Chương 9: Files

1. Giới thiệu

- C/C++ hỗ trợ 2 hệ thống nhập xuất.
  - Một hệ thống thừa kế từ ngôn ngữ C
  - Một hệ thống nhập xuất hướng đối tượng của C++.

2. Stream và file

- Hệ thống nhập xuất của C cung cấp một giao diện (*interface*) nhất quán - mức độ trừu tượng giữa lập trình viên và thiết bị nhập xuất. Sự trừu tượng này được gọi là stream và thiết bị thật sự được gọi là file

3.1. Text Stream

- Một text stream là một chuỗi các ký tự. Trong một text stream, một số ký tự có thể bị chuyển đổi (được hiểu như là một ký tự khác) tùy thuộc môi trường.
- Ví dụ, ký tự newline ('\n') có thể bị đổi thành cặp ký tự carriage return/linefeed (*ký tự xuống dòng và về đầu dòng*).

3.2. Binary Stream

- Một binary stream là một chuỗi bytes mà có sự tương ứng một-một với chuỗi bytes trên thiết bị ngoài. Nghĩa là không có sự chuyển đổi xảy ra. Do đó, số bytes được viết (hay đọc) thì bằng với số bytes trên thiết bị ngoài.

4. Files

- Một file có thể là một tập tin trên đĩa, một terminal, hay máy in.
- Để tạo kết nối (*associate*) giữa một stream với một file ta dùng hoạt động mở (*open*). Một khi một file được mở, thông tin có thể được trao đổi giữa nó và chương trình.

4. Files

- Không phải tất cả file đều có cùng khả năng như nhau.
- Ví dụ, một tập tin trên đĩa (*file*) có thể hỗ trợ truy xuất ngẫu nhiên trong khi đó máy in (cũng là file) thì không thể. Việc này đưa đến một kết luận là: "Tất cả stream là như nhau nhưng file thì không".

4. Files

- Để ngắt kết nối giữa một stream với một file ta dùng hoạt động đóng (*close*). Nếu đóng một file đang mở cho xuất (*output*) thì nội dung (nếu có) của stream tương ứng được viết ra thiết bị ngoài. Quá trình này được gọi là flushing và đảm bảo là không có thông tin bị để lại trong vùng đệm (*buffer*).
- Tất cả file được tự động đóng khi chương trình mở chúng kết thúc bình

4. Files

- Files không được đóng khi chương trình mở chúng bị kết thúc bất thường như bị treo (*halt*) hay khi chương trình thực hiện hàm abort().
- Mỗi stream liên đới với một file có một cấu trúc kiểu FILE

4.1. Cơ bản về hệ thống Files

- Các hàm hoạt động trên file nằm trong thư viện stdio.h.

Tên hàm	Chức năng
fopen( )	Mở một file
fclose( )	Đóng một file.
putc( )	Viết một ký tự đến một file.
fputc( )	Giống như putc() .
getc( )	Đọc một ký tự từ một file.
fgetc( )	Giống như getc() .
fgets( )	Đọc một chuỗi từ một file.

4.1. Cơ bản về hệ thống Files

Tên hàm	Chức năng
fputs( )	Viết một chuỗi đến một file.
fseek( )	Tìm một byte trong một file.
ftell( )	Trả về vị trí hiện hành của của file indicator.
feof( )	Trả về true nếu duyệt đến cuối file (end-of-file).
ferror( )	Trả về true nếu một lỗi xảy ra.
rewind( )	Đưa indicator về đầu.
remove( )	Xóa một file.
fflush( )	Xả hết vùng đệm của file.

