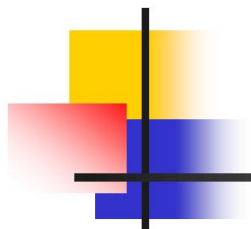




# Xử lý tập tin

Chỉ dành cho Trung tâm Aptech sử dụng

## Phiên 12



# Mục tiêu

---

Giải thích về luồng và tệp

Thảo luận về luồng văn bản và luồng nhị phân Chỉ dành cho Trung tâm Aptech sử dụng

Giải thích các chức năng tệp khác nhau

Giải thích về con trỏ

tệp Thảo luận về con trỏ đang hoạt

động hiện tại Giải thích về các đối số dòng lệnh



# Đầu vào/Đầu ra tập tin

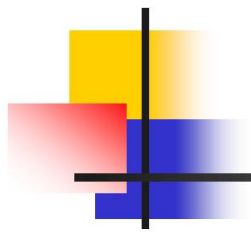
---

Tất cả các hoạt động I/O trong C được thực hiện bằng cách sử dụng các hàm từ thư viện chuẩn

Chỉ dành cho Trung tâm Aptech sử dụng

Cách tiếp cận này làm cho hệ thống tập tin C trở nên rất mạnh mẽ và linh hoạt

I/O trong C là duy nhất vì dữ liệu có thể được truyền dưới dạng biểu diễn nhị phân bên trong hoặc dưới dạng văn bản mà con người có thể đọc được



# Các luồng

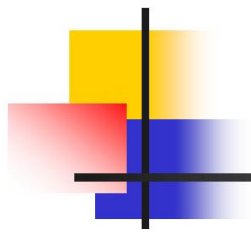
---

Hệ thống tập tin C hoạt động với nhiều loại thiết bị khác nhau bao gồm máy in, ổ đĩa, ổ băng và thiết bị đầu cuối

Mặc dù tất cả các thiết bị này rất khác nhau, hệ thống tập tin đệm chuyển đổi mỗi thiết bị thành một thiết bị logic được gọi là luồng

Vì tất cả các luồng đều hoạt động tương tự nhau nên việc xử lý rất dễ dàng các thiết bị khác nhau

Có hai loại luồng - luồng văn bản và luồng nhị phân  
suối



# Luồng văn bản

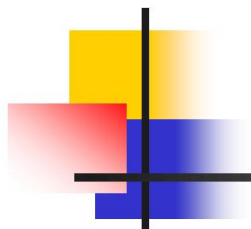
---

Một luồng văn bản là một chuỗi các ký tự có thể được sắp xếp vào các dòng được kết thúc bằng ký tự xuống dòng

Trong luồng văn bản, một số bản dịch ký tự có thể xảy ra theo yêu cầu của môi trường

Do đó, có thể không có mối quan hệ một-một giữa các ký tự được viết (hoặc đọc) và các ký tự trong thiết bị bên ngoài

Ngoài ra, do có thể có các bản dịch, số lượng ký tự được viết (hoặc đọc) có thể không giống với số lượng ký tự trong thiết bị bên ngoài



# Luồng nhị phân

---

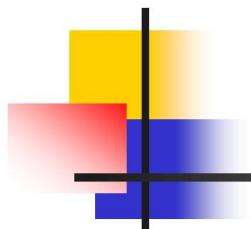
Luồng nhị phân là một chuỗi các byte có sự tương ứng một-một với các byte trong thiết bị bên ngoài, nghĩa là không có sự dịch chuyển ký tự

Chỉ dành cho Trung tâm Aptech sử dụng

Số lượng byte được ghi (hoặc đọc) giống như số trên thiết bị bên ngoài

Luồng nhị phân là một chuỗi byte phẳng, không có cờ nào để chỉ ra kết thúc tệp hoặc kết thúc bản ghi

Phần cuối của tệp được xác định bởi kích thước của tệp



# Tập tin

---

Một tập tin có thể tham chiếu đến bất cứ thứ gì từ một tập tin đĩa đến một thiết bị đầu cuối hoặc một máy in

Một tệp được liên kết với một luồng bằng cách thực hiện một thao tác mở và được hủy liên kết bằng một thao tác đóng

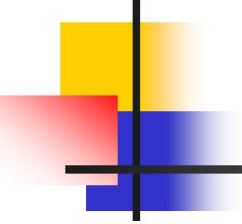
Khi một chương trình kết thúc bình thường, tất cả các tập tin sẽ tự động đóng lại

