**《互联网编程》**

**作业报告**

**作业名称： 作业2 TCP服务器日志快速并发处理服务程序**

**授课教师： 毛斐巧**

**报告人： 吴嘉楷 学号： 2022150168 班级： 国际班**

**报告提交时间： 2024年4月17日**

**成 绩：**

**1.作业内容与要求**：

思考指出实验2中你设计实现的线程池TCP服务器日志程序有何设计缺点？若让你编程处理服务器上的日志文件，该如何设计程序？需考虑这些问题：日志记录项的快速写入，快速读出分析处理，成千上万、甚至百万条日志记录文件的打开占用大量内存问题。试为你实验2中编写的TCP服务器，写一个能解决上述问题的日志处理服务程序，编程实现日志信息的格式化定向输出存储时，请使用log4j开源插件。请简述你的设计思路，给出相应的完整程序源码和运行结果。

====================================================================

**2.解答报告正文**

**缺点：**

1. 写入日志时的性能问题：如果每个日志记录项都需要单独打开文件进行写入，会造成频繁的磁盘 I/O 操作，影响性能。
2. 日志文件的读取和分析处理可能不高效：如果日志文件过大，加载整个文件到内存中可能会导致内存溢出或者性能下降。
3. 日志文件的并发写入可能会出现竞争条件，导致数据丢失或者写入错误。
4. 日志文件大小：如果日志文件不断增长而没有适当的轮转（rotation）策略，最终可能会耗尽磁盘空间，并且大型文件难以管理和分析。
5. 缺乏日志级别：如果日志记录没有区分不同的级别（如DEBUG、INFO、WARN、ERROR），将难以根据日志的紧急程度和重要性进行过滤和分析。

**解决思路：**

1. 针对日志记录项的写入性能，可以将日志信息先写入到内存缓冲区，然后定期批量写入到日志文件中，减少频繁的磁盘 I/O 操作。
2. 对于日志文件的快速读取和分析处理，可以使用流式读取方式，逐行读取日志文件，避免一次性加载整个文件到内存中。
3. 针对并发安全问题，我们可以选择使用线程安全的日志库log4j，利用日志框架提供的并发写入功能，安全地处理多线程环境下的日志记录。
4. 日志轮转：定期或根据日志文件大小来轮转日志文件，避免单个文件过大。对于大量日志记录文件占用大量内存的问题，还可以采用按时间或大小切割日志文件的方式，将日志文件分割成多个小文件，降低内存压力。
5. 针对日志级别问题，log4j日志框架支持不同级别的日志记录，我们可以使用log4j设置日志级别，以便于日志管理和分析。

**设计思路**

1. 先将实验二的代码文件拷贝到项目src目录下

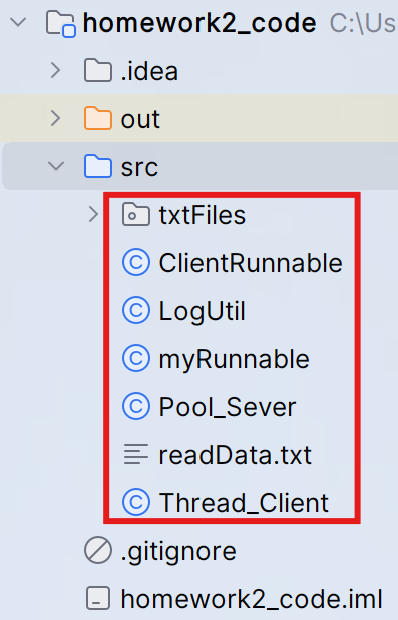


图1 项目目录结构

1. 修改源代码里面的一些文件写入读出的路径
2. 下载log4j的jar包

官网资源url：**<https://dlcdn.apache.org/logging/log4j/2.23.1/apache-log4j-2.23.1-bin.zip>**

