Hầu hết các chương trình đều yêu cầu đọc và ghi dữ liệu vào các hệ thống lưu trữ trên đĩa, các chương trình xử lý văn bản cần lưu các tập tin văn bản, các chương trình xử lý bảng tính cần lưu trữ nội dung vào các ô

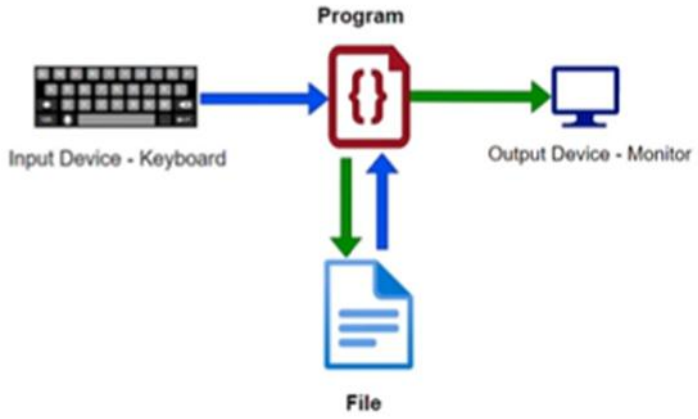
Chương trình cơ sở dữ liệu thì cần lưu trữ các mẫu tin

Các tiện ích của C dành cho việc thao tác xử lý và làm việc với các dữ liệu tập tin

Các nội dung là hiểu được các thuật ngữ file và stream

Thế nào là text streams và binary streams

Con trỏ file là gì



Cách sử dụng stream cơ bản để đọc ghi file

Các thuật ngữ và định nghĩa

File, strea, và kiểu dữ liệu file

Text stream và binary stream

Stream không còn là từ khoá xa lạ, live stream bán hàng, streamer. Stream trong TA là luồng hay sóng. Stream trong bài học thực chất là

Với ngôn ngữ C những người thiết kế muốn xây dựng 1 cách giao tiếp thống nhất các nguồn dữ liệu tuần tự khác nhau như là file, bàn phím, cổng USB, máy in

Giao diện có thể áp dụng cho tất cả các chương trình sử dụng thuộc tính và trừu tượng được đặt với tên chung là stream

Các hàm thư viện trong C hiện nay xử lý các đối tượng kiểu file nên cấu trúc dữ liệu C đại diện cho 1 stream được gọi là file chứ ko phải là 1 stream

Kiểu dữ liệu file:

*FILE ** **fopen** *(const char *filename, const char *opentype)*

File trong TV là tập tin là tập hợp các dữ liệu do người dùng tạo ra từ máy tính giúp người sử dụng máy tính có thể lưu trữ dữ liệu 1 cách dễ dàng và thuận lợi

File được đặt tên và lưu trữ trên rất nhiều các công cụ khác biệt ví dụ như đĩa cứng, đĩa mềm, CD, DVD, USB

Cấu trúc của 1 file bao gồm

Tên và phần đuôi mở rộng được ngăn cách nhau bằng dấu . phần đuôi được dùng để phân loại file và phân đuôi sau dấu chấm của file được sử dụng để định dạng cho tệp tin đó, nó xác định file đó được sử dụng cho mục đích gì, sử dụng cho hành động nào, mỗi hệ điều hành sẽ dùng phần đuôi của tệp tin không giống nhau. Các tệp tin có hệ điều hành này không thể dùng cho hệ điều hành khác được hoặc file này lưu trữ thì không thể chạy cho phần mềm sử dụng các file âm thanh.

File system hay hệ thống tập tin là 1 phương pháp lưu trữ và tổ chức các loại tệp của máy tính và dữ liệu của chúng

Computer file là 1 cái nguồn tài nguyên để lưu trữ thông tin

Kiểu dữ liệu file và con trỏ file

Là kiểu dữ liệu đại diện cho đối tượng stream. 1 đối tượng có kiểu dữ liệu file thì giữ toàn bộ thông tin trạng thái nội bộ về kết nối với tập tin được liên kết

Đối tượng có kiểu dữ liệu file được cấp phát và quản lý bởi chức năng của thư viện vào ra trong C.

