

Shell in C

Design and Implementation of linux shell in C

2018

# PES University,

Bangalore

# 

# Introduction & Background

Shell is a UNIX term for the interactive user interface with an operating system. The shell is the layer of programming that understands and executes the commands a user enters. The shell is also called a command line interpreter. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by

creating text scripts of one or more such commands. This project was divided in three phases with the following goals

1. Basic Shell Implementation
   1. Start-up
   2. Read input, parse and execute commands (execvp)
   3. Exit shell
2. Implementation of features
   1. Piping and I/O Redirection
   2. History
   3. Aliasing
   4. Editor
3. Implementation of custom functions

# 

# Implementation details

## Basic Shell Implementation

* The input was taken from stdin and parsed to split as per the needs.
* Display of the prompt.
* Setting the environment for the shell to run.
* Fork a child process.
* Used exec function to run the process in the child.
* Parent waits for the child to execute.

Most of the commands executed were system commands using exec function.

But the command cd was implemented using chdir and kept track of the path using custom functions.

The shell is terminated by the command ‘exit’.

The exec command uses the path defined in the environment variable ‘PATH’.

## Implementation of Features

1. Piping and redirection-

This was done with the use of pipes and dup system call. The output of one command will be put on one end of the pipe and the command after the pipe reads from the other end of the pipe taking it as the input.

Using dup system call, close the file descriptors which makes the output to be written to stdout.

Redirection uses idea as above and concatenates the text with what was already present in the file if already present else the file is created.

2. History-

Every time there is an input, it is put in the queue where each node in it is a structure keeping info about the pid, timestamp and the command. The queue has a size of 25 which can be changed by changing the definition in the header file.

3. Editor (Shell and Text)-

The Text editor is invoked by using m3p as that is its name and the name of the file is passed. The file is created if it is not already present and concatenates to the existing file if it is already present. This is implemented by using a buffer and reading and writing to files.

The Shell editor provides to give the command as input across multiple lines and perform all the actions as a single job. This is done by using backslash (‘\’) character to take more input even after end of line is reached.

4. Aliasing-

The aliases are specified in a file. Aliases can be dynamically added.

Whenever the aliases are to be used, the file is executing using the source command. Source takes the input and checks the file and maps it to the actual decoded command and executes.

Background processes are generated by making the parent not wait for the child process and the child process being the one to be run in the background.

## Implementation of Custom functions

1. Sgown -

This command takes in a keyword as an argument and returns all the occurences of the keyword in the present directory, subdirectories and files along with the line numbers.

2. Enhancing functionality of ps -

* Using the -z option, lists all the zombie processes
* Using the -cpu option, lists the top 10 cpu consuming processes.
* Using -memry option, lists the top 10 memory consuming processes.
* Using -ph option, gives the process hierarchy for the current process.

3. Enhancing the functionality of ls-

* Use the -z option to list all the files of size zero
* Use the -u option to list all the files in increasing order of their uid

# 

# References

1. Stack Overflow.com
2. Tutorialspoint.com
3. Lab2 - Implementing a shell
4. Writing your own shell
5. ibm.com

# 