Khi một chương trình bị sập, các tập tin vẫn mở

# Các chức năng cơ bản của tập tin

Tên	Chức năng
<code>mở()</code>	Mở một tập tin
<code>đóng( )</code>	Đóng một tập tin
<code>fputc( )</code>	Ghi một ký tự vào một tập tin
<code>fgetc( )</code>	Đọc một ký tự từ một tập tin
<code>đọc()</code>	Đọc từ một tập tin vào bộ đệm
<code>viết()</code>	Ghi từ bộ đệm vào tệp
<code>tìm kiếm()</code>	Tìm kiếm một vị trí cụ thể trong tập tin
<code>inf( )</code>	Hoạt động giống như <code>printf()</code> , nhưng trên một tập tin
<code>fscanf()</code>	Hoạt động giống như <code>scanf()</code> , nhưng trên một tập tin
<code>của ( )</code>	Trả về true nếu đạt đến cuối tệp
<code>lỗi( )</code>	Trả về true nếu có lỗi xảy ra
<code>tua lại( )</code>	Đặt lại vị trí tệp về đầu tệp
<code>di dời( )</code>	Xóa một tập tin
<code>xả( )</code>	Ghi dữ liệu từ bộ đệm bên trong vào một tệp được chỉ định





# Con trỏ tập tin

---

Con trỏ tệp là cần thiết để đọc hoặc ghi tệp

Nó là con trỏ đến một cấu trúc chứa tên tệp, vị trí hiện tại của tệp, tệp đang được đọc hay ghi và có xảy ra lỗi hoặc kết thúc tệp không

Chỉ dành cho Trung tâm Aptech sử dụng

Các định nghĩa thu được từ `stdio.h` bao gồm một cấu trúc khai báo được gọi là `FILE`

Khai báo duy nhất cần thiết cho một con trỏ tệp là:

**TỆP \*fp**



# Mở một tập tin văn bản

Hàm `fopen()` mở một luồng để sử dụng và liên kết một tệp với luồng đó suốt

Hàm `fopen()` trả về một con trỏ tệp được liên kết với tệp  
Nguyên mẫu cho hàm `fopen()` là:

**TẬP TIN** `*fopen(const char *tên tệp, const char *chế độ);`

Cách thức	Nghĩa
<code>r</code>	Mở một tập tin văn bản để đọc
<code>w</code>	Tạo một tập tin văn bản để viết
<code>a</code>	Thêm vào một tập tin văn bản
<code>r+</code>	Mở một tập tin văn bản để đọc/ghi
<code>w+</code>	Tạo một tập tin văn bản để đọc/ghi
<code>a+</code>	Thêm hoặc tạo một tập tin văn bản để đọc/ghi



# Đóng một tập tin văn bản

---

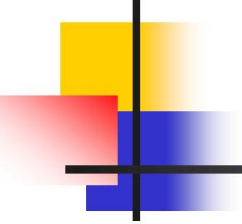
Điều quan trọng là phải đóng một tệp sau khi đã sử dụng  
Điều này giải phóng tài nguyên hệ thống và giảm nguy cơ vượt quá giới hạn số tệp  
có thể mở. Đóng một luồng sẽ xóa sạch mọi bộ đệm liên  
quan, một thao tác quan trọng giúp ngăn ngừa mất dữ liệu khi ghi vào đĩa.

Chỉ sử dụng tại Trung tâm Aptech

Hàm `fclose()` đóng một luồng được mở bằng lệnh gọi tới `fopen()`. Nguyên mẫu  
của `fclose()`  
là:

```
int fclose(TỆP *fp);
```

Hàm `fcloseall()` đóng tất cả các luồng đang mở.



