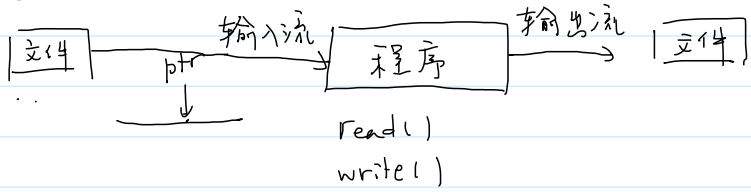
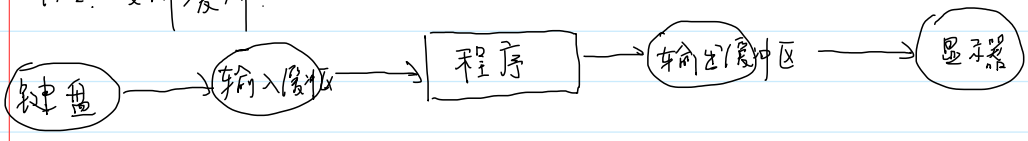
文件

Unix 哲学：一切皆文件

流：表示任意输入的源或输出的目的地。 (stream)



#2. 文件缓冲



缓冲区的分类：

fflush synchronizes an output stream with the actual file (function)

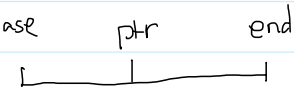
满缓冲区：

行缓冲区：

无缓冲区：

#3. 标准流

流 → FILE*



stdin	标准输入流	键盘
stdout	标准输出流	显示器
stderr	标准错误流	显示器

这三个标准流可以直接使用，使用完毕之后也不需关闭。

#4. 文本文件和二进制文件

文本文件有两个特殊的性质：

① 文本文件有行的概念：

Linux → '\n'

Windows → '\r''\n'

② 文本文件可能包一个特殊的文件末尾 EOF

Windows → '\x1a' (ctrl+z)

文本文件存储数据，方便人类阅读和编辑。 "32767" → 5字节

缺点：占用空间高

二进制文件存储数据：缺点：看不懂

优点：占用空间小 32767 → 2字节

#5. fopen / fclose

FILE

Defined in header <stdio.h>

```
FILE *fopen( const char *filename, const char *mode );
```

流 ←

filename，文件路径

mode，打开文件的方式.

文件路径 { 绝对路径，从根目录（盘符）开始，一直到文件所在的位置. "c:/project/test.dat"
相对路径，从当前工作目录开始，一直到文件所在的位置. "test.dat"

模式.

"rt"	"r"	→ read 只读，要求文件事先存在.
"wt"	"w"	→ write 只写，（不要求文件存在，如果文件存在，写之前会清空原文件的内容）
"at"	"a"	→ append 追加（不要求文件存在，如果文件存在，不会清空原文件的内容）
"rt+"	"r+"	→ 要求文件存在，如果含数据会清空数据.
"wt+"	"w+"	→ 不要求文件存在.
"at+"	"a+"	→ 不要文件存在，不会清空原有数据.

读各文本文件

"rb"
"wb"
"ab"
"rb+"或"r+b"
"wb+"或"w+b"
"ab+"或"a+b"

Defined in header <stdio.h>

```
int fclose( FILE *stream );
```

如果成功关闭，返回零；否则返回EOF

文件的读和写.

文本文件：fgetc/fputc，fgets/fputs，fscanf/fprintf

Defined in header <stdio.h>

```
int fgetc( FILE *stream );   (1)
```

Defined in header <stdio.h>

```
int fputc( int ch, FILE *stream );
```

fgets

Defined in header <stdio.h>

```
char *fgets( char          *str, int count, FILE          *stream );   (until C99)
char *fgets( char *restrict str, int count, FILE *restrict stream );   (since C99)
```

Reads at most `count - 1` characters from the given file stream and stores them in the character array pointed to by `str`. Parsing stops if a newline character is found, in which case `str` will contain that newline character, or if end-of-file occurs. If bytes are read and no errors occur, writes a null character at the position immediately after the last character written to `str`.

