Introduction to C Programming Language PART II

Input & Output Files

Min Zhang
zhangmin@sei.ecnu.edu.cn

2020.12.28

Software Engineering Institute

10:00-11:40, Monday, Room 319
Software Engineering Institute, East China Normal University

Functions on input/output

Input:

- `scanf`: formatted input
- `getchar`: read a char, including whitespace, enter, etc.
- `gets`: read a line, including whitespace and enter

Output:

- `printf`: formatted output
- `putchar`: output a char.
- `gets`: output a string ended with `'\0'`

Functions on input/output

Input:

- `scanf`: formatted input
- `getchar`: read a char, including whitespace, enter, etc.
- `gets`: read a line, including whitespace and enter

Output:

- `printf`: formatted output
- `putchar`: output a char.
- `gets`: output a string ended with `'\0'`

Where do you read: keyboard
Where do you write: screen

More than that

Besides keyboard, we can read from: **files**

More than that

Besides keyboard, we can read from: **files**

Besides screen, we can write to: **files**

How do YOU read/write from/to files?

How do YOU read/write from/to files?

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`

How do YOU read/write from/to files?

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., double click

How do YOU read/write from/to files?

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., double click
3. Read/Write

How do YOU read/write from/to files?

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., double click
3. Read/Write
4. Save

How do YOU read/write from/to files?

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., double click
3. Read/Write
4. Save
5. Close, e.g., `Alt+F4`

How to read/write from/to files (by program)

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`

How to read/write from/to files (by program)

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., `fopen()`

How to read/write from/to files (by program)

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., `fopen()`
3. Read/Write, e.g., `fscanf(), fgetc(), fgets()/ fprintf(), fputc(), fputs()`

How to read/write from/to files (by program)

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., `fopen()`
3. Read/Write, e.g.,
   `fscanf()`, `fgetc()`, `fgets()`/ `fprintf()`, `fputc()`, `fputs()`
4. ~~Save~~ (no needed anymore)

How to read/write from/to files (by program)

1. Find the location of the file, e.g., `"C:\Users\ABC\Desktop\MyFiles\abc.txt"`
2. Open it, e.g., `fopen()`
3. Read/Write, e.g.,
   `fscanf()`, `fgetc()`, `fgets()`/ `fprintf()`, `fputc()`, `fputs()`
4. ~~Save~~ (no needed anymore)
5. Close, e.g., `fclose()`

fopen()

Usage

```c
FILE *fopen(const char *path, const char *mode)
```

- FILE: a type
- path: a char pointer, **the path of the file to access**
- mode: a char pointer, **the access mode**
  - "r": read only
  - "r+": read and write, **error if file not exists**
  - "w": overwrite
  - "w+": read and write, **create the file if not exist**
  - "a": write at the end file
  - "a+": read from beginning and write at the end

fopen(): an example

```
FILE *fp = fopen("C:\Users\ABC\Desktop\MyFiles\abc.txt", "r");
```

fgetc: read a character from an opened file

```
FILE *fp = fopen("C:\Users\ABC\Desktop\MyFiles\abc.txt", "r");
```

fgetc: read a character from an opened file

```c
FILE *fp = fopen("C:\Users\ABC\Desktop\MyFiles\abc.txt", "r");
```

```c
char c;
c=fgetc(fp); // read a character (can be whitespace, return, etc)
```

fgetc: read a character from an opened file

```c
FILE *fp = fopen("C:\Users\ABC\Desktop\MyFiles\abc.txt", "r");
```

```c
char c;
c=fgetc(fp); // read a character (can be whitespace, return, etc)
```

Example (I)

```c
while((c=fgetc(fp))!=EOF){
  putchar(c); // output to the screen
}
```

fgetc: read a character from an opened file

```
FILE *fp = fopen("C:\Users\ABC\Desktop\MyFiles\abc.txt", "r");
```

```
char c;
c=fgetc(fp); // read a character (can be whitespace, return, etc)
```

Example (I)

```
while((c=fgetc(fp))!=EOF){
  putchar(c); // output to the screen
}
```

Example (II): copy a file

```
FILE *fp2 = fopen("C:\Users\ABC\Desktop\MyFiles\abc2.txt", "w");
while((c=fgetc(fp))!=EOF){
   fput(c,fp2); // output to the screen
}
```

fgets: read a line from an opened file

```c
char *fgets(char *s, int size, FILE *stream);
```

- s: the pointer to a char array to store fetched chars
- size: the maximal number of chars to read one time
- stream: the opened file
- **return NULL if reading to the end**

fgets: read a line from an opened file

```c
char *fgets(char *s, int size, FILE *stream);
```

- s: the pointer to a char array to store fetched chars
- size: the maximal number of chars to read one time
- stream: the opened file
- **return NULL if reading to the end**

```c
char str[1001];
char *sp=str;
fgets(str,1000,fp)
```

fgets: read a line from an opened file

Example (I)

```
while((sp=fgets(str,1000,fp))!=NULL){
  puts(sp); // output to the screen
}
```

fgets: read a line from an opened file

Example (I)

```c
while((sp=fgets(str,1000,fp))!=NULL){
  puts(sp); // output to the screen
}
```

Example (II): copy a file

```c
FILE *fp2 = fopen("C:\Users\ABC\Desktop\MyFiles\abc2.txt", "w");
while((sp=fgets(str,1000,fp))!=EOF){
   fputs(sp,fp2); // output to the screen
}
```

fclose()

Remember to close your opened files when leaving!

