KỸ THUẬT LẬP TRÌNH C

Chương 1: Tập tin

1. Khái niệm tập tin
2. Phân loại tập tin
3. Các bước xử lý tập tin
4. Con trỏ tập tin
5. Vùng đệm
6. Các hàm thao tác tập tin

1. Khái niệm tập tin

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong bộ nhớ RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này C cung cấp cho ta các hàm để lưu trữ và truy xuất tập tin, đó là kiểu **FILE** .

2. Phân loại tập tin

a. Tập tin văn bản (text file)

Nội dung tập tin được lưu trữ dưới dạng văn bản gồm nhiều dòng. Hai dòng liên tiếp nhau được ngăn cách bởi ký hiệu ngăn cách dòng (gồm hai ký tự '\r' (mã 13) và '\n' (mã 10)).

Ví dụ: Các tập tin: *.cpp, *.txt, ... là các tập tin văn bản.

b. Tập tin nhị phân (binary file)

Nội dung tập tin được lưu trữ theo một cấu trúc riêng. Mỗi tập tin nhị phân do một phần mềm nào tạo ra thì sẽ có cấu trúc do phần mềm đó quy định.

Ví dụ: Các tập tin: *.exe, *.doc, *.xls, *.bmp, ... là các tập tin nhị phân

3. Các bước xử lý tập tin

Bước 1: Mở tập tin để chuẩn bị làm việc.

Bước 2: Thực hiện thao tác đọc/ghi tập tin: Đọc dữ liệu từ tập tin vào biến nhớ hoặc ghi dữ liệu từ biến nhớ vào tập tin.

Bước 3: Đóng tập tin.

4. Con trỏ tập tin

- Khi mở tập tin để chuẩn bị làm việc, tùy theo cách thức mở tập tin một con trỏ tập tin sẽ được đặt tại vi trí đầu hoặc cuối tập tin.
- Mỗi thao tác đọc/ghi tập tin sẽ tác động lên vị trí hiện hành của con trỏ tập tin. Sau khi thực hiện xong một thao tác đọc/ghi con trỏ tập tin sẽ tự động dời đi một số byte đúng bằng số byte đọc/ghi.

5. Vùng đệm

- Trong C, tập tin được gắn liền với một vùng đệm. Mỗi thao tác đọc/ghi tập tin thường được tiến hành trên vùng đệm chứ không hẳn trên tập tin.
- Khi ghi, dữ liệu từ biến nhớ được đưa vào vùng đệm và khi nào vùng đệm đầy thì vùng đệm mới được đẩy lên đĩa.
- Khi đọc, dữ liệu được lấy từ vùng đệm đưa vào biến nhớ và chỉ khi nào vùng đệm đã trống rỗng thì máy mới lấy dữ liệu từ đĩa đưa vào vùng đệm.

6. Các hàm thao tác tập tin

a. Các hàm đóng mở tập tin

- FILE *fopen(char *tentaptin, char *kiểu);

  Hàm mở tập tin để chuẩn bị làm việc. Nếu thành công hàm trả về một con trỏ kiểu FILE ứng với tập tin vừa mở, nếu thất bại hàm trả về giá trị NULL.

  Kiểu có thể nhận các giá trị sau:

- “rt”: Mở một tập tin văn bản để đọc. Tập tin cần tồn tại nếu không sẽ có lỗi.
- “wt”: Mở một tập tin văn bản mới để ghi. Nếu tập tin đã tồn tại nó sẽ bị xóa.
- “at”: Mở một tập tin văn bản để ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới.
- “r+t” : Mở một tập tin văn bản để đọc/ghi. Tập tin cần tồn tại nếu không sẽ có lỗi.

- “w+t”: Mở một tập tin văn bản mới để đọc/ghi. Nếu tập tin đã tồn tại nó sẽ bị xóa.
- “a+t”: Mở một tập tin văn bản để đọc/ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới.
- “rb”: Mở một tập tin nhị phân để đọc. Tập tin cần tồn tại nếu không sẽ có lỗi.
- “wb”: Mở một tập tin nhị phân mới để ghi. Nếu tập tin đã tồn tại nó sẽ bị xóa.

- “ab”: Mở một tập tin nhị phân để ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới.
- “r+b” : Mở một tập tin nhị phân để đọc/ghi. Tập tin cần tồn tại nếu không sẽ có lỗi.
- “w+b”: Mở một tập tin nhị phân mới để đọc/ghi. Nếu tập tin đã tồn tại nó sẽ bị xóa.
- “a+b”: Mở một tập tin nhị phân để đọc/ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới

- int fclose(FILE *fp);

  Hàm dùng để đóng tập tin fp. Nếu thành công hàm trả về giá trị 0, nếu thất bại hàm trả về giá trị EOF (số -1).