# Viết một ký tự - Tập văn bản

---

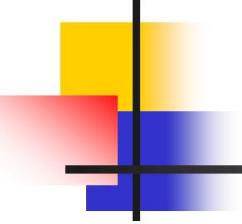
Các luồng có thể đư ợc ghi vào từng ký tự hoặc dư ới dạng chuỗi

Chỉ dành cho Trung tâm Aptech sử dụng

Hàm `fputc()` đư ợc sử dụng để ghi các ký tự vào một tệp đã đư ợc mở trư ớc đó bởi `fopen()`

Nguyên mẫu là:

```
int fputc(int ch, TẬP *fp);
```



# Đọc một ký tự – Tập văn bản

---

Hàm `fgetc()` đư ợc sử dụng để đọc các ký tự từ một tệp đư ợc mở ở chế độ đọc, sử dụng `fopen()`

Nguyên mẫu là: `int`  
`fgetc(int ch, FILE *fp);`

Hàm `fgetc()` trả về ký tự tiếp theo từ vị trí hiện tại trong luồng đầu vào và tăng chỉ báo vị trí tệp



# Chuỗi 1/0

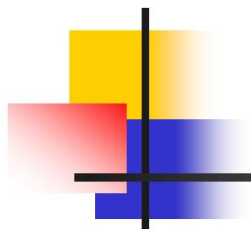
---

Các hàm `fputs()` và `fgets()` ghi và đọc các chuỗi ký tự vào và ra khỏi một tệp đĩa. Hàm `fputs()` ghi toàn bộ chuỗi vào luồng được chỉ định. Hàm `fgets()` đọc một chuỗi từ luồng được chỉ định cho đến khi một ký tự xuống dòng được đọc hoặc các ký tự có độ dài 1 được đọc. Các nguyên mẫu là:

Chỉ sử dụng tại Trung tâm Aptech

`int fputs(const char *str, TỆP *fp);`

`char *fgets(char *str, int độ dài, TỆP *fp);`



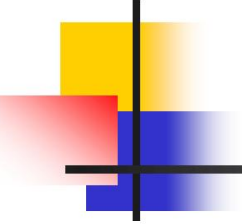
# Mở một File-Binary

Hàm `fopen()` mở một luồng để sử dụng và liên kết một tệp với luồng đó suốt

Hàm `fopen()` trả về một con trỏ tệp được liên kết với tệp  
Nguyên mẫu cho hàm `fopen()` là:

**TẬP TIN** `*fopen(const char *tên tệp, const char *chế độ);`

Cách thức	Nghĩa
rb	Mở một tập tin nhị phân để đọc
wb	Tạo một tập tin nhị phân để ghi
b	Thêm vào một tập tin nhị phân
r+b	Mở một tập tin nhị phân để đọc/ghi
w+b	Tạo một tập tin nhị phân để đọc/ghi
a+b	Thêm một tệp nhị phân để đọc/ghi



# Đóng một tập tin nhị phân

---

Hàm `fclose()` đóng một luồng đư ợc mở bằng lệnh gọi `fopen()`

Chỉ dành cho Trung tâm Aptech sử dụng

Nguyên mẫu cho `fclose()` là:

```
int fclose(TỆP *fp);
```



# Các hàm fread() và fwrite()

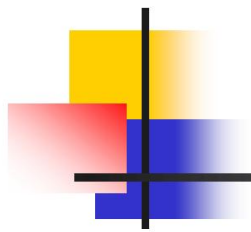
Các hàm fread() và fwrite() đư ợc gọi là đọc không định dạng hoặc viết các hàm

Chúng đư ợc sử dụng để đọc và ghi toàn bộ khối dữ liệu vào và ra khỏi một tài liệu

Ứng dụng hữu ích nhất liên quan đến việc đọc và ghi các kiểu dữ liệu do ngư ời dùng định nghĩa, đặc biệt là các cấu trúc Các nguyên mẫu cho các hàm là:

```
size_t fread(void *buffer, size_t số lượng byte, size_t số lượng, TẬP *fp);
```

```
size_t fwrite(const void *buffer, size_t số byte, size_t số lượng, TẬP *fp);
```



# Sử dụng feof()

---

Hàm feof() trả về true nếu đã đến cuối tệp ,  
nếu không thì trả về false (0)

Chỉ dành cho Trung tâm Aptech sử dụng

Hàm này đư ợc sử dụng khi đọc dữ liệu nhị phân

Nguyên mẫu là:

```
int feof (TỆP *fp);
```



# Hàm `rewind()`

---

Hàm `rewind()` đặt lại chỉ báo vị trí tệp về đầu tệp

Chỉ dành cho Trung tâm Aptech sử dụng  
Nó lấy con trỏ tệp làm đối số của nó

Cú pháp:

```
tua lại(fp);
```



# Hàm `error()`

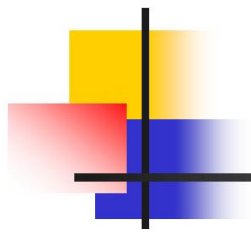
---

Hàm `error()` xác định xem thao tác tệp có tạo ra lỗi hay không

Vì mỗi thao tác đặt điều kiện lỗi, nên `error()` phải được gọi ngay sau mỗi thao tác; nếu không, lỗi có thể bị mất