How? `fclose(fp); // to close the opened file fp`

Why? `In case of data loss. Others can use that file!`

Try: `try deleting a file when you open it!`

The type: FILE

```c
FILE *fp;
```

What is FILE?

The type: FILE

```c
FILE *fp;
```

What is FILE? FILE is defined in stdio.h

```c
struct _iobuf {
  char *_ptr; //文件输入的下一个位置
  int _cnt; //当前缓冲区的相对位置
  char *_base; //指基础位置(即是文件的其始位置)
  int _flag; //文件标志
  int _file; //文件的有效性验证
  int _charbuf; //检查缓冲区状况,如果无缓冲区则不读取
  int _bufsiz; //缓冲区的大小
  char *_tmpfname; //临时文件名
};
typedef struct _iobuf FILE。
```

Input/Output redirection

Three FILE type pointers:

1. stdin: a constant of FILE*, pointing to the keyboard;
2. stdout: a constant of FILE*, pointing to the screen;
3. stderr a constant of FILE*, pointing to the screen;

```c
putc('A', stdout); // equal to putchar('A')
char c = getc(stdio); // read from keyboard
fputs("something error", stderr); // output error message to screen
```

Redirection

Make stdin and stdout point to other input source and output destination.

```
Prompt> ./a.out < aa.txt // stdin points to aa.txt
Prompt> ./a.out > aa.txt // stdout points to aa.txt
Prompt> ./a.out | ./b.out // stdout of a.out points to the stdin of b.out
```

Error handling & Exit

When error occurs, you may want to output some message to the screen!
You need `stderr`, e.g., `fprintf(stderr, "Error, RUN\n");`

Why do we need `stderr`

Error handling & Exit

When error occurs, you may want to output some message to the screen!
You need `stderr`, e.g., `fprintf(stderr, "Error, RUN\n");`

Why do we need `stderr`
Because stdout may be redirected from screen.

Error handling & Exit

When error occurs, you may want to output some message to the screen!
You need `stderr`, e.g., `fprintf(stderr, "Error, RUN\n");`

Why do we need `stderr`
Because stdout may be redirected from screen.

When error occurs, you may want to make program terminate!
How: to use `exit()` function

```c
if(x!=0){
  x=y%x;
}else{
  exit(1); // terminate the program, 1 means an error occurs
}
```

Miscellaneous functions

string.h `char s[100],t[100];`

```
int n;
char c;
```

Function	Meaning
strcat(s,t)	concatenate t to end of s
strncat(s,t,n)	concatenate n characters of t to end of s
strcmp(s,t)	return negative, zero, or positive for s $<$ t , s $==$ t , s $>$ t
strncmp(s,t,n)	same as strcmp but only in first n characters
strcpy(s,t)	copy t to s
strncpy(s,t,n)	copy at most n characters of t to s
strlen(s)	return length of s
strchr(s,c)	return pointer to first c in s , or NULL if not present
strrchr(s,c)	return pointer to last c in s , or NULL if not present

Miscellaneous functions

ctype.h

`char c;`

Function	Meaning
`isalpha(c)`	non-zero if c is alphabetic, 0 if not
`isupper(c)`	non-zero if c is upper case, 0 if not
`islower(c)`	non-zero if c is lower case, 0 if not
`isdigit(c)`	non-zero if c is digit, 0 if not
`isalnum(c)`	non-zero if isalpha(c) or isdigit(c) , 0 if not
`isspace(c)`	non-zero if c is blank, tab, newline, return, formfeed, vertical tab
`toupper(c)`	return c converted to upper case
`tolower(c)`	return c converted to lower case

Miscellaneous functions

Command Execution

Function	Meaning
`system(char *s)`	executes the command contained in the character string s , then resumes execution of the current program

Storage Management

Function	Meaning
`void *malloc(size_t n)`	returns a pointer to n bytes of uninitialized storage
`void *calloc(size_t n, size_t size)`	returns a pointer to enough free space for an array of n objects of the specified size

Miscellaneous functions

math.h

Function	Meaning
`sin(x)`	sine of x , x in radians
`cos(x)`	cosine of x , x in radians
`atan2(y,x)`	arctangent of y/x , in radians
`exp(x)`	exponential function e x
`log(x)`	natural (base e) logarithm of x (x>0)
`log10(x)`	common (base 10) logarithm of x (x>0)
`pow(x,y)`	x^y
`sqrt(x)`	square root of x (x>0)
`fabs(x)`	absolute value of x

Miscellaneous functions

Random Number generation

Function	Meaning
`rand()`	computes a sequence of pseudo-random integers in the range zero to `RAND_MAX`
`srand(unsigned)`	sets the seed for rand()

ENJOY PROGRAMMING!