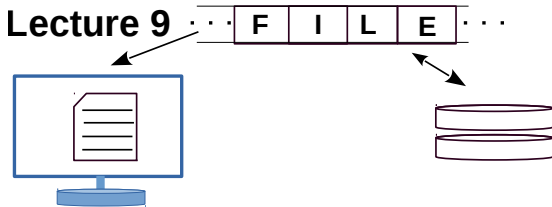


# C Programming

## Lecture 9



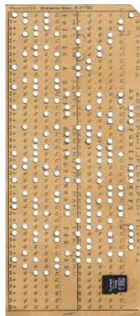
Lecturer: *Dr. Wan-Lei Zhao*  
*Spring Semester 2022*

1 Overview about Computer Storage

2 Operation on Files

- 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

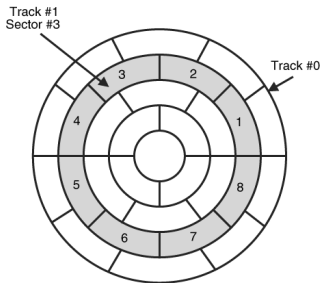
# Storage Device (1)



- 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**

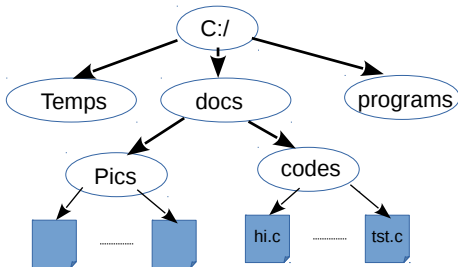
# Storage Device (2)

- 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



1 Overview about Computer Storage

2 Operation on Files

# Flow of File Operation in C

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

```
int main()
```

```
{  
    FILE *fp = fopen("hi.c", "r");
```

```
    ...  
    return 0;
```

```
}
```



Cache

Disc controller



Transparent to you

- 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

```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     int a = 0, b = 0;
6     FILE *fp = fopen(fn, "r");
7     if(fp == NULL) // in case that open file failed
8     { // required all the time
9         printf("Open file %d\n failed", fn);
10        return 0;
11    }
12    fscanf(fp, "%d%d", &a, &b);
13    fclose(fp); // This is required all the time
14    printf("%d %d\n", a, b);
15    return 0;
16 }
```

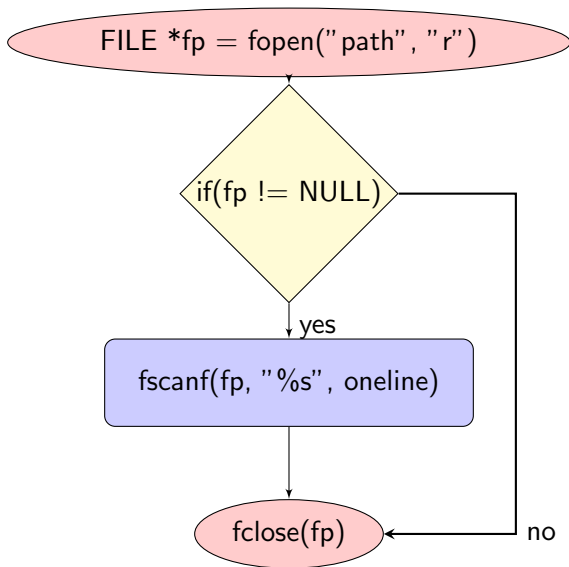
# FILE struct type

```
1 #include <stdio.h>
2 int main()
3 {
4     FILE *fp = fopen(fn, "r");
5 }
```

```
1 typedef struct
2 {
3     short level;
4     short token;
5     short bsize;
6     char fd;
7     unsigned flags;
8     unsigned char hold;
9     unsigned char *buffer;
10    unsigned char *curp;
11    unsigned istemp;
12 } FILE;
```