1 con trỏ tập tin file pointer rất cần thiết cho việc đọc và ghi các tập tin. Nó là 1 con trỏ trỏ đến 1 cấu trúc chứa thông tin về tập tin bao gồm tên tập tin, vị trí hiện tại của tập tin. Tập tin đó thì đang được đọc hay ghi và có bất kỳ lỗi nào xuất hiện hay là đã đến cuối tập tin rồi. người dùng không cần thiết phải biết chi tiết

Các định nghĩa đã được lấy từ trong thư viện chuẩn của C bao gồm khai báo cấu trúc trên file và câu lệnh khai báo cần thiết cho 1 con trỏ tập tin là

FILE *fptr; là 1 con trỏ trỏ đến file

Hệ thống tập tin của C làm việc được với rất nhiều thiết bị khác nhau bao gồm máy in, ổ đĩa, ổ băng từ và các thiết bị đầu cuối khác nhau. Mặc dù các thiết bị đều khác nhau nhưng những hệ thống tập tin có vùng đệm sẽ chuyển mỗi thiết bị về mỗi cái thiết bị logic được gọi là 1 stream vì mọi stream hoạt động tương tự trên việc quản lý các thiết bị rất dễ dàng và vì vậy mà ta có 2 loại stream cơ bản là text stream và binary stream.

Text stream

1 text stream văn bản là 1 chuỗi các ký tự, các stream văn bản thì có thể được tổ chức thành các dòng, mỗi dòng có thể kết thúc bằng ký tự sang dòng mới, ký tự sang dòng mới là tuỳ chọn trong dòng cuối và được quyết định khi cài đặt. hầu hết các trình biên dịch của C thì không kết thúc stream văn bản với ký tự sang dòng mới.

Trên 1 số hệ thống thì tệp văn bản có thể chỉ chứa các ký tự in được, các ký tự tab ngang và dòng mới, do đó thì các văn bản có thể ko hỗ trợ cái ký tự khác.

Trong 1 stream văn bản có thể xảy ra 1 vài sự chuyển đổi ký tự khi môi trường yêu cầu

Ví dụ ký tự sang dòng mới có thể chuyển thành 1 cặp ký tự để đầu dòng hoặc là nhảy đến dòng kế tiếp. vì vậy mối quan hệ giữa các ký tự được ghi hay là đọc và những ký tự ở ngoại vi có thể không phải là mối quan hệ 1 1 và cũng vì sự chuyển đổi có thể xảy ra này mà số lượng ký tự được ghi hay đọc có thể ko giống với ký tự ở thiết bị ngoại vi.

Binary stream

Là 1 chuỗi các byte với sự tương ứng 1 1 với thiết bị ngoại vi, nghĩa là ko có sự chuyển đổi ký tự, cũng vì vậy là số byte đọc ghi cũng sẽ giống như số byte ở thiết bị ngoại vi. Các stream nhị phân là các chuỗi byte thuần tuý mà không có bất kỳ 1 ký hiệu nào được dùng để chỉ ra điểm kết thúc của tập tin hay kết thúc của 1 bản ghi.

Tóm lại 1 binary stream là 1 chuỗi dài các ký tự đơn giản.

Binary stream có thể xử lý bất kỳ các giá trị ký tự nào. Luôn có nhiều khả năng hơn và dễ dự đoán hơn là text stream

Có các kiểu stream khác như là custom, string

Thế nào để làm việc với 1 stream cơ bản giống như file

Làm việc với stream:

Đóng mở 1 stream

Bộ đệm stream là gì, stream vào ra thế nào, định vị thông tin ra sao, điểm kết thúc của file và các text lỗi. Phân biệt stream và thread.

Standard stream

Ngôn ngữ lập trình C xem file như là 1 dòng stream các byte với các thiết bị xuất nhập theo từng byte cũng được xem là file thì C định nghĩa sẵn các tên cho các thiết bị này mở sẵn cho ta truy xuất ngay khi mở máy tính thì khi chức năng chính trong chương trình được thực hiện thì đã có 5 stream được xác định đã mở và có sẵn để sử dụng

Đầu vào tiêu chuẩn – Standard input (stdin)

Đầu ra tiêu chuẩn – Standard output (sdtout)

Standard error – stderr

Standard printer – stdprn

Standard auxiliary – stdaux (các thiết bị phụ trợ tiêu chuẩn)

Để mở 1 stream thì ta mở 1 tệp với hàm fopen sẽ tạo ra 1 stream mới và thiết lập kết nối giữa stream với tập tin. Điều này có thể liên quan đến việc tạo 1 tập tin mới. hàm fopen mở 1 stream cho việc nhập xuất tới tên file và trả về 1 con trỏ biến stream

Nếu mở file ko thành công thì hàm fopen sẽ trả về con trỏ NULL và có thể mở nhiều stream để cùng trỏ đến 1 file đang mở cùng 1 lúc.

