Lab3实验报告

PartA

汉明距离计算函数

两个等长二进制字符串之间的汉明距离是两个字符串对应位置的不同字符的个数

```
public int hammingDistance(int x, int y) {
    //获取x、y不同的位，如果不同就是1
    int hamming = x ^ y;
    int cnt = 0;
    while(hamming > 0){
        hamming = hamming & (hamming - 1);//将最低位的1置0
        cnt++;
    }
    return cnt;
}
```

计算机系统的可靠性

计算机系统可靠性指在规定条件下和给定时间内计算机系统正确运行(计算)的概率。而容错技术是指在一定程度上容忍故障的技术，主要靠冗余设计来实现，以增加资源的办法换取可靠性，TCP协议就采取了容错技术。为了保证可靠性传输，TCP协议通过检验和（利用在数据中外加的一部分信息位来检测或纠正信息在运算或传输中的错误）、确认应答机制（ACK）、超时重传机制等机制来进行差错检测及纠正。

PartB

设计决策

单个事务：

在开始进程后，增加“start” log记录。随后在每次write之前调用log记录写的行数，在进程完成所有write后增加“commit” log记录。在下一次重新运行程序update之前，运行recover，若记录最后一行为commit则说明上次事务执行成功，否则找到记录中最后一行start，恢复上次事务之前的状态。

任何一个事务要修改数据的时候，先将修改的数据写入另一个文件，提交时再写回原文件，从而保证写入失败的话原数据没变。

多个事务：

在提交之前检查修改的原内容是否一致，如果有冲突则不commit。

在每个进程对文件进行修改之前加锁，防止其它进程同时修改。

代码实现

单个事务：

```
private void recover() {
    try {
        File file =new File( pathname: "src/lab3/log.txt");
        if(!file.exists()) {
            file.createNewFile();
        }
        BufferedReader br = new BufferedReader(new FileReader(file));
        String temp = "";
        String start_text = "";
        String final_text = "";
        int num = 0;
        while ( (temp = br.readLine()) != null) {
            num++;
            if (num == 1){
                start_text = temp;
            }else {
                final_text = temp;
            }
        }
        if (final_text.equals("commit")){
            System.out.println("上次修改成功");
        }else{
            String[] a = start_text.split( regex: " ");
            System.out.println("上次将数据修改为"+a[a.length-1]+"失败，恢复初始数据");
        }
```

每次recover前判断log内是否已commit，若commit了则说明上次修改成功

```
public synchronized void update(char ch) {
    recover();

    log( text: "start update "+ch);
    copy( srcPath: "src/lab3/data.txt", destPath: "src/lab3/temp.txt");

    for (int i=1;i<=10;i++){
        log( text: "line "+i);
        write(i, ch);
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    //将temp.txt的内容替代data.txt
    copy( srcPath: "src/lab3/temp.txt",  destPath: "src/lab3/data.txt");
    log( text: "commit");
}
```

若运行到一半终止了，此时还没有将temp文件写入data文件，因此data文件不会改变

多个事务：

在commit之前添加一个判断，看文件内容是否还未变

```
//将temp.txt的内容替代data.txt,若原文件内容不符，则不commit
String char2 = readOriginalFile();
if (!char2.equals(original_char)){
    System.out.println("数据冲突，放弃commit");
}else {
    copy( srcPath: "src/lab3/temp.txt",  destPath: "src/lab3/data.txt");
    log( text: "commit");
}
```

给进程加锁

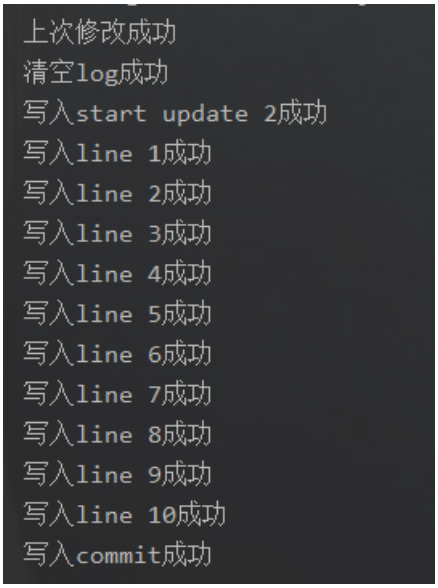
```
int threadNumber = 10;

for (int i = 0;i<threadNumber;i++){
    int finalI = i;
    new Thread(()->{
        lock.lock();
        MyAtomicity atomicity = new MyAtomicity();
        atomicity.update(Integer.toString(finalI).charAt(0));
        lock.unlock();
    }).start();
}
```

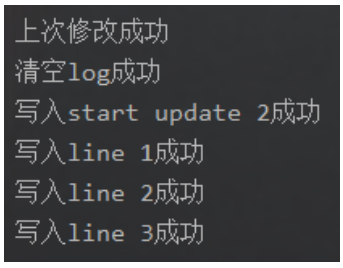
测试截图

单个事务：

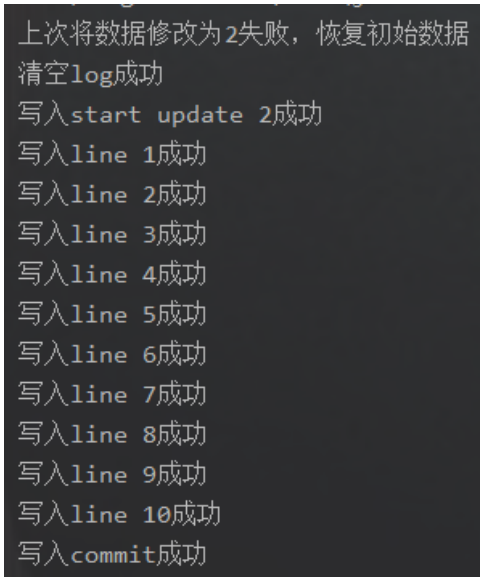
修改成功



运行到一半退出



再次运行



多个事务：

```
当前线程id为14
上次将数据修改为4失败，恢复初始数据
清空log成功
start update 0
commit 成功
当前线程id为18
上次修改成功
清空log成功
start update 4
commit 成功
当前线程id为21
上次修改成功
清空log成功
start update 7
commit 成功
当前线程id为17
上次修改成功
清空log成功
start update 3
commit 成功
当前线程id为19
上次修改成功
清空log成功
start update 5
commit 成功
```