电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

实验五 | 线程控制实验

主讲老师：杨珊

实验五 | 线程控制实验目的

目的一： 掌握UNIX/Linux系统创建线程的方法

目的二： 理解线程启动例程的设计思想

目的三： 掌握线程终止的方式

目的四： 掌握线程之间共享数据的方法

目的五： 掌握等待线程终止的方法

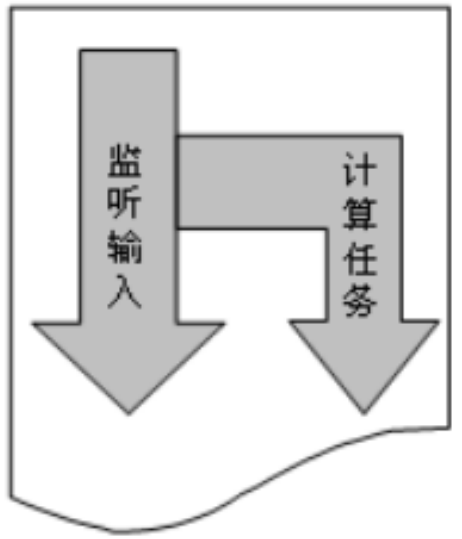
实验五 | 线程设计思想

多线程程序主要用于需要并发执行的场合

- 游戏场景中：玩家通过鼠标键盘输入操作指令，控制游戏进行

  无多线程，则程序等待玩家指令输入，程序的动画效果停止

- 在线视频播放存在同样的问题



实验五｜实验原理｜创建线程

头文件： pthread.h

函数： int pthread_create(pthread_t *restrict tidp,
const pthread_attr_t *restrict attr,
void *(*start_rtn)(void *),
void *restrict arg);

作用： 创建一个线程

返回值： 出错返回错误码；成功返回0

实验五｜实验原理｜创建线程函数的参数

参数	参数的意义
tidp	指向线程ID的指针，当函数成功返回时将存储所创建的子线程ID
attr	用于指定所创建线程的属性（一般直接传入空指针NULL，表示线程属性采用默认值）
Start_rtn	创建函数的子线程
arg	NULL标识线程属性为默认值
	Childth序中的
	向childthread传入str参数

例：pthread_create(&tid , NULL , (void *) childthread , str);

实验五 | 线程例程函数

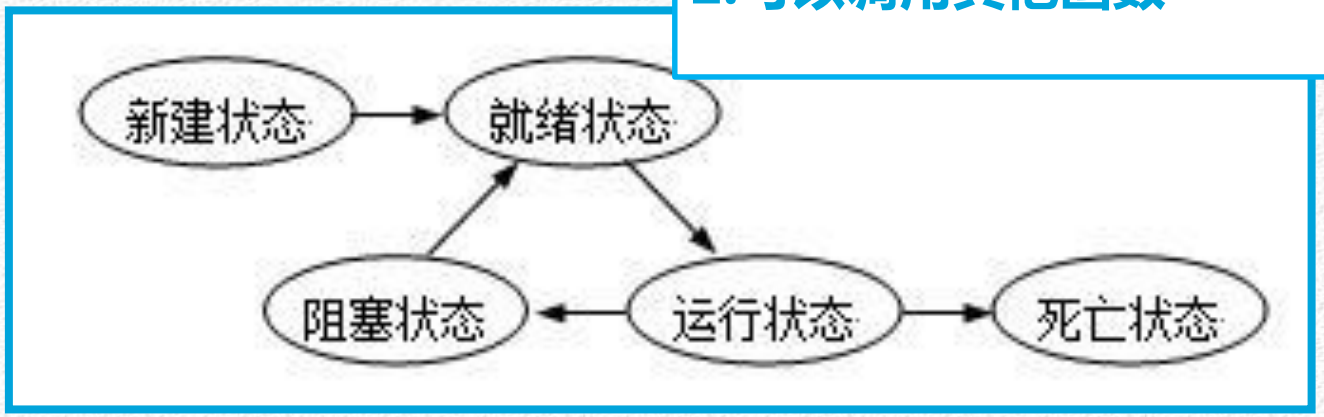
线程的启动

✓线程创建后等待系统调度，被调度后从线程**启动例程函数**开始执行

✓一次性创建多个线程，调度

1.可以执行各种业务流程（循环、条件、分支）

2.可以调用其他函数



实验五｜启动例程函数

关于启动例程： pthread_create(&tid , NULL , (void *) childthread , str);

