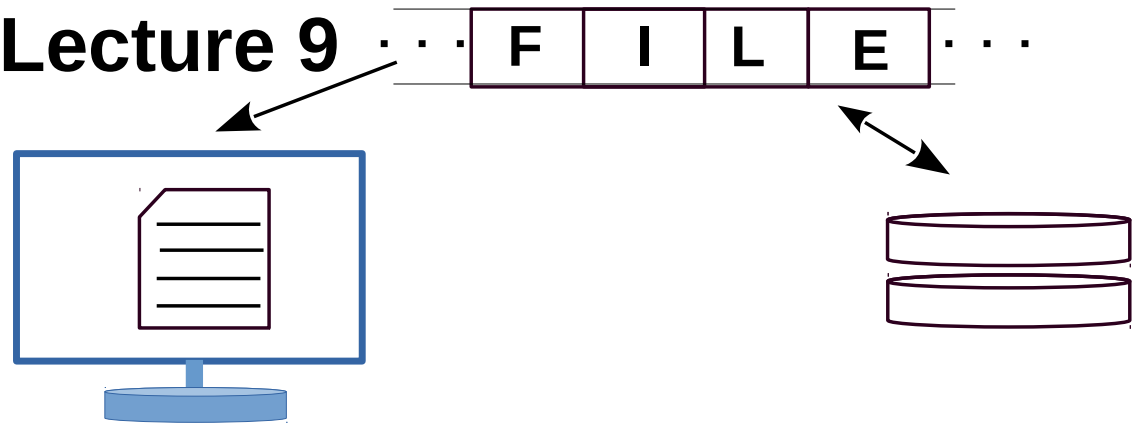
C Programming

Lecture 9



Lecturer: *Dr*. Wan-Lei Zhao

Spring Semester 2022

Email: wlzhao@xmu.edu.cn,

Outline

Overview

- When we doing programming
- Or run our code
- Two main components of a computer we work with
  1. CPU: where our codes are executed
  2. Memory: where our codes are temporarily kept
- When computer is switched off
- Everything resides in memory disappear
- External storage is where we can keep things permanently

Storage Device (1)



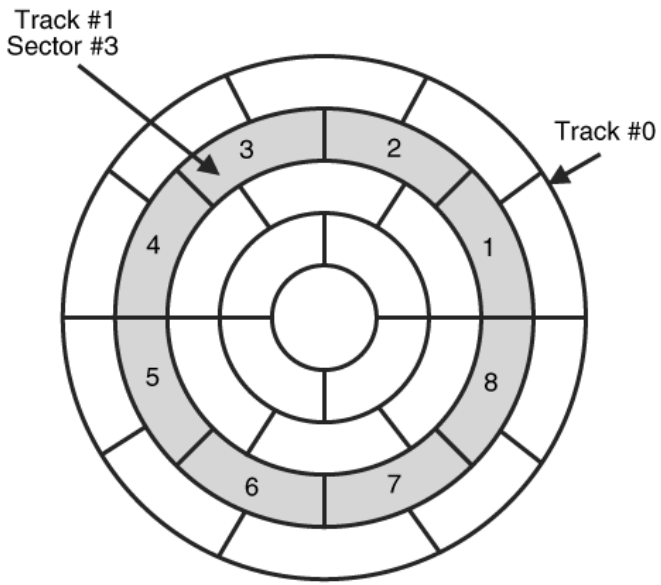




- There are all kinds of storage devices in the history of computer
- Currently, hard disc (it is hard), DVD and USB are popular
- Things are kept there in the unit of **file**

