第三章：标准I/O

目标：
本章旨在向学员介绍Linux系统I/O相关函数的使用：
1）掌握I/O相关函数的特点及使用方法
2）了解I/O与系统调用相关的函数之间的区别

时间：3 学时

教学方法：讲授PPT

3.1 关于标准I/O库

功能

标准I/O库的主要目的是提供高效的、扩展的和快捷的文件访问方式。

区别

库提供了比系统调用更多的功能函数，例如格式化输出和数据转变等。

特点

标准库是快捷的，不固定哪一个操作系统，实际上它已经成为独立与UNIX/LINUX系统外C语言的ANSI标准的一部分

3.2 文件操作

标准I/O库中有下列库函数：

fopen、fclose
fread、fwrite
fflush
fseek
fgetc、getc、getchar
fputc、putc、putchar
fgets、gets
printf、fprintf和sprintf
scanf、fscanf和sscanf

3.2.1 文件操作

- fopen函数

类似与底层的open系统调用。主要用于文件的输入输出

```
#include <stdio.h>

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

filename　指定打开的文件

mode参数：

“r” 或 “rb”：　以只读方式打开文件

“w” 或 “wb”:：　以写方式打开，并把文件长度截短为零

“a” 或 “ab”：以写方式打开，新内容追加在文件尾

“r+” 或 “rb+” 或 “r+b”：以修改方式打开(读和写)

“w+” 或 “wb+” 或 “w+b”：以修改方式打开，并把文件长度截短为零

“a+” 或 ”ab+“ 或 ”a+b“：以修改方式打开，新内容追加在文件尾

3.2.1 文件操作

- fclose函数

关闭指定的文件流stream，使所有尚未写出的数据都写出。

```
#include <stdio.h>

int fclose(FILE *stream);
```

3.2.1 文件操作

- fopen例程：

```
#include <stdio.h>

int main()
{
   FILE *fp;
   fp = fopen("file","w");
   fclose(fp);
   return 0;
}
```

3.2.1 文件操作

- fflush函数

把文件流里所有未写出的数据立刻写出。

```
#include <stdio.h>

int fflush(FILE *stream);
```

3.2.1 文件操作

- fflush例程：

```
void flush(FILE *stream)
{
    int duphandle;
    fflush(stream);
    duphandle =  dup(fileno(stream));
    close(duphandle);
}
```

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <io.h>
void flush(FILE *stream);
int main(void)
{
   FILE *stream;
   char msg[] = "This is a test";
   stream = fopen("DUMMY.FIL", "w");
   fwrite(msg, strlen(msg), 1, stream);
   clrscr();
   printf("Press any key to flush DUMMY.FIL:");
   getch();
   flush(stream);
   printf("\nFile was flushed, Press any key to quit:");
   getch();
   return 0;
}
```

3.2.1 文件操作

- fread函数

从一个文件流里读取数据。数据从文件流stream读到ptr指定的数据缓冲区里,函数返回值是成功读到数据缓冲区里的记录个数(不是字节数)

```
#include <stdio.h>

size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
```

size参数指定每个数据记录的长度
nitems给出要传输的记录个数

3.2.1 文件操作

- fwrite函数

从指定的缓冲区里读取数据记录，并把他们写到输出流中，返回值为成功写入的记录个数。

```
#include <stdio.h>

size_t fwrite (const void *ptr, size_t size, size_t nitems, FILE *stream);
```

3.2.1 文件操作

- fseek函数

与lseek系统调用等价的文件流函数。它在文件流里为下一次读写操作指定位置，但函数返回值是一个整数，表示成功与否

```
#include <stdio.h>

int fseek(FILE *stream, long int offset, int whence);
```

3.2.1 文件操作

例程：

```
#include <string.h>
#include <stdio.h>
int main(void)
{
   FILE *stream;
   char msg[] = "this is a test";
   char buf[20];
   if ((stream = fopen("DUMMY.FIL", "w+")) == NULL)
   {
      fprintf(stderr, "Cannot open output file.\n");
      return 1;
    }
    fwrite(msg, strlen(msg)+1, 1, stream);
    fseek(stream, 0, SEEK_SET);
    fread(buf, 1, strlen(msg)+1, stream);
    printf("%s\n", buf);
    fclose(stream);
    return 0;
}
```

3.2.1 文件操作

- fgets、getc和getchar函数

从文件流里读取下一个字节并把它作为一个字符返回。当达到文件尾时，返回EOF。

```
#include <stdio.h>

int fgetc(FILE *stream);
int getc(FILE *stream);
int getchar();
```

3.2.1 文件操作

- fputc、putc和putchar函数

把一个字符写到一个输出文件流中。返回写入的值，如果失败，返回EOF。

```
#include <stdio.h>

int fputc(int c, FILE *stream);
int putc(int c, FILE *stream);
int putchar(int c);
```

