Assignment 1: Design

October 16, 2017

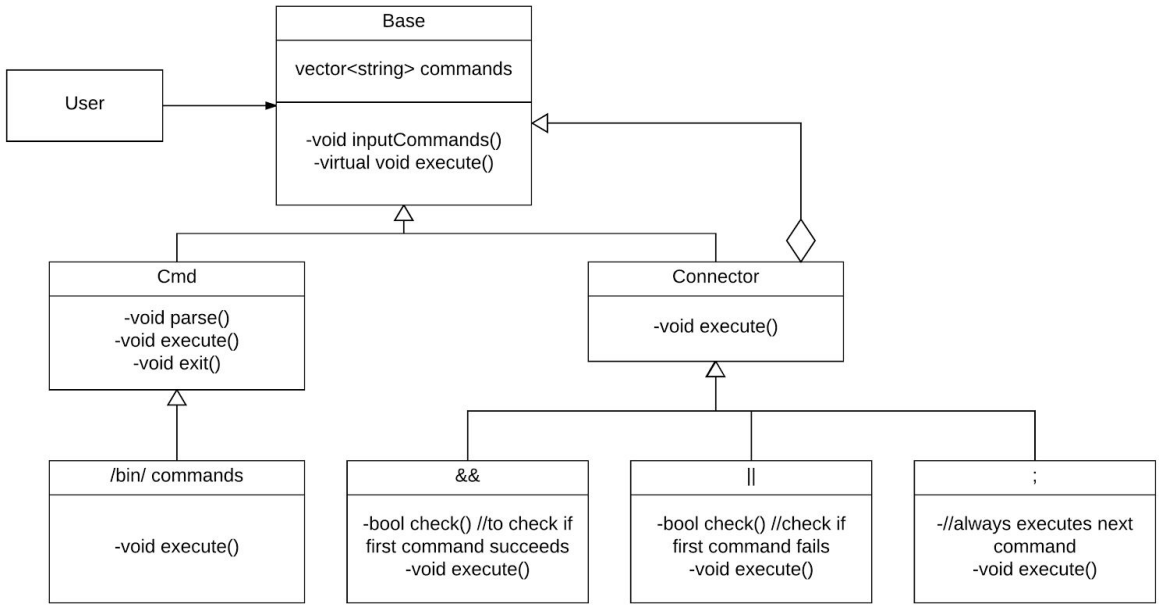
Fall 2017

Alic Lien
Daniel Li

Introduction

The project is a basic command shell, rshell, that will be utilized in a terminal. We will be writing a C++ program that will be used to execute shell commands. We will have the base class, which contains execute(), and the inherited classes “Cmd” and “Connector.” Cmd, or command, is the class that will read in user inputs for the shell (calling functions within the /bin/ or /usr/bin/). Connector contains 3 classes &&, ||, and ; (And, Or, Next), which will be used to extend commands from the Cmd class. Cmd class will also contain an exit() function which ends the programs.

Diagrams



Classes/Class Group

Base

Base class contains a virtual function, which other classes will inherit, and use.

Cmd

Command class takes in the user's input (i.e. commands) and runs it through execute() to run the shell commands.

- Will utilize C++ syscalls: `fork`, `execvp`, and `waitpid`.
- Parse will split up the string to chunks, and store in the vector.
- Exit() command is included to allow the user to exit the program (this isn't Sword Art Online, we're not trying to trap people in it).

Connector

Operator overload functions to extend the execute() functions.

&& - If a command is followed by &&, then the next command is executed only if the first one succeeds (lhs && rhs)

|| - If a command is followed by ||, then the next command is executed only if the first one fails.

; - If a command is followed by ; , then the next command is always executed

Coding Strategy

Current plan is to divide and conquer, we each work on sections that will be completed faster based on our strengths. One focus on the C++ user input implementation, and the other focuses on implementing /bin/ cmds into the C++ program.

Team requires a little more understanding of git branches, but will most likely utilize them throughout the coding process to avoid any collisions. Implementations will be tested before branches are merged.

Rough schedule so far for the specifications will go with:

- Efficient user input implementation
- Shell functions such as cd, ls, mkdir, etc
- Test out single shell command
- Work on multiple commands, utilizing connectors

Roadblocks

Schedule conflicts
As students, we cannot dedicate our entire day on to one program, on top other time conflicts and personal matters.

Deadline extensions
Deadline extensions are most likely impossible, so assignment must be completed early.

Coding errors
Errors may occur throughout the coding process, and there might be a possibility in which the errors take longer to fix than expected. Other problems that may involve coding are different coding style/syntax, but should be a minor problem.

Implementation errors
There is a possibility that our implementation might diverge from our original plan, causing either a fix in the specifications, or redoing the implementation.