Design Document

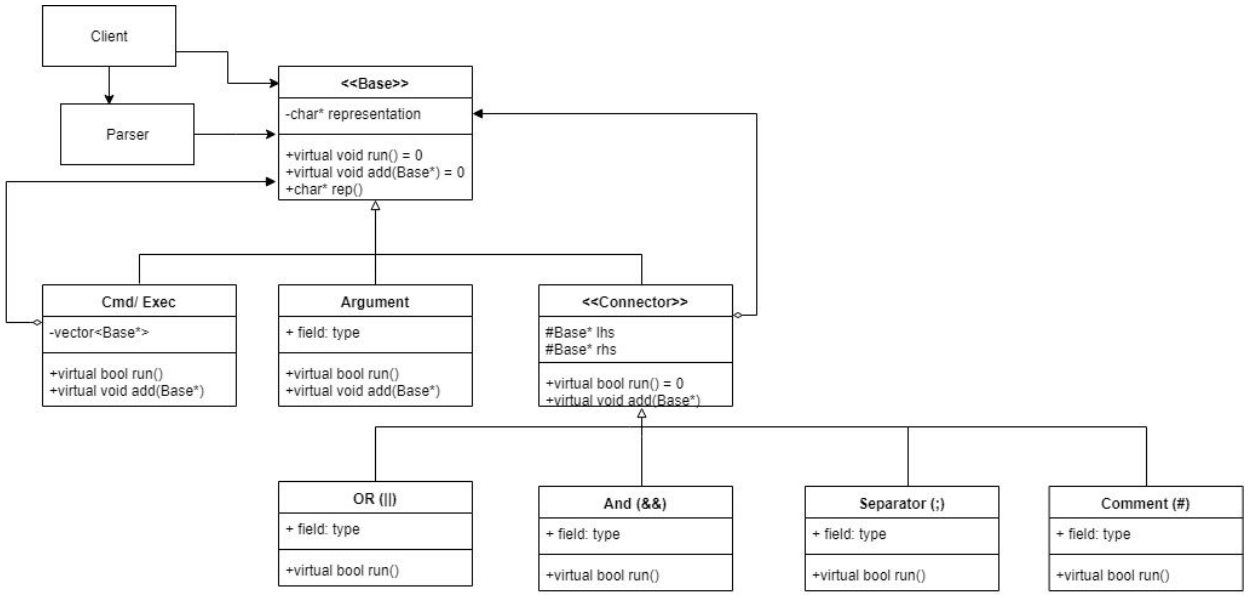
Nathan Brennan
Abdelrahim Hentabli

11/20/18
Fall 2018

Introduction:

We will be creating a command shell using C++. It will be able to read commands and execute them. We will also be allowing the use of connectors ";", "&&", and "||". We will use syscalls "fork", "execvp", and "waitpid" to execute the commands the program receives.

UML Diagram:



Classes/ Class Group:

Client:

The client will display the machine and username as well as a "$", then take a string as an input. Will pass the entire input to parser, then call the returned Base* run() command.

Parser:

The parser will first input the first "word" as a cmd and every subsequent word as an argument in that command. If it reaches an operator, it will create an operator, then add the cmd before and after it as its children. Finally, the parser will return a base* to the head of the "tree" which contains our full command.

Base:

This will just have a char array representation of its command name, argument name, or operator representation. And is an abstract parent class for all the remaining classes, with the functions do(), and add().

Argument:

Just has empty do() and add() functions that do nothing.

Cmd/Exec:

Will add in arguments into the vector with add. And will execvp in do()

Operator:

Will add into lhs than rhs in add() while do() is still virtual.

OR:

Will fork in do(), if the lhs.do() fails, it will run the rhs.do(). If lhs.do() doesn't fail however, it will not run the rhs.

AND:

Will fork in do(), will only run rhs if lhs runs.

Separator:

Will fork in do(), will run the lhs than rhs. rhs may be null, in that case it will only run lhs.

Comment:

Will fork in do(), will run the lhs, and skip the rhs.

Coding Strategy:

One member will create a parser, while the other creates the base class and cmd/Exec class. This will insure that we have a framework to test using googletest. Then we each work on either Argument or Operator. Finally, we will each create 2 out of the 4 operators, this will insure that we both know how to work with fork() and waitpid() for fixing any issues we may have find in google testing our project.

Roadblocks:

One of the first roadblocks that we foresee is the usage of fork(), waitpid(), and execvp() in our Operator and Execute classes. We have already looked through tutorials and shell documentation, and haven't completely understood the usage of it. We plan to overcome this through trial and error while using the functions in our program.

Another roadblock we may encounter is with the Argument class, we may need to restructure the classes because Argument has to implement the do() function as well as the add(Base*) function, but it does not do or add anything and is just a character array.