

# 文件同步工具 ——大作业程序说明报告

信息学院 2013202487 白非凡

## 说明：

本次大作业主要针对在Linux平台下的文件同步工具的开发，而在windows平台下只需将Linux下的函数换成windows平台的相应函数，原理基本相同，故不赘述。

## 作业要求：

有2个文件夹A和B，经过之前的同步操作，文件夹B中的内容与文件夹A中内容完全一致，现在用户对A文件夹中部分文件进行了编辑、移动、删除或者新增的操作，编写程序根据文件夹A中的内容变化对文件夹B进行增量更新。

## 基本要求：

1. 文件夹A中保持不变的文件在文件夹B中保持不变，不更新。
2. 给出文件夹A中文件的更新情况列表，即列出与上次同步后文件夹A中所有发生变化的文件列表，包括：

- a) 与上次更新相比，所有新增文件的列表（包括文件的信息）
- b) 与上次更新相比，所有删除文件的列表
- c) 与上次更新相比，所有发生变动的文件的列表

只对文件夹A中发生变化的文件在文件夹B中更新，即针对上述3中变化的文件，删除文件夹B中的相应原文件，并将A中最新文件复制到文件夹B中。

注意：删除时需要提示用户是否确定删除源文件。

## 提高要求：

1. 将上述工具移植到windows平台下。
2. 在文件夹B中保留更新前的文件（比如，重新以特定的命名方式命名或者保存在特定的文件夹中的方式）。
3. 开发图形界面

## 问题分析：

本次大作业要实现的功能是两个文件夹中文件的同步。首先要将新、旧文件底下的目录进行比较，然后进行增添、删除、复制等操作使两个文件夹内文件相同。其中最关键的步骤就是文件夹的遍历和目录存储。判断两个文件是否相同可以先比对文件名，然后比对最后修改时间。而增添、删除、复制等操作则可以调用适用于Linux平台的库函数实现。而实现文件夹的遍历和

目录存储有如下两种方式：

- 1、利用system函数调用系统命令
- 2、使用Linux系统提供的API

考虑到操作的简便性和安全性，本次大作业代码采用的是Linux系统提供的API

## 实现方案：

本次大作业采用的是Linux系统提供的API，在查阅相关资料后，用到了如下相关Linux相关库函数和API

1、文件夹的遍历 ( opendir(),readdir(),mkdir(),Istat(),access(),closedir() )

①opendir()

**头文件：**#include<sys/types.h> #include<dirent.h>

**函数原型：**DIR\* opendir (const char \* path );

**功能：**打开一个目录，在失败的时候返回一个空的指针。

②readdir()

**函数原型：**struct dirent\* readdir(DIR\* dir\_handle);

**返回值：**dirent 的结构类型

**功能：**读取目录，返回 dirent 结构体指针

③mkdir()

**头文件库：**#include<direct.h>

**函数原型：**int \_mkdir(const char \*dirname );

**功能：**创建一个目录，若成功则返回 0，否则返回-1

④Istat()

**函数原型：**int Istat(const char \*path, struct stat \*buf);

**参数：**path：文件路径名。 filedес：文件描述词。 buf：结构体的指针

**功 能：**获取文件相关的信息，成功返回 0，失败返回-1

⑤access()

**头文件库：**#include<unistd.h>

**函数原型：**int access(const char \*filename, int amode);

int \_access(const char \*path,int mode) ;

**功能：**确定文件的访问权限，如果指定的存取方式有效，即存在，则函数返回 0，否则返回-1。

⑥closedir()

**头文件库：**#include<sys/types.h>#include<dirent.h>

**函数原型** : int closedir(DIR \*dir);

**功能** : 关闭参数dir所指的目录流。关闭成功则返回0，失败返回-1建立目录

2、文件夹下目录信息存储（构造结构体stat，存储用户信息，最后访问时间，最后修改时间，最后状态改变时间等）

①<sys/stat.h>中的结构体stat：

```
struct stat {  
    time_t    st_atime; /* 最后访问时间 */  
    time_t    st_mtime; /* 最后修改时间 */  
    time_t    st_ctime; /* 最后状态改变时间 */  
};
```

3、删除(remove(),S\_ISDIR())

①remove()

**函数原型** : int remove(char \*filename);

**功能** : 删除一个文件

②S\_ISDIR()

**函数原型** : S\_ISDIR(stat.st\_mode)

**功能** : 判断一个路径是否为目录

4、复制 ( fopen(),fgetc(),fclose() )

