一、非标准 IO 通过文件描述符 fd 来进行读写。　　In linux　fd: 0 1 2

<fcntl.h>　-----　open,　create

<unistd.h>　----　close,　lseek,　read(fd, buf, size),　write(fd, buf, size),

原子性 pread,　pwrite

Buf_size 设置为 4096 最为合适。

二、标准 IO 操作是围绕流对象（FILE）来进行操作的。

<cstdio>中的所有操作函数都是针对流对象的，有些因为标准的定义隐藏了流对象的显示声明。

FILE *stdout, *stdin, *stderr;

Fread,　fwrite,(可以作为二进制批量读取，因为其指定了 buf 以及 buf_size，以 char 为单位的读 写)　fseek,　fopen,　fclose,　freopen,　fflush,　setbuf,　setvbuf,　fgetc/getc, fgetchar/getchar,　fputc/putc,　fgets/gets,　fputs/puts,　perror/ferror

格式化输入输出：

Defined in header <cstdio>

```
int scanf( const char* format, ... );                                  (1)
int fscanf( std::FILE* stream, const char* format, ... );              (2)
int sscanf( const char* buffer, const char* format, ... );             (3)
```

```
int printf( const char* format, ... );                                 (1)
int fprintf( std::FILE* stream, const char* format, ... );             (2)
int sprintf( char* buffer, const char* format, ... );                  (3)
int snprintf( char* buffer, int buf_size, const char* format, ... );   (4)  (since C++11)
```

File positioning

ftell	returns the current file position indicator (function)
fgetpos	gets the file position indicator (function)
fseek	moves the file position indicator to a specific location in a file (function)
fsetpos	moves the file position indicator to a specific location in a file (function)
rewind	moves the file position indicator to the beginning in a file (function)

对于流的读写分为两种方式：字符流读写、二进制流读写。 -------　这些在打开流的时候通过参数来确定，一般二进制流带上参数‘b’或者‘+’。

三、C++ 的流操作：

- IO 流：

Iostream 以及继承子它的 fstream, sstream, 且每个流中都有一个 streambuf 由 iostream 头文件包含）就是流的缓冲区对象。 可以通过 rdbuf()来获取。Pubsetbuf(buf, size);来重新设置 buf。

Istream:　read (buf, size);

```
basic_istream& getline( char_type* s, std::streamsize count );                (1)
basic_istream& getline( char_type* s, std::streamsize count, char_type delim ); (2)
```

```
int_type get();                                                          (1)
basic_istream& get( char_type& ch );                                     (2)
basic_istream& get( char_type* s, std::streamsize count );               (3)
basic_istream& get( char_type* s, std::streamsize count, char_type delim );  (4)
basic_istream& get( basic_streambuf& strbuf );                           (5)
basic_istream& get( basic_streambuf& strbuf, char_type delim );          (6)
```

Ostream: put(char), write(buf, size);

ends	outputs '\0' (function template)
flush	flushes the output stream (function template)
endl	outputs '\n' and flushes the output stream (function template)

- 文本流：《基本的读写操作继承自 iostream》

File operations

is_open	checks if the stream has an associated file (public member function)
open	opens a file and associates it with the stream (public member function)
close	closes the associated file (public member function)

Miscellaneous

flush	synchronizes with the underlying storage device (public member function of std::basic_ostream)

See also

sync	synchronizes with the underlying storage device (public member function of std::basic_istream)
flush	flushes the output stream (function template)
endl	outputs '\n' and flushes the output stream (function template)

- 字符流：《基本的读写操作继承自 iostream》

Miscellaneous

flush	synchronizes with the underlying storage device (public member function of std::basic_ostream)

Get/set 方法

```
std::basic_string<CharT,Traits,Allocator> str() const;          (1)
void str(const std::basic_string<CharT,Traits,Allocator>& new_str);  (2)
```

四、高级 IO ：

<sys/uio.h>

memcached 就是利用这两个数据结构配合上 sendmsg 来完成数据的传输。

<sys/socket.h>

sendmsg(Socketfd, struct msghdr *, int flags);

recvmsg(Socketfd, struct msghdr *, int flags);

```
struct msghdr
{
	void *	msg_name;	/* Socket name			*/
	int		msg_namelen;	/* Length of name		*/
	struct iovec *	msg_iov;	/* Data blocks		*/
	int		msg_iovlen;	/* Number of blocks		*/
	void *	msg_accrights;	/* Per protocol magic (eg BSD file descriptor passing) */
	int		msg_accrightslen;	/* Length of rights list */
};
```

In the msghdr structure, the msg_name and msg_namelen members specify the source address if the socket is unconnected. If the socket is connected, the msg_name and msg_namelen members shall be ignored.

```
<sys/uio.h>
struct iovec {
	void *iov_base; /* Starting address */
	size_t iov_len; /* Length in bytes */
};
Size_t readv(fd, iovec *, iovcnt)
Size_t writev(fd, iovec *, iovcnt)
```