4.2. Con trỏ file (file pointer)

- Một con trỏ file là một cấu trúc kiểu FILE.
- Con trỏ file:
  - Trỏ đến thông tin mà định nghĩa về file như tên file, trạng thái, và vị trí hiện hành của file.
  - Được dùng bởi stream tương ứng để thực hiện các hoạt động nhập xuất trên file.
- Để đọc hay viết file, chương trình phải

4.3. Mở file

- Hàm fopen(): mở một stream để dùng và liên kết một file với stream đó. Hàm trả về một con trỏ file liên đới với tập tin được mở.
  - Cú pháp:

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

- **filename**: Là một hằng chuỗi chứa tên (và đường dẫn) của file
- **mode**: Là một hằng chuỗi cho biết mở file theo mode nào

4.3. Mở file

- Hàm fopen():
  - **Các mode để mở file:**
    - **“r”**: Nếu tập tin được mở thành công, hàm fopen() nạp nó vào trong bộ nhớ và trả về một con trỏ trỏ đến ký tự đầu tiên của tập tin. Nếu không thể mở tập tin, hàm fopen() trả về NULL
    - **“w”:** Nếu tập tin tồn tại, nội dung của nó sẽ bị viết đè. Nếu tập tin không tồn tại, một tập tin mới được tạo. Trả về NULL nếu không thể mở tập tin.

4.3. Mở file

- Hàm fopen():
  - **Các mode để mở file:**
    - "**a**"**:** Nếu tập tin được mở thành công, hàm fopen() nạp nó vào trong bộ nhớ và trả về một con trỏ trỏ đến ký tự cuối cùng của tập tin. Nếu tập tin không tồn tại, một tập tin mới được tạo. Trả về NULL nếu không thể mở tập tin.
    - **"r+"** Nếu tập tin được mở thành công, hàm fopen() nạp nó vào trong bộ nhớ. Trả về NULL nếu không thể mở tập tin.
    - **"a+":** Nếu tập tin được mở thành công, hàm fopen() nạp nó vào trong bộ nhớ và trả về một con trỏ trỏ đến ký tự đầu tiên của tập tin. Nếu tập

4.3. Mở file

- Ví dụ:

```
FILE *fp;
if((fp = fopen("test.txt","w")) ==
NULL)
{   cout << "Cannot open file";
    exit(1);
}
```

4.4. Đóng file

- Hàm fclose(): đóng stream được mở bởi hàm fopen(). Khi hàm được gọi, nó sẽ viết bất kỳ dữ liệu nào vẫn còn trong buffer đến file rồi đóng file.
  - Cú pháp:

    int fclose(FILE *fp);

  - **fp:** là con trỏ file trả về bởi hàm fopen()
  - Nếu đóng file thành công, hàm trả về giá trị zero. Nếu một lỗi xảy ra khi đóng file, hàm trả về EOF.

4.4. Viết một ký tự vào file

- Có hai hàm xuất ký tự đến file là putc() và fputc(). Hai hàm này là tương đương nhau. Hàm putc() viết một ký tự đến một file đã được mở bởi hàm fopen().
  - Cú pháp:

int putc(int ch, FILE *fp);

- **fp** là con trỏ file trả về bởi hàm fopen()
- **ch** là ký tự được viết đến file
- Nếu hoạt động putc() thành công, nó trả về ký tự

4.4. Đọc một ký tự vào file

- Có hai hàm tương đương để đọc một ký tự từ file là getc() và fgetc(). Hàm getc() đọc mỗi lần một ký tự từ file được mở bởi hàm fopen() ở chế độ đọc (read).
  - Cú pháp:

    int getc(FILE *fp);

    - fp là con trỏ file kiểu FILE trả về bởi hàm fopen().
    - Hàm getc() trả về một số nguyên là giá trị của ký tự được đọc. Hàm getc() trả về EOF nếu một lỗi xảy ra.

4.5. Ví dụ các thao tác trên file

- Ví dụ 1: Viết chương trình cho người dùng nhập ký tự từ bàn phím và ghi chúng vào một tập tin và dừng khi người dùng nhập ký tự $

```
void main()
{
	FILE *fp;
	char ch;
	if((fp=fopen(“test.txt”, "w"))==NULL)
	{
		cout << "Cannot open file.\n";
		exit(1);
	}
	do {
	ch = getchar();
	putc(ch, fp);
	}while (ch != '$');
	fclose(fp);
}
```

4.5. Ví dụ các thao tác trên file

- Ví dụ 2: Viết chương trình đọc một file văn bản và xuất chúng ra màn hình.

```
void main()
{
	FILE *fp;
	char ch;
	if((fp=fopen(”test.txt”, "r"))==NULL)
	{
	cout << "Cannot open file.\n";
	exit(1);
	}
	ch = getc(fp); // read one character
	while (ch!=EOF)
	{
	putchar(ch); // print on screen
	ch = getc(fp);}
	fclose(fp);
}
```

