实验一　标准 I/O 操作编程

1.1 标准 IO 实验（一）

【实验内容】

本实验通过一个简单的程序计算默认缓冲区的大小，理解标准 I/O 提供的三种类型的缓存。

【实验目的】

深入了解标准 IO 的三种类型，即全缓存，行缓存，不带缓存。

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验提示】

全缓冲：当填满 I/O 缓存后才进行实际的 I/O 操作；

行缓冲：当在输入和输出中遇到新换行符(‘\n’)时，进行 I/O 操作；

不带缓冲：标准 I/O 库不对字符进行缓冲，例如 stderr；

【实验步骤】

1、向缓冲区中输入字符

2、当往缓冲区中写的字符数目超过缓冲区大小的时候，才执行输出。

3、可以手工计算出缓冲区大小（341*3+1=1024）

参考代码：

```
#include <stdio.h>
int main()
{
int i=0;
for(i=0;i<379;i++)//每次向缓冲区内写三个字符
{
    if(i>=100)
        fprintf(stdout,"%d",i);
    else if (i>=10)
        fprintf(stdout,"0%d",i);
    else if (i>=0)
        fprintf(stdout,"00%d",i);
}
while(1);//强制执行，如果取消，程序结束时将会输出所有字符，看不到效果了。
}
```

1.2 标准 IO 实验（二）

【实验内容】

本实验通过一个简单的 copy 程序，完成文件的复制程序，了解基本的标准 I/O 文件读写的基本步骤。

【实验目的】

深入了解标准 IO 文件读写的基本原理。

1.学习如何判断文件是否结束

2.熟练掌握标准 I/O 函数

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验提示】

1、判断文件结束有三种方式：

a)
```
char p[1024];
while (fgets(p,1024,src)!= NULL)
    fputs(p,des1);
```

b)
```
int c;
while((c=fgetc(src))!=EOF)
        fputc(c,des2);
```

c)
```
while((n=fread(s,1,20,src)) == 20)
{
//      printf("----------%d\n",t++);
    fwrite(s,1,n,des3);
}
fwrite(s,1,n,des3);
```

2、要拷贝一个文件，即要有源文件和目标文件，使用 fopen 分别打开两个文件，一个文件被读，一个文件被写。完成文件的复制后须将这两个文件都关闭。

参考代码：

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#define maxsize 5

int main(int argc,char *argv[])
{
  FILE *fp1,*fp2;
  char ch[maxsize];
  if(argc!=3)
  {
    printf("command error!\n");
        return -1;// exit(-1);
  }
```

```
 if( (fp1=fopen(argv[1],"r"))==NULL)
  {
     printf("file %s cannot open",argv[1]);
          return -1;//exit(-1);
  }

 if ((fp2=fopen(argv[2],"wa+"))==NULL)
  {
         printf("cannot creat file %s",argv[1]);
         return -1;// exit(-1);
   }
  while(fgets(ch,maxsize,fp1)!=NULL)
     fputs(ch,fp2);

  fclose(fp1);
  fclose(fp2);
  return 0;
}
```

【实验思考】

如果要将一个文件拷贝到两个文件中，怎么办？

提示：

用 fseek()或 rewind（）定位；

1.3 标准 IO 实验（三）

【实验目的】

通过本实验掌握标准 I/O 的使用

【实验内容】

编程读写一个文件 test.txt，每隔 1 秒向文件中写入一行数据，类似这样

1, 2007-7-30 15:16:42

2, 2007-7-30 15:16:43

该程序应该无限循环，直到按 Ctrl-C 中断程序。下次再启动程序写文件时可以追加到原文件之后，并且序号能够接续上次的序号，比如：

1, 2007-7-30 15:16:42

2, 2007-7-30 15:16:43

3, 2007-7-30 15:19:02

4, 2007-7-30 15:19:03

5, 2007-7-30 15:19:04

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验提示】

- 要追加写入文件，同时要读取该文件的内容以决定下一个序号是几，应该用什么模式打开文件？
- 首先判断一下打开的文件是否为新文件，如果是新文件，就从序号 1 开始写入；如果不是新文件，则统计原来有多少行，比如有 n 行，然后从序号 n+1 开始写入。以后每写一行就把行号加 1。
- 获取当前的系统时间需要调用函数 time()，得到的结果是一个 time_t 类型，其实就是一个大整数，其值表示从 UTC 时间 1970 年 1 月 1 日 00:00:00（称为 UNIX 的 Epoch 时间）到当前时刻的秒钟数。然后调用 localtime()将 time_t 所表示的 UTC 时间转换为本地时间（我们是+8 区，比 UTC 多 8 个小时）并转成 struct tm 类型，该类型的各数据成员分别表示年月日时分秒，请自己写出转换格式的代码，不要使用 ctime()或 asctime()函数。具体用法请查阅 man page。time 和 localtime 函数需要头文件 time.h。
- 调用 sleep(n)可使程序睡眠 n 秒，该函数需要头文件 unistd.h。

1.4 标准 IO 实验（四）

【实验目的】

通过本实验掌握标准 I/O 的使用

【实验内容】

一个文件 test.txt，文件内容为：

1

2

4

5

编程读写这个文件，修改其内容，添加一行，将文件内容变成：

1

2

3

4

5

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验提示】

把正确的内容写入一个临时文件，把临时文件重命名即可；

实验二　文件 I/O 操作编程

【实验内容】

本实验通过一个简单的 copy 程序，完成文件的复制程序，了解基本的文件 I/O 文件读写的基本步骤。

【实验目的】

通过本实验掌握文件 I/O 的基本用法

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验步骤】

编写代码，实现相应的功能

打开源文件

打开目标文件

循环读取源文件并写入目标文件

关闭源文件

关闭目标文件

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

#define maxsize 256                          //定义一次从文件读字符的最大数
int main(int argc,char *argv[])
    {

    int fd1,fd2;                            //定义源文件和目的文件的文件描述符
    char buff[maxsize];                      //缓冲
    int i;
    if(argc!=3)                              //如果命令格式不正确
    {
      printf("command error!\n");
          return -1;// exit(-1);
    }

    fd1=open(argv[1],O_RDONLY);              //以只读的方式打开源文件
    if(fd1==-1)
    {
      printf("file %s cannot open",argv[1]);
          return -1;//exit(-1);
    }

    if((fd2=open(argv[2],O_WRONLY|O_CREAT|O_APPEND))==-1)// 以追加的方式创
```

```
建目的文件
        {           printf("cannot creat file %s",argv[1]);
                    return -1;// exit(-1);
                }

         while(1)
         {
         i=read(fd1,buff,maxsize);
         write(fd2,buff,i);
         if(i!=maxsize)  break;                    //如果读到的字节数不是希望的 bufsize，结束
文件读写
         }
         close(fd1);
         close(fd2);
    }
```

