The Persistence of Memory

C程序设计

T13

文件输入输出

厦门大学信息学院软件工程系

黄炜 副教授

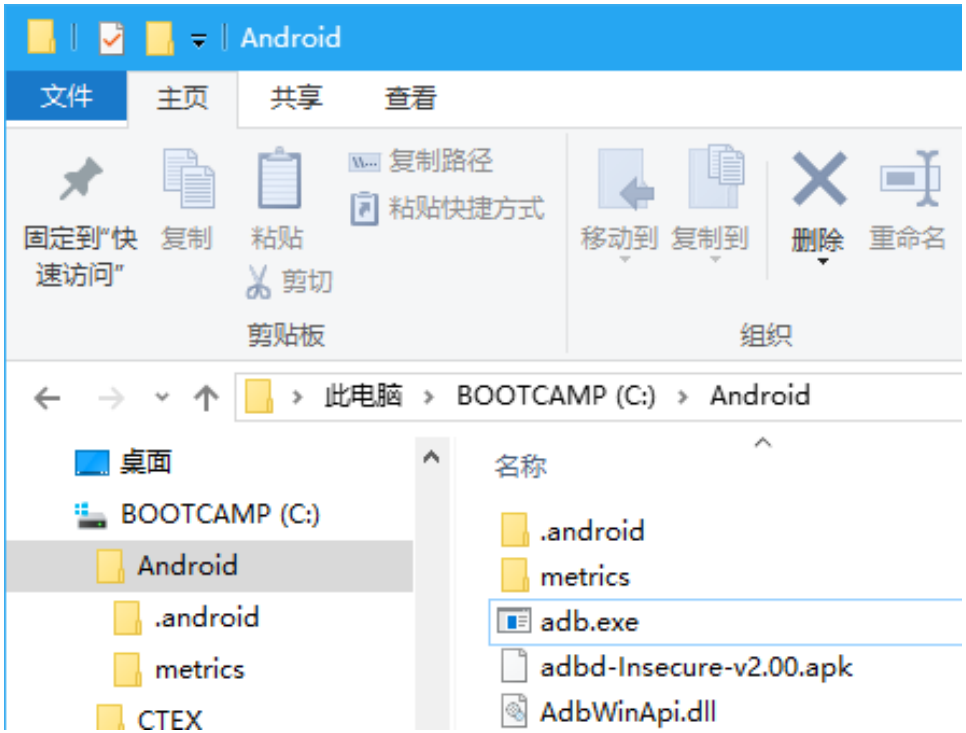
主要内容

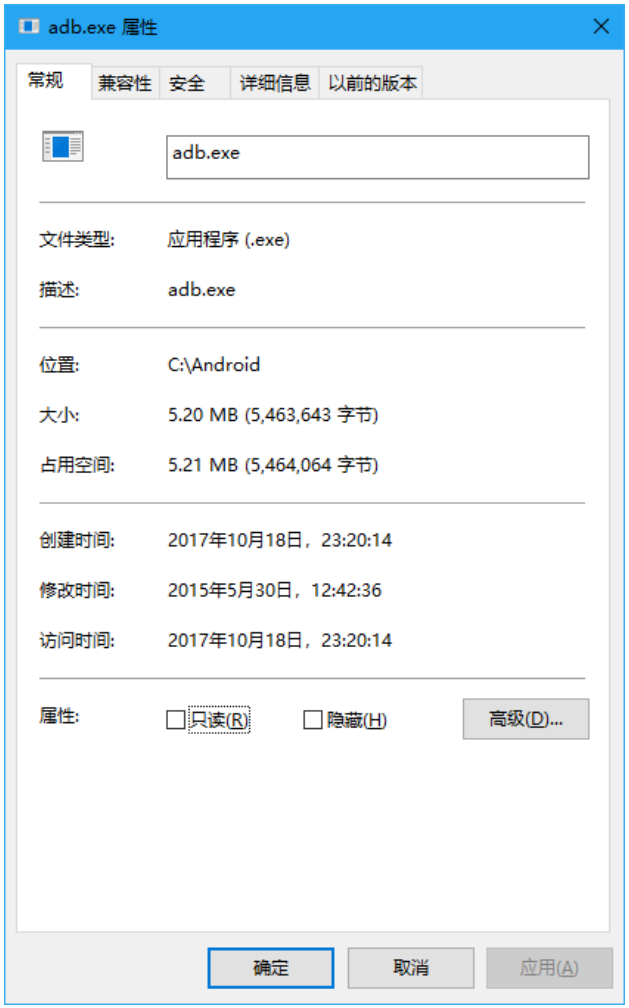
- 文件的基本概念
- 文件的操作
  - 开、关、读、写
- 其它文件函数

1. 文件的基本概念

文件的基本概念

- 驱动器、文件夹、文件
- 文件的属性





文件（File）

- 文件是磁盘上一段命名的存储区
  - 物理上：可能是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
- 文件的两种视图
  - 二进制视图存取的最小单位是字节
  - 文本视图存取的最小单位是字符
    - 回车问题的本地化：读写时会将回车转换为本地类型

