**“shell项目简要开发文档”**

**开发人员：**徐彩宏 2021051615080

何卓玲 2021051615068

**日期：2024年6月29日**

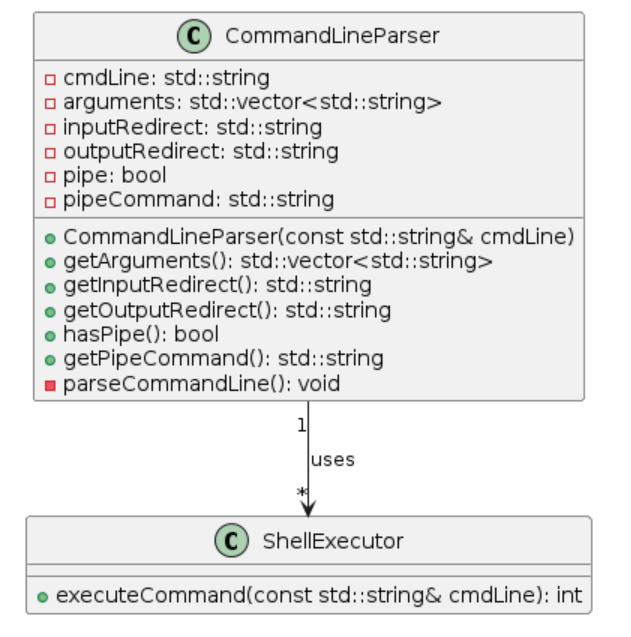
# 项目概述

Shell是一个面向C++开发的shell项目，旨在提供一个高效、灵活的命令行工具，满足程序员和系统管理员的多样需求。它不仅能够快速执行命令，并且支持脚本编程，使得复杂的操作可以以简单的代码片段实现。管道功能将大幅度提高命令之间的协作效率，而I/O重定向则为数据的流动提供了更多的控制可能性。我们相信，通过StellarShell强大的功能集合，能够极大地提升用户的工作效率和系统管理能力。

# 2. UML 模型图

在开发过程中使用的UML模型图如下：

* **类图（Class Diagram）**：



* + **CommandLineParser.h和CommandLineParser.cpp**

**目的：** 解析命令行输入，提取参数，处理输入/输出重定向，并检测管道命令。

**关键方法：**

getArguments()：返回解析后的命令参数。

getInputRedirect() 和 getOutputRedirect()：返回输入和输出文件路径，用于重定向。

hasPipe() 和 getPipeCommand()：检查是否存在管道，并获取管道符号后的命令。

**实现：**

使用 std::istringstream 进行分词。

使用标志位 (inputRedirectFlag 和 outputRedirectFlag) 管理重定向状态。

parseCommandLine() 方法基于标记检查 (<, >, |)填充成员变量。

* + **ShellExecutor.h 和 ShellExecutor.cpp**

**目的：** 使用 fork() 和 execvp() 系统调用执行由 CommandLineParser 解析的命令。

**关键方法：**

executeCommand(const std::string& cmdLine)：使用 fork() 创建子进程，并使用 execvp() 替换子进程为新程序。

