

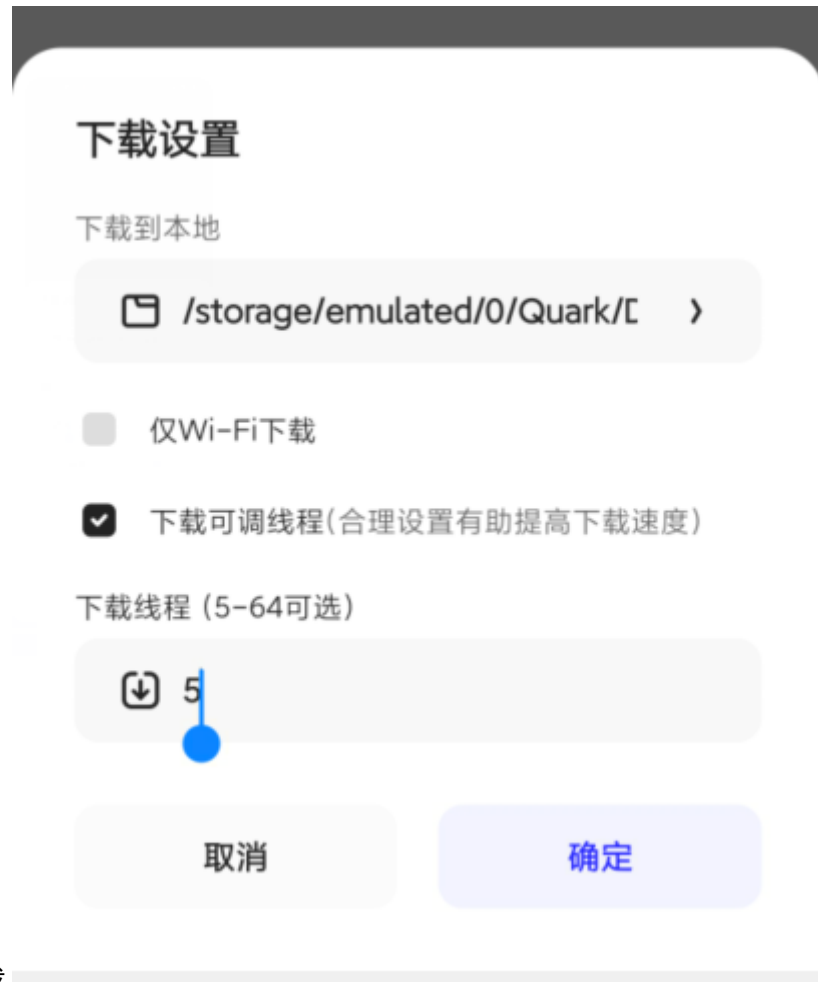
## Task 1

Java中实现多线程的方法有哪些？

- 继承Thread类，重写run方法。
- 实现Runnable接口，再`Thread xiancheng = new Thread(实现接口的类)`

举例说明身边用到了多线程的3个例子（来自廖老的question）

(身边的多线程？不一定指计算机方面吧 = =)



1. 下资源时使用多线程下载

2. 高中做作业的时候一些人做选择题，一些人做填空题，最后合起来提高了效率

3. 点开一个软件，一般是由很多线程一起运行的。

任务管理器

文件(E) 选项(O) 查看(V)

进程 性能 应用历史记录 启动 用户 详细信息 服务

名称	状态	4% CPU	56% 内存	2% 磁盘	0% 网络
Steam Client WebHelper		0%	0.9 MB	0 MB/秒	0 Mbps
Steam Client WebHelper		0%	16.9 MB	0 MB/秒	0 Mbps
Visual Studio Code (14)		0.1%	594.8 MB	0 MB/秒	0 Mbps
2021091201012-李子诺-...		0%	41.5 MB	0 MB/秒	0 Mbps
C/C++ Extension for Visual ...		0%	112.7 MB	0 MB/秒	0 Mbps
CodeHelper (32 位)		0%	2.6 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	89.6 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	23.8 MB	0 MB/秒	0 Mbps
Visual Studio Code		0.1%	154.3 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	11.4 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	12.5 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	118.3 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	14.8 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	3.0 MB	0 MB/秒	0 Mbps
Visual Studio Code		0%	1.1 MB	0 MB/秒	0 Mbps
Visual Studio Code Setup (...)		0%	3.7 MB	0 MB/秒	0 Mbps
控制台窗口主进程		0%	5.7 MB	0 MB/秒	0 Mbps