实验三　文件目录操作编程

3.1 目录遍历

【实验内容】

本实验通过一个简单的 ls 程序，完成读目录内容程序，了解基本的读目录读的基本步骤。

【实验目的】

目录读的基本原理与 linux 目录操作的原理

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验步骤】

程序代码：

```
#include <stddef.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>          // stat  函数所在的文件
#include <dirent.h>

int main(void)
{
    DIR *dp;
```

```
    struct dirent *ep;
    struct stat st;
    char dirp[50];

    printf("Please input a dir name:\n");

    scanf("%s",&dirp);              //读入目录名

    dp=opendir(dirp);          //打开所给目录

    printf("filename:\ttype:\tPermission\taccesstime\tlastmodtime\tsize\t");

    if(dp!=NULL)                //如果打开目录成功，则进行操作。
    {
        while(ep = readdir(dp))         //读每一个目录项的循环
        {
            if(ep->d_name[0]!='.')       //判断文件名称的第一个字符是否'.'，如果是，表明是隐含文件，我们不动，否则操作
    {//用 stat 函数打开文件的信息，第一个参数是文件的路径，第二个参数存放文件的信息
            if(stat(ep->d_name,&st)!=-1)   //读成功
    {
                printf("%s\t",ep->d_name);

                 if((st.st_mode&S_IFMT)==S_IFDIR)            //判断文件的类型
                   printf("Directory\t");                          //目录
                  else if((st.st_mode&S_IFMT)==S_IFBLK)     //块文件
                   printf("Block special file\t");
                  else if((st.st_mode&S_IFMT)==S_IFCHR)     //特殊字符文件
                      printf("character special file\t");
                  else if((st.st_mode&S_IFMT)==S_IFREG)     //普通文件
                      printf("Ordinary file\t");
                  else if((st.st_mode&S_IFMT)==S_IFIFO)     //管道文件
                      printf("pipefile file\t");
                printf("%o\t",st.st_mode&0x1ff);               //文件的权限
                printf("%15s\t",ctime(st.st_atime));          //文件创建时间
                printf("%15s\t",ctime(st.st_mtime));          //文件上次修改时间
                printf("%ld\n",st.st_size);                     //文件的大小
```

```
            }
    }
            }
        closedir(dp)                    //关闭目录
        }
        else
            puts("Couldn't open the directory.\n");      //打开不成功时，输出不能打开路径

        return 0;

    }
```

3.2 文件信息的遍历

【实验内容】

编写一个程序实现“ls –l FILEIO”的功能，使其显示

drwxr-xr-x 2 linux linux 4096 2009-08-12 09:50 FILEIO

通过此次实验掌握如何获取文件信息的系统调用及 stat（）的使用方法

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验步骤】

要求实现以下功能：

列出的信息包括模式字段，链接数，用户名，组名，文件大小，文件建立时间，文件名，以上字段从左到右一次显示。

【实验提示】

用 stat 获得文件信息，存储在 struct stat 结构中，/usr/include/sys/stat.h 描述了 struct stat 的成员变量，这里只列出要用到的成员变量

ushort st_mode; 文件类型和许可权限

ushort st_nlink; 连接数

ushort st_uid; 属主的用户 ID

ushort st_gid; 属主的组 ID

ulong st_size; 普通文件的字节长度

ulong st_ctime; 最后文件状态更改时间

用 ctime 将文件状态的最后修改时间转换为字符串；

用/usr/include/sys/stat.h 中定义的宏将 st_mode 的位分离出来，并填充到一个字符数组中，等待输出；

用 getpwuid()和 getgrgid()将用户 ID 和组 ID 然后转换成字符串。

【实验思考】

1、在以上的基础上如果输入的是文件目录怎么办？将以上的函数封装起来作为一个功

能函数。利用 opendir（）、readdir（）、closedir（）来操作目录，并调用前面编写的函数处理和输出文件信息。

2、如何实现类似于“ls”的一个 myls 的命令，即可以实现包含-l 、-a 参数的 myls 命令。

参数处理可以利用 getopt 函数来实现。