1. 将jar包添加到项目结构下

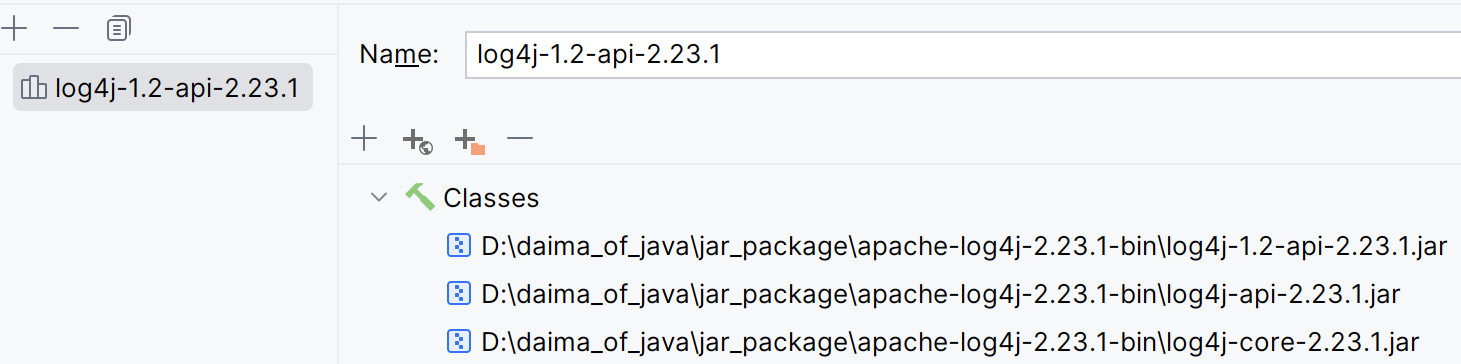


图2 添加log4j的jar包

1. 设计日志处理服务程序（在原本的LogUtil下进行修改）
2. 配置日志输出格式以及日志文件的轮转
3. 设置日志级别为 DEBUG，并将将Appender添加到Logger实例
4. 编写普通日志函数和报错日志函数

源代码：

*//服务器日志功能模块*import java.io.IOException;  
  
import org.apache.log4j.Level;  
import org.apache.log4j.Logger;  
import org.apache.log4j.PatternLayout;  
import org.apache.log4j.RollingFileAppender;  
public class LogUtil {  
 String filePath;  
 *//通过类名创建Logger对象* private static final Logger *logger* = Logger.*getLogger*(LogUtil.class);  
  
 public LogUtil(String filePath) throws IOException {  
 this.filePath = filePath;  
 *// 配置日志输出格式* PatternLayout layout = new PatternLayout("%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n");  
 *// 配置日志文件轮转* RollingFileAppender appender = new RollingFileAppender();  
 appender.setFile(this.filePath);*//日志文件路径* appender.setMaxFileSize("10MB");*//文件最大容量* appender.setMaxBackupIndex(5);*//设置轮转文件的最大数量* appender.setLayout(layout);*//日志输出格式* appender.activateOptions();*//激活配置选项  
 // 将Appender添加到Logger  
 logger*.addAppender(appender);  
 *logger*.setLevel(Level.*DEBUG*); *// 设置日志级别为 DEBUG* }  
 *//正常日志* public void logInfo(String message) {  
 *logger*.info(message);  
 }  
 *//报错日志* public void logError(String message, Throwable t) {  
 *logger*.error(message, t);  
 }  
}

1. 在线程池TCP服务器程序中修改日志处理模块，改为调用上述日志处理服务程序。

注意：创建Logger对象时 必须定义为static变量，不能对Logger对象重复做初始化操作，否则会使Appender 添加多次，从而导致日志内容的重复。

//创建Logger实例  
static LogUtil logUtil;  
static {  
 try {  
 logUtil = new LogUtil("src/logs/mySeverLog.log");  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
}

