The Persistence of Memory

厦門大學 XIAMEN UNIVERSITY
信息学院 黄 炜
(特色化示范性软件学院) 博士·副教授
School of Informatics Wei Huang

C 程 序 设 计

C Programming

13

文件输入输出

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院 黄 煒
(特色化示范性软件学院) 博士,副教授
School of Informatics Wei Huang

知识框架

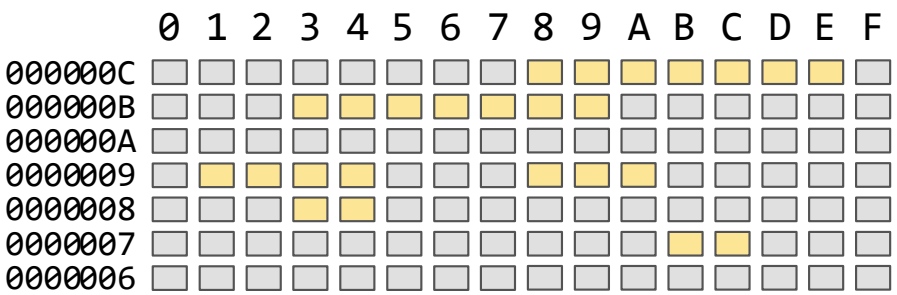
- 文件的基本概念
- 文件的操作
  - 开、关、读、写
- 其它文件函数

内容纲要

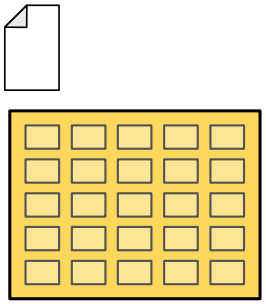
1	文件的基本概念
2	文件的操作
3	其它文件函数

文件（File）

- 文件是磁盘上一段命名的存储区
  - 物理上：可能分散在磁盘的不同区域里（操作系统管的）
  - 逻辑上：程序可以认为这是一个连续的字节序列（文件流）
- 文件存取（读写）的最小单位是字节



文件的物理存储



文件的逻辑存储

文件

- 文件的两种视图
  - 二进制视图：按字节存取

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	54	68	65	20	71	75	69	63	6B	20	62	72	6F	77	6E	20
0010	66	6F	78	20	6A	75	6D	70	73	20	6F	76	65	72	20	61
0020	20	6C	61	7A	79	20	64	6F	67	2E	0D	0A	55			

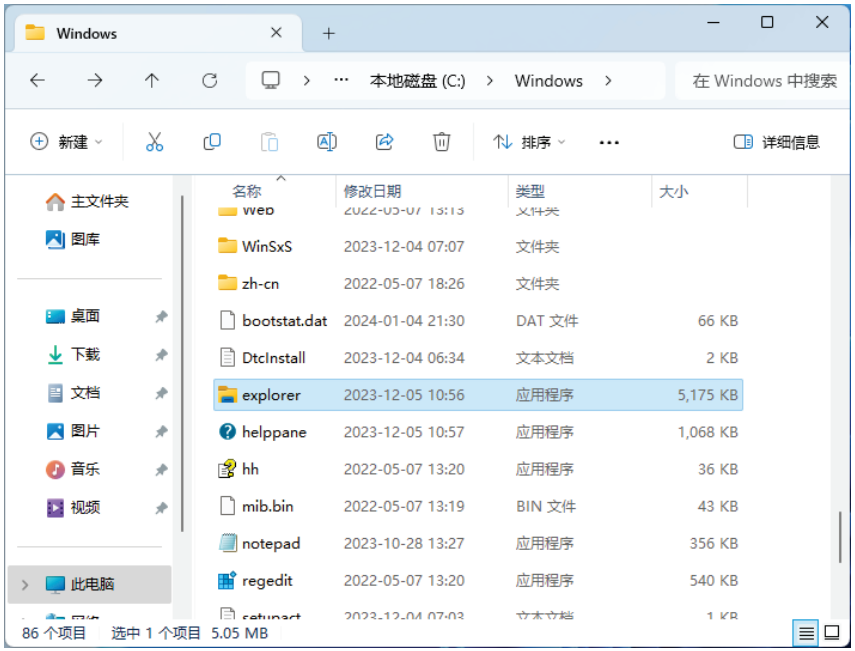
  - 文本视图：按字符存取，且处理回车问题的本地化
    - 读写时会遇到回车转换为本地操作系统的类型