Cú pháp mở 1 tập tin đơn giản:

FILE *fopen (const char *filename, const char *opentype)

Hàm fopen gồm 2 tham số là filename và opentype

Trong đó filename là chuỗi chứa tên file được mở còn open file là chuỗi chứa 1 trong các chế độ truy cập file bao gồm các chế độ

OpenType	
'r'	Open an existing file for reading only.
'w'	Open the file for writing only. If the file already exists, it is truncated to zero length. Otherwise a new file is created.
'a'	Open a file for append access; that is, writing at the end of file only. If the file already exists, its initial contents are unchanged and output to the stream is appended to the end of the file. Otherwise, a new, empty file is created.
'r+'	Open an existing file for both reading and writing. The initial contents of the file are unchanged and the initial file position is at the beginning of the file.
'w+'	Open a file for both reading and writing. If the file already exists, it is truncated to zero length. Otherwise, a new file is created.
'a+'	Open or create file for both reading and appending. If the file exists, its initial contents are unchanged. Otherwise, a new file is created. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

Ngoài ra còn 1 số chế độ khác

Close 1 stream

Ngược lại với open ta đóng stream bằng hàm fclose. Khi 1 stream được đóng với hàm fclose 1 kết nối giữa stream và file sẽ bị huỷ bỏ. sau khi đóng 1 stream thì ta ko thể thực hiện bất kỳ thêm 1 thao tác bổ sung nào trên stream này nữa

Hàm fclose sẽ trả về giá trị 0 nếu file này được đóng thành công và EOF nếu có lỗi phát sinh trong quá trình close

Int fclose (FILE *stream)

Ngoài ra có hàm fcloseall(void) để đóng toàn bộ các tệp tin

Thông thường thì các tệp tin được đóng riêng lẻ để dễ kiểm soát các lỗi phát sinh của trong từng stream một.

Khi close 1 string thì nếu chức năng chính trong chương trình là trả về hoặc là gọi hàm thoát thì tất cả các stream đang mở đều tự động đóng lại đúng cách

Nếu chương trình mà kết thúc theo bất kỳ cách nào khác chẳng hạn gọi hàm huỷ bỏ hoặc từ 1 tín hiệu nghiêm trọng nào đó mà các stream mở có thể không được đóng đúng cách và các bộ đệm không được xoá và các tệp có thể không hoàn chỉnh hoặc có thể gây lỗi

Stream buffering: các ký tự được ghi vào 1 stream thường được tích luỹ và truyền không đồng bộ tới tệp trong 1 khối thay vì xuất hiện ngay khi chúng được xuất ra bởi chương trình ứng dụng thì nó được gọi là 1 vùng đệm.

Hiện tại có 3 kiểu vùng đệm: unbuffered, line buffered, fully buffered

Unbuffered hay còn gọi là không dùng vùng đệm thì các ký tự được ghi và đọc từ luồng không có bộ đệm sẽ được truyền riêng lẻ từ tệp càng sớm càng tốt

Với linebuffered thì các ký tự được ghi vào 1 stream có điện được truyền tới 1 file theo khối khi gặp bất kỳ ký tự dòng mới

Cuối cùng là fully buffered thì các ký tự được ghi và đọc từ 1 stream được điền đẩy đủ và được truyền đến tệp trong các khối có kích thước tuỳ ý.

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

int main ()
{
   FILE *fp;

   fp = fopen("D:\\examples.txt", "w");

   if(fp == NULL)
   {
      printf("Error!");
      exit(1);
   }

   fputs("Learning about File Handling", fp);

   fclose(fp);