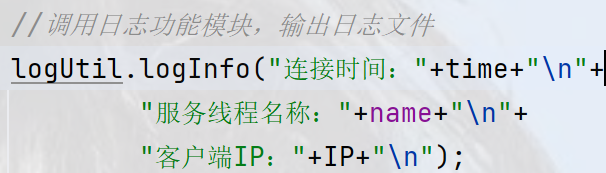


图3 输出日志

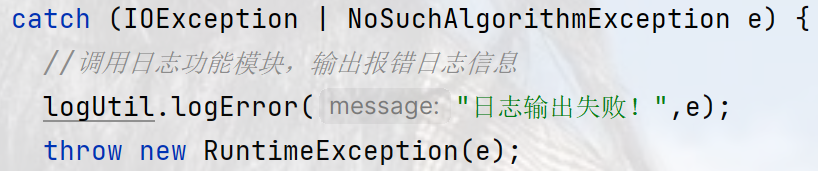


图4 记录报错信息

1. 运行结果

运行程序后，在目录文件下会产生一个日志文件：

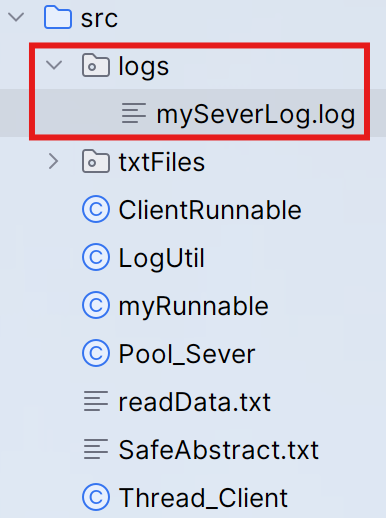


图5 最终目录结构

日志文件中保存着log4j格式化记录的日志信息和报错信息等：

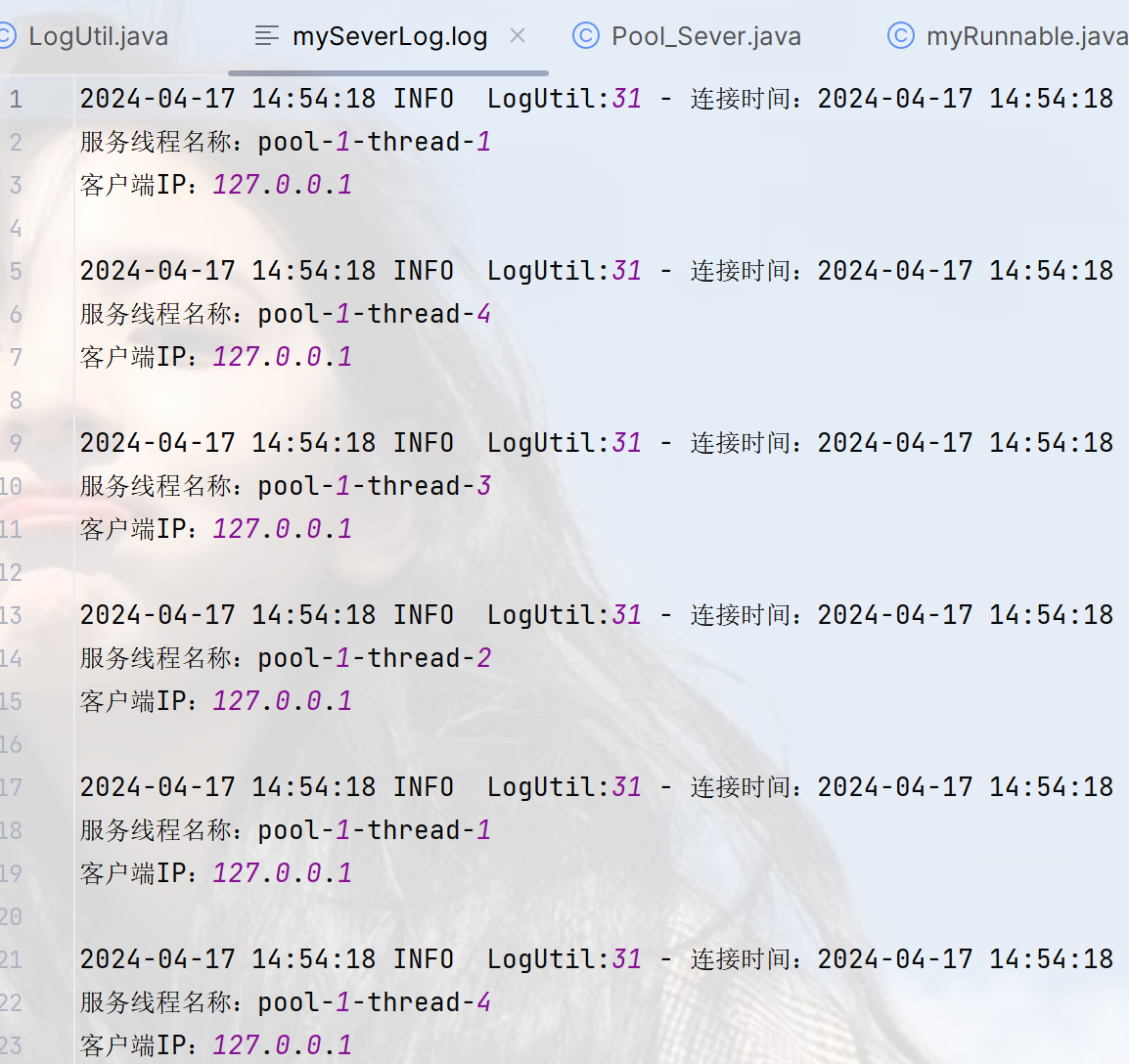


图6 日志文件内容详情展示