T	h	e		q	u	i	c	k		b	r	o	w	n	
f	o	x		j	u	m	p	s		o	v	e	r		a
	l	a	z	y		d	o	g	.	\n	U				

操作系统	换行符	数值
Windows	\r\n	0D 0A
Unix	\n	0A
Mac	\r	0D

文件的基本概念

- 卷、文件夹、文件
  - Windows下，硬盘划分为卷（驱动器、分区）
  - 文件夹下可以包含多个子文件夹和文件
- 文件的属性
  - 文件名扩展名
  - 大小
  - 时间
  - 只读、隐藏



文件I/O接口

- 低级I/O
  - 提供对文件和设备的直接访问
  - 使用“文件描述符”来跟踪文件的状态
  - 复杂（缓冲区管理由程序员完成）
- 高级I/O
  - 隐藏了底层细节，提供统一接口（ C库函数包）
  - 提供了缓存功能，提高数据传输效率

```c
/* count.c -- using standard I/O */
#include <stdio.h>
#include <stdlib.h> // exit() prototype
int main(int argc, char *argv[]) {
    int ch;          // place to store each character as read
    FILE *fp;        // "file pointer"
    unsigned long count = 0;
    if (argc != 2) {
        printf("Usage: %s filename\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if ((fp = fopen(argv[1], "r")) == NULL) {
        printf("Can't open %s\n", argv[1]);
        exit(EXIT_FAILURE);
    }
    while ((ch = getc(fp)) != EOF) {
        putc(ch,stdout);  // same
        count++;
    }
    fclose(fp);
    printf("File %s has %lu characters\n", argv[1], count);
    return 0;
}
```

主函数可以接收命令行参数

打开文件

同于putchar(ch);

关闭文件

```
D:\***\Ch13>count wordy↲
The fabulous programmer enchanted
the large
File wordy has 44 characters
```

程序说明

- 检查命令行参数

```
int main(int argc, char *argv[])
```

- 返回值
  - 主函数通过return语句将信息传回操作系统并终止程序
  - 任何函数通过exit函数将信息传回操作系统并终止程序
    - 需共享的函数应慎重调用exit函数，以免造成程序莫名其妙终止
  - ANSI C规定0或宏EXIT_SUCCESS表示程序成功终止，EXIT_FAILURE表示程序非成功终止

内容纲要

文件打开

- 格式

```
FILE *fopen(const char *filename, const char *mode);
```

- 参量
  - 文件名：字符串，需要操作的文件
  - 模式：字符串，由1～3个字符组成

基本模式	加强模式	文本模式
r：读	+：兼顾读写	t：文本模式，默认
w：清空写	空：无操作，默认	b：二进制模式
a：追加		

- 返回值
  - 操作成功返回文件指针，操作失败返回NULL（0）

文件关闭

- 格式

```
int fclose(FILE *stream);
```

- 参量
  - 文件：指针，需要操作的文件
- 返回值
  - 操作成功返回0，操作失败返回EOF（-1）
  - 应根据文件开关（甚至读写）的返回值进行处理
    - 对返回不成功的情况应对用户进行提示

标准流

- 标准流

名称	定义	设备	打开关闭
标准输入流	extern struct _IO_FILE *stdin;	键盘	自动打开，可关闭
标准输出流	extern struct _IO_FILE *stdout;	屏幕	自动打开，不可关闭
标准错误流	extern struct _IO_FILE *stderr;	屏幕	自动打开，不可关闭