4.6. Đọc và viết chuỗi trên file

- C/C++ hỗ trợ hai hàm fgets() và fputs() để đọc và viết chuỗi ký tự trên file.
- Hàm fputs(): viết một chuỗi trỏ đến bởi con trỏ str vào stream trỏ đến bởi con trỏ file fp. Hàm trả về EOF nếu một lỗi xảy ra.
- Cú pháp:
  - **int fputs(const char *str, FILE *fp);**

4.6. Đọc và viết chuỗi trên file

- Hàm fgets(): đọc một chuỗi từ stream tương ứng cho đến khi gặp ký tự newline hay đã đọc được length-1 ký tự. Hàm trả về str nếu đọc thành công và một con trỏ null nếu không.
- Cú pháp:
  - **char *fgets(char *str, int length, FILE *fp);**

4.6. Đọc và viết chuỗi trên file

- Ví dụ: Viết chương trình đọc các chuỗi từ bàn phím và viết chúng đến file tên teststr.txt. Để kết thúc chương trình, nhập một dòng trống.

4.6. Đọc và viết chuỗi trên file

```
void main(void)
{
	char str[80];
	FILE *fp;
	if((fp = fopen("teststr.txt", "w"))==NULL)
	{
		cout << "Cannot open file.\n";
		exit(1);
	}
	do
	{
		cout << "Enter a string (CR to quit):\n";
		gets(str);
		strcat(str, "\n"); /* add a newline */
		fputs(str, fp);
	} while(*str!='\n');
}
```

4.7. Hàm fread() và fwrite()

- Để đọc và viết các kiểu dữ liệu có kích thước lớn hơn 1 byte, C/C++ cung cấp hai hàm fread() và fwrite().
- Cú pháp:

```
size_t fread(void *buffer, size_t numbytes, size_t count, FILE *fp);
```

```
size_t fwrite(const void *buffer, size_t numbytes, size_t count, FILE *fp);
```

4.8. Hàm rewind()

- Hàm rewind() di chuyển indicator đến điểm bắt đầu của file.
- Cú pháp:

  void rewind(FILE *fp);

- Hàm trả về true nếu một lỗi đã xảy ra với hoạt động trên file trước khi gọi hàm ferror(), ngược lại trả về false.

4.9. Hàm ferror()

- Hàm ferror() cho biết một hoạt động trên file đã gây ra lỗi.
- Cú pháp:

int ferror(FILE *fp);

4.10. Xóa file

- Hàm remove() dùng để xóa tập tin.
- Cú pháp:

  int remove(const char *filename);

  - Hàm trả về zero nếu xóa thành công, ngược lại trả về nonzero

4.11. Flushing a stream

- Hàm fflush() dùng để xuất tất cả nội dung còn lại trong buffer của stream.

- Cú pháp:

  int fflush(FILE *fp);

  - Hàm viết nội dung còn trong buffer đến file liên đới với fp. Nếu ta gọi hàm fflush() không có đối số thì tất cả flush tất cả file đang mở.
  - Hàm trả về 0 nếu thành công, ngược lại trả về EOF

4.11. Truy xuất file ngẫu nhiên

- Hàm fseek: dùng để dùng để di chuyển chỉ báo file.
- Cú pháp:

```
int fseek(FILE *fp, long numbytes, int origin);
```

  - fp: là con trỏ trả về bởi hàm fopen()
  - origin: là một trong các giá trị sau: SEEK_SET (từ đầu file), SEEK_CUR (từ vị trí hiện hành), và SEEK_END (từ cuối file).
  - numbytes: số byte mà indicator di chuyển tùy thuộc origin cung cấp.

5. Các stream chuẩn

- Khi một chương trình thực thi, ba stream được mở tự động:
  - stdin (standard input): đọc từ bàn phím
  - stdout (standard output): xuất lên màn hình
  - stderr (standard error).
- Bởi vì standard streams là các con trỏ file nên có thể dùng các hàm nhập xuất trên chúng.