   return(0);
}
```

Đối tượng ở đây là 1 file example.txt với kiểu làm việc là write only với write only thì chúng ta biết là nếu file này chưa có thì sẽ được tạo mới, còn nếu mà đã có rồi thì write only sẽ xoá nội dung trong file và ghi các dữ liệu mới vào file thì chương trình sẽ kiểm tra xem là việc mở file có lỗi gì xảy ra hay không. Nếu có thì sẽ in ra lỗi và thoát ra khỏi chương trình, nếu ko thì sẽ put 1 đoạn string vào trong nội dung của file này và sau đó đóng file

Flusing Buffer

Trong quá trình làm việc với buffer thì buffer sẽ liên tục được vào ra các dữ liệu thì việc đẩy vào buffer trên 1 stream có nghĩa là chúng ta sẽ truyền những dữ liệu data sẽ được tích luỹ dần thành ký tự và được chuyển dần đến file.

Việc chuyển đổi tích luỹ này sẽ được tự động đẩy lên khi mà đầu ra của buffer là đầy hoặc là khi chúng ta đóng stream hoặc là chúng ta sẽ gọi kết thúc stream của chúng ta bằng cách gọi hàm exit, …

Thì nó sẽ tự động đẩy các data vào buffer và ở đây chúng ta đang muốn đẩy data buffer tại bất kỳ thời điểm nào thì chúng ta có thể gọi hàm fflush

Int fflush (FILE *stream)

Hàm này đồng thời sẽ xoá toàn bộ data trên buffer để làm cho chúng ta có 1 buffer trống chúng ta gọi

Formatted Input/Output là chuẩn rất hay làm việc với stream với các hàm fprint và fscanf đều là những hàm dễ sử dụng nhất với input và output trong quá trình chúng ta vào ra với stream nhưng nó ko phải lúc nào cũng là hiệu quả nhất trong công việc đọc ghi

Qua 1 số dữ liệu thì sẽ chuyển đổi thành mã ASCII thay vì formatted binary

Vấn đề về kích thước file cũng như là tốc độ mà nó xử lý cần được quan tâm thì chúng ta sẽ có các hàm fread fwrite là những lựa chọn tốt hơn trong quá trình input output

```
int fprintf(FILE * stream, const char *control_string,..);
int fscanf(FILE *stream, const char *control_string,...);
```

Khác ở printf và scanf là thêm con trỏ FILE vào trong cái tham số truyền vào đó thôi

Ngoài hàm fprintf và fscanf

Character Input/Output

Prototype	Function
int **fgetc** *(FILE *stream)*	Reads the next character.
int **getc** *(FILE *stream)*	Just like **fgetc**, except that it is implemented as a macro
int **getchar** *(void)*	Equivalent to **getc** with stdin as the value of the *stream* argument.
int **fputc** *(int c, FILE *stream)*	Converts the character *c* to type unsigned char, and writes it to the *stream*.
int **putc** *(int c, FILE *stream)*	Just like **fputc**, except that it can be implement as a macro, making it faster.
int **putchar** *(int c)*	Equivalent to **putc** with *stdout* as the value of the *stream* argument.
int **fputs** *(const char *s, FILE *stream)*	Writes the string *s* to the *stream*

Fgetc và getc là những hàm làm việc với các ký tự trong stream

Để làm việc với string thì chúng ta có hàm puts

Khi làm việc với file thì chúng ta có biến tham số đầu vào của chúng ta là con trỏ file nữa thôi.

Block Input/Output

- *You can use these functions to read and write binary data, as well as to read and write text in fixed-size blocks instead of by characters or lines.*

  *size_t fread (void *data, size_t size, size_t count, FILE *stream)*

  *size_t fwrite (const void *data, size_t size, size_t count, FILE *stream)*

Khi chúng ta quan tâm tới vấn đề kích thước file và tốc độ xử lý thì ta có thể sử dụng các hàm fread và fwrite để làm việc với các dữ liệu binary thì sẽ hiệu quả hơn so với fprintf và fscanf

```c
int main()
{
    FILE *fptr;
    int id;
    char name[30];
    int salary;

    fptr = fopen("D:\\Infor.txt", "w+");
    if (fptr == NULL) {
        printf("File khong ton tai.\n");
        exit(1);
    }

    printf("Nhap id: ");
    scanf("%d", &id);
    fprintf(fptr, "Id = %d\n", id);

    printf("Nhap name: ");
    scanf("%s", name);
    fprintf(fptr, "Name = %s\n", name);

    fclose(fptr);
    return 0;
}
```

Nhập ID và tên vào file.txt sẽ được lưu ở trong ổ D

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

int main()
{
   char str1[10], str2[10], str3[10];
   int year;
   FILE * fp;

   fp = fopen ("D:\\baitapc.txt", "w+");
   fputs("Nam nay la 2022", fp);

   rewind(fp);

   fscanf(fp, "%s %s %s %d", str1, str2, str3, &year);

   printf("Doc chuoi 1: |%s|\n", str1 );
   printf("Doc chuoi 2: |%s|\n", str2 );
   printf("Doc chuoi 3: |%s|\n", str3 );
   printf("Doc so nguyen: |%d|\n", year );

   fclose(fp);

   return(0);
}
```