**实现：**

使用 CommandLineParser 获取命令参数和重定向路径。

使用 freopen() 处理输入/输出重定向。

管理子进程的执行并使用 waitpid() 等待完成。

* + **main.cpp**

**目的：** 实现交互式Shell功能的主循环。

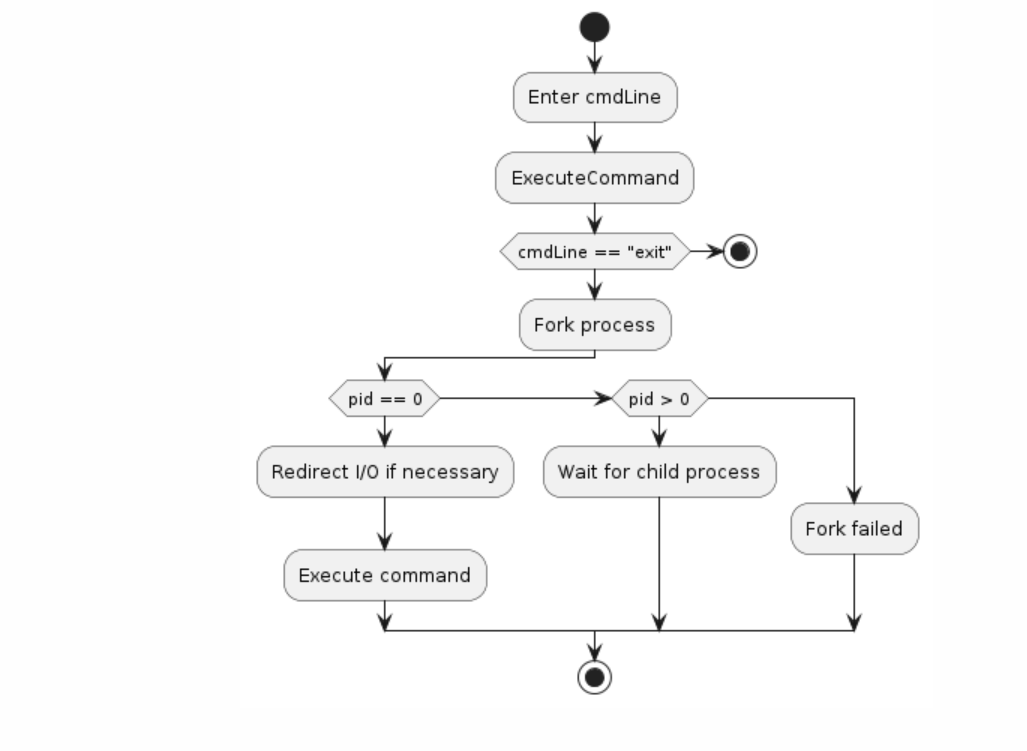
**关键特性：**

持续提示用户输入 (> )。

使用 std::getline() 从标准输入读取命令。

使用 ShellExecutor::executeCommand(cmdLine) 执行命令，直到用户输入 “exit”。

* **活动图（Activity Diagram）**：
  + 描述了命令执行过程中的流程和控制流，包括命令解析、执行和结果显示的流程。



# 3. 文本材料

项目中使用的其他文本材料包括：

* **需求文档**：详细定义了StellarShell的核心功能需求，包括命令执行、脚本编程、I/O重定向和管道功能。这些功能将确保用户能够通过简单的命令和脚本自动化任务，灵活控制输入和输出流，并通过管道连接多个命令进行数据处理。这些功能不仅满足了程序员和系统管理员的基本需求，还提升了工作效率和系统管理的灵活性。
* **设计文档**：细描述了StellarShell的整体架构，包括核心类如何组织以及它们之间的关系。文档中还涵盖了所选用的数据结构和算法，以确保Shell程序在执行命令、处理脚本、进行I/O重定向和管道操作时能够高效运行和灵活应对各种情况。这些设计决策旨在保证StellarShell在性能和可扩展性方面达到最佳表现，满足广泛的用户需求和使用场景。

# 4. 技术栈

主要使用的技术和工具包括：

* **编程语言：**C++
* **开发框架：**标准库和Boost库（可能用于处理管道和进程通信）
* **数据库：**未涉及

# 5. 开发进展

实现I/O重定向和管道功能是迈向Unix/Linux shell脚本的重要一步。这些功能不仅增加了程序的灵活性和功能性，还能让用户更轻松地管理输入输出和执行复杂的管道操作。在实现这些功能时，确保考虑到各种使用场景和边界条件，这样可以提高程序的稳定性和可靠性。

# 6. 其他说明

目前的设计和实现主要面向Unix/Linux系统，Windows系统下可能需要额外适配。