- int feof(FILE *fp);

  Hàm kiểm tra cuối tập tin fp. Nếu gặp cuối tập tin khi đọc hàm trả về giá trị khác 0, ngược lại hàm trả về giá tri 0.

- int fflush(FILE *fp);

  Hàm dùng làm sạch vùng đệm của tập tin fp. Nếu thành công hàm trả về giá trị 0, ngược lại hàm trả về giá trị EOF.

b. Các hàm đọc/ghi tập tin văn bản

- int fprintf(FILE *fp, “chuỗi định dạng” [, danh sách các tham số]);

  Giá trị của các tham số được ghi lên tập tin fp theo khuôn dạng xác định trong chuỗi định dạng. Nếu thành công hàm trả về một giá trị nguyên bằng số byte ghi lên tập tin, nếu thất bại hàm trả về giá trị EOF.

- int fscanf(FILE *fp, “chuỗi định dạng” [,danh sách các tham số]);

  Đọc dữ liệu từ fp, biến đổi theo khuôn dạng trong chuỗi định dạng và lưu kết quả vào danh sách các tham số. Hàm trả về một giá trị bằng số trường đọc được.

- int fputs(char *s, FILE *fp);

  Ghi chuỗi s lên tập tin fp (ký tự '\0' không ghi lên tập tin). Nếu thành công hàm trả về mã ký tự cuối cùng ghi lên tập tin, nếu thất bại hàm trả về giá tri EOF.

- char *fgets(char *s, int n, FILE *fp);

  Đọc một dãy ký tự từ tập tin fp và lưu vào chuỗi s. Việc đọc kết thúc khi hoặc đã được đọc n-1 ký tự hoặc gặp dấu xuống dòng hoặc kết thúc tập tin. Nếu thành công hàm trả về con trỏ tới chuỗi s, nếu thất bại hoặc gặp cuối tập tin hàm trả về giá trị NULL.

Ví dụ 1: Tạo một tập tin văn bản có tên là "Tho.txt" nằm trong thư mực hiện hành có nội dung như sau:

Quê hương là chùm khế ngọt

Cho con trèo hái mỗi ngày

```
int TaoTT(char *tentt);
void main() {
    int kq = TaoTT("Tho.txt");
    if(kq ==0)
        printf("Không tạo được tập tin\n");
}
```

```
int TaoTT(char *tentt) {
        FILE *fp; int ret;
        fp = fopen(tentt, “wt”);
        if(fp != NULL) {
                fprintf(fp, “Quê hương …\n”);
                fprintf(fp, “Cho con …\n”);
                fclose(fp);
                ret = 1; //Thành công
        }
        else ret = 0; //Thất bại
        return ret;
}
```

Ví dụ 2: Đọc từng dòng của tập tin văn bản “Tho.txt” mới vừa tạo ở trên và xuất ra màn hinh.

```
#define MAX_LEN 256
int DocTT(char *tentt);
void main() {
        int kq = DocTT(“Tho.txt”);
        if(kq ==0)
                printf(“Tập tin không có trên đĩa”);
}
```

```
int DocTT(char *tentt) {
        FILE *fp; char s[MAX_LEN]; int ret;
        fp = fopen(tentt, “rt”);
        if(fp != NULL) {
                while(!feof(fp))  {
                        if(fgets(s, MAX_LEN, fp) != NULL)
                                printf(“%s”, s); }
                fclose(fp);
                ret = 1;
        }
        else ret = 0;
        return ret;
}
```

Ví dụ 3: Ghi thêm vào tập tin văn bản "Tho.txt" mới vừa tạo ở trên hai câu thơ được nhập từ bàn phím có nội dung sau:

Quê hương là đường đi học

Con về rợp bướm vàng bay

```
void main() {
    int kq = GhiThemTT("Tho.txt");
    if(kq == 1) {
            printf("** Noi dung tap tin sau khi ghi them
                    **\n");
            DocTT("Tho.txt"); }
    else printf("Tập tin không có trên đĩa\n");
}
```

```
int GhiThemTT(char *tentt) {
        FILE *fp; int ret, i; char s[MAX_LEN];
        fp = fopen(tentt, “at”);
        if(fp != NULL) {
                printf(“Nhập dữ liệu muốn thêm:\n”);
                for(i = 0; i < 2; i++) {
                        gets(s);
                        fprintf(fp, “%s”, s); }
                fclose(fp);
                ret = 1;
        }
        else ret = 0;
        return ret;
}
```