Tạo 1 file txt và sau đó put string vào file này và bắt đầu get các ký tự và các số nguyên trong file bài tập c

File Positioning

- *The file position of a stream describes where in the file the stream is currently reading or writing. I/O on the stream advances the file position through the file.*

Vị trí của con trỏ FILE

Vị trí của con trỏ file trong string được mô tả là nơi mà con trỏ đó đang đọc hay là ghi. mỗi vị trí mà ta sẽ xác định xem là cái stream đang làm việc ở đâu

Vị trí con trỏ file được biểu diễn dưới dạng số nguyên thì nó sẽ đếm số byte tính từ đầu của file thì chúng ta sẽ có các hàm để hỗ trợ cách xác định vị trí của con trỏ file là hàm ftell

Hàm tiếp theo sẽ trả về một vị trí hiện tại của con trỏ stream của chúng ta

Hàm này cũng có thể xảy ra lỗi nếu như stream của chúng ta không hỗ trợ vị trí của con trỏ file hoặc là nếu vị trí của con trỏ file không được thể hiện bằng kiểu nguyên

Nếu có bất kỳ lỗi gì xảy ra thì hàm này sẽ trả về giá trị -1

- *The file position is represented as an integer, which counts the number of bytes from the beginning of the file.*

  *long int ftell (FILE *stream)*

  *int fseek (FILE *stream, long int offset, int whence)*

  whence : SEEK_SET, SEEK_CUR, or SEEK_END

  *void rewind (FILE *stream)*

Tiếp theo là hàm fseek của chúng ta sẽ sử dụng để thay đổi vị trí của con trỏ file trong stream, giá trị của nó được xác định từ offset nghĩa là vị trí mà chúng ta muốn thay đổi còn whence là vị trí mà chúng ta xét làm điểm gốc

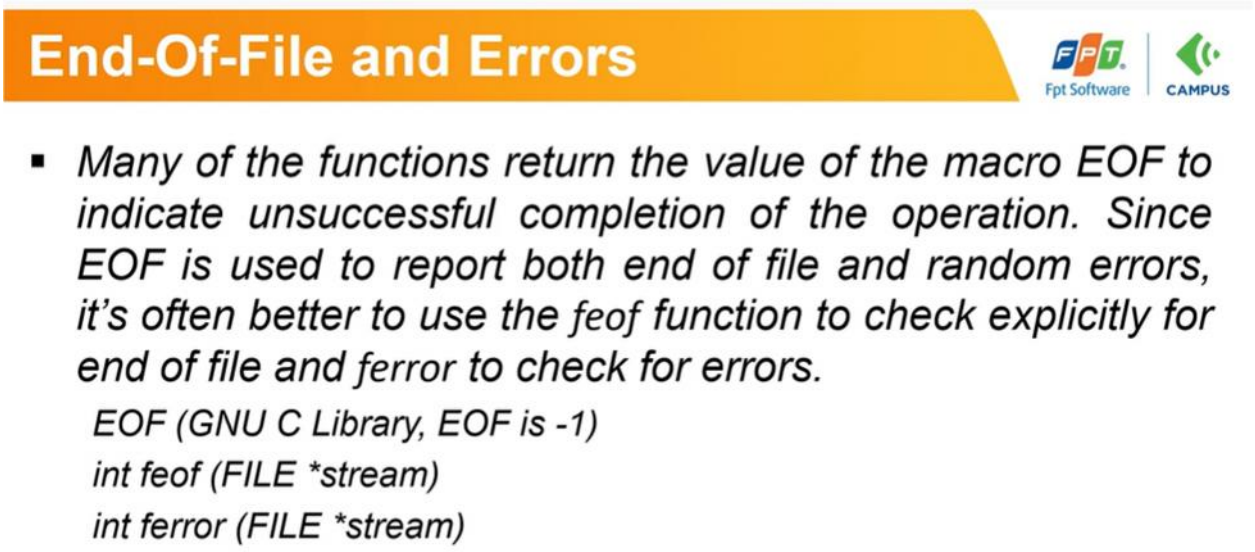
Chẳng hạn như là whence của chúng ta có 3 thông số SEEK_SET, SEEK_CUR, SEEK_END.

