# C Programming

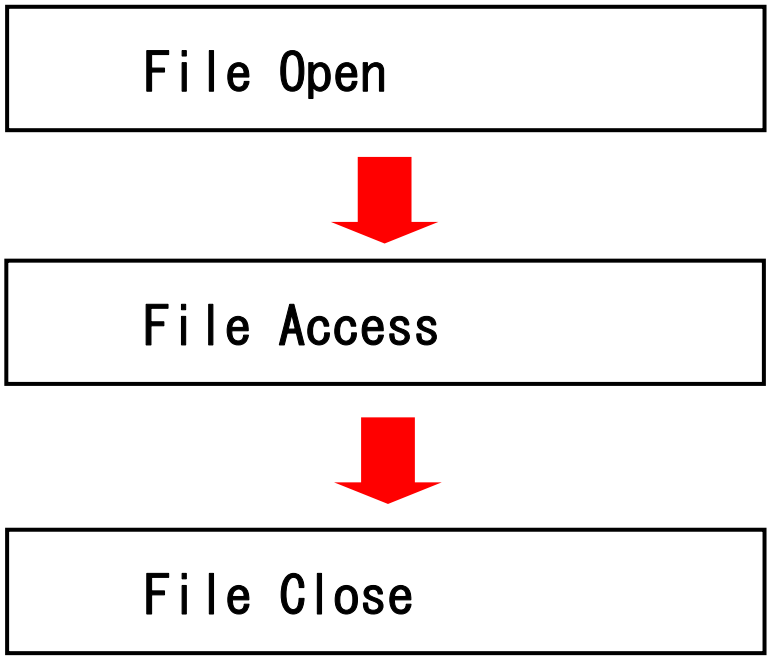
## — File Input & Output—

# Outline

- File Input & Output
  - File Open
  - File Close
  - File Input & Output
- Command Line and Arguments
- Excercise

# File Input & Output

Procedure




"fopen()"

File Input & Output

"fclose()"

# File Access

- File pointer points to a structure that contains information about the file

- Declaration :

```
FILE *name of file pointer;
```

Example: FILE *fp;
FILE *fin, *fout;

- To open 「sample.txt」

```
fp = fopen("sample.txt", "r");
```

fp is the pointer to a FILE
fopen returns a pointer to a FILE

# File Open

fopen(*filename*, *"mode"*)

- *filename*:a character string containing the name of file.
- mode:a character string indicates how one intends to use the file
- NULL:a pointer to show the result of file access.
- when file open is failed, fp== NULL.

Ex fp = fopen("sample.txt", "r");
fp = fopen(file, "w");

- *mode* :

| | |
|---|---|
| r | read<br>error(if no such file exits) |
| w | write<br>re-write(if the file exits),otherwise,creat a new file |
| a | add<br>add after the end position(if the file exits),otherwise,creat a new file |

# File Open

If file open is failed, the program will be stopped

Ex

```
#include <stdio.h>
#include <stdlib.h>                          ← exit()is neccesary

int main(void)
{
    FILE *fp;

    fp = fopen("file1.c", "r");              ← o p e n 「file1.c」
    if (fp == NULL) {                        ← fp=NULL: open failed
        printf(Cannot open the file¥n);
        exit(1);                             ← stop the program
    }

    ...
}
```

# File Close

fclose(*fp*)

- close the file pointed by fp.
- Successful:0;Failed: EOF.

Ex

```
int main(void)
{
    FILE *fp;

    fp = fopen("file1.c", "r");
    …

    fclose(fp);

    return 0;
}
```

# File Input & Output

there are lots of functions：

fgetc(*fp*)

- Get one character from the file (pointed by fp) and read as an int.
- Return EOF to indicate the end of file.
- Ex:ch = fgetc(fp);

fputc(*ch*, *fp*)

- Write one character (ch) to the file (pointed by fp).

# File Input & Output

fgets(*str*, *size*, *fp*)

- Read one line from the file (pointed by fp) and store it in str.
- the maximum number of charactors in one line is decided by *size* .
- Return NULL to indicate the end of file.
- Ex:fgets(str, 256, fp);

fputs(*str*, *fp* )

- Put one line of charactors (str) to the file (pointed by fp).

# File Input & Output

fscanf(*fp, format, arguments*)

- Read contents of the file (pointed by fp) with format and arguments.
- *format, arguments* are the same as scanf().
- Ex:fscanf(fp, "%d", &a);

fprintf(*fp,format, arguments*)

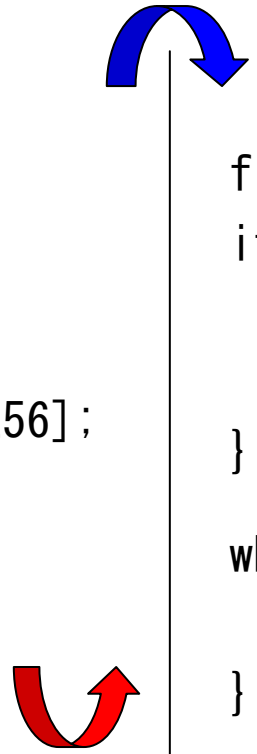
- Write contents to the file (pointed by fp) with format and arguments.
- *format, arguments* are the same as printf().
- Ex:fprintf(fp, "%d¥n", &a);