3.2.1 文件操作

- fgets和gets函数

从输入文件流stream里读取一个字符串。

```
#include <stdio.h>

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

3.2.2 格式化输入输出

- printf、fprintf和sprintf函数

能够对各种不同类型的参数进行格式编排和输出。

```
#include <stdio.h>

int printf(const char *format, …);
int sprintf(char *s, const char *format, …);
int fprintf(FILE *stream, const char *format, …);
```

每个参数在输出流中的表示形式是由格式参数format控制的

3.2.2 格式化输入输出

- scanf、fscanf和sscanf函数

从一个文件流读取数据，并把数据值放到传递过来的指针参数指向的地指出的变量中。

```
#include <stdio.h>

int scanf(const char *format, …);
int fscanf(FILE *stream, const char *format, …);
int sscanf(const char *s, const char *format, …);
```

3.2.3 其他流函数

- remove函数

相当于unlink函数，但如果它的path参数是一个目录的话，作用相当于rmdir函数

```
#include <stdio.h>

int remove(char *filename);
```

3.2.3 其他流函数

- remove例程：

```
#include <stdio.h>

int main(void)
{
    char file[80];
    printf("File to delete: ");
    gets(file);
    if (remove(file) == 0)
      printf("Removed %s.\n",file);
    else
      perror("remove");
    return 0;
}
```

3.2.3 其他流函数

- fgetpos函数　获得文件流的当前(读写)位置
- fsetpos函数　设置文件流的当前(读写)位置

```
#include <stdio.h>

int fgetpos(FILE *fp, fpos_t *pos);
int fsetpos(FILE *fp, const fpos_t *pos);
```

3.2.3 其他流函数

- fgetpos例程：

```
#include <string.h>
#include <stdio.h>

int main(void)
{
   FILE *stream;
   char string[] = "This is a test";
   fpos_t filepos;
   stream = fopen("DUMMY.FIL", "w+");
   fwrite(string, strlen(string), 1, stream);
   fgetpos(stream, &filepos);
   printf("The file pointer is at byte %ld\n", filepos);
   fclose(stream);
   return 0;
}
```

3.2.3 其他流函数

- ftell函数　　返回文件流当前(读写)位置的偏移值
- rewind函数　　重置文件流里的读写位置

```
#include <stdio.h>

long ftell(FILE *fp);
void rewind(FILE *fp);
```

3.2.3 其他流函数

- setbuf函数　　设置文件流的缓冲区
- setvbuf函数　设置文件流的缓冲机制

```
#include <stdio.h>

void setbuf(FILE *fp, char *buf);
int setvbuf(FILE *fp, char *buf, int mode, size_t size);
```

setbuf buf参数：
　　设置用户缓冲区大小，如果为NULL关闭缓冲区
setvbuf mode参数：
　　_IOFBF　完全缓冲
　　_IOLBF　线性缓冲
　　_IONBF　不缓冲

3.2.3 其他流函数

- setbuf例程：

```
#include <stdio.h>
 char outbuf;
 int main(void)
{
    setbuf(stdout, outbuf);
    puts("This is a test of buffered output.\n\n");
    puts("This output will go into outbuf\n");
    puts("and won't appear until the buffer\n");
    puts("fills up or we flush the stream.\n");
    fflush(stdout);
    return 0;
}
```

3.2.3 其他流函数

- setvbuf例程

```
#include <stdio.h>
int main(void)
{
  FILE *input, *output;
  char bufr[512];
  input = fopen("file.in", "r+b");
  output = fopen("file.out", "w");
  if (setvbuf(input, bufr, _IOFBF, 512) != 0)
      printf("failed to set up buffer for input file\n");
  else
      printf("buffer set up for input file\n");
  if (setvbuf(output, NULL, _IOLBF, 132) != 0)
     printf("failed to set up buffer for output file\n");
  else
     printf("buffer set up for output file\n");
  fclose(input);
  fclose(output);
  return 0;
}
```

3.2.3 其他流函数

- freopen函数　重新使用一个文件流
- fdopen函数　将文件描述符转为文件流

```
#include <stdio.h>

FILE *freopen(const char *pathname, const char *type, FILE *fp);
FILE *fdopen(int fildes, const char *type);
```

3.2.3 其他流函数

- freopen例程

```
#include <stdio.h>

int main(void)
{
   if (freopen(“file.out", "w", stdout)== NULL)
     fprintf(stderr, "error redirecting stdout\n");
   printf("This will go into a file.");
   fclose(stdout);
   exit(0);
}
```

3.2.3 其他流函数

- fdopen例程：

```
#include<stdio.h>

main()
{
    FILE * fp =fdopen(0,”w+”);
    fprintf(fp,”%s\n”,”hello!”);
    fclose(fp);
}
```

3.3 实验：文件拷贝程序

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int c;
    FILE *in, *out;
    in = fopen(“file.in”,”r”);
    out = fopen(“file.out”,”w”);
    while ((c = fgetc(in)!= EOF)
        fputc(c,out);
    exit(0);
}
```

Copyright © 2008 版权所有 东软集团