Tương ứng với đầu file, vị trí hiện tại và vị trí cuối file

Khi chúng ta điền vào offset là 2 chẳng hạn còn cái whence của chúng ta là SEEK_SET thì có nghĩa là tại vị trí đó trong file chúng ta sẽ muốn con trỏ của chúng ta dịch đi 2 byte nữa so với vị trí đầu và nó trỏ tới vị trí offset hiện tại

Còn hàm rewind sẽ thay đổi vị trí con trỏ file lên đầu file

Nó tương tự như việc gọi hàm fseek với stream và offset là 0 và whence là SEEK_SET



Có rất nhiều các hàm được trả về giá trị là end of file để chỉ ra rằng hệ thống không được chính xác và gặp lỗi trong quá trình hoạt động và end of file thường được sử dụng để report cho hàm check kiểm tra là đã đến cuối file hay chưa hoặc là những lỗi thông thường trong quá trình chạy và chúng ta sẽ làm quen với 2 hàm này

Khi làm việc với file thì chúng ta cần phải biết là đâu là điểm cuối kết thúc của file, thì hàm feof sẽ trả về giá trị khác 0 nếu như đã đến cuối của file đó rồi và trả về giá trị 0 trong các trường hợp còn lại

Hàm feof sẽ giúp chúng ta xác định được xem là đã đến cuối file hay chưa để kết thúc công việc của chúng ta hay không. Điều này rất hữu ích khi chúng ta làm công việc là đọc dòng nhỏ hay những phần nhỏ của 1 file hoặc là check, kiểm tra xem là file đó đã đến cuối hay chưa

Hàm ferror, trong 1 file làm việc với stream thì chúng ta không biết là liệu các thao tác với stream có lỗi phát sinh hay không hoặc liệu có sai sót gì trong quá trình chúng ta chạy chương trình hay không thì hàm ferror sẽ giúp chúng ta kiểm tra điều đó. Các thao tác stream có lỗi gì trước đó hay không và hàm này sẽ trả về giá trị khác 0 nếu mà có 1 lỗi xảy ra trong quá trình làm việc với stream và trả về giá trị bằng 0 trong những trường hợp còn lại. có 1 điểm hạn chế của hàm ferror là hàm này sẽ không cho chúng ta biết cụ thể là

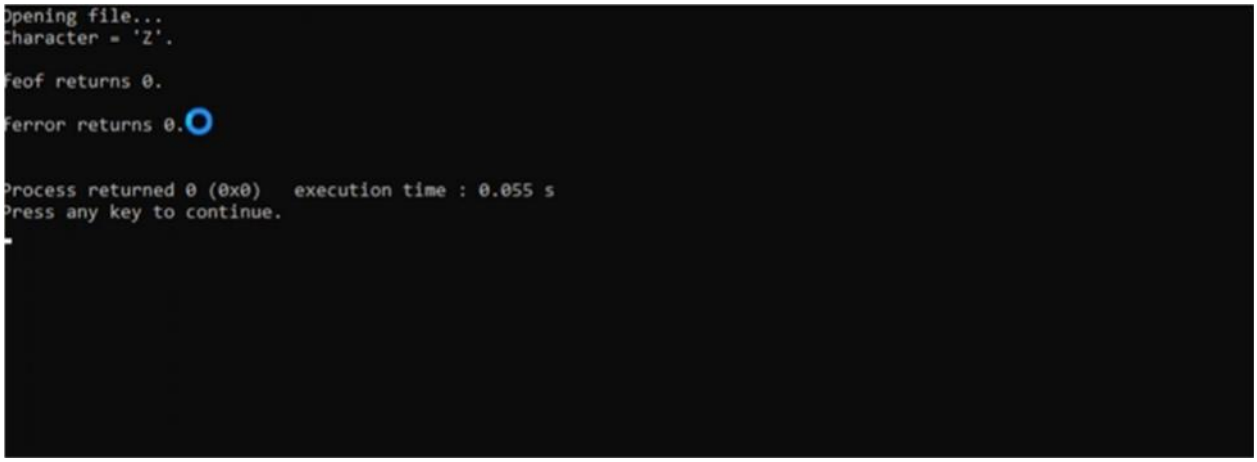
lỗi xảy ra là lỗi gì hoặc là xảy ra ở đâu và chỉ cho chúng ta biết là những lỗi đó có hay không thôi.

