1，需求分析

1. 通过线程池技术和内核链表实现根目录或者其他目录的拷贝，如果遇到同名的文件 则覆盖原来的文件或者目录
2. 先遍历目录，然后提取目录的内容
3. 在拷贝文件之前，先把目录建立好，拷贝文件的时候可以是任意位置的拷贝

2，设计阶段

1. 数据结构的设计

//目录节点的数据结构

typedef struct directory

{

char dir\_name[300]; //存放目录的绝对路径

struct list\_head list; //内核链表成员

}directory;

//文件节点的数据结构

typedef struct file

{

char file\_name[300]; //存放文件的绝对路径

struct list\_head list; //内核链表成员

}file;

//线程池的数据结构

typedef struct thread\_pool

{

pthread\_mutex\_t lock; //互斥锁，保护任务链表

pthread\_cond\_t cond; //条件锁，保持线程同步

struct task \*task\_list;

pthread\_t \*tids; //任务链表

unsigned waiting\_tasks; //等待处理任务的数量

unsigned active\_threads; //活跃的线程

bool shutdown; //线程池总开关

}thread\_pool;

//任务的数据结构

struct task

{

void \*(\*task)(void \*arg); //自定义任务函数列表

void \*arg; //任务函数的参数

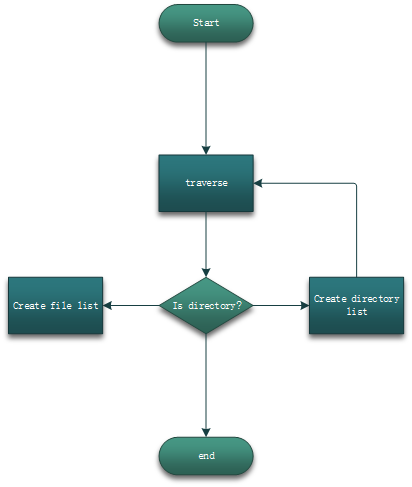
struct task \*next; //连接下一个节点

};

1. 接口的设计

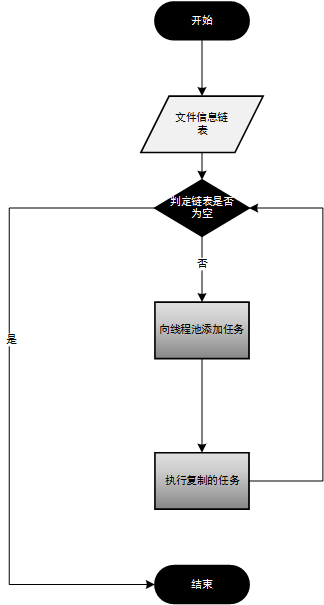
（1）//递归遍历源路径，提取目录位置和文件信息

bool traverse\_src(char \*src\_path)



（2）//复制文件

bool copy\_file(file \*head, char \*dest\_path)



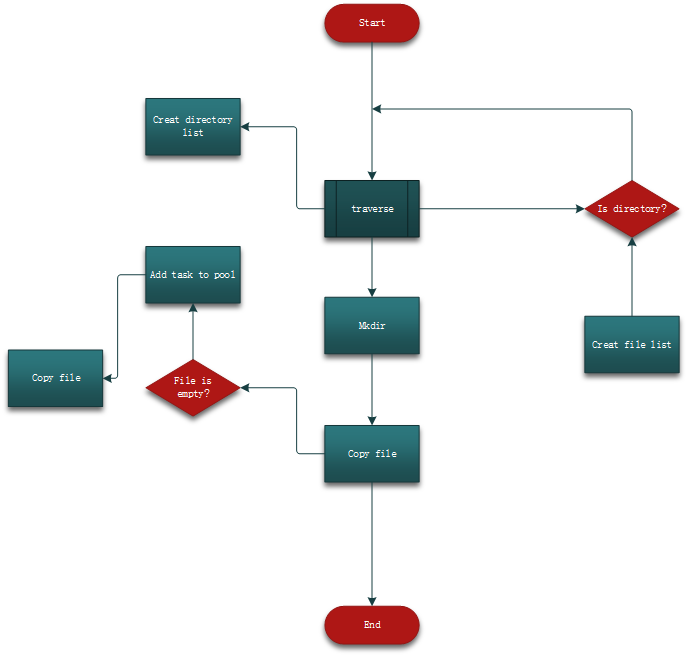
（3）//复制目录

bool copy\_directory(directory \*head, char \*dest\_path)

（4）//复制的任务

void \*copy\_task(void \*arg)

3，程序流程图



4，核心代码段

//递归遍历源路径，提取目录位置和文件信息

bool traverse\_src(char \*src\_path)

{

if(src\_path == NULL)

return false;

DIR \*dirp = NULL;

dirp = opendir(src\_path);

if(dirp == NULL)

{

perror("src directory not exist\n");

return false;

}

else

printf("OPEN --> %-25s\n", src\_path);

struct dirent \* entry;

char path\_buf[300];

bzero(path\_buf, sizeof(path\_buf));

char ch, \*p = NULL;

char file\_buf[300];

bzero(file\_buf, sizeof(file\_buf));

char ch1, \*p1 = NULL;

while(entry = readdir(dirp))

{

if(entry->d\_type == DT\_DIR) //判断是否为目录

{

if(!strcmp(entry->d\_name, ".") || !strcmp(entry->d\_name, ".."))

continue;

//继续递归调用,源路径的末尾不可以有'/'

//if( strcasecmp(src\_path, ))

if( !strcasecmp(src\_path, "/") || !strcasecmp(src\_path, "./"))

sprintf(path\_buf, "%s%s", src\_path, entry->d\_name);

else

sprintf(path\_buf, "%s/%s", src\_path, entry->d\_name);

printf("DIR --> %-25s\n", path\_buf);

//统计目录数

++real\_directory\_count;

//创建目录链表

add\_directory\_node(directory\_head, path\_buf);

traverse\_src(path\_buf);

}

if(entry->d\_type == DT\_REG) //判断是否为普通文件

{

ch1 = '/';

p1 = strrchr(src\_path, ch1);

int flag1 = strcasecmp(p1, "/");

if(flag1 ==0)

{

sprintf(file\_buf, "%s%s", src\_path, entry->d\_name);

}

else

sprintf(file\_buf, "%s/%s", src\_path, entry->d\_name);

printf("REG --> %-25s\n", file\_buf);

//统计文件数

++real\_file\_count;

//创建文件信息链表

add\_file\_node(file\_head, file\_buf);

}

}

closedir(dirp); //必须关闭打开的目录，否则会出现打开文件或者目录太多 这个错误

return true;

}