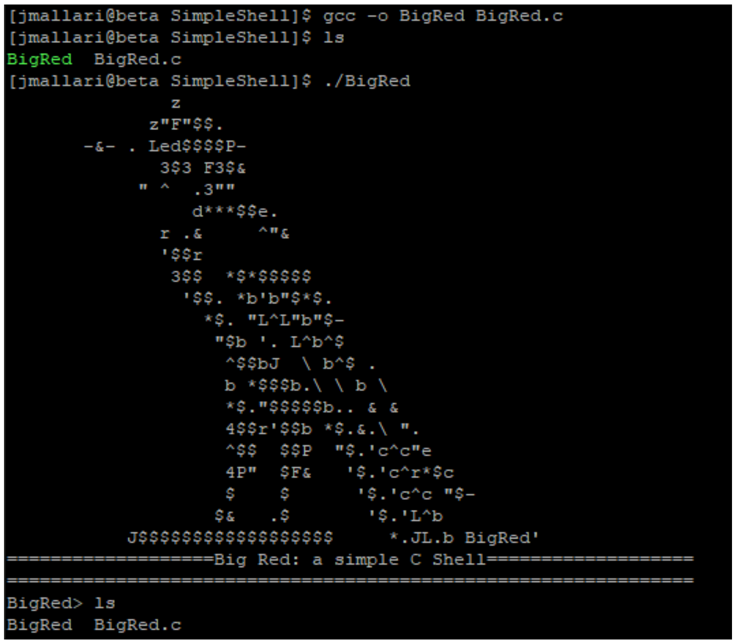
Group Members:
Joshua Mallari
Jennifer Tsan
Nicholas Wade

Shell Project Report

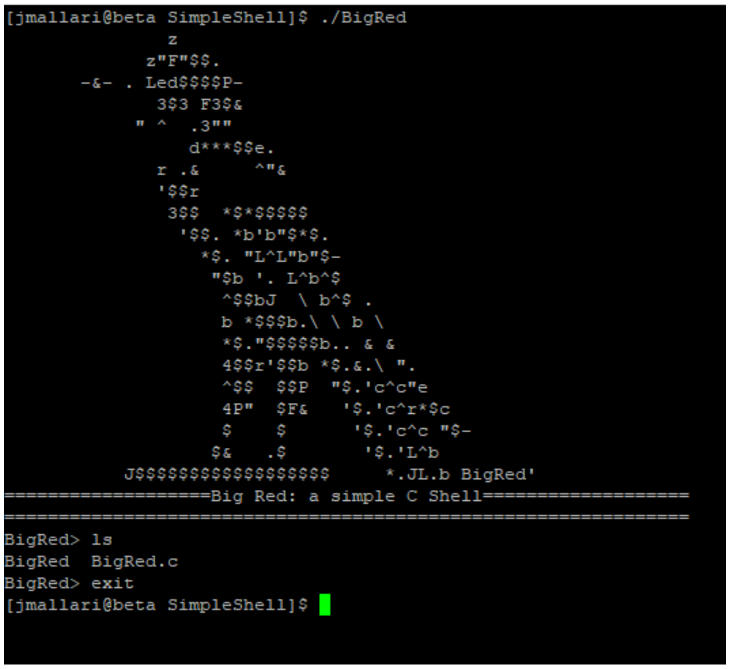
Introduction:
A shell is a command line interpreter. The shell is a text-based program that the user can use to send commands to the OS, in which the OS will respond directly with the interactive user. It is the outermost layer of the operating system which allows the user to control processes and files, with the ability to also start and control other programs.

Technical Content and Diagrams:
BigRed is based in the C language, which allows for a lower-level implementation that is closer to the OS. The main body of the shell runs within a while-loop within the main() function, with auxiliary functions defined around the main() function. To execute functions, these commands are either built-in or accessible from the OS via the execv function. When running built-ins, the main() function runs it as a part of the parent function, while the OS commands are executed via use of fork() and a child process. Parsing is done by dedicated parse functions, which allow for the program to better read input commands.

How to Run the BigRed Shell:
To compile BigRed Shell, type in *gcc -o BigRed BigRedShell.c* into the command line.
To execute BigRed Shell, type in *./BigRed*
Type *ls* into the command line to have the terminal display all files in the current directory



```
[jmallari@beta SimpleShell]$ gcc -o BigRed BigRed.c
[jmallari@beta SimpleShell]$ ls
BigRed  BigRed.c
[jmallari@beta SimpleShell]$ ./BigRed
               z
             z"F"$$.
       -&- . Led$$$$P-
              3$3 F3$&
            " ^  .3""
                 d***$$e.
              r .&     ^"&
              '$$r
               3$$  *$*$$$$$
                '$$. *b'b"$*$.
                  *$. "L^L"b"$-
                   "$b '. L^b^$
                    ^$$bJ  \ b^$ .
                    b *$$$b.\ \ b \
                    *$."$$$$$b.. & &
                    4$$r'$$b *$.&.\ ".
                    ^$$  $$P  "$.'c^c"e
                    4P"  $F&   '$.'c^r*$c
                    $    $      '$.'c^c "$-
                   $&   .$       '$.'L^b
           J$$$$$$$$$$$$$$$$$$     *.JL.b BigRed'
===================Big Red: a simple C Shell===================
===============================================================
BigRed> ls
BigRed  BigRed.c
```



```
[jmallari@beta SimpleShell]$ ./BigRed
               z
             z"F"$$.
       -&- . Led$$$$P-
              3$3 F3$&
            " ^  .3""
                 d***$$e.
              r .&     ^"&
              '$$r
               3$$  *$*$$$$$
                '$$. *b'b"$*$.
                  *$. "L^L"b"$-
                   "$b '. L^b^$
                    ^$$bJ  \ b^$ .
                    b *$$$b.\ \ b \
                    *$."$$$$$b.. & &
                    4$$r'$$b *$.&.\ ".
                    ^$$  $$P  "$.'c^c"e
                    4P"  $F&   '$.'c^r*$c
                    $    $      '$.'c^c "$-
                   $&   .$       '$.'L^b
           J$$$$$$$$$$$$$$$$$$     *.JL.b BigRed'
===================Big Red: a simple C Shell===================
===============================================================
BigRed> ls
BigRed  BigRed.c
BigRed> exit
[jmallari@beta SimpleShell]$
```

Type *exit* into the command line to exit the BigRed shell program.
Type *cd [name of directory that you want to change to]* into the command line to change the current directory.
Type *help* into the command line to display the list of commands that BigRed Shell can execute.

Issues Encountered:
Implementing and running the built-in function execv( ) was a recurring issue throughout the project. execv( ) takes in two arguments. The first argument is the path as a string to the program that is needing to be executed. The second argument is an array of strings that will be used as arguments of the program that the user wants to execute. The issue was getting the path of the program.

Conclusion:
Despite a minor struggle with implementing execv(), the silver lining from working through its issues is that we now have a deeper insight into the exec family of functions. Furthermore, the main takeaway from this shell project was a basic understanding of how a UNIX shell functions and a foundational-level comprehension of its basic design principles.