Assignment 1: Design

October 20, 2017

Fall 2017

Daniel Tan, Christian Grayson

Introduction

Our design is one that is object oriented. With our design being object oriented, you can expect all object oriented features such as inheritance, polymorphism, abstract classes, and so forth to be prevalent in our design.

Classes/Class Group

We have two classes in our design. A base class for inheritance and the class that is derived from the base class. Our base class is simple and straight forward. It is an abstract class with two functions that are abstract as well, requiring the derived class to do its implementation. Close to all of the functionality of our project will consist within the derived class. The derived class will have a char array for user input and a char pointer array to store the users input as a command list. The first function of the class will parse the users input into a list of commands that can be executed. It does this by taking both of the classes attributes as parameters. It will loop through the pointer array making tokens after each semicolon and place each token in the double pointer array. The second and last function will pick up where the first function left off and take the newly created list as a parameter. It forks a child class and executes each token in

the child class by the execvp function. The parent class is also put to wait with the wait function until the child class has finished its operation.

Coding Strategy

In this assignment, we will split the assignment into the functions and objects that parse the char arrays into tokens, and other functions and objects that take in char array and call the appropriate functions mapped to the char arrays. Throughout the iterative design of the code, we will review each other's code to check for readability, share different ideas and methodologies for optimizing and organizing code, and respond to changes in plans when necessary. We will create a protocol, agree on a set of functions which will be called, and create a common interface in which the two different parts can be integrated.

Roadblocks

Some of the roadblocks that we face include learning how to call the shell commands from C/C++ code, parsing the complicated char array input correctly, organizing code to make it more modular, and managing messages being sent between different objects. When inheriting from base class to create derived class, we need to be careful that when changing base classes, the derived classes that depend on the base classes do not end up with too many problems.

When these roadblocks, we will read up and learn more about how to call the shell commands from C/C++ code, look up algorithms implemented by others to help with char array parsing and apply what we learnt from those algorithms to our own project, and when organizing code, we can look at how other professionals organize their code and implement their design. *Design*

Patterns: Elements of Reusable Object-Oriented Software will be especially helpful in determining good/bad practices, and help implement software in a more agile way.