例程函数原型： void * start_rtn(void *arg)

参数 :void *arg

不需要传递

pthread_c

，此时

返回值

void *，可

、结构体、

缓冲区等

不需要传参时，则pthread_create第四个参数为Null

arg可以是可以是指向任意数据结构的指针，包括整形、结构体、缓冲区等

实验五 |线程的三种退出方式

线程的终止三种方式

- ✓ 线程从启动例程函数中返回，函数返回值作为线程的退出码
- ✓ 线程调用pthread_exit函数终止自身执行
- ✓ 线程被同一进程中的其他线程取消

实验五 | 实验原理 | 线程的终止

头文件： pthread.h

函数： void pthread_exit(void *rval_ptr);

作用： 线程调用后将会终止自身

参数： rval_ptr指针将会传递给等待线程终止函数

实验五 | 实验原理 | 父线程等待子线程终止

头文件： pth

函数： int pt **rval_ptr);

作用： 父线程 止

返回值： ✓

✓出错返回错误编号

- 线程从**启动例程返回**，rval_ptr将指向返回值
- 线程**被**其他线程**取消**，rval_ptr指向的值置为PTHREAD_CANCELED
- 线程通过**调用pthread_exit终止**，rval_ptr就是pthread_exit函数中指定的参数
- 不关注返回值，则该参数使用时使用NULL