- 标准错误流的使用
  - 应将错误提示输出到标准错误流，不应和输出流相互干扰

```c
// reducto.c -- reduces your files by two-thirds!
#include <stdio.h>
#include <stdlib.h>     // for exit()
#include <string.h>     // for strcpy(), strcat()
#define LEN 40
int main(int argc, char *argv[]) {
    FILE  *in, *out;   // declare two FILE pointers
    int ch;
    char name[LEN];    // storage for output filename
    int count = 0;
    if (argc < 2) {    // check for command-line arguments
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if ((in = fopen(argv[1], "r")) == NULL) { // set up input
        fprintf(stderr, "I couldn't open the file \"%s\"\n", argv[1]);
        exit(EXIT_FAILURE);
    }
    // set up output
    strncpy(name,argv[1], LEN - 5); // copy filename
```

```
    name[LEN - 5] = '\0';
    strcat(name,".red");                    // append .red
    if ((out = fopen(name, "w")) == NULL) {// open file for writing
        fprintf(stderr,"Can't create output file.\n");
        exit(3);
    }
    while ((ch = getc(in)) != EOF) // copy data
        if (count++ % 3 == 0)
            putc(ch, out);  // print every 3rd char
    if (fclose(in) != 0 || fclose(out) != 0) // clean up
        fprintf(stderr,"Error in closing files\n");
    return 0;
}
```

```
D:\***\Ch13>reducto wordy↲
```

wordy 的内容	wordy.red 的内容
The fabulous programmer enchanted the large	T bo oaeeht eae

文件读写字符

- 函数

函数	格式	参量	返回值
读字符	int getc(FILE *stream);	文件指针	成功返回字符，失败返回EOF
写字符	int putc(int c, FILE *stream);	字符，文件指针	成功返回字符，失败返回EOF

- 判定读写结束应用

```
while ((ch = getc(fp))!= EOF)
{
    ...
}
```

```
while ( ch != EOF )
{
    ch = getc(fp);
}
```

文件格式化读写

- 格式
- 参量

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

  - 文件：指针，需要操作的文件
  - 格式字符串：字符串，参考scanf/printf的第一个参量
  - 参数列表：参考scanf/printf的第二个以上参量
- 返回值
  - 参考scanf/printf的返回值

文件格式化读写

- 格式

```
char *fgets (char * s, int n, FILE * stream);
```

- 参量
  - 字符串：指针，输入字符串存储的位置
  - 长度：整数，接受字符串最大的字符数
  - 文件：指针，需要操作的文件
- 返回值：操作成功返回s，操作失败返回NULL
  - 规则：文件剩余m个字符
    - m>=n-1，读取n-1个，第n个为0
    - m<=n-2，读取m个，第m+1个为0x0A，第m+2个为0
    - 读取前触及文件末尾，不改变s的内容

文件格式化读写

- 格式

```
int fputs(const char * s, FILE * stream);
```

- 参量
  - 字符串：指针，输出字符串存储的位置
  - 文件：指针，需要操作的文件
- 返回值
  - 操作成功返回非负整数（一般为0），操作失败返回EOF
  - 规则：输出字符串，不添加回车

```c
/* addaword.c -- uses fprintf(), fscanf(), and rewind() */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 41

int main(void)
{
    FILE *fp;
    char words[MAX];

    if ((fp = fopen("wordy", "a+")) == NULL)
    {
        fprintf(stdout,"Can't open \"wordy\" file.\n");
        exit(EXIT_FAILURE);
    }

    puts("Enter words to add to the file; press the #");
    puts("key at the beginning of a line to terminate.");
```

```
    while ((fscanf(stdin,"%40s", words) == 1)  &&
(words[0] != '#'))
        fprintf(fp, "%s\n", words);

    puts("File contents:");
    rewind(fp);           /* go back to beginning of file */
    while (fscanf(fp,"%s",words) == 1)
        puts(words);
    puts("Done!");
    if (fclose(fp) != 0)
        fprintf(stderr,"Error closing file\n");

    return 0;
}
```

文件随机存取寻址

- 格式 `int fseek(FILE *stream, long int off, int whence);`
- 参量
  - 文件：指针，需要操作的文件
  - 偏移量：整数，从*whence*计算的偏移量
  - 参照点：整数