①fopen()

**函数原型** : FILE \* fopen(const char \* path,const char \* mode);

**功能** : 打开一个文件，文件顺利打开后，指向该流的文件指针就会被返回。如果打开失败则返回 NULL，并把错误代码存在 errno 中。

**rb** : 读写打开一个二进制文件，允许读写数据，文件必须存在。

**wb** : 只写打开或新建一个二进制文件；只允许写数据。

②fgetc()

**函数原型** : int fgetc(FILE \*stream);

**功能** : 返回所读取的一个字节。如果读到文件末尾或者读取出错时返回 EOF。

③fclose()

**函数原型** : int fclose(FILE \*stream);

**功能** : 关闭一个打开文件。

注：本次大作业的实现，主要分为遍历、删除、复制三个模块：

```
void List(char *path, char *path2,int indent) ;
```

```
void Delete(char *path2, char *path, int indent) ;  
int copy(char *path1,char *path2) ;
```

### 算法描述：

```
#include<unistd.h>  
#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#include<fcntl.h>  
#include<dirent.h>  
#include<string.h>  
#include<sys/stat.h>  
#include<sys/types.h>  
#include <errno.h>
```

```
#define BUFSIZE 1000
```

```
int copy(char *path1,char *path2);  
void List(char *path, char *path2,int indent)  
{  
    struct    dirent*ent = NULL;  
    DIR        *pDir;  
    char    dir[512],dir2[512];  
    struct stat statbuf;  
  
    if( (pDir=opendir(path))==NULL )  
    {  
        fprintf(stderr, "Cannot open directory:%s\n", path);  
        return;  
    }  
    while( (ent=readdir(pDir))!=NULL )  
    {  
        //得到读取文件的绝对路径名  
        snprintf(dir, 512,"%s/%s",path,ent->d_name);  
        snprintf(dir2, 512,"%s/%s",path2,ent->d_name);  
        //得到文件信息  
        lstat(dir, &statbuf);  
        //判断是目录还是文件，文件则返回1  
        if(S_ISDIR(statbuf.st_mode) )  
        {  
            //排除当前目录和上级目录  
            if(strcmp(".",ent->d_name) == 0 || strcmp("..",ent->d_name) == 0)
```

```

    {
        continue;
    }
    //如果是子目录,递归调用函数本身,实现子目录中文件遍历
    // 确定文件是否存在, 存在则返回0, 否则返回-1
    if (access(dir2,0)==-1)
    {
        mkdir(dir2, S_IRWXU);//建立一个目录, 拥有读, 写和操作权限
    }
    //递归调用, 遍历子目录中文件
    List(dir, dir2,indent+4 );
}
else
{
    if (access(dir2, 0)==-1)
    {
        copy(dir,dir2);
        printf("%s has been copied.\n",dir);
    }
    else
    {
        struct stat st1,st2;
        stat(dir,&st1);
        stat(dir2,&st2);
        if (st1.st_mtime!=st2.st_mtime)
        {
            copy(dir,dir2);
            printf("%s has been copied.\n",dir);
        }
    }
}
closedir(pDir);
}

```

**void Delete(char \*path2, char \*path, int indent)**

```

{
    struct    dirent*ent = NULL;
    DIR      *pDir;
    char      dir[512],dir2[512];
    struct stat statbuf;
    char c;
    if( (pDir=opendir(path))==NULL )

```

```

    {
        fprintf(stderr, "Cannot open directory: %s\n", path );
        return;
    }
while( (ent=readdir(pDir))!=NULL )
{
    snprintf( dir,512,"%s/%s", path, ent->d_name );
    snprintf( dir2,512,"%s/%s", path2, ent->d_name );
    //lstat函数获取一些文件相关信息
    lstat(dir, &statbuf);
    if(S_ISDIR(statbuf.st_mode) )//是目录的情况
    {
        if(strcmp( ".",ent->d_name) == 0||strcmp( "..",ent->d_name) == 0)
        {
            continue;
        }
        if (access(dir2,0)==-1)
        {
            Delete(dir2,dir,indent+4 );
            remove(dir);//删除一个文件
            printf("%s has been deleted.\n",dir);
        }
        else
            Delete(dir2,dir,indent+4);
    }
    else
    {
        if (access(dir2, 0)==-1)
        {
            remove(dir);
            printf("%s has been deleted.\n",dir);
        }
    }
}
closedir(pDir);
}