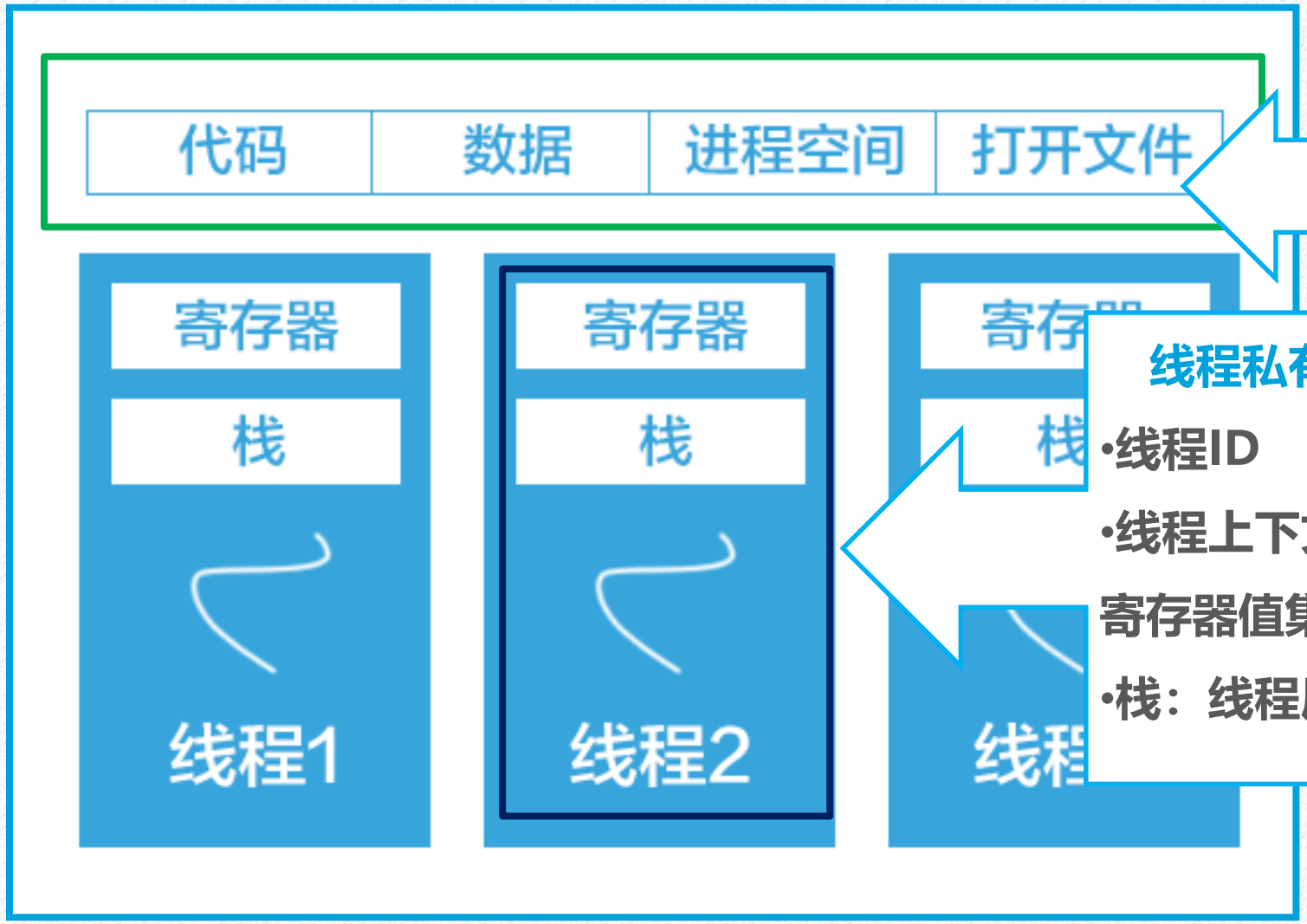
实验五｜演示程序

testthread.c文件有childthread函数和main函数

```
void *childthread(void *arg){
    int i;
    for(i=0;i<10;i++){
        printf("childthread
                message:%s\n",
                (char *)arg);
        sleep(1);}
    return 0;}
```

```
int main(int argc,char** argv){
        pthread_t tid;
        char str[100];
        printf("create childthread\n");
        sprint(str,"str from parent");
  pthread_create(&tid,NULL,(void *)
                        childthread,str);
        pthread_join(tid,NULL);
        printf("childthread exit\n");
        return 0;}
```