宏名	整数值	含义
SEEK_SET	0	文件头
SEEK_CUR	1	当前位置
SEEK_END	2	文件尾

- 返回值
  - 操作成功返回0，操作失败返回非0
  - 用perror函数查看具体错误代码

文件随机存取查址

- 格式 `long int ftell (FILE *stream);`
- 参量
  - 文件：指针，需要操作的文件
- 返回值
  - 操作成功返回当前文件读写位置
  - 文本和二进制模式处理回车的差异导致返回值可能不同
- 大尺寸版本文件寻址查址
  - fgetpos()和fsetpos()

```c
/* reverse.c -- displays a file in reverse order */
#include <stdio.h>
#include <stdlib.h>
#define CNTL_Z '\032'   /* eof marker in DOS text files */
#define SLEN 81
int main(void)
{
    char file[SLEN];
    char ch;
    FILE *fp;
    long count, last;

    puts("Enter the name of the file to be processed:");
    scanf("%80s", file);
    if ((fp = fopen(file,"rb")) == NULL)
    {                                   /* read-only mode   */
        printf("reverse can't open %s\n", file);
        exit(EXIT_FAILURE);
    }
```

```
    fseek(fp, 0L, SEEK_END);          /* go to end of file */
    last = ftell(fp);
    for (count = 1L; count <= last; count++)
    {
        fseek(fp, -count, SEEK_END); /* go backward       */
        ch = getc(fp);
             if (ch != CNTL_Z && ch != '\r')  /* MS-DOS files */
            putchar(ch);
    }
    putchar('\n');
    fclose(fp);

    return 0;
}
```

```
Enter the name of the file to be processed:
eddy↲


.ydaer nevo emac yddE neve oS
```

文件二进制读写

- 格式

```
size_t fread(void *buffer, size_t size, size_t count,
    FILE *stream);
```

- 参量
  - 字节数组：指针，输入对象数组存储的位置
  - 单位：整数，该对象内每个元素的字节数，影响端序
  - 项度：整数，读取的数量，单位乘以项度等于总字节数
  - 文件：指针，需要操作的文件
- 返回值
  - 操作成功返回已读的项数

文件二进制读写

- 格式

```
size_t fwrite(const void *buffer, size_t size,
    size_t count, FILE *stream);
```

- 参量
  - 字节数组：指针，输出对象数组存储的位置
  - 单位：整数，该对象内每个元素的字节数，影响端序
  - 项度：整数，读取的数量，单位乘以项度等于总字节数
  - 文件：指针，需要操作的文件
- 返回值
  - 操作成功返回已写的项数

```c
/* randbin.c -- random access, binary i/o */
#include <stdio.h>
#include <stdlib.h>
#define ARSIZE 1000
int main() {
    double numbers[ARSIZE];
    double value;
    const char * file = "numbers.dat";
    int i;
    long pos;
    FILE *iofile;
    // create a set of double values
    for(i = 0; i < ARSIZE; i++)
        numbers[i] = 100.0 * i + 1.0 / (i + 1);
    // attempt to open file
    if ((iofile = fopen(file, "wb")) == NULL) {
        fprintf(stderr, "Could not open %s for output.\n", file);
        exit(EXIT_FAILURE);
    }
    // write array in binary format to file
    fwrite(numbers, sizeof (double), ARSIZE, iofile);
```

```
    fclose(iofile);
    if ((iofile = fopen(file, "rb")) == NULL){
        fprintf(stderr,
                "Could not open %s for random access.\n", file);
        exit(EXIT_FAILURE);
    }
    // read selected items from file
    printf("Enter an index in the range 0-%d.\n", ARSIZE - 1);
    while (scanf("%d", &i) == 1 && i >= 0 && i < ARSIZE) {
        pos = (long) i * sizeof(double); // calculate offset
        fseek(iofile, pos, SEEK_SET);    // go there
        fread(&value, sizeof (double), 1, iofile);
        printf("The value there is %f.\n", value);
        printf("Next index (out of range to quit):\n");
    }
    fclose(iofile); // fin
    puts("Bye!");
    return 0;
}
```

```
Enter an index in the range 0-999.
123↲
The value there is 12300.008065.
Next index (out of range to quit):
5↲
The value there is 500.166667.
Next index (out of range to quit):
q↲
Bye!
```