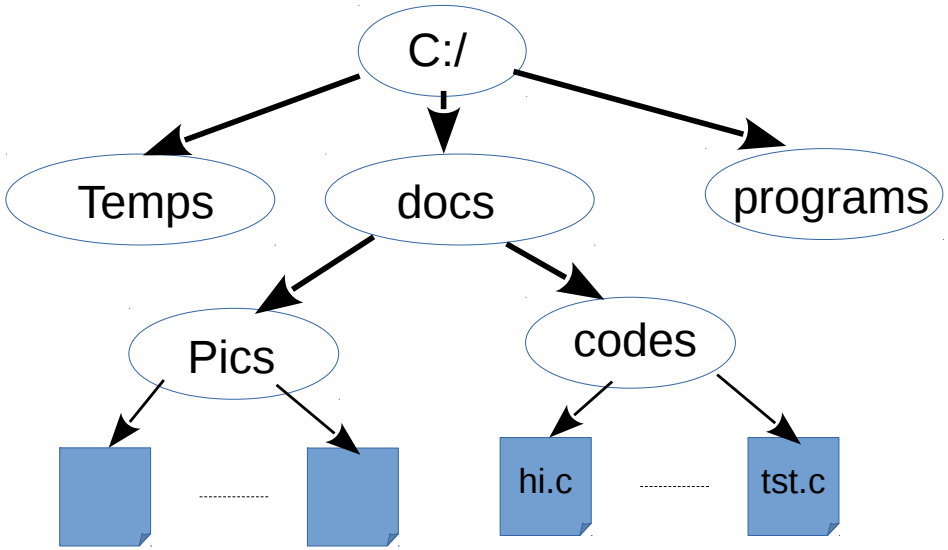
Storage Device (2)

- There are several basic thing should be kept for a file
  1. File name
  2. File size
  3. Location in the disc: which cylinder, which track and which sector?
  4. Time the file created, updated and the last time file is read
  5. Anything else??



Storage Device (3)

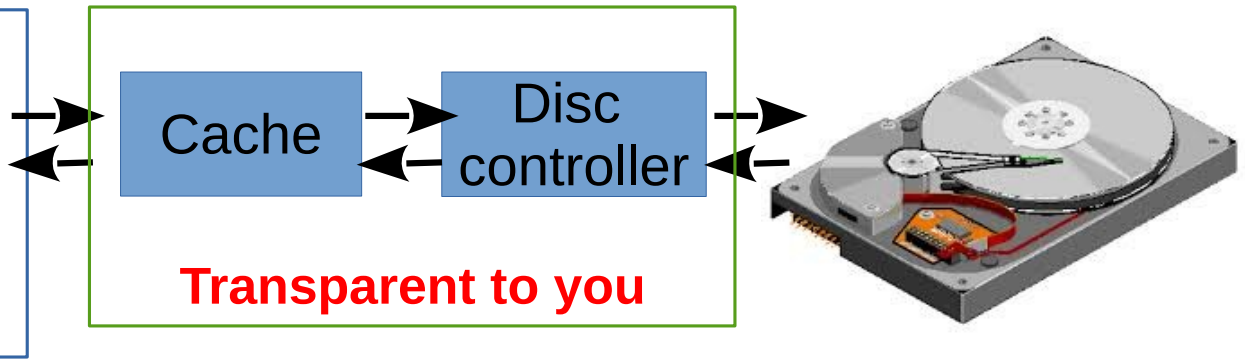
- There are several basic thing should be kept for a file
  1. File name
  2. File size
  3. Location in the disc: which cylinder, which track and which sector?
  4. Time the file created, updated and the last time file is read
  5. Logic location: full path of the file: "c:/docs/codes/hi.c"
- We use a struct type to define such kind of information



Outline

Flow of File Operation in C

```
#include <stdio.h>
int main()
{
   FILE *fp = fopen("hi.c", "r");
    ...
   return 0;
}
```



- When file is open (for write/read)
- A cache is created in the memory
- Data are put to cache from either side first
- Transfer to another side later

Read File in C (1)