实验五 | 线程的数据共享



一个进程内的所有线程共享进程的多种数据和资源

引发：同步问题

线程私有内容：

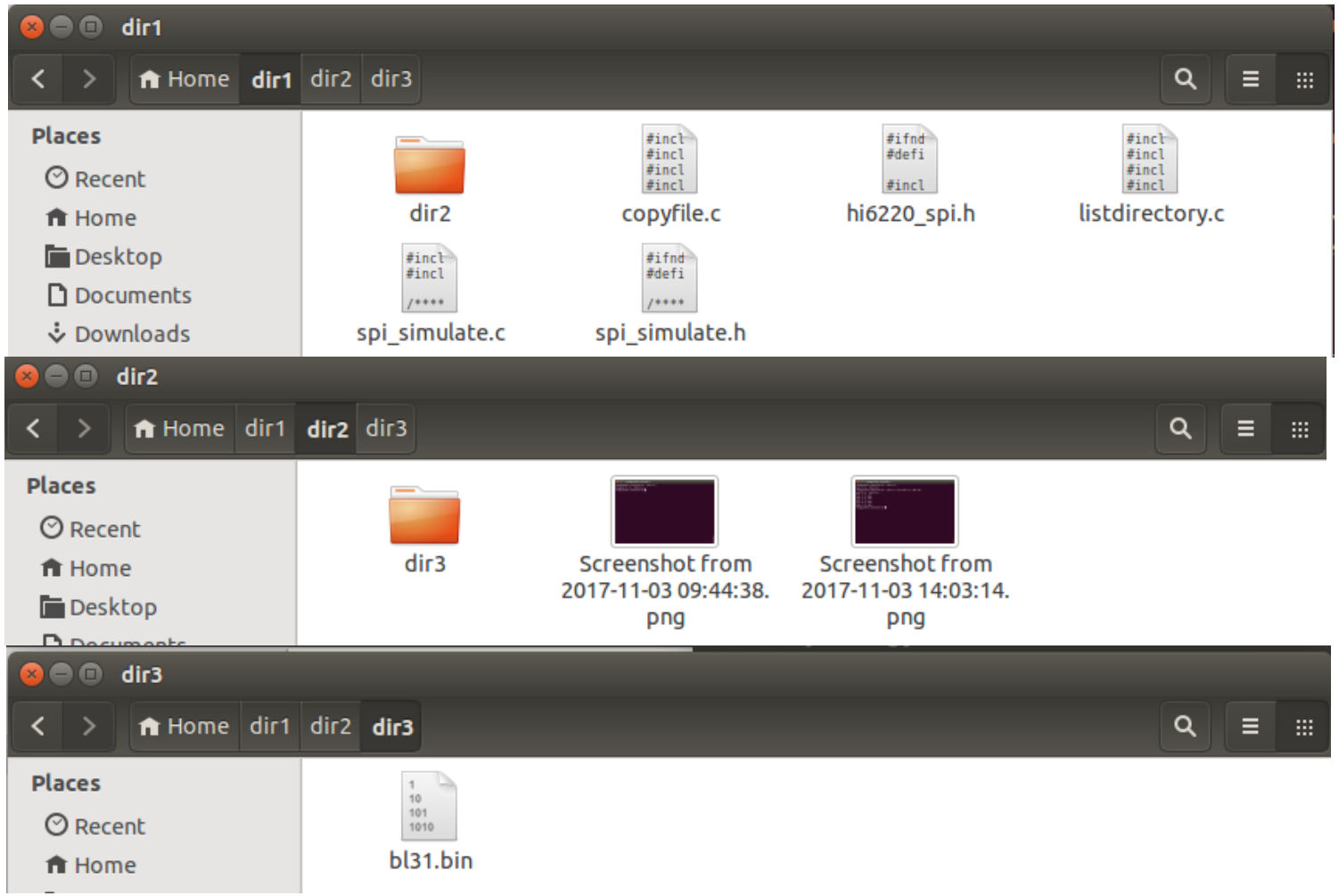
- 线程ID
- 线程上下文：一组寄存器值集合
- 栈：线程局部变量

实验五 | 线程控制实验内容

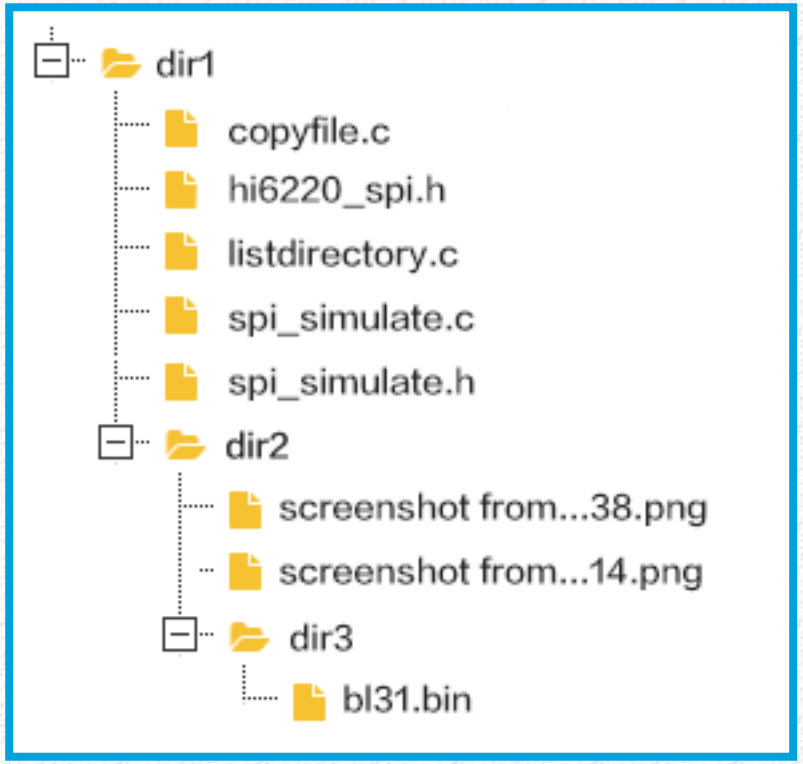
基于已经实现的实验二文件拷贝（mycp）以及实验三目录遍历（myls）的内容（与实验四的区别为并发单位变为线程）

1. 改造myls程序作为从属子线程，其在遍历目录时，对非目录文件再次创建子线程运行mycp程序。
2. mycp源文件路径是父主体线程myls遍历所获取的文件的路径名（通过命令行参数传递给子进程mycp），并将源文件拷贝到指定目录下（在/home目录下以自己的名字的汉语拼音创建一个目录）。
3. 线程myls等待线程mycp运行结束，回收其内核空间资源，main线程等待myls遍历完成，程序结束。

