So in Linux, **fopen**, **fclose**, **printf**, **scanf**, **fprintf**, **fscanf**, **fread** and **fwrite** are implemented through lower-level APIs:
  - open
  - read
  - write
  - close
- Now, we will use these lower-level API to access files.

In Linux, Everything is a File: Socket



In Linux, Everything is a File: Device Driver



So if you know how to use these lower-level APIs,
it will be very helpful.

Now we will use these APIs to access a file.

open, close

int open(const char *pathname, int flags);

Given a pathname for a file, open() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls. open() returns the new file descriptor, or -1 if an error occurred

The argument flags must include one of the following access modes: **O_RDONLY**, **O_WRONLY**, or **O_RDWR**.

int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused. close() returns zero on success. On error, -1 is returned, and errno is set appropriately.

For more flags, please read manual.

read, write

ssize_t read(int fd, void *buf, size_t count);

read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf. On files that support seeking, the read operation commences at the current file offset, and the file offset is incremented by the number of bytes read. If the current file offset is at or past the end of file, no bytes are read, and read() returns zero.

ssize_t write(int fd, const void *buf, size_t count);

write() writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd.

read, write

ssize_t read(int fd, void *buf, size_t count);

read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf. On files that support seeking, the read operation commences at the current file offset, and the file offset is incremented by the number of bytes read. If the current file offset is at or past the end of file, no bytes are read, and read() returns zero.

ssize_t write(int fd, const void *buf, size_t count);

write() writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd.

How about seek? use **lseek**.

Example

Please see example.10.fd/mycopy.c

We Have Some More Tools

int fstat(int fd, struct stat *buf);

These functions return information about a file, in the buffer pointed to by buf.

- Q: What is "struct stat"?
- A: You can read manual.

Example

Please see example.10.fd/file_size_v2.c

Which one do you prefer?

mmap

What if I want to read a LARGE file?

- How to read a file?
  1. Create a buffer.
  2. Open a file.
  3. Read data to the buffer many many times till EOF.
  4. Close the file.
- I think it is too annoying ...

OK, What Do You Want?

- I dream that someone will load a file into a **BIG ARRAY** so that I do not need to read a file many many times.
- How about that?
  1. Get file size first.
  2. Allocate a memory block with that size.
  3. Read file data into the memory.

OK, What Do You Want?

- I dream that someone will load a file into a **BIG ARRAY** so that I do not need to read a file many many times.
- How about that?
  1. Get file size first.
  2. Allocate a memory block with that size.
  3. Read file data into the memory. I am a Genius!

OK, What Do You Want?

- I dream that someone will load a file into a **BIG ARRAY** so that I do not need to read a file many many times.
- How about that?
  1. Get file size first.
  2. Allocate a memory block with that size.
  3. Read file data into the memory. I am a Genius!
- How about a blueray movie?

Dream Comes True

- Now you can use mmap.

void *mmap(void *addr, size_t length, int prot, int flags,int fd, off_t offset);

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in addr. The length argument specifies the length of the mapping.

If addr is NULL, then the kernel chooses the address at which to create the mapping.