```

**int copy(char \*path1,char \*path2)**

```

{
    int c;
    FILE *fpSrc, *fpDest; //定义两个指向文件的指针
    fpSrc = fopen(path1, "rb"); //以读取二进制的方式打开源文件
    if(fpSrc==NULL){

```

```

        printf("Unable to open source file.\n"); //源文件不存在的时候提示错误
        return 0;
    }
    fpDest = fopen(path2, "wb"); // //以写入二进制的方式打开目标文件
    if(fpDest==NULL){
        printf("Destination file open failure.\n");
        return 0;
    }
    while((c=fgetc(fpSrc))!=EOF){ //从源文件中读取数据知道结尾
        fputc(c, fpDest);
    }
    fclose(fpSrc); //关闭文件指针，释放内存
    fclose(fpDest);
    return 0;
}

```

```

int main()
{
    char    path1[512],path2[512];
    printf("Please input path1 path2.\n");
    scanf("%s %s",path1,path2);
    List(path1,path2,2);
    Delete(path1,path2,2);
    printf("All work has been done.\n");
    return 0;
}

```

## 遇到的问题以及解决：

1、**遍历函数**：一开始写遍历函数的时候，不知道通过什么方式来存储文件夹下的目录信息。后来在老师给的遍历函数的基础上，采用了access()函数，可以确定文件是否存在。这样只需要对file1中的文件进行一一遍历，再在file2的中依次查找file2中是否存在一样的文件，再进行删除、复制等操作就能实现文件同步。

参考资料：百度知道：《C 语言 检查文件是否存在》

<http://zhidao.baidu.com/link?url=oC4jNnEsM05R1eUtIIoJsGG3VQvf2hA0RjGYZaSlebAfhTtsikZoR3g735rxkcSGddTBDrDFdXLXJtg9F89DdUpfrGwdP6eu-rJOlljN7bG>

2、**删除函数**：删除文件操作一开始不知道怎么写，后来在网上找到了remove()函数（其实一共有unlink()rmdir()remove()三个函数，其中remove()适用最广，最便捷），可以直接调用删除一个文件，十分简洁方便（在test文件夹的file1，file2之间就涉及了图片和文字之间的复制）

参考资料：《Linux 下 C 编程删除一个文件,该如何解决》

<http://www.myexception.cn/linux-unix/252009.html>

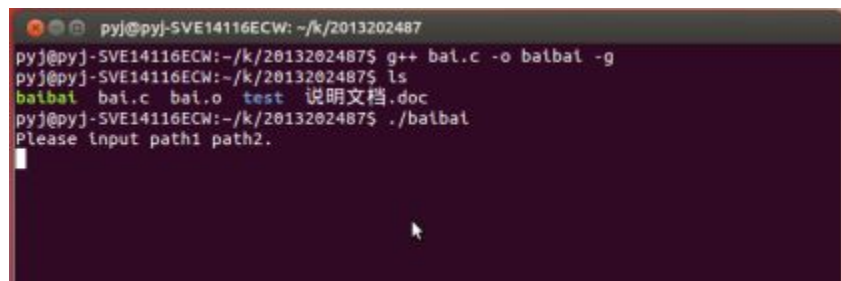
3、**复制函数**：一开始考虑对不同类型的文件利用文件扩展名进行分类再复制的方式（如：jpg、word、txt等文件）但是工作量繁琐，无从操作，后来采用以二进制的方式读取目标文件并写入的方式，因为所有文档在电脑上都是以二进制代码的方式存储的，这样就构成了统一，完美地解决了这一问题（在本次大作业测试数据的test文件夹的file1，file2之间就涉及了图片和文字之间的复制）。

代码参考：《如何用 c 语言实现文件的复制》

[http://blog.sina.com.cn/s/blog\\_a3f0d68201013f0u.html](http://blog.sina.com.cn/s/blog_a3f0d68201013f0u.html)

## 使用说明：

- 1、打开终端，利用g++编译器编译好的代码如图所示，运行编译好的代码“baibai”，出现：“Please input path1 path2.” 的界面。



```
pyj@pyj-SVE14116ECW: ~/k/2013202487
pyj@pyj-SVE14116ECW:~/k/2013202487$ g++ bai.c -o baibai -g
pyj@pyj-SVE14116ECW:~/k/2013202487$ ls
baibai  bai.c  bai.o  test  说明文档.doc
pyj@pyj-SVE14116ECW:~/k/2013202487$ ./baibai
Please input path1 path2.
```

