Basic C Programming
Bài 2
(Lớp học lần 2)

Nội dung

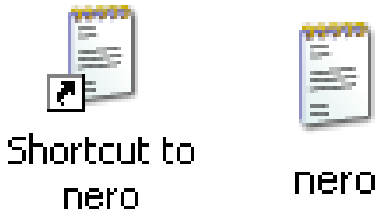
- File văn bản
- File nhị phân

Nội dung

- 1. Khái niệm và phân loại file
- 2. Các thao tác với file
  - 2.1. Khai báo
  - 2.2. Mở file
  - 2.3. Đóng file
  - 2.4. Truy nhập file nhị phân
  - 2.5. Truy nhập file văn bản

1. Khái niệm và phân loại file

- Khái niệm
  - file dữ liệu là một tập hợp các dữ liệu có liên quan đến nhau, được lưu ở bộ nhớ ngoài



Biểu tượng file

1. Khái niệm và phân loại tệp

- Phân loại
  - File văn bản (text file)
    - Các phần tử là các kí tự: chữ cái, chữ số, dấu câu, dấu cách, kí tự điều khiển
    - Khi mở bằng các phần mềm soạn thảo văn bản, có thể xem được nội dung file
  - File nhị phân (binary file)
    - Các phần tử là các số nhị phân 0, 1 mã hóa thông tin.
    - Thông tin được mã hóa: số nguyên, kí tự…
    - Tệp văn bản là trường hợp riêng của tệp nhị phân

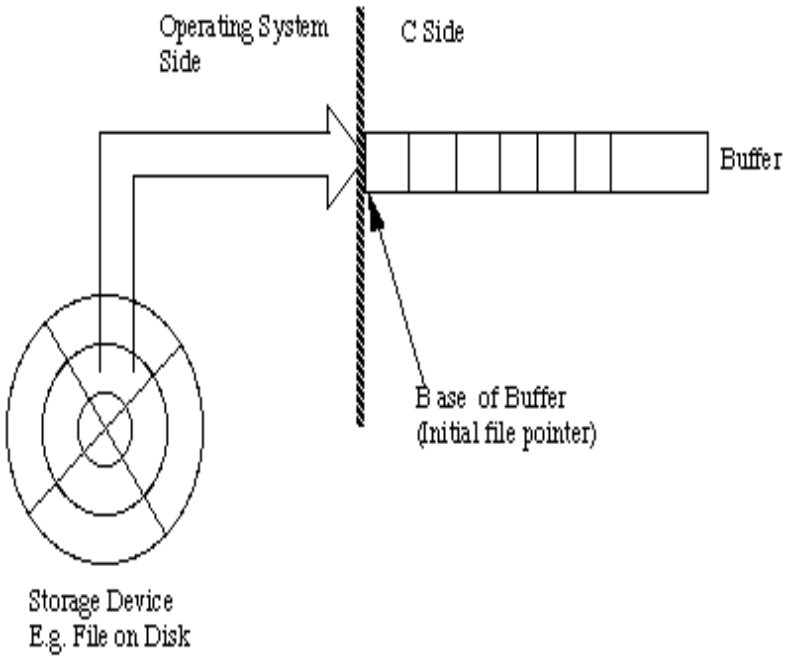
1. Khái niệm và phân loại file

- Ý nghĩa của file
  - Cất giữ dữ liệu lâu dài
- Phân biệt file và mảng
  - Khác
    - Mảng: lưu trữ ở bộ nhớ trong, kích thước bị hạn chế
    - file: lưu trữ ở bộ nhớ ngoài, kích thước lớn hơn mảng rất nhiều