Parameters

str - pointer to an element of a char array
count - maximum number of characters to write (typically the length of `str`)
stream - file stream to read the data from

Return value

`str` on success, null pointer on failure.

If the end-of-file condition is encountered, sets the *eof* indicator on `stream` (see `feof()`). This is only a failure if it causes

fputs

Defined in header <stdio.h>

```
int fputs( const char          *str, FILE          *stream );   (until C99)
int fputs( const char *restrict str, FILE *restrict stream );   (since C99)
```

Writes every character from the null-terminated string `str` to the output stream `stream`, as if by repeatedly executing `fputc`.

The terminating null character from `str` is not written.

Parameters

str - null-terminated character string to be written

stream - output stream

Return value

On success, returns a non-negative value

On failure, returns EOF and sets the *error* indicator (see `ferror()`) on `stream`.

二进制文件的读写.

fread

Defined in header <stdio.h>

```
size_t fread( void          *buffer, size_t size, size_t count,   (until C99)
              FILE          *stream );
size_t fread( void *restrict buffer, size_t size, size_t count,   (since C99)
              FILE *restrict stream );
```

Reads up to `count` objects into the array `buffer` from the given input stream `stream` as if by calling `fgetc` `size` times for each object, and storing the results, in the order obtained, into the successive positions of `buffer`, which is reinterpreted as an array of `unsigned char`. The file position indicator for the stream is advanced by the number of characters read.

If an error occurs, the resulting value of the file position indicator for the stream is indeterminate. If a partial element is read, its value is indeterminate.

Parameters

buffer - pointer to the array where the read objects are stored

size - size of each object in bytes

count - the number of the objects to be read

stream - the stream to read

Return value

Number of objects read successfully, which may be less than `count` if an error or end-of-file condition occurs.

If `size` or `count` is zero, `fread` returns zero and performs no other action.

`fread` does not distinguish between end-of-file and error, and callers must use `feof` and `ferror` to determine which occurred.

fwrite

Defined in header <stdio.h>

```
size_t fwrite( const void *buffer, size_t size, size_t count,   (until C99)
               FILE *stream );
size_t fwrite( const void *restrict buffer, size_t size, size_t count,   (since C99)
               FILE *restrict stream );
```

Writes `count` of objects from the given array `buffer` to the output stream `stream`. The objects are written as if by reinterpreting each object as an array of `unsigned char` and calling `fputc` `size` times for each object to write those `unsigned char`s into `stream`, in order. The file position indicator for the stream is advanced by the number of characters written.

If an error occurs, the resulting value of the file position indicator for the stream is indeterminate.

Parameters

buffer - pointer to the first object in the array to be written

size - size of each object

count - the number of the objects to be written

stream - pointer to the output stream

Return value

The number of objects written successfully, which may be less than `count` if an error occurs.

If `size` or `count` is zero, `fwrite` returns zero and performs no other action.

7. 文件定位.

```
int fseek ( FILE* stream , long int offset , int whence );
long ftell ( FILE* stream );
void rewind ( FILE* stream );
```

offset: 以字节为单位计数

whence: 参照点
- SEEK_SET: 文件的起始位置
- SEEK_CUR: 文件的当前位置
- SEEK_END: 文件的末尾位置

移动到文件的开始

fseek(stream, 0L, SEEK_SET) ⇔ rewind(stream)

往回移动10个字节.

```
fseek(stream, -10L, SEEK_CUR);
```

移动到文件的末尾.

```
fseek(stream, 0L, SEEK_END);
```

ftell返回当前文件的位置(相对SEEK_SET而言)

错误处理

2022年3月25日 17:52

errno

Defined in header <errno.h>

→ 系统调用（system call）

数值计算、文件读写发生错误，会把errno设置为对应的值.