简略信息(D)

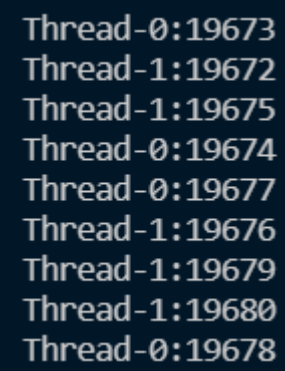
结束任务(E)

线程池的优势有哪些？

增加了cpu的利用效率，减少了线程的响应时间。

Task 2

```
public class App {
    public static void main(String[] args) throws Exception {
        Count ct = new Count();
        Thread t1 = new Thread(ct);
        Thread t2 = new Thread(ct);
        t1.start();
        t2.start();
    }
}
```



```
Thread-0:19673
Thread-1:19672
Thread-1:19675
Thread-0:19674
Thread-0:19677
Thread-1:19676
Thread-1:19679
Thread-1:19680
Thread-0:19678
```

运行截图

可以看到在中间明显乱了

## 发生问题的原因

```
for(int i = 0; i < 1000; i++){
    num++;
    System.out.println(Thread.currentThread().getName() + ":" + num);
}
```

假如thread\_1在执行完成num++后, thread\_2获得cpu执行权, thread\_1被阻塞, 延迟进行输出语句。

解决方法:

**加同步锁**锁死操作共享数据的代码块

循环里面改成下面的代码

```
synchronized(obj) {
    num++;
    System.out.println(Thread.currentThread().getName() + ":" + num);
}
```

```
Thread-1:1532
Thread-1:1533
Thread-1:1534
Thread-0:1535
Thread-0:1536
Thread-0:1537
Thread-0:1538
Thread-1:1539
Thread-1:1540
Thread-1:1541
Thread-1:1542
Thread-1:1543
Thread-1:1544
Thread-1:1545
Thread-1:1546
Thread-1:1547
Thread-1:1548
Thread-1:1549
Thread-1:1550
Thread-0:1551
Thread-0:1552
Thread-0:1553
Thread-0:1554
```

再次进行运行，就按照顺序数数了  
也可以进行**方法同步**

```
public void run() {
    for (int i = 0; i < 1000; i++) {
        cnt();
    }
}
private synchronized void cnt() {
    num++;
    System.out.println(Thread.currentThread().getName() + ":" + num);
}
```

1. 用`thread_1.setPriority(10)`修改优先级，优先级在 $1 - 10$ 之间. 优先级只是增加概率，并不能严格按照优先级执行

## Task 3

思路





不难看出这个任务对Java的File类和IO流要点理解。

## 实现

1. 可以通过字节输入输出流来复制文件
2. 遍历并嵌套函数，实现多级文件夹复制。

## 脑测一下困难

1. IO流与File类的运用
2. 多线程同步有点难写,如何给不同线程分配文件?。
3. 一堆bug

## 实现

- 通过字节流copy
- 通过字节大小来计算进度
- 遇到文件，加一个copy进程到池中，遇到文件夹递归
- 本来还想写个计时器的，但是没精力调了

```
PS D:\Code\微光\T4pro\src> cd "d:\Code\微光\T4pro\src\" ; if ($?) { javac App.java } ; if ($?) { java App }
Input A Folder PATH
D:\Code\Java
Input A Folder PATH
E:\diuwashfdiuoshaefiuwsaefiuheas
Thread-18 Copying ThreadTest.java
Thread-20 Copying Ticket.java
Thread-9 Copying P1001.class
Thread-17 Copying ThreadTest.class
Thread-1 Copying 2.png
Thread-6 Copying Test.class
Thread-19 Copying Ticket.class
Thread-10 Copying SellTicket.class
Thread-12 Copying Test3.class
```

## 具体代码

放在github上了，因为代码不够健壮，所以测试前请一定阅读README

[github链接](#)

使用clone <https://github.com/Malelk/FolderCopyTool.git> 下载

## 一些bug

- 字节计数器加不到100，大概是因为int类型转化的原因？
- 我发现在调用work方法后，无法输出东西，没找到原因。
- 没给cnt加同步，所以进度出现得并不均匀。甚至还有字节显示往后跳的情况。但是我怕加了锁多进程降低效率。。。所以没改了。