1. Khái niệm và phân loại file

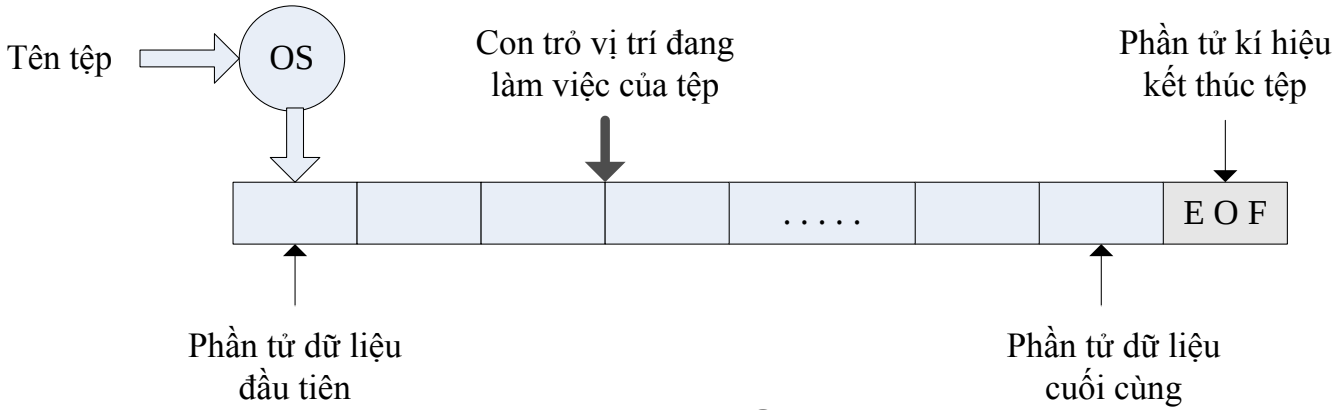
Kênh xuất nhập trong C

- Là một vùng đệm dùng cho việc nhập xuất dữ liệu ở mức cao
- Chương trình chỉ đọc và ghi dữ liệu trên vùng đệm, do đó các hàm vào ra độc lập với thiết bị đầu cuối
- Hệ điều hành đảm nhiệm việc đồng bộ hoá dữ liệu trên vùng đệm với thiết bị vào ra



1. Khái niệm và phân loại file

- Tổ chức
  - Phần tử kết thúc file: EOF (End Of File idicator)
  - EOF: -1 trong stdio.h
  - Con trỏ file: con trỏ xác định vị trí đang làm việc của file



1. Khái niệm và phân loại file

- Quy trình thao tác với file
  - Khai báo file
  - Mở file để làm việc
  - Truy nhập file
  - Đóng file

2. Các thao tác với file

- 2.1. Khai báo
  - Truy nhập file thông qua con trỏ file
  - ***Cú pháp*:**

    FILE *tên_con_trỏ_file;
  - Ví dụ: FILE * f1, * f2;

2.2. Mở file

- ***Cú pháp***

tên_con_trỏ_file = fopen(tên_file, chế_độ_mở_file);

- Chế độ mở file
  - Phụ thuộc vào mục đích sử dụng file: read (r), write (w), read write…
  - Loại file: văn bản (t), nhị phân (b)

2.2. Mở file

“r”	Đọc (tệp đã tồn tại)
“w”	Ghi (tạo mới nếu chưa có)
“a”	Ghi thêm vào cuối (tạo mới nếu chưa có)
“r+”	Đọc và ghi (tệp đã tồn tại)
“w+”	Đọc và ghi (tạo mới nếu không có tệp)
“a+”	Đọc và ghi thêm vào cuối (tạo mới nếu chưa có)

2.2. Mở file

- Loại file

Kí hiệu	Loại file
“b”	file nhị phân
“t”	file văn bản

- Ví dụ: FILE * f1, * f2, *f3;
  - Để mở file *c:\abc.txt* để đọc ta dùng lệnh

    f1 = fopen("c:\\abc.txt", "rt");
  - Để mở file *c:\ho_so.dat* để ghi ta dùng lệnh

    f2 = fopen("c:\\ho_so.dat", "wt");
  - Để mở file *c:\abc.txt* để vừa đọc và ghi ta dùng lệnh

    f3 = fopen("c:\\abc.txt", "r+t");

2.2. Mở file

- Chú ý:
  - Trong C ngầm định là file văn bản. Do vậy có thể bỏ qua "t" trong chế độ mở file nếu mở file văn bản
  - Để bắt lỗi mở file không thành công

```
if((con_trỏ_file = fopen(tên_file, chế_độ_mở_file)) ==NULL)
{
    <Xử lí cho trường hợp mở file không thành công>
}else // Trường hợp mở file thành công
{
    <Xử lí khi mở file thành công>
}
```

2.3. Đóng file

- Đảm bảo những thay đổi dữ liệu được ghi lại trên file
- ***int fclose(FILE* <tên con trỏ file>);***
  - Thành công: 0
  - Ngược lại: EOF