- A file name (with path) should be given
  - full path: "c:/docs/codes/hi.c"
  - relative path: "../codes/hi.c"
- Tell the system, you are going to read ("r") the file

```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   int a = 0, b = 0;
   FILE *fp = fopen(fn, "r");
   if(fp == NULL)//in case that open file failed
   { //required all the time
       printf("Open file %d\n failed", fn);
       return 0;
   }
   fscanf(fp, "%d%d", &a, &b);
   fclose(fp); //This is required all the time
   printf("%d %d\n", a, b);
   return 0;
}
```

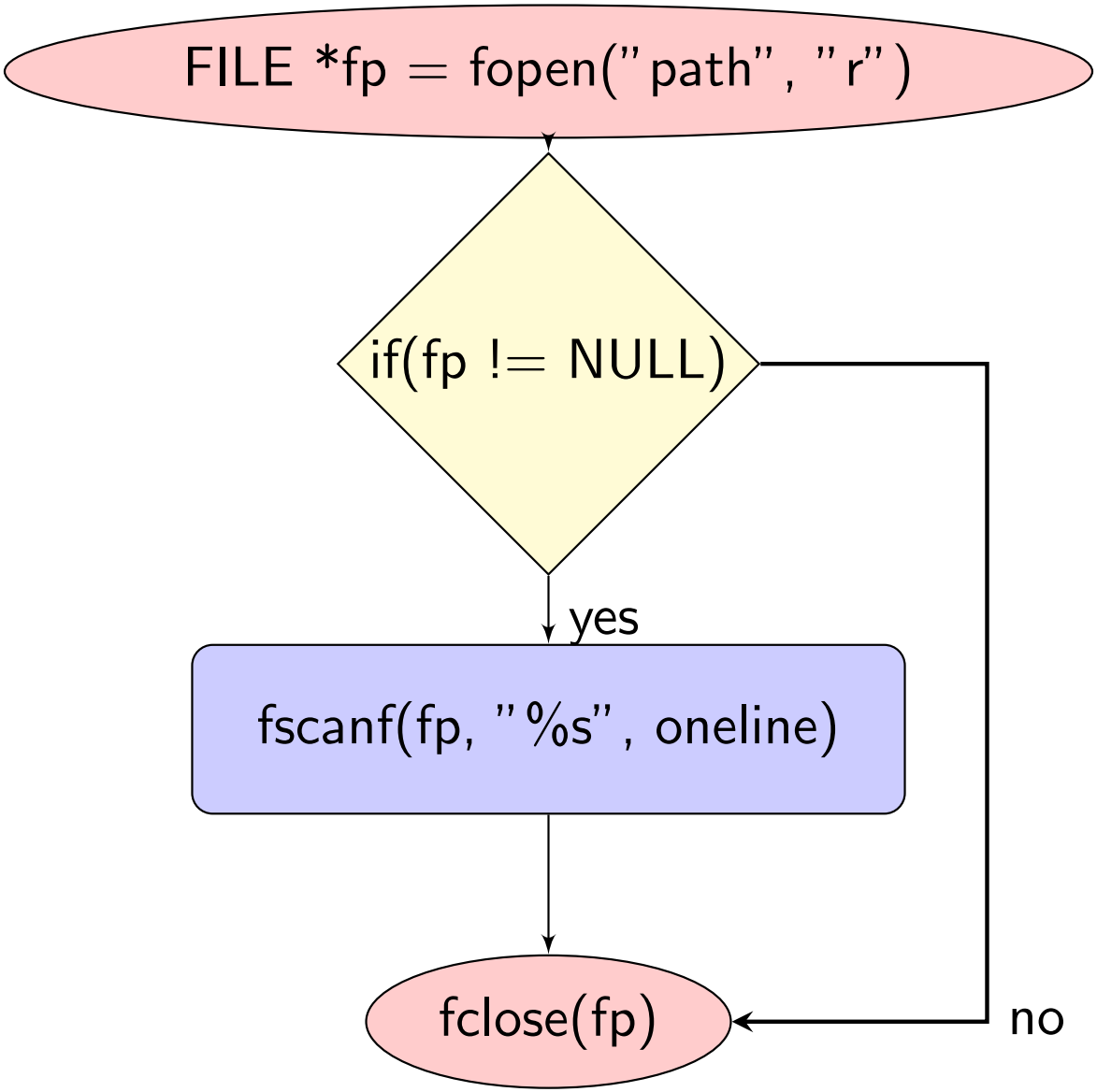
FILE struct type

```c
#include <stdio.h>
int main()
{
   FILE *fp = fopen(fn, "r");
}
```

```c
typedef struct
{
 short level;
 short token;
 short bsize;
 char fd;
 unsigned flags;
 unsigned char hold;
 unsigned char *buffer;
 unsigned char * curp;
 unsigned istemp;
}FILE;
```