实验五 | 源文件目录结构

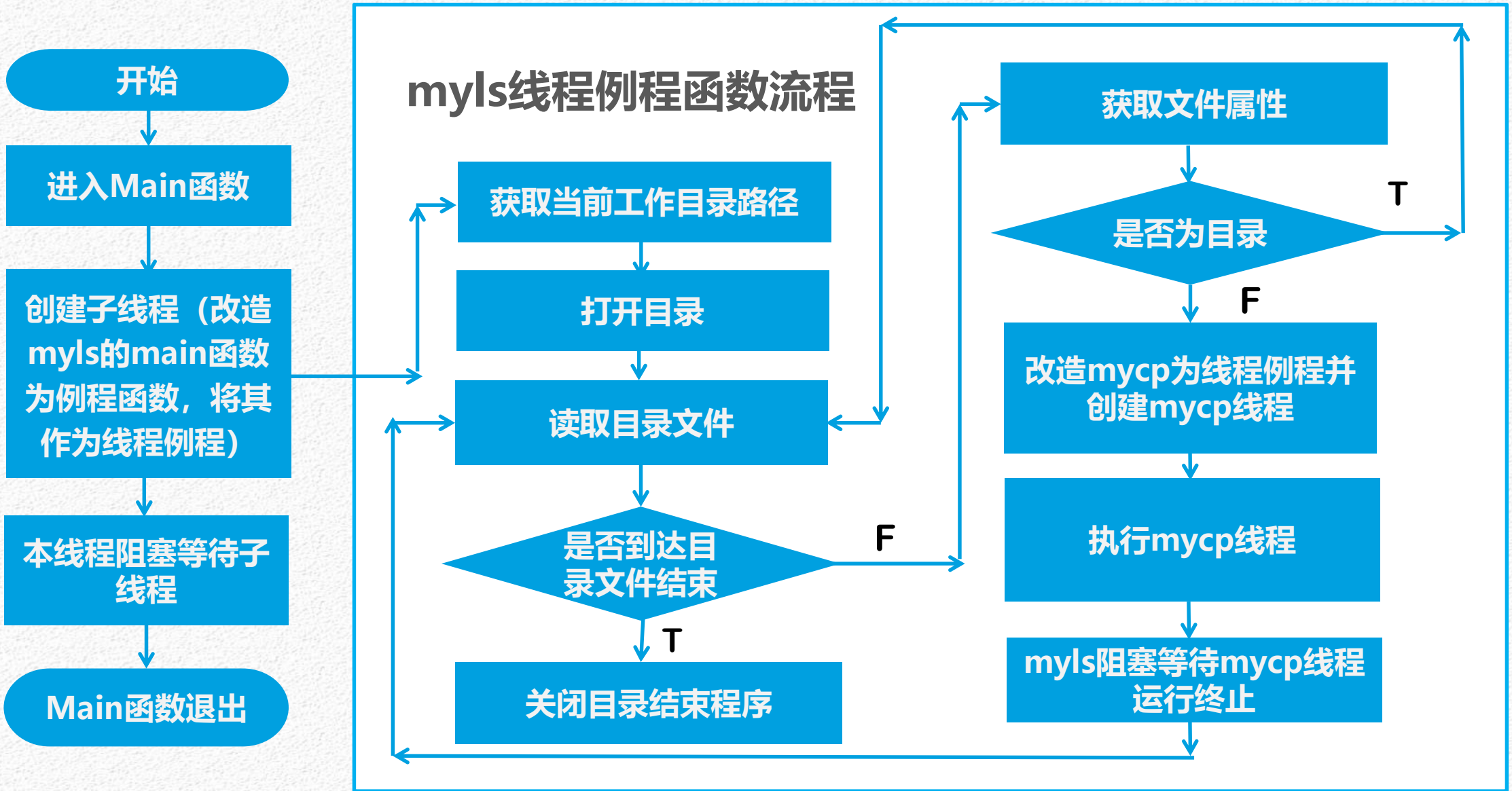


源目录结构如下所示：



实验五 | 实现对非目录文件拷贝

实验五 | 程序流程

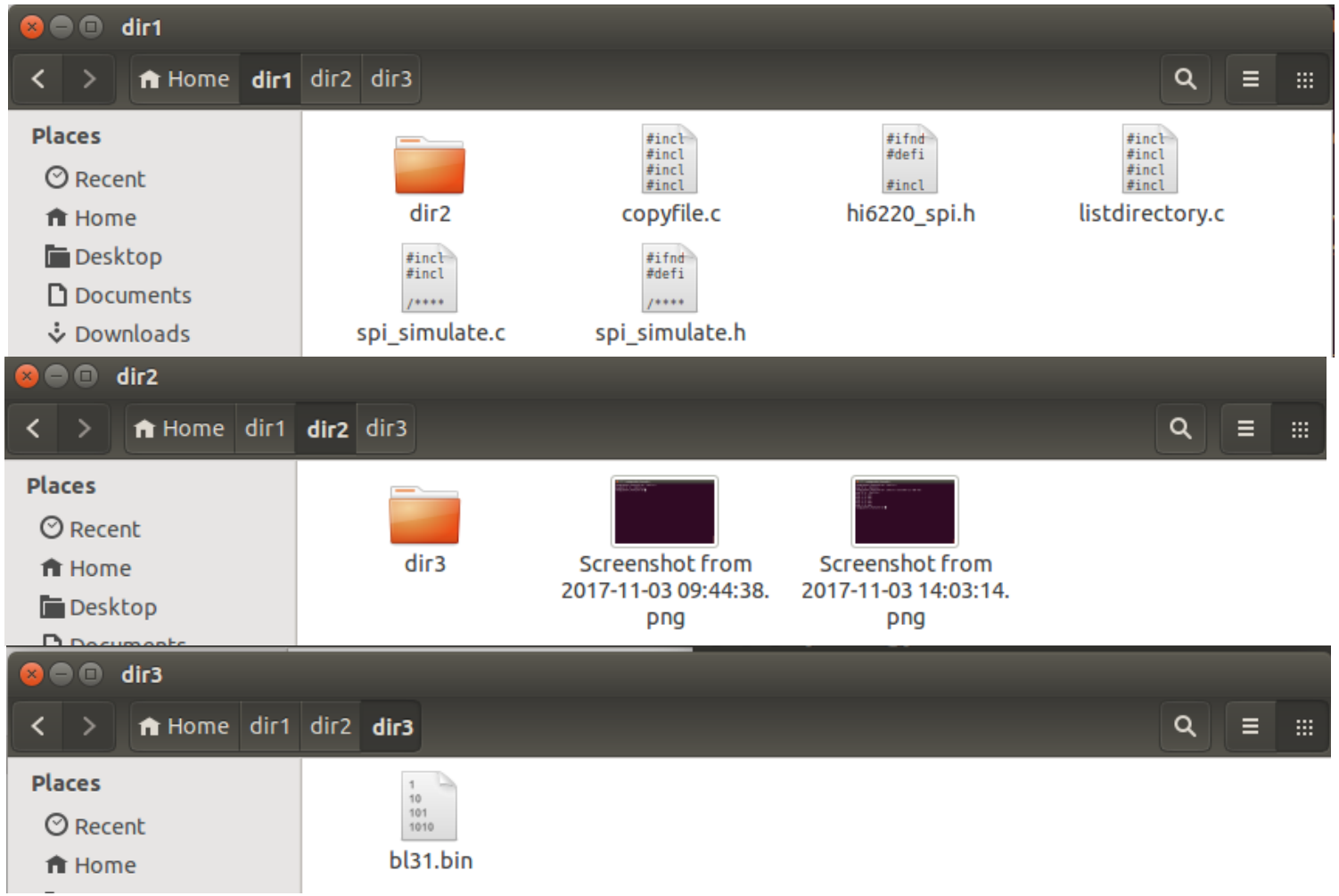

开始
进入Main函数
创建子线程（改造myls的main函数为例程函数，将其作为线程例程）
本线程阻塞等待子线程
Main函数退出
myls线程例程函数流程
获取当前工作目录路径
打开目录
读取目录文件
是否到达目录文件结束
F
T
关闭目录结束程序
获取文件属性
是否为目录
T
F
改造mycp为线程例程并创建mycp线程
执行mycp线程
myls阻塞等待mycp线程运行终止