Ví dụ 4: Nhập vào tên hai tập tin văn bản. Hãy ghép nội dung tập tin thứ hai vào tập tin thứ nhất.

```
void main() {
    char tentt1[MAX_LEN], tentt2[MAX_LEN];
    printf("Nhap ten tap tin thu 1:");
    gets(tentt1);
    printf("Nhap ten tap tin thu 2:");
    gets(tentt2);
    int kq = GhepTT(tentt1, tentt2);
```

```
 if(kq == -2)
        printf("Tap tin %s và %s khong co tren dia\n",
               tentt1, tentt2);
else if(kq == -1)
        printf("Tap tin %s khong co tren dia\n", tentt1);
else if(kq == 0)
        printf("Tap tin %s khong co tren dia\n", tentt2);
else {
        printf("Tap tin %s sau khi ghep\n", tentt1);
        DocTT(tentt1);
}
}
```

```
int GhepTT(char *tentt1, char *tentt2) {
    FILE* fp1, *fp2; char s[MAX_LEN]; int ret;
    fp1 = fopen(tentt1, “at”); fp2 = fopen(tentt2, “rt”);
    if(fp1 == NULL && fp2 == NULL)
        ret = -2;
    else if(fp1 == NULL) {
        ret = -1; if(fp2 != NULL) fclose(fp2); }
    else if(fp2 == NULL) {
        ret = 0; fclose(fp1); }
    else {
        while(!feof(fp2)) {
            if(fgets(s, MAX_LEN, fp2) != NULL)
                fprintf(fp1, “%s”, s); }
        fclose(fp1); fclose(fp2);
        ret = 1; }
    return ret; }
```

Ví dụ 5: Nhập một ma trận từ một tập tin văn bản có tên "MaTran.txt" nằm trong thư mục hiện hành, tập tin có nội dung như sau:

- Dòng đầu tập tin lưu giá tri sd và sc (số dòng và số cột của ma trận), hai số này ngăn cách nhau bởi khoảng trắng.
- Các dòng tiếp theo, mỗi dòng tập tin lưu một dòng của ma trận, hai số kề nhau ngăn cách nhau bởi khoảng trắng.

```
#define SIZE1 5
#define SIZE2 6
struct MaTran{
	int sd, sc;
	int arr[SIZE1][SIZE2];
};
void main() {
	MaTran mt;
	int kq = DocMT("MaTran.txt", &mt);
	if(kq == 1) {
		printf("** Ma tran sau khi doc **\n");
		XuatMT(mt);
	}
	else printf("Tap tin MaTran.txt khong co tren dia\n");
}
```

```
int DocMT(char *tentt, MaTran *u) {
        FILE *fp;
        int i, j, tam, ret;
        fp = fopen(tentt, “rt”);
        if(fp != NULL) {
                fscanf(fp, “%d”, &(u->sd));
                fscanf(fp, “%d”, &(u->sc));
                if(u->sd > SIZE1) u->sd = SIZE1;
                if(u->sc > SIZE2) u->sc = SIZE2;
```

```
                for(i = 0 ; i < u->sd ; i++)
                        for(j = 0 ; j < u->sc ; j++) {
                                if(!feof(fp)) {
                                        fscanf(fp, “%d”, &tam);
                                        u->arr[i][j] = tam;
                                }
                                else u->arr[i][j] = 0;
                        }
                fclose(fp);
                ret = 1;
        }
        else ret = 0;
        return ret;
}
```

```
void XuatMT(MaTran u) {
        int i, j;
        printf("So dong: %d\n", u.sd);
        printf("So cot: %d\n", u.sc);
        for(i = 0 ; i < u.sd ; i++) {
                for(j = 0 ; j < u.sc ; j++)
                        printf("%d\t", u.arr[i][j]);
                printf("\n");
        }
}
```

c. Các hàm đọc/ghi tập tin nhị phân

- int fwrite(void *ptr, int size, int n, FILE *fp);

  Ghi n phần tử, mỗi phần tử có kích thước là size byte, từ vùng nhớ được trỏ bởi con trỏ ptr lên tập tin fp. Hàm trả về một giá trị bằng số phần tử thực sự ghi được.

- **int fread(void *ptr, int size, int n, FILE *fp);**

  Đọc n phần tử, mỗi phần tử có kích thước là size byte, từ tập tin fp và lưu vào vùng nhớ được trỏ bởi con trỏ ptr. Hàm trả về một giá trị bằng số phần tử thực sự đọc được.

- **void rewind(FILE *fp);**

  Di chuyển con trỏ tập tin về đầu tập tin fp. Khi đó việc truy xuất trên tập tin được thực hiện từ đầu tập tin.

