



Xử lý tập tin

Phiên 12

Chỉ dành cho Trung tâm Aptech sử dụng



Mục tiêu

- Giải thích các luồng và tập tin
- Thảo luận về luồng văn bản và luồng nhị phân
- Giải thích các chức năng khác nhau của tập tin
- Giải thích con trỏ tập tin
- Thảo luận về con trỏ đang hoạt động hiện tại
- Giải thích 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
- 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 các thiết bị khác nhau rất dễ dàng
- Có hai loại luồng - luồng văn bản và luồng nhị phân



Luồng văn bản

- Một luồng văn bản là một chuỗi các ký tự có thể được sắp xếp thành các dòng kết thúc bằng một ký tự xuống dòng
- Trong một luồng văn bản, một số bản dịch ký tự nhất định có thể xảy ra tùy 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 bản dịch ký tự
- Số byte được ghi (hoặc đọc) giống với số trên thiết bị ngoài
- Luồng nhị phân là một chuỗi byte phẳng, không có bất kỳ cờ nào để chỉ ra kết thúc của tệp hoặc kết thúc của bản ghi
- Phần cuối của tệp tin được xác định bởi kích thước của tệp tin

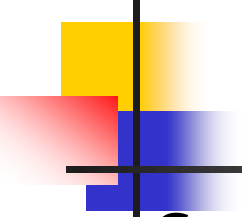


Tập tin

- 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 thao tác mở và hủy liên kết bằng 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
mở()	Mở một tập tin
đóng()	Đóng một tập tin
fputc ()	Ghi một ký tự vào một tập tin
fgetc ()	Đọc một ký tự từ một tập tin
đọc()	Đọc từ một tập tin vào bộ đệm
viết()	Ghi từ bộ đệm vào tệp
tìm kiếm()	Tìm kiếm một vị trí cụ thể trong tập tin
inf()	Hoạt động giống như printf(), nhưng trên một tập tin
fscanf()	Hoạt động giống như scanf(), nhưng trên một tập tin
của ()	Trở về true nếu đạt đến cuối tệp
lỗi()	Trả về true nếu có lỗi xảy ra
tua lại()	Đặt lại vị trí tệp về đầu tệp
di dời()	Xóa một tập tin
xả()	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
- Đây 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 hay không.
- Các định nghĩa thu được từ stdio.h bao gồm một khai báo cấu trúc 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
- 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
r	Mở một tập tin văn bản để đọc
w	Tạo một tập tin văn bản để viết
a	Thêm vào một tập tin văn bản
r+	Mở một tập tin văn bản để đọc/ghi
w+	Tạo một tập tin văn bản để đọc/ghi
a+	Thêm hoặc tạo một tập tin văn bản để đọc/ghi

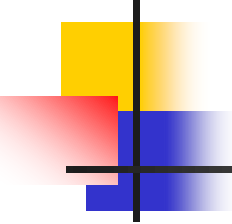


Đóng một tập tin văn bản

- Điều quan trọng là phải đóng một tập tin sau khi đã sử dụng
- Điều này giải phóng tài nguyên hệ thống và giảm nguy cơ vượt quá giới hạn số tập có thể mở
- Đóng một luồng sẽ xóa sạch mọi bộ đệm liên quan, một hoạt động quan trọng giúp ngăn ngừa mất dữ liệu khi ghi vào đĩa
- Hàm `fclose()` đóng một luồng được mở bằng lệnh gọi `fopen()`
- Nguyên mẫu cho `fclose()` là:

`int fclose(TẬP *fp);`

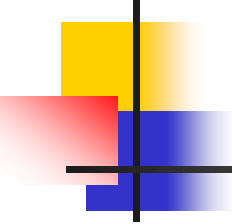
- Hàm `fcloseall()` đóng tất cả các luồng đang mở



Viết một ký tự – Tập văn bản

- Các luồng có thể được ghi vào từng ký tự hoặc thành chuỗi
- Hàm `fputc()` được sử dụng để ghi các ký tự vào một tệp đã được mở trước đó bởi `fopen()`
- Nguyên mẫu là:

`int fputc(int ch, TỆP *fp);`



Đọc một ký tự – Tập văn bản

- Hàm `fgetc()` được sử dụng để đọc các ký tự từ một tệp được mở ở chế độ đọc, sử dụng `fopen()`
- Nguyên mẫu là:
`int fgetc(int ch, TẬP *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 I/O

- 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 đã chỉ định
- Hàm fgets() đọc một chuỗi từ luồng được chỉ định cho đến khi một ký tự dòng mới được đọc hoặc các ký tự có độ dài 1 đã được đọc
- Các nguyên mẫu là:

```
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
- 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 được mở bằng lệnh gọi `fopen()`
- Nguyên mẫu cho `fclose()` là:

`int fclose(TỆP *fp);`



fread() và fwrite() chức năng

- Các hàm fread() và fwrite() được gọi là các hàm đọc hoặc ghi không định dạng
- Chúng được sử dụng để đọc và ghi toàn bộ khối dữ liệu vào và ra khỏi một tệp
- Ứng dụng hữu ích nhất liên quan đến việc đọc và ghi các kiểu dữ liệu do người dùng xác định, đặc biệt là các cấu trúc
- Nguyên mẫu cho các chức năng là:

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

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



Sử dụng feof()

- Hàm feof() trả về true nếu đã đến cuối tệp, nếu không thì trả về false (0)
- Chức năng này được 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
- Nó lấy con trỏ tệp làm đối số của nó
- Cú pháp:

tua lại(fp);



Hàm ferror()

- Hàm ferror() xác định xem thao tác tệp có tạo ra lỗi hay không
- Vì mỗi thao tác đặt điều kiện lỗi, nên ferror() phải được gọi ngay sau mỗi thao tác; nếu không, lỗi có thể bị mất
- Nguyên mẫu của nó là:

int ferror(TỆP *fp);



Xóa tập tin

- Hàm `remove()` xóa một tập tin được chỉ định
- Nguyên mẫu của nó là:

`int remove(char *tên tệp);`



Dòng nước xả

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

`int fflush(TỆP *fp);`

- Hàm `fflush()`, với giá trị null, sẽ xóa tất cả các tệp được 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

- 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 được 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 được đọc từ hoặc ghi vào luồng, con trỏ đang hoạt động hiện tại (được gọi là curp) sẽ được nâng cao
- Vị trí hiện tại của con trỏ đang hoạt động có thể được 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()`
- 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 tập tin hiện tại
SEEK_END hoặc 2	Kết thúc tập tin



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(TỆP *fp, const char *chuỗi_điều_khiển,...);



fprintf() và fscanf()-2

- Mặc dù fprintf() và fscanf() là dễ nhất, nhưng không phải lúc nào cũng hiệu quả nhất
- Chi phí phát sinh thêm cho 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