MaSH

Mathematical Shell

GROUP 16
SLOT 11
STAGE 2 SUBMISSION
SRS (Software Requirement Specifications)

Project – Unix Shell in C++ with some mathematical functions
Project Name – MaSH

Introduction to Shell :

Shell is an command language interpreter or a command line interpreter that executes commands, which is read from the standard input device such as keyboard or from a file. Shell is not part of system kernel, but uses the system kernel to execute programs, create files etc. Simply put, the shell is a program that takes your commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix computer. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.

On most linux system, bash (Bourne again Shell) and the C shell (csh) are the most influential shell. These shells have both been used as the coding base and model for many derivative and work-alike shells with extended feature sets. The korn shell (ksh) is also used sometimes. Some other shells are tcsh, zsh.

Objective :

Our objective is to create a Unix shell and emulates its basic tasks. Generally we are trying to implement basic I/O redirection among files and commands, basic piping between commands and files and running basic *nix commands.

While our shell will behave like a normal Unix shell, it will also have some other features. As students of the mathematics department, we will try to implement basic mathematical operation in our shell.

List of our objectives :

- Run basic *nix commands
- Emulate basic shell behaviour
- Basic mathematical commands including integration and differentiation

We have till now decided upon these three broad objectives and have started towards fullfilling them.

<u>Functional Requirement</u> :

Different software packages and non-standard header files used till now :-

- "fparser" library for parsing mathematical functions in shell terminal.
- GNU readline library for reading user input at terminal. It was chosen because it makes tab-completion and command history implementing easy.
- GNU GCC/G++ compiler package
- gedit, code::blocks and sublime text editors for editing codes.
- Ubuntu 14.04 Operating system, Windows 7 Operating System

<u>Sample Code for a Shell (Unix) (Main Function)</u> :

```
int main (int argc, char **argv)
{
    while (true)
    {
        int childPid;
        char * cmdInput;

        printPrompt();

        cmdInput= readline(); //or GNU
        cmd = parseCommand(cmdInput);

        add_history (cmdInput) ;

        if ( checkBuiltIn (cmd))
            executeBuiltIn (cmd);
        else
        {
            childPid = fork(); //Unix specific function

            if (childPid == 0)
                executeCommand (cmd)
                else
                {
                    if (backgroundJob(cmd))
                        record_Job () ;
                    else
                        waitpid (childPid);
                }
            }
        }
}
```

<u>Required features</u> :

Some of the required features that are required for the Shell :-

- The prompt you print should indicate the current working directory. For example:
  `The directory:~/usr/foo/bar/baz #`

- Maintain a history of commands previously issued. So that the user can access previously used commands using a single key.

- A built-in command is one for which no new process is created but instead the functionality is build directly into the shell itself. We are supporting the following built-in commands : `cd, history, version, exit and math`.

- The prompt should print system name and system user name. For example
  `user@system:~/usr/foo/bar/baz #`

- Mathematical functions are to be included, some of the required mathematical functions are

1. Integrating a function given as input by the user. User will also supply lower limit and the upper limit for integrating. If the function cannot be cannot be integrated then the program must print out error message and also

a reason for the error.

2. Differentiating a given function. The program should differentiate at a given point, which is inputted by the user. Now it must also show if the function is not differentiable at a given point, printing an error message and a reason for the error.
3. Greatest common divisor of 'n' number should be implemented. It should first ask the user to input the number of integers that is to be given. Then it should print out as output the greatest common divisor of the given integers.
4. It should contain a program that prints out the value of a given function at a given point. If the function is not defined at that point then it must print an error message.
5. Given any function and an approximate root, it must contain a program that gives us the real root of the function based on the approximate root given by the user.

- Shell should not exit if user presses CTRl + C or CTRL + Z and should handle these signals properly.

- Tab completion and command prompt editing. The GNU readline library is being used for this.

- Up and down errors to scroll through the history list. The GNU readline library is being used for this.

- Basic Piping between two processes must be present. Codes like `ps | grep bash` should work.

Optional Requirement

- One can implement I/O redirection.
- Similarly chaning of pipes and redirection may be implemented
- `History` can be implemented as a command and `history -c` should delete all commands previously stored in history.
- Scripting can be implemented using parser, lexical analyzer etc.
- Other useful statistical and mathematical commands may be added, like adding, multiplying and finding determinant of matrices.