- "FILE" is a struct designed for file operation
- It is defined in header "<stdio.h>"

Read File in C (2): the flow

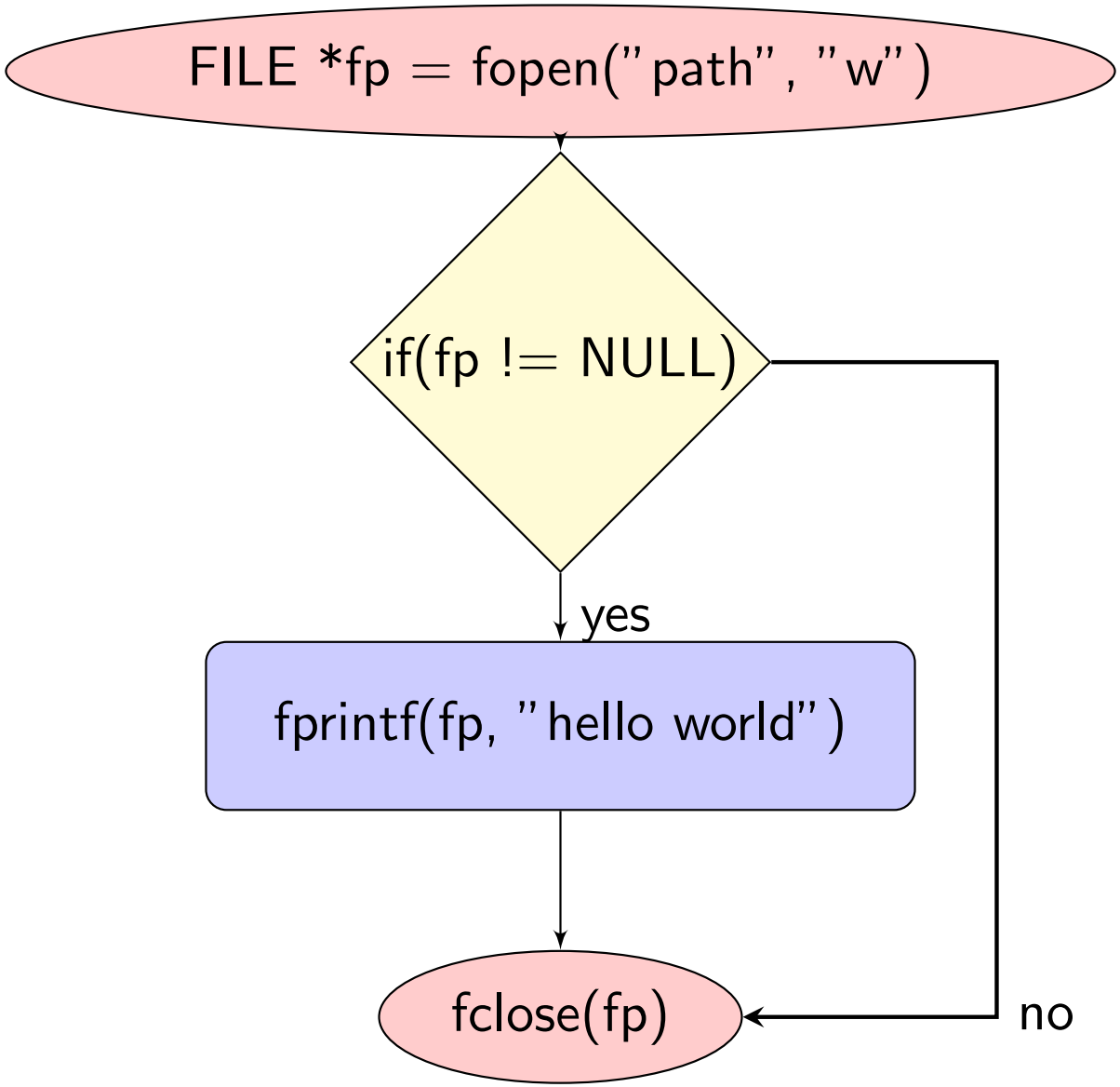


Read File in C (3)

```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   char line[1024];
   FILE *fp = fopen(fn, "r");
   if(fp == NULL){
       printf("Open file %d\n failed", fn);
       return ;
   }
   fscanf(fp, "%s", line);
   fclose(fp); //This is required all the time
   printf("%s", line);
}
```

- It is possible that file does not exist
- Checking whether "fp" is NULL is safe to handle exceptions
- Call "fclose" all the time when you finish reading