Cách làm việc với vị trí của con trỏ file này

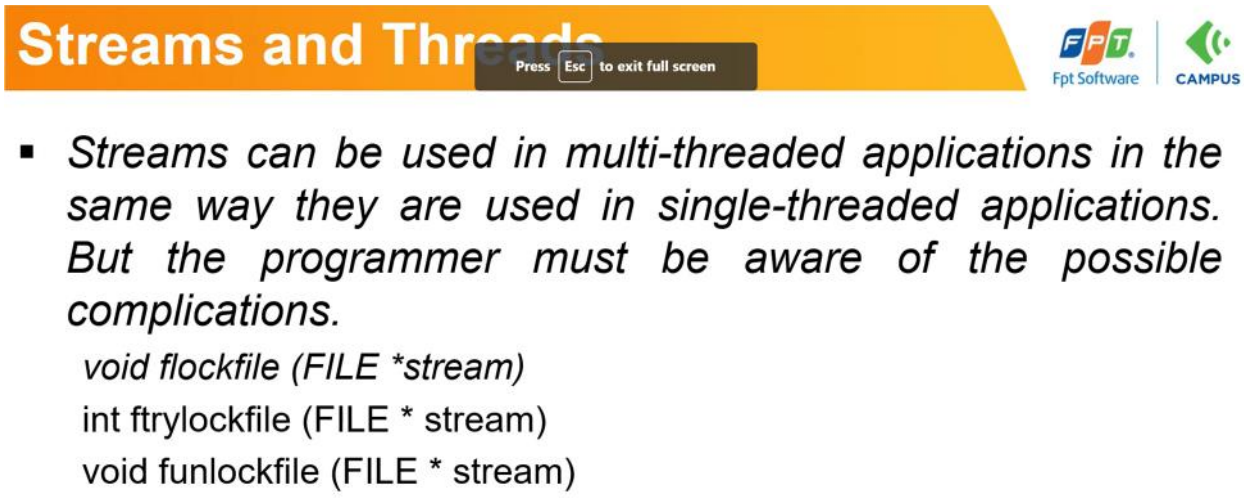
#include <stdio.h>

```
int main()
{
  int input_char;
  FILE *my_stream;
  char my_filename[] = "D:\\example.txt";
  int eof_status, error_status;

  printf ("Opening file...\n");
  my_stream = fopen (my_filename, "w+");
  fprintf (my_stream, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");

  /* Seeking position 25 = 'Z'  */
  fseek (my_stream, 25, SEEK_SET);
  input_char = getc (my_stream);
  printf ("Character = '%c'.\n\n", input_char);

  /* check EOF*/
  eof_status = feof (my_stream);
  printf ("feof returns %d.\n\n", eof_status);

  /* Check error*/
  error_status = ferror (my_stream);
  printf ("ferror returns %d.\n\n", error_status);

  fclose (my_stream);
  return 0;
}
```

Sau đó chúng ta sẽ kiểm tra xem là đã đến cuối file hay chưa và check các lỗi thì như thông thường chúng ta sẽ làm việc với example.txt ở trong ổ D và mở file ghi 1 chuỗi ký tự từ A đến Z và sau đó sẽ dùng hàm fseek để set vị trí con trỏ đến tới vị trí thứ 25 tính theo thứ tự đầu file và sẽ get ký tự tại vị trí con trỏ 25 ra đồng thời là kiểm tra xem là tại vị trí đó thì đã là cuối file hay chưa.



In ra ký tự z, tại vị trí con trỏ, tại vị trí con trỏ 25 thì sẽ ko phải là vị trí cuối file nên giá trị trả về là bằng 0