实验五 | 线程控制实验内容扩展

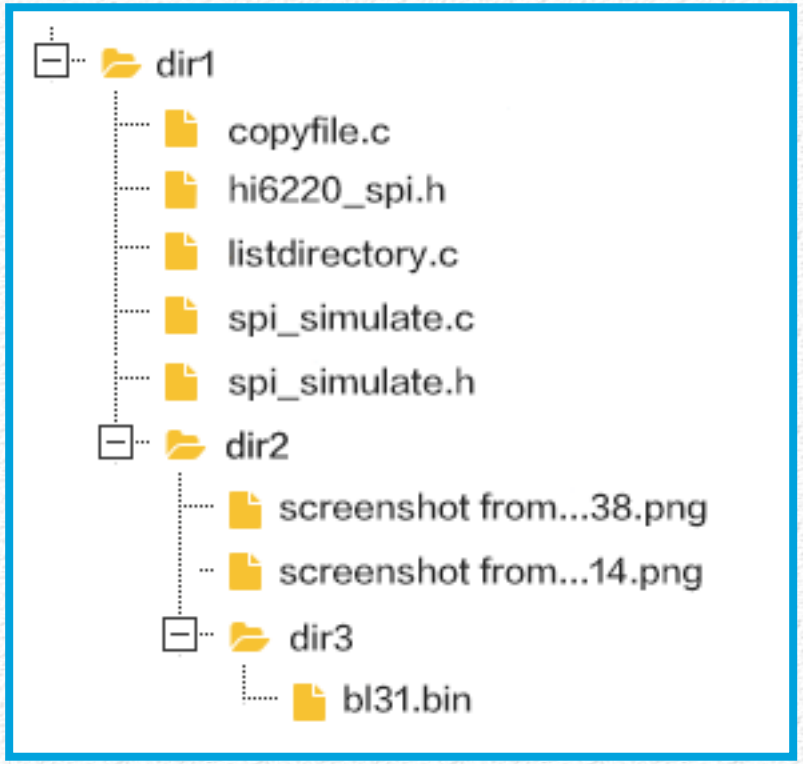
扩展一：在实现基本功能实现的基础上，对程序功能进行扩展，以支持对目录的递归遍历并将**所有目录**及**子目录中的文件**拷贝到一个目录中

- ✓ 如果没有命令行参数则通过getcwd获取当前工作目录
- ✓ 如果包含一个命令行参数则通过该参数传递需要遍历的目录
- ✓ 如果有超过一个命令行参数则出错
- ✓ 拷贝文件及子目录下的文件