操作系统	换行符
Windows	\r\n
Unix	\n
Mac	\r

文件（File）

- 低级I/O
  - 操作系统提供的基本I/O服务，因系统而异
- 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);
    }
```

主函数可以接收命令行参数

```
    while ((ch = getc(fp)) != EOF)
    {
        putc(ch,stdout);  // same as putchar(ch);
        count++;
    }
    fclose(fp);
    printf("File %s has %lu characters\n", argv[1], count);

    return 0;
}
```

```
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

2. 文件的操作

文件打开

- 打开文件的函数fopen()

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

  - 返回值：文件指针FILE *
    - 不同操作系统文件类型不同

- 打开模式（mode）：基本模式+加强模式+文本模式
  - 基本模式：r（读）w（写，文件在则清空）a（追加）
  - 加强模式：r+（读写）w+（清空读写） a+（追加读写）
  - 文本模式：t（文本模式） b（二进制模式）

文件关闭

- 关闭文件的函数fclose()

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

  - 返回值：0表示成功，EOF表示失败。

- 应对文件开关（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

标准文件

- 标准文件流

名称	定义	设备	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;

    // check for command-line arguments
    if (argc < 2)
    {
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        exit(EXIT_FAILURE);
    }
```

```
    // set up input
    if ((in = fopen(argv[1], "r")) == NULL)
    {
        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);
    }
```

```
    // copy data
    while ((ch = getc(in)) != EOF)
        if (count++ % 3 == 0)
            putc(ch, out);  // print every 3rd char
    // clean up
    if (fclose(in) != 0 || fclose(out) != 0)
        fprintf(stderr,"Error in closing files\n");

    return 0;
}
```

```
D:\***\Ch13>reducto wordy↲
```

wordy 的内容

```
The
fabulous
programmer
enchanted
the
large
```

wordy.red 的内容

```
T
bo
oaeeht
eae
```

文件读写

- 读字符的函数getc()
  - ch=getc(stdin)同于ch=getchar()
- 写字符的函数putc()
  - putc(ch, stdout)同于putchar(ch)

```
int getc( FILE *stream );
int fputc( int c, FILE *stream );
```

- 判定读写结束应用

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

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

文件读写

- 文件输入函数

```
int fscanf (FILE *__restrict __stream, const char *__restrict __format, ...) ;
char *fgets (char *__restrict __s, int __n, FILE *__restrict __stream);
```

- 文件输出函数

```
int fprintf (FILE *__restrict __stream, const char *__restrict __format, ...);
int fputs (const char *__restrict __s, FILE *__restrict __stream);
```

- 使用方法
  - 类似非 f 版本，需传入文件指针
  - 注意：不定参数的函数，文件指针在最前

```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

- 文件随机存取寻址fseek
- 文件随机存取查址ftell

```
int fseek (FILE *__stream, long int __off, int __whence);
long int ftell (FILE *__stream) ;
```

  - 参数whence的取值

```
# define SEEK_SET   0      /* Seek from beginning of file.  */
# define SEEK_CUR   1      /* Seek from current position.  */
# define SEEK_END   2      /* Seek from end of file.  */
```

  - ftell()返回的值对MSDOS里的文本和二进制模式不相同

- 大尺寸版：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
```

标准I/O的流程

- 打开文件，建立缓冲区
  - 缓冲区满则输出（类似标准I/O的缓冲区）
- 文件的输入与输出
- 关闭文件

3. 其它文件函数

其它标准I/O函数

- 放回输入流函数：ungetc()
- 刷新缓冲区函数：fflush()
- 替换缓冲区函数：fsetvbuf()
- 二进制I/O函数：fread()和fwrite()
- 文件尾函数：feof()
- 文件错函数：ferror()
- 使用二进制I/O进行随机存取

```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);
    exit(EXIT_FAILURE);
}
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);
}
```

```c
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
/* 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");
    }
    // finish up
    fclose(iofile);
    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!
```

C程序设计

T13

谢谢

厦门大学信息学院软件工程系

黄炜 副教授