- The **cache** will be always there

Read File in C (1): the flow



Write File in C (2)

- A file name (with path) should be given
  - full path: "c:/docs/codes/hi.c"
  - relative path: "../codes/hi.c"
- Tell the system, you are going to write ("w") the file

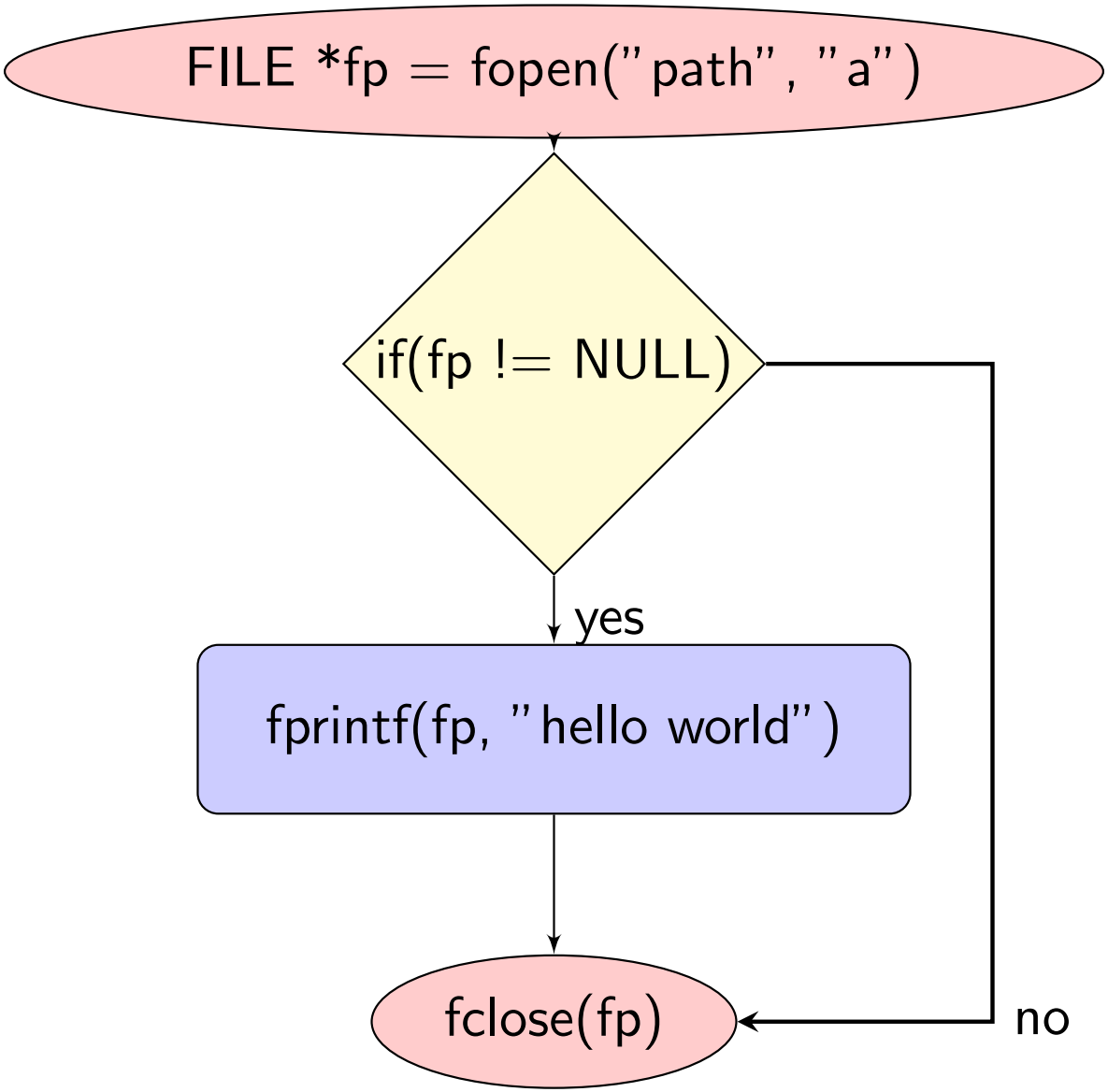
```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   char line[1024];
   FILE *fp = fopen(fn, "w");
   if(fp == NULL)//in case that open file failed
   { //required all the time
       printf("Open file %d\n failed", fn);
       return 0;
   }
   fprintf(fp, "Hello world %d\n", 1);
   fclose(fp);
   return 0;
}
```