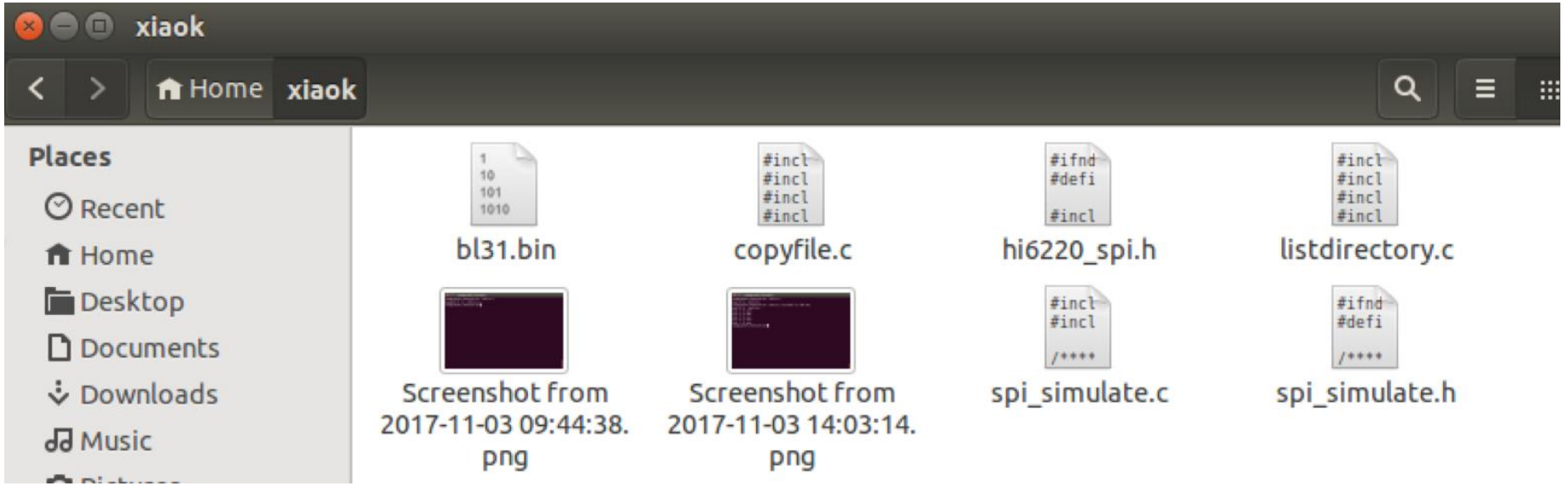
实验五 | 源文件目录结构



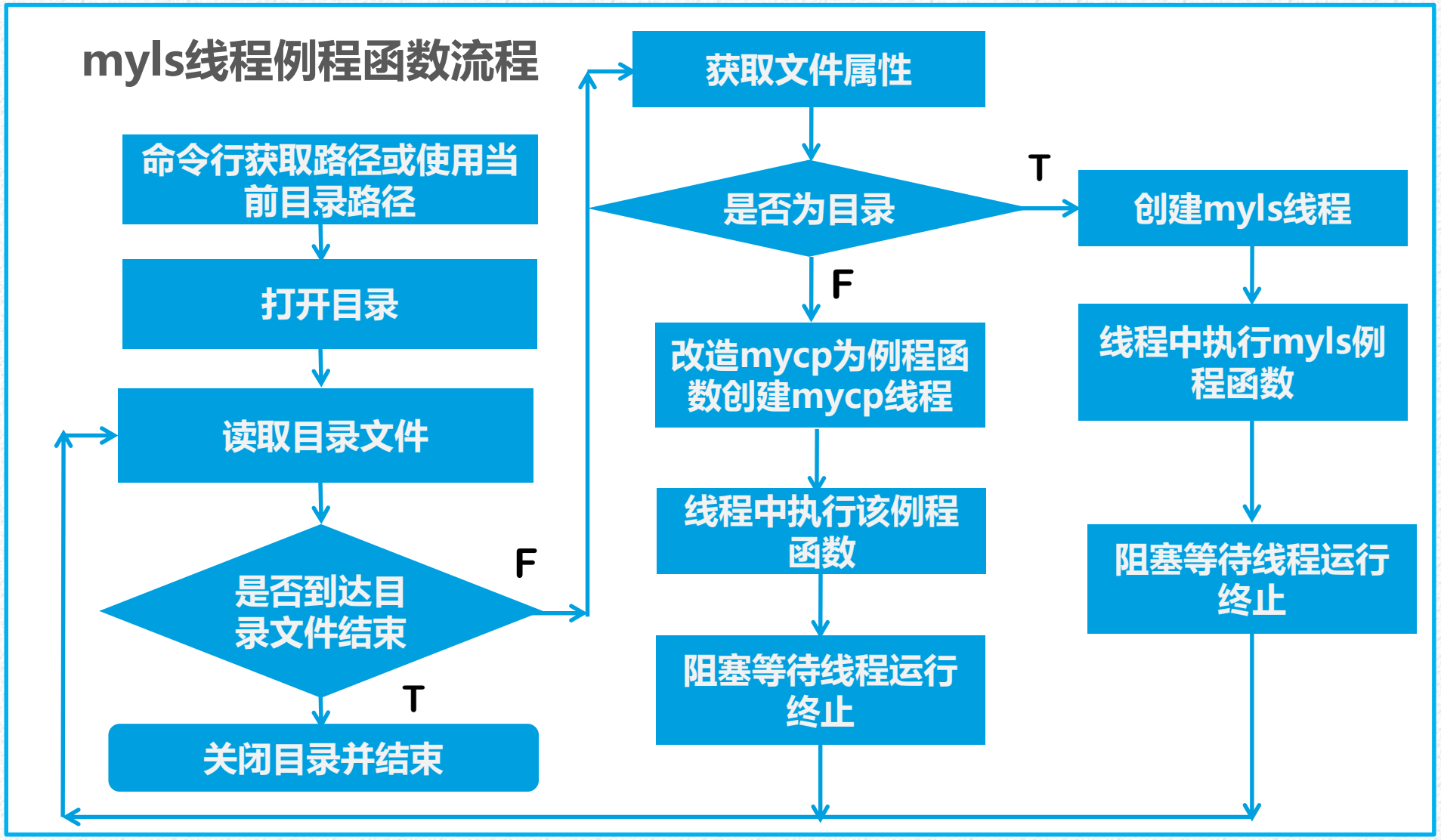
源目录结构如下所示：



实验五｜实现对所有非目录文件拷贝



实验五 | 扩展内容程序流程



实验五 | 线程控制实验内容扩展

扩展二：

✓myls与mycp线程并发执行（线程不等待所创建线程运行终止）遍历同时则创建多个mycp线程拷贝文件

✓增加源文件目录结构，观察文件较多的情况下，是否能提升程序运行效率

实验五 | 线程文件编译

多个源文件编译为一个可执行文件：

✓gcc file1.c file2.c -o objfile.o

外部库的链接

✓线程控制API实现在线程库中lpthread

✓gcc file1.c file2.c –o objfile.o -lpthread

电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

感谢观看

主讲老师：杨珊