- **long ftell(FILE *fp);**

  Nếu thành công hàm trả về vị trí hiện tại của con trỏ tập tin (byte thứ mấy trên tập tin, số thứ tự byte được tính từ 0), Nếu có lỗi hàm trả về giá trị -1L.

- **int fseek(FILE *fp, long sb, int xp);**

  Di chuyển con trỏ tập tin xuất phát từ vị trí xp qua một số byte bằng giá trị tuyệt đối của sb. Chiều di chuyển là về cuối tập tin nếu sb dương, về đầu tập tin nếu sb âm. Nếu thành công hàm trả về giá trị 0, nếu thất bại hàm trả về giá trị khác 0.

xp nhận một trong 3 giá trị sau:

- 0 (hoặc SEEK_SET): xuất phát từ đầu tập tin.
- 1(hoặc SEEK_CUR): xuất phát từ vị trí hiện hành của con trỏ tập tin.
- 2(hoặc SEEK_END): xuất phát từ cuối tập tin.

Ví dụ 1: Tạo một tập tin chứa danh sách các sinh viên, danh sách này được lưu sẵn trong một mảng cấu trúc.

```
#define SIZE 20
struct SINHVIEN {
        char ma[10];
        char ten[30];
        int namsinh;
        float diem;
};
struct DSSV{
        int n;
         SINHVIEN arr[SIZE];
};
```

```
int TaoTTDSSV(char *tentt, DSSV u)  {
        FILE *fp;
        int ret;
        fp = fopen(tentt, “wb”);
        if(fp != NULL) {
                /*Ghi u.n cấu trúc, mỗi cấu trúc kích thước
                sizeof(SINHVIEN ), chứa trong mảng u.arr
                vào tập tin fp*/
                fwrite(u.arr, sizeof(SINHVIEN), u.n, fp);
                fclose(fp);
                ret = 1;
        }
        else ret = 0;
        return ret;
}
```

```
void main() {
	DSSV dssv;
	char tentt[MAX_LEN];
	printf(“Nhap ten tap tin muon tạo:”);
	gets(tentt);
	NhapDSSV(&dssv); //Xem lại chương cấu trúc
	int kq = TaoTTDSSV(tentt, dssv);
	if(kq == 0)
		printf(“Khong the tao tap tin\n”);
}
```

Ví dụ 2: Đọc danh sách các sinh viên chứa trong tập tin vừa tạo ở trên và xuất ra màn hình.

```
int DocTTDSSV(char *tentt);
void main() {
        char tentt[MAX_LEN];
        <Nhap ten tap tin muon doc>
        int kq = DocTTDSSV(tentt);
        if(kq == 0)
                printf("Tap tin %s khong ton tai tren dia\n",
                        tentt);
}
```

```
int DocTTDSSV(char *tentt) {
    FILE *fp; SINHVIEN sv; int ret;
    fp = fopen(tentt, “rb”);
    if(fp != NULL) {
        while(!feof(fp)) {
            /*Đọc một cấu trúc có kích thước
            sizeof(SINHVIEN) từ tập tin fp vào biến sv*/
            if(fread(&sv, sizeof(SINHVIEN), 1, fp) == 1)
                Xuat1SV(sv);
        }
        fclose(fp);
        ret = 1;
    }
    else ret = 0;
    return ret; }
```

Ví dụ 3: Tương tự như ví dụ trên nhưng danh sách các sinh viên đọc từ tập tin được lưu vào một mảng cấu trúc (số sinh viên đọc được không được vượt quá kích thước mảng cấu trúc), sau đó xuất mảng cấu trúc này ra màn hình.

```
int DocTTDSSV2(char *tentt, DSSV *u);
void main() {
	char tentt[MAX_LEN];
	DSSV dssv;
	<Nhap ten tap tin muon doc>
	int kq = DocTTDSSV2(tentt, &dssv);
	if(kq == 1)
		XuatDSSV(dssv); //Xem lại chương cấu trúc
	else
		printf("Tap tin %s khong co tren dia\n", tentt);
}
```

```
int DocTTDSSV2(char *tentt, DSSV *u) {
    FILE *fp; int ret;
    fp = fopen(tentt, “rb”);
    if(fp != NULL) {
        u->n = 0;
        while(!feof(fp) && u -> n < SIZE) {
            if(fread(&u->arr[u->n], sizeof(SinhVien), 1, fp)
                == 1)
            u->n++;
        }
        fclose(fp)
        ret = 1;
    }
    else ret = 0;
    return ret; }
```