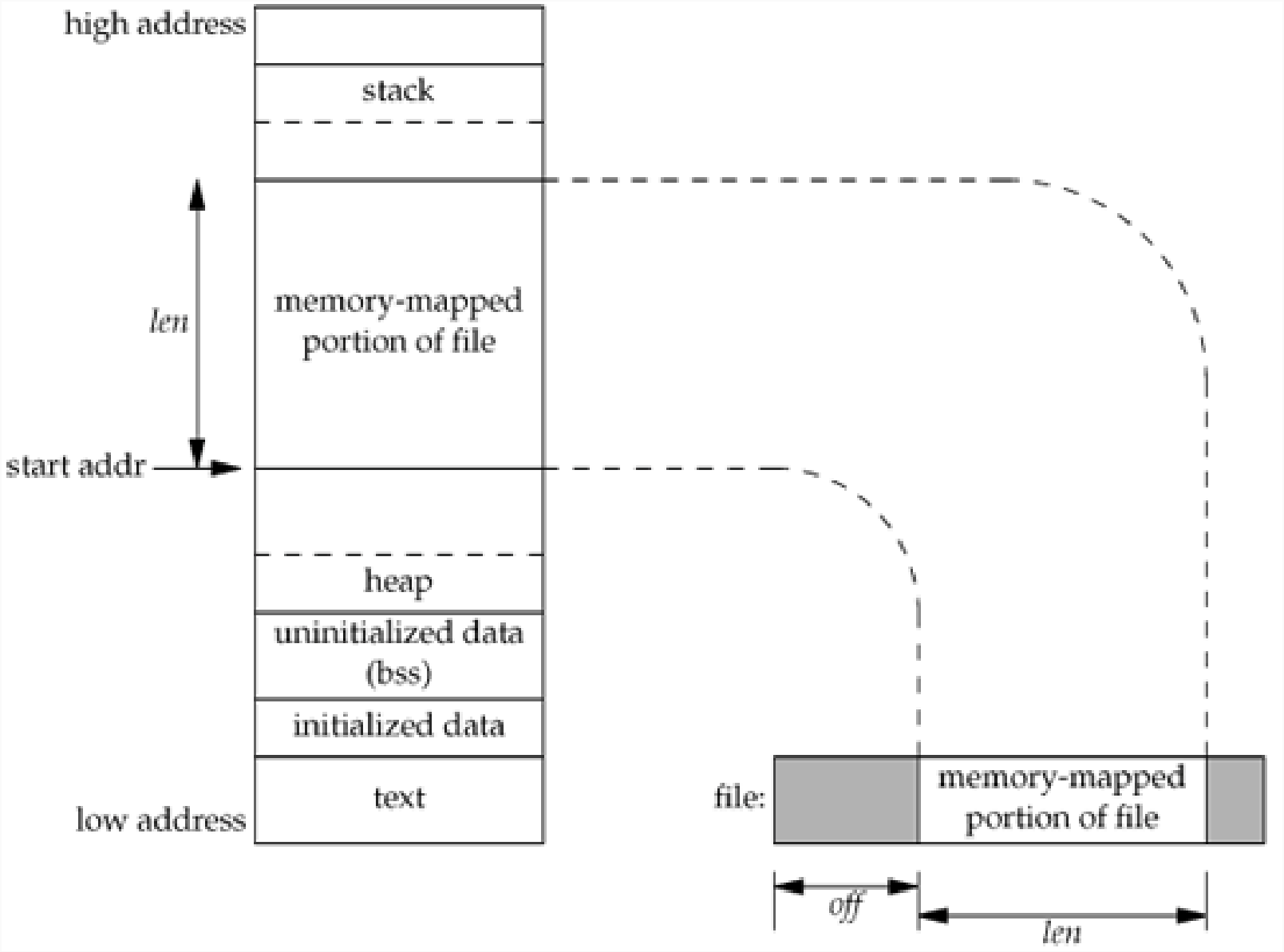
Now you can treat the return address as a BIG array and you can directly access data, including modifying values.

Example

Please see example.10.mmap .

This is an example upper case and lower case conversion.

mmap


high address
stack
len
memory-mapped
portion of file
start addr
heap
uninitialized data
(bss)
initialized data
text
low address
file:
memory-mapped
portion of file
off
len

IMPORTANT

After using mmap, please remember to munmap.

Note

- **mmap()** is not a standard C function.
- **MapViewOfFile** function is somewhat equivalent to mmap in Windows.

Practice

- Please write a data hiding program.
- You should let a user to input a string and embed the string to a picture.
- For example, if I want to embed a string "hello", the ascii code is

'h'	'e'	'l'	'l'	'o'
0x68	0x65	0x6C	0x6C	0x6F
0110 1000	0110 0101	0110 1100	0110 1100	0110 1111

- For each color byte, replace the least significant bit with the secret string bit.
- Example: Red 0000FF
  - 0000 0000 $\rightarrow$ 0000 0000
  - 0000 0000 $\rightarrow$ 0000 0001
  - 1111 1111 $\rightarrow$ 1111 1111