# File Input & Output

Output the contents of the program to the screen :

file_io1.c

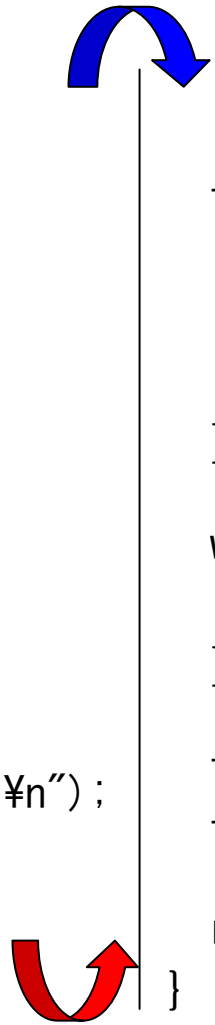
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  FILE *fin;
  char filename[40], str[256];

  printf("File name : ");
  scanf("%s", filename);

  fin = fopen(filename, "r");
  if (fin == NULL) {
    printf("Cannot open the file.¥n");
    exit(1);
  }

  while (fgets(str, 256, fin) != NULL){
    printf("%s", str);
  }

  fclose(fin);

  return 0;
}
```



Copy one program to another program :

file_io2.c

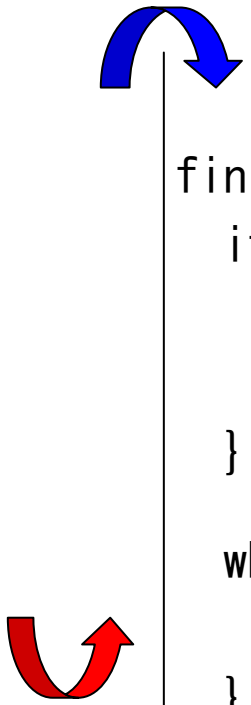
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  FILE *fin, *fout;
  char infile[40], outfile[40];
  char str[256];

  printf("input file : ");
  scanf("%s", infile);
  printf("output file : ");
  scanf("%s", outfile);

  fin = fopen(infile, "r");
  if (fin == NULL) {
    printf("Cannot open input file.¥n");
    exit(1);
  }

  fout = fopen(outfile, "w");
  if (fout == NULL) {
    printf("Cannot open output file.¥n");
    exit(1);
  }

  while (fgets(str, 256, fin) != NULL) {
    fputs(str, fout);
  }

  fclose(fin);
  fclose(fout);

  return 0;
}
```



Output the contents of 「file_io1.c」 to the screen by using fgetc :

file_io3.c

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

int main(void)
{
  FILE *fin;
  char filename[40];
  char ch;

  printf("File name : ");
  scanf("%s", filename);

  fin = fopen(filename, "r");
  if (fin == NULL) {
    printf("Cannot open the file.¥n");
    exit(1);
  }

  while ((ch = fgetc(fin)) != EOF) {
    printf("%c", ch);
  }

  fclose(fin);

  return 0;
}
```



# Command Line and Arguments

Up to now ,all programs are executed as follows :

```
$ ./file_io1↵
File name : filename ↵
...
```

Generally, arguments are wanted to be input from the command line directly.

```
$ ./file_io1 filename ↵
...
```

⇒ how to do?

# Command Line and Arguments

old main() :

```
int main(void)
{
    ...
}
```

change :

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

# Command Line and Arguments

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

Ex

```
$ ./file_io4  abc  def ↵
```

Status :

| | | |
|---|---|---|
| argc | 3 | number of arguments including file-name |
| argv[0] | "./file_io4" | string of file-name |
| argv[1] | "abc" | string of the 1st argument |
| argv[2] | "def" | string of the 2nd argument |

# Command Line and Arguments

file_io4.c

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int i;

    printf("argc : %d¥n", argc);
    for (i = 0; i < argc; i++) {
        printf("argv[%d] : %s¥n", i, argv[i]);
    }

    return 0;
}
```

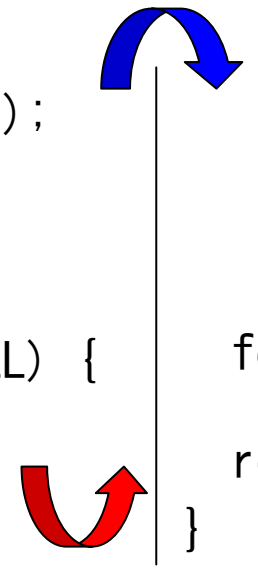
execute :

```
$ ./file_io4  abc  def  ghi ↵
...
```

# change「file_io1.c」with command line and arguments

file_io1_1.c

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

int main(int argc, char *argv[])
{
  FILE *fin;
  char str[256];

  fin = fopen(argv[1], "r");
  if (fin == NULL) {
    printf("Cannot open the file.¥n");
    exit(1);
  }

  while (fgets(str, 256, fin) != NULL) {
    printf("%s", str);
  }

  fclose(fin);

  return 0;
}
```



# file_io1_1.c

execute :

```
$ ./file_io1 file_io4.c ↵
#include <stdio.h>

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

# Excercise

Change all lowercases in one file into all uppercases.
And input all arguments from the command line

Ex

```
$ ./file_io5  file_io4.c ↵
#INCLUDE <STDIO.H>

INT MAIN(INT ARGC, *CHAR ARGV[])
...
```