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

## Read File in C (2): the flow

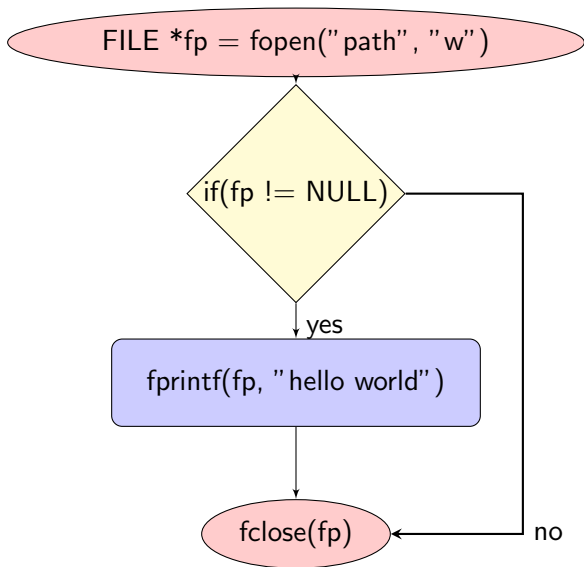


## Read File in C (3)

```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     char line[1024];
6     FILE *fp = fopen(fn, "r");
7     if(fp == NULL){
8         printf("Open file %d\n failed", fn);
9         return ;
10    }
11    fscanf(fp, "%s", line);
12    fclose(fp); //This is required all the time
13    printf("%s", line);
14 }
```

- 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

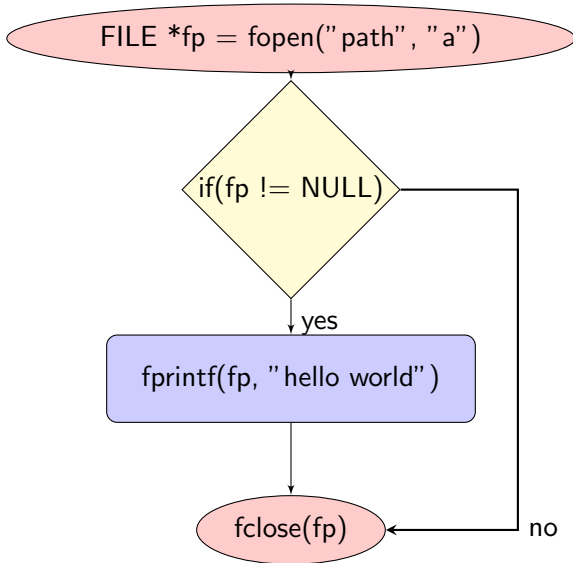
```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     char line[1024];
6     FILE *fp = fopen(fn, "w");
7     if(fp == NULL) // in case that open file failed
8     { // required all the time
9         printf("Open file %d\n failed", fn);
10        return 0;
11    }
12    fprintf(fp, "Hello world %d\n", 1);
13    fclose(fp);
14    return 0;
15 }
```

# Write File in C (3)

```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     char line[1024];
6     FILE *fp = fopen(fn, "w");
7     if(fp == NULL){
8         printf("Open file %d\n failed", fn);
9         return ;
10    }
11    fprintf(fp, "Hello world %d\n", 1);
12    fclose(fp);
13 }
```

- 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

```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     char line[1024];
6     FILE *fp = fopen(fn, "a");
7     if(fp == NULL) //in case that open file failed
8     { //required all the time
9         printf("Open file %d\n failed", fn);
10        return 0;
11    }
12    fprintf(fp, "Hello world %d\n", 2);
13    fclose(fp); //This is required all the time
14    return 0;
15 }
```

# Append File in C (3)

```
1 #include <stdio.h>
2 int main()
3 {
4     const char *fn = "c:/docs/codes/hi.txt";
5     char line[1024];
6     FILE *fp = fopen(fn, "a");
7     if(fp == NULL) //in case that open file failed
8     { //required all the time
9         printf("Open file %d\n failed", fn);
10        return 0;
11    }
12    fprintf(fp, "Hello world %d\n", 2);
13    fclose(fp); //This is required all the time
14    return 0;
15 }
```

- 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	<code>fopen(const char *fname, "r w a")</code>	
Read	<code>fscanf(FILE *fp, char *buffer)</code>	"r"
Write	<code>fprintf(FILE *fp, "format string")</code>	"w"
Append	<code>fprintf(FILE *fp, "format string")</code>	"a"
Close	<code>fclose(FILE *fp)</code>	

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

**Thanks for your commitment  
and support!**

Best wishes to your study and  
future career!