2.4. Truy nhập file văn bản

- Ghi dữ liệu lên file
- Sử dụng: **fprintf(), fputs(), putc()**
- **fprintf()**
  - int fprintf(FILE* con_trỏ_file, xâu_định_dạng, [danh_sách_tham_số]);
  - Khác: printf in ra thiết bị ra chuẩn là màn hình (stdout), fprintf() phải chỉ ra con trỏ file ghi dữ liệu.
  - Thành công: số bytes ghi dữ liệu
  - Thất bại: EOF
  - Ví dụ: fprintf(fptr, “%d”, a);

2.4. Truy nhập file văn bản

- fputs()
  - int fputs(char* xâu_kí_tự, FILE* con_trỏ_file);
  - Ghi nội dung của xâu_kí_tự lên con_trỏ_file.
  - Khác puts ở chỗ, puts thêm kí tự xuống dòng sau khi hiển thị dữ liệu
  - Thành công: kí tự cuối cùng được ghi
  - Thất bại: EOF
  - Ví dụ fputs(“Ha Noi”,fptr);

2.3. Truy nhập file văn bản

- **putc()**
  - int putc(int ch, FILE* con_trỏ_file);
  - Ghi kí tự được chứa trong biến ch lên file
  - Thành công: số nguyên là mã ASCII của kí tự
  - Ngược lại: EOF
  - Ví dụ: putc('a',fptr);
  - Demo: dF.c

Bài tập

- Viết chương trình cho phép:
  - Nhập từ bàn phím xâu kí tự str
  - Nhập từ bàn phím 2 số nguyên a, b
  - Nhập từ bàn phím kí tự ch
  - Ghi ra file demo.txt theo định dạng
    - Dòng 1. Ghi a-dấu cách-kí tự ch-dấu cách-b
    - Dòng 2. Ghi xâu str
  - Khi kết thúc chương trình, mở file demo.txt để xem kết quả

2.4. Truy nhập file văn bản

- Đọc dữ liệu từ file.
- Sử dụng hàm: **fscanf(), fgets(), getc();**
- **fscanf()**
  - int fscanf(FILE* con_trỏ_file, xâu_định_dạng, [danh_sách_địa_chỉ]);
  - Đọc dữ liệu từ file: con trỏ file
  - Định dạng đọc: xâu định dạng; Lưu vào dsách địa chỉ
  - Thành công: số byte đọc được. Thất bại: EOF
  - Ví dụ: fscanf(fptr, “%d %c %d”,&a, &ch,&b);

2.4. Truy nhập file văn bản

- **fgets()**
  - char* fgets(char* xâu_kí_tự, int n, FILE* con_trỏ_file);
  - Đọc từ file một xâu kí tự và gán cho biến xâu_kí_tự
  - Việc đọc dừng khi đọc được đủ n-1 kí tự hoặc gặp dấu xuống dòng
  - Thành công: xâu kí tự được trỏ bởi xâu_kí_tự.
  - Thất bại: trả về con trỏ NULL
  - Ví dụ: fgets(hoten, 20, fptr);

2.4. Truy nhập file văn bản

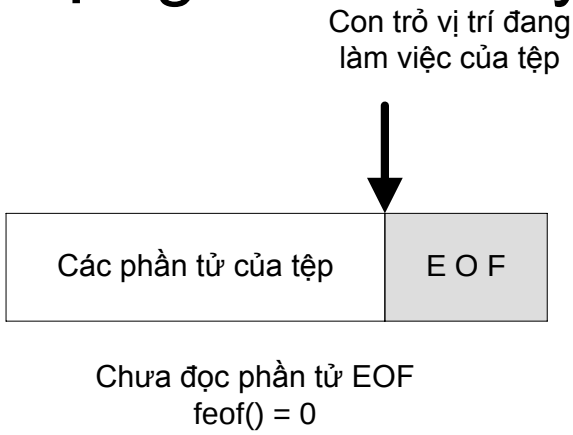
- **getc()**
  - int getc(FILE* con trỏ file);
  - Đọc một kí tự từ file và trả về một số nguyên tương ứng
  - Thành công: kí tự được đọc (dạng int)
  - Thất bại: trả về EOF
  - Demo: dF.c