Nguyên mẫu của nó là:

```
int error(TỆP *fp);
```



# Xóa tập tin

---

Hàm `remove()` xóa một tập tin được chỉ định

Nguyên mẫu của nó là: *Chỉ dành cho Trung tâm Aptech sử dụng*

```
int remove(char *tên tệp);
```



# Dòng nư ớc xả

---

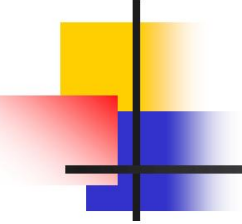
Hàm `fflush()` xóa bộ đệm tùy thuộc vào loại tệp. Một tệp đư ợc mở để đọc sẽ xóa bộ đệm đầu vào, trong khi một tệp đư ợc mở để ghi sẽ ghi bộ đệm đầu ra vào các tệp. Nguyên mẫu của nó là:

Chỉ sử dụng tại Trung tâm Aptech

Vì

```
int fflush(TỆP *fp);
```

Hàm `fflush()`, với giá trị `null`, sẽ xóa tất cả các tệp đư ợc mở để xuất ra.



# Các luồng tiêu chuẩn

---

Bất cứ khi nào một chương trình C bắt đầu thực thi dưới DOS, năm luồng đặc biệt sẽ được hệ điều hành tự động mở

Chỉ dành cho Trung tâm Aptech sử dụng

- Đầu vào chuẩn (stdin)
- Đầu ra chuẩn (stdout)
- Lỗi chuẩn (stderr)
- Máy in chuẩn (stdprn)
- Trợ động từ chuẩn (stdaux)



# Con trỏ đang hoạt động hiện tại

---

Một con trỏ đư ợc duy trì trong cấu trúc FILE để theo dõi vị trí diễn ra các hoạt động I/O

Bất cứ khi nào một ký tự đư ợc đọc từ hoặc ghi vào luồng, con trỏ đang hoạt động hiện tại (đư ợc gọi là curp) sẽ đư ợc nâng cao

Vị trí hiện tại của con trỏ đang hoạt động có thể đư ợc tìm thấy với sự trợ giúp của hàm `ftell()`

Nguyên mẫu là:

```
int dài ftell(FILE *fp);
```





# Thiết lập vị trí hiện tại-1

---

Hàm `fseek()` định vị lại curp theo số byte được chỉ định từ vị trí bắt đầu, vị trí hiện tại hoặc vị trí cuối của luồng tùy thuộc vào vị trí được chỉ định trong hàm `fseek()`

Chỉ dành cho Trung tâm Aptech sử dụng

Nguyên mẫu là:

```
int fseek (FILE *fp, độ lệch int dài, gốc int);
```



# Thiết lập vị trí hiện tại-2

Nguồn gốc chỉ ra vị trí bắt đầu của tìm kiếm và có giá trị như sau:

Nguồn gốc	Vị trí tập tin
SEEK_SET hoặc 0	Bắt đầu tập tin
SEEK_CUR hoặc 1	Vị trí con trỏ tệp hiện tại
SEEK_END hoặc 2	Kết thúc tệp



# fprintf() và fscanf()-1

---

Hệ thống I/O đệm bao gồm các hàm fprintf() và fscanf() tương tự như printf() và scanf() ngoại trừ việc chúng hoạt động với các tệp

Các nguyên mẫu của là:

```
int fprintf(TẬP TIN * fp, const char *chuỗi_điều_khiển,...);
```

```
int fscanf(TẬP *fp, const char *chuỗi_điều_khiển,...);
```



# `fprintf()` và `fscanf()` -2

---

`fprintf()` và `fscanf()` mặc dù dễ nhất nhưng không phải lúc nào cũng hiệu quả nhất

Chỉ dành cho Trung tâm Aptech sử dụng

Chi phí phát sinh thêm với mỗi cuộc gọi vì dữ liệu được ghi ở dạng dữ liệu ASCII được định dạng thay vì định dạng nhị phân

Vì vậy, nếu tốc độ hoặc kích thước tệp là mối quan tâm, `fread()` và `fwrite()` là lựa chọn tốt hơn