Ví dụ 4: Thêm một sinh viên vào tập tin chứa danh sách sinh viên đã tạo ở trên.

```
int ThemSVVaoTTDSSV(char *tentt, SINHVIEN sv) {
    FILE *fp; int ret;
    fp = fopen(tentt, “ab”);
    if(fp != NULL) {
        /*Ghi cấu trúc có kích thước sizeof(SINHVIEN
        ) chứa trong biến sv lên tập tin fp*/
        fwrite(&sv, sizeof(SINHVIEN), 1, fp);
        fclose(fp);
        ret = 1;
    }
    else ret = 0;
    return ret;
}
```

```
void main() {
     SINHVIEN sv;
    char tentt[MAX_LEN];
    printf("Nhap ten tap tin muon them:");
    gets(tentt);
    printf("Nhap thong tin sinh vien muon them\n");
    Nhap1SV(&sv); //Xem lại chương cấu trúc
    int kq = ThemSVVaoTTDSSV (tentt, sv);
    if(kq == 1) {
          printf("Danh sach sinh vien sau khi them\n");
          DocTTDSSV(tentt);
    }
    else printf("Tap tin %s khong ton tai tren dia\n", tentt);
}
```

Ví dụ 5: Nhập vào tên tập tin chứa danh sách sinh viên, mã sinh viên và năm sinh mới. Hãy tìm trong tập tin một sinh viên có mã trùng với mã nhập vào, nếu tìm thấy thì sửa năm sinh cũ thành năm sinh mới.

```
void main() {
    char ma[10]; int namsinh; char tentt[MAX_LEN];
    <Nhap ten tt muon sua>
    < Nhap ma sinh vien cua sinh vien can sua>
    <Nhap nam sinh moi>
    int kq = SuaTTDSSV(tentt, ma, namsinh);
    if(kq == 0)
            printf("Tap tin %s khong co tren dia\n", tentt);
    else if(kq == -1)
             printf("Khong tim thay sv co ma %s\n", ma);
    else {
            printf("** Danh sach sinh vien moi **\n");
            DocTTDSSV(tentt);
    }
}
```

```
int SuaTTDSSV(char *tentt, char *ma, int namsinh) {
    FILE *fp ; SINHVIEN sv; int timthay, vitri, ret;
    fp = fopen(tentt, “r+b”);
    if(fp != NULL) { timthay = 0;
          while(!feof(fp)) {
                vitri = ftell(fp);
                if(fread(&sv, sizeof(SINHVIEN), 1, fp) == 1) {
                if(strcmp(sv.ma, ma) == 0) {
                      sv.namsinh = namsinh;
                      fseek(fp, vitri, SEEK_SET);
                      fwrite(&sv, sizeof(SINHVIEN), 1, fp);
                      timthay = 1; break;
                }}}
                fclose(fp);
                ret = (timthay == 1 ? 1:-1);}
    else ret = 0; return ret; }
```

d. Các hàm đọc/ghi tập tin dùng chung cho cả tập tin văn bản và nhị phân

- **int fputc(int ch, FILE *fp);**

  Hàm ghi lên tập tin một ký tự có mã là ch%256 trong đó ch được xem là số nguyên không dấu. Nếu thành công hàm trả về mã ký tự được ghi, nếu thất bại hàm trả về giá trị EOF.

- **int fgetc(FILE *fp);**

  Hàm đọc một ký tự từ tập tin fp. Nếu thành công hàm trả về mã đọc được (có giá trị từ 0 đến 255), nếu thất bại hoặc gặp cuối tập tin hàm trả về giá trị EOF.

  Ví dụ: Nhập vào tên một tập tin văn bản. Hãy đọc từng ký tự của tập tin, nếu ký tự đọc được là chữ thường thì đổi thành chữ hoa và ghi ký tự trở lại tập tin.

```
void main() {
        char tentt[MAX_LEN];
        <Nhap ten tap tin muon sua>
        int kq = DoiHoaTT(tentt);
        if(kq == 1) {
                printf("** Tap tin sau khi doi hoa **\n");
                DocTT(tentt);
        }
        else printf("Tap tin %s khong co tren dia\n", tentt);
}
```

```
int DoiHoaTT(char *tentt) {
        FILE *fp; char ch; int vitri, ret;
        fp = fopen(tentt, “r+b”);
        if(fp != NULL) {
                while(!feof(fp)) {
                        if((ch = fgetc(fp)) != EOF) {
                        if(ch >= ‘a’ && ch <= ‘z’) {
                                ch -= 32;
                                fseek(fp, -1, SEEK_CUR);
                                fputc(ch, fp);
                                fseek(fp, 1, SEEK_CUR); }}
                }
                fclose(fp);
                ret = 1; }
        else ret = 0;return ret; }
```

Hết