- 2、按照提示，依次输入要同步的两个文件夹的路径（将path1的内容同步到path2）



```
pyj@pyj-SVE14116ECW: ~/k/2013202487
pyj@pyj-SVE14116ECW:~/k/2013202487$ g++ bai.c -o baibai -g
pyj@pyj-SVE14116ECW:~/k/2013202487$ ls
baibai  bai.c  bai.o  test  说明文档.doc
pyj@pyj-SVE14116ECW:~/k/2013202487$ ./baibai
Please input path1 path2.
./test/file1 ./test/file2
```

- 3、轻敲回车，代码自动运行，在运行过程中，界面上会打印出所有变更信息。最后，当所有操作执行完毕后，界面上会显示“All work has been done.” 的提示语，同步结束，轻敲回车退出。



```
pyj@pyj-SVE14116ECW: ~/k/2013202487
pyj@pyj-SVE14116ECW:~/k/2013202487$ g++ bai.c -o bai -g
pyj@pyj-SVE14116ECW:~/k/2013202487$ ls
baibai  bai.c  bai.o  test  说明文档.doc
pyj@pyj-SVE14116ECW:~/k/2013202487$ ./baibai
Please input path1 path2.
./test/file1 ./test/file2
./test/file1/7.doc has been copied.
./test/file1/027.JPG has been copied.
./test/file1/5.doc has been copied.
./test/file2/2.doc has been deleted.
./test/file2/6.doc has been deleted.
All work has been done.
pyj@pyj-SVE14116ECW:~/k/2013202487$
```

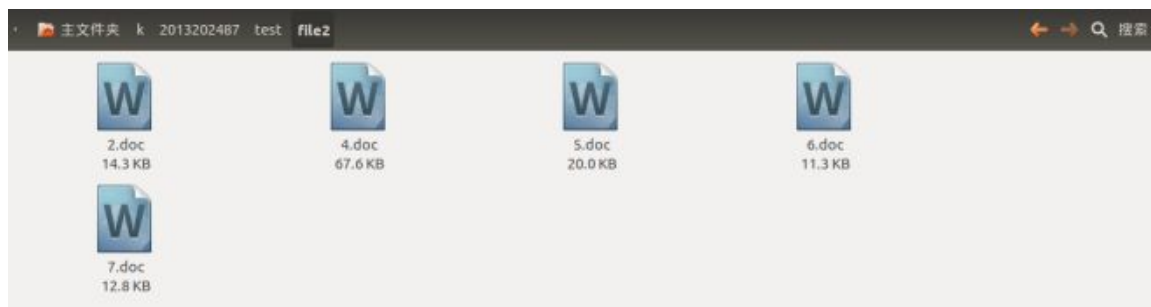
测试文件夹：



测试前：file1



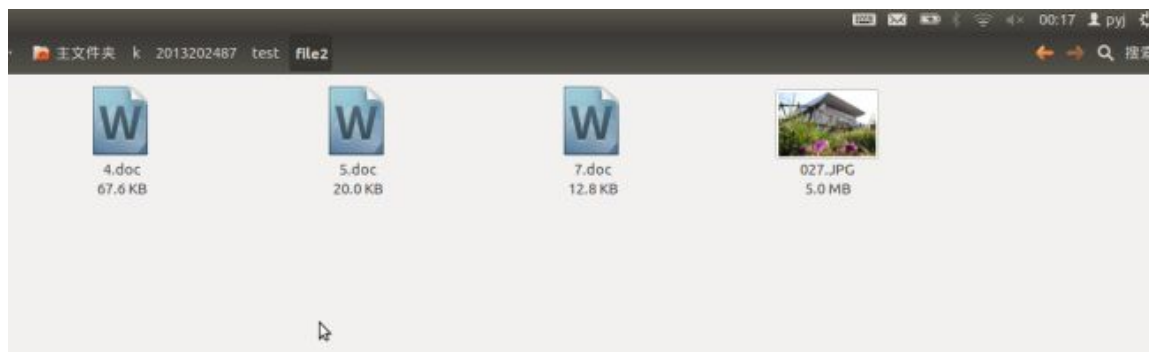
file2



测试后：file1



file2



**注：**经测试，本文件同步工具不仅能实现文档之间的同步，还适用于文档和图片之间，其他格式的文件同理可进行相应的同步过程，不一一赘述。