内容纲要

1	文件的基本概念
2	文件的操作
3	其它文件函数

其它标准I/O函数

功能	格式	参数	返回值
文件尾判断	`int feof(FILE *stream);`	指针，需要操作的文件。	到文件尾，返回非零；否则，返回0。
文件错误代码	`int ferror(FILE *stream);`	同上。	最近一次文件操作有错误，返回具体错误代码；否则，返回0。
刷新缓冲区	`int fflush(FILE *stream);`	同上。	已成功刷新缓冲区，将返回 0。
放回输入流	`int ungetc(int c, FILE *stream);`	字符，要推送的字符。 指针，需要操作的文件。	操作成功返回c；否则，返回EOF。
自定义缓冲区	`int setvbuf(FILE *stream, char *buffer, int mode, size_t size);`	指针，需要操作的文件。 字符数组，用户缓冲区。 整数，缓冲模式。 整数，缓冲区大小。	操作成功返回0；否则，返回EOF。

```c
/* append.c -- appends files to a file */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BUFSIZE 4096
#define SLEN 81
void append(FILE *source, FILE *dest);
char * s_gets(char * st, int n);
int main(void) {
    FILE *fa, *fs;    // fa for append file, fs for source file
    int files = 0;  // number of files appended
    char file_app[SLEN];  // name of append file
    char file_src[SLEN];  // name of source file
    int ch;
    puts("Enter name of destination file:");
    s_gets(file_app, SLEN);
    if ((fa = fopen(file_app, "a+")) == NULL) {
        fprintf(stderr, "Can't open %s\n", file_app);
        exit(EXIT_FAILURE);
    }
    if (setvbuf(fa, NULL, _IOFBF, BUFSIZE) != 0) {
        fputs("Can't create output buffer\n", stderr);
```

```
puts("Enter name of first source file (empty line to quit):");
while (s_gets(file_src, SLEN) && file_src[0] != '\0') {
    if (strcmp(file_src, file_app) == 0)
        fputs("Can't append file to itself\n",stderr);
    else if ((fs = fopen(file_src, "r")) == NULL)
        fprintf(stderr, "Can't open %s\n", file_src);
    else {
        if (setvbuf(fs, NULL, _IOFBF, BUFSIZE) != 0) {
            fputs("Can't create input buffer\n",stderr);
            continue;
        }
        append(fs, fa);
        if (ferror(fs) != 0)
            fprintf(stderr,"Error in reading file %s.\n", file_src);
        if (ferror(fa) != 0)
            fprintf(stderr,"Error in writing file %s.\n", file_app);
        fclose(fs);
        files++;
        printf("File %s appended.\n", file_src);
        puts("Next file (empty line to quit):");
    }
}
```

```
    printf("Done appending. %d files appended.\n", files);
    rewind(fa);
    printf("%s contents:\n", file_app);
    while ((ch = getc(fa)) != EOF)
        putchar(ch);
    puts("Done displaying.");
    fclose(fa);
    return 0;
}
void append(FILE *source, FILE *dest) {
    size_t bytes;
    static char temp[BUFSIZE]; // allocate once
    while ((bytes = fread(temp,sizeof(char),BUFSIZE,source)) > 0)
        fwrite(temp, sizeof (char), bytes, dest);
}
char * s_gets(char * st, int n) {
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val) {
        find = strchr(st, '\n'); // look for newline
        if (find)                // if the address is not NULL,
            *find = '\0';        // place a null character there
```

```
        else
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Enter name of destination file:
eddy↲
Enter name of first source file (empty line to quit):
wordy↲
File wordy appended.
Next file (empty line to quit):
↲
Done appending. 1 files appended.
eddy contents:
So even Eddy came oven ready.

The
fabulous
programmer
enchanted
the
large
Good
Done displaying.
```

C 程 序 设 计

C Programming

13

谢谢观看

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 煒

(特色化示范性软件学院) 博士,副教授

School of Informatics Wei Huang