Write File in C (3)

```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   char line[1024];
   FILE *fp = fopen(fn, "w");
   if(fp == NULL){
      printf("Open file %d\n failed", fn);
      return ;
   }
   fprintf(fp, "Hello world %d\n", 1);
   fclose(fp);
}
```

- It is possible that file does not exist
- Checking whether "fp" is NULL is safe to handle exceptions
- Call "fclose" all the time when you finish writing
- Otherwise no one can write/read the file

Append File in C (1): the flow



Append File in C (2)

- You allowed to put something more on the end of an existing file
- Tell the system, you are going to append ("a") the file

```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   char line[1024];
   FILE *fp = fopen(fn, "a");
   if(fp == NULL)//in case that open file failed
   { //required all the time
       printf("Open file %d\n failed", fn);
       return 0;
   }
   fprintf(fp, "Hello world %d\n", 2);
   fclose(fp); //This is required all the time
   return 0;
}
```

Append File in C (3)

```c
#include <stdio.h>
int main()
{
   const char *fn = "c:/docs/codes/hi.txt";
   char line[1024];
   FILE *fp = fopen(fn, "a");
   if(fp == NULL)//in case that open file failed
   { //required all the time
       printf("Open file %d\n failed", fn);
       return 0;
   }
   fprintf(fp, "Hello world %d\n", 2);
   fclose(fp); //This is required all the time
   return 0;
}
```

- If the file does not exist
- It will be created

Append File in C (3): the result

```
1 Hello world 1
2 Hello world 2
```

Summary on File Operation

Operation	function	instruction
Open	fopen(const char *fname, "r|w|a")	
Read	fscanf(FILE *fp, char *buffer)	"r"
Write	fprintf(FILE *fp, "format string")	"w"
Append	fprintf(FILE *fp, "format string")	"a"
Close	fclose(FILE *fp)	

- Open and close operations are always required
- All above functions are defined in "<stdio.h>"