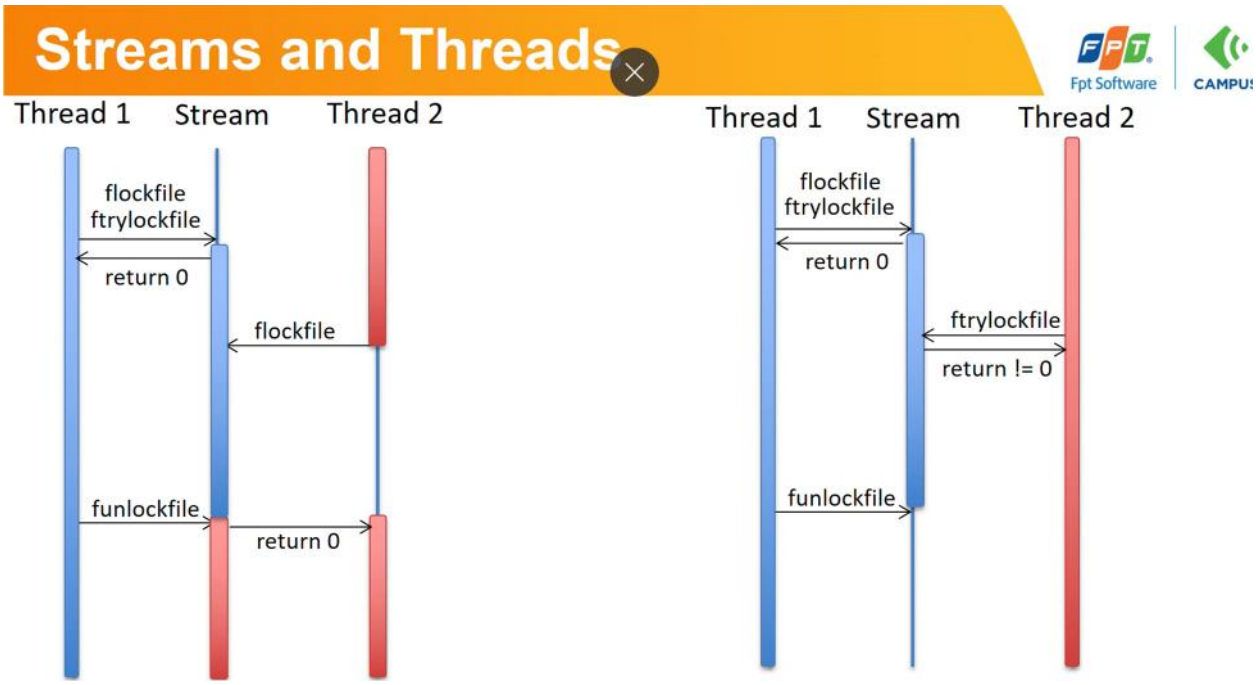


Trong quá trình làm việc của 1 stream trong các hệ điều hành khác nhau như linux thì sẽ xảy ra 1 số vấn đề đó là khi mà nhiều chương trình hoặc là nhiều ứng dụng cùng sử dụng stream đó cho việc đọc viết chẳng hạn sẽ gây ra xung đột vì lúc đó có nhiều luồng cùng vào sửa thay đổi thông tin và dẫn đến làm sai lệch dữ liệu của stream đó và để tránh việc đó xảy ra thì ta có hàm trên để đảm bào khi nhiều luồng ứng dụng cùng làm việc với stream thì sẽ ko gây ra các xung đột trong quá trình làm việc.

Hàm flockfile sẽ đợi cho đến khi stream của chúng ta không bị khoá nữa và nó sẽ chiếm lấy khoá và làm chủ cái stream đó, đồng thời tăng cái account lock lên.

Thì bình thường các stream bị khoá thì sẽ có account lock bằng 0, thứ 2 là hàm ftrylockfile tương tự như hàm flockfile khác ở chỗ với hàm này thì nó sẽ không đợi cho đến stream được mở khoá mà sẽ trả về ngay lập tức còn không, nếu như lấy được khoá và làm chủ được stream còn sẽ trả về giá trị khác 0 nếu stream đang bị khoá bởi thread khác

Cuối cùng là hàm unlockfile mở khoá và đồng thời giảm cái count lock



Cả 2 hình đều có 2 luồng, thread làm việc trên cùng với 1 stream

Hình bên trái, luồng thread 1 sẽ dùng hàm lockfile và try lock file để làm chủ stream này trước

Khi hàm trylock trả về 0 thì có nghĩa là thread1 đã lấy được khoá và làm chủ stream này.

Khi đó thread 2 muốn lấy khoá và làm chủ stream bằng cách gọi hàm lockfile thì phải đợi cho đến khi cái thread1 unlock cái stream thì thread mới làm việc tiếp được

Hình bên phải, thread 2 thay vì dùng hàm lockfile thì sẽ dùng hàm trylock file và với trylock file thì hàm này sẽ trả ngay về cái kết quả trả về khác 0 và nghĩa là stream này đang bị lock bởi 1 thread khác.

Thread 2 này sẽ không thể nào mà lock được stream này và làm chủ được stream, nó sẽ thực hiện tiếp các công việc tiếp theo khi mà thread 2 mà ko phải delay hay là chờ đến khi mà thread1 unlock file nữa. đó là sự khác biệt

Đó là cách các thread cùng làm việc với stream