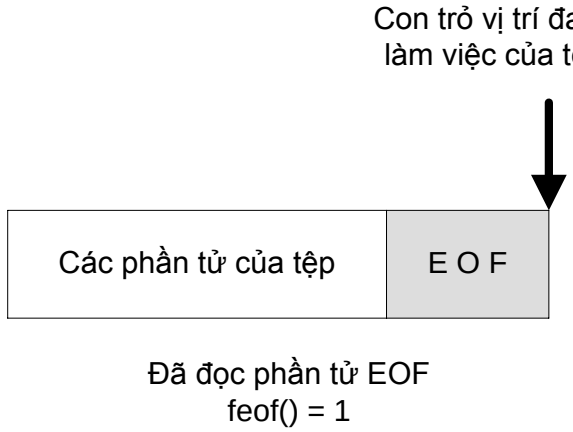
Bài tập

- Tiếp theo bài tập đã làm
- Mở file demo.txt
  - Đọc nội dung dòng 1 để hiển thị a, b, ch
  - Đọc xâu có độ dài tối đa 20 kí tự ở dòng tiếp theo và hiển thị ra màn hình

2.4. Truy nhập file văn bản

- **feof()**
  - int feof(FILE* con_trỏ_file);
  - Kiểm tra xem đã duyệt đến cuối file hay chưa
  - Kiểm tra phần tử EOF đã được đọc trong lần đọc gần nhất hay chưa





2.4. Truy nhập file văn bản

- Đọc
  - fscanf()
  - fgets()
  - getc()
- Thất bại
  - Trả về EOF (trừ fgets)

- Ghi
  - fprintf
  - fputs
  - putc
- Thất bại
  - Trả về EOF

2.5. Truy nhập file nhị phân

- Ghi dữ liệu
  - int fwrite(void *địa_chỉ_biến, int số_byte, int số_mục, FILE *con trỏ file);
  - Đọc một vùng dữ liệu có địa chỉ bắt đầu là địa_chỉ_biến và có kích thước số_byte * số_mục bytes rồi ghi lên file
  - Thành công: số mục ghi lên file
  - Thất bại: trả về 0.

2.5. Truy nhập file nhị phân

- Đọc dữ liệu trên file
  - int fread(void *địa_chỉ_biến, int số_byte, int số_mục, FILE *con_trỏ_file);
  - Đọc một khối dữ liệu kích thước số_byte * số_mục bytes rồi ghi lên vùng nhớ có địa chỉ là địa_chỉ_biến.
  - Thành công: trả về số mục đọc được
  - Thất bại: trả về 0
  - Demo: dFb.c

Bài tập

- Tạo file **demo.dat** và ghi những nội dung sau
  - Nhập từ bàn phím xâu kí tự str
  - Nhập từ bàn phím 2 số nguyên a, b
  - Nhập từ bàn phím kí tự ch
  - Ghi ra file theo định dạng
    - Dòng 1. Ghi a, rồi đến ch rồi đến b
    - Dòng 2. Ghi xâu str
  - Xậu khi kết thúc chương trình, mở file **demo.dat** để xem kết quả
  - Sau đó đọc lại dữ liệu từ file vào lại các biến a, b, ch, str. Hiển thị các biến này ra màn hình.

2.5. Truy nhập file nhị phân

- Dịch chuyển con trỏ file
  - Ghi nhớ các cặp hàm có chức năng đối ngẫu nhau
    - fread() – fwrite()
    - fscanf() – fprintf()
    - fputs() – fgets()
    - getc() – putc()
  - Thất bại trong file văn bản: EOF, file nhị phân: 0 trừ fgets trả về NULL

Lỗi cần tránh

- Sử dụng hàm feof chưa đúng
- Hiện tượng: hiển thị (đọc) thừa dữ liệu cuối cùng
- Ví dụ dErrorEOF.c
- Cách khắc phục
  - Phổ biến: sử dụng các hàm đọc dữ liệu. Đọc chừng nào còn thành công

Nên sử dụng

- Sử dụng file nhị phân khi có thể
- Ưu điểm
  - Hỗ trợ nhiều hàm đọc/ghi hơn file văn bản
  - Hàm fread, fwrite cho phép đọc/ghi cùng lúc một cấu trúc/mảng các phần tử cùng kiểu

Thảo luận