EMERTXE TRAINING PROJECT DOCUMENTATION FRAMEWORK

REQUIREMENTS & DESIGN DOCUMENT

Module – Linux Internals

Mini Shell

Contents

1 Abstract

Implement a minimalist shell, mini-shell (msh) as part of the Linux Internal module. The objective is to understand and use the system calls w.r.t process creation, signal handling, process synchronization, exit status, text parsing etc..

2 Requirements

Provide a prompt for the user to enter commands

- Display the default prompt as msh>
- Prompt should be customizable using environmental variable PS1
  - To change the prompt user will do PS1=NEW_PROMPT
  - Make sure that you do not allow whitespaces between =,

    i.e., do not allow PS1 = NEW_PROMPT
  - In the above case, it should be treated like a normal command

Execute the command entered by the user

- User will enter a command to execute
- If it is an external command
  - Create a child process and execute the command
  - Parent should wait for the child to complete
  - Only on completion, msh prompt should be displayed
  - If user entering without a command should show the prompt again

Special Variables:

- Exit status of the last command (echo $?)
  - After executing a command the exit status should be available
  - echo $? should print the exit status of the last command executed
- PID of msh (echo $$)

  echo $$: should print msh's PID
- Shell name (echo $SHELL)

  echo $SHELL: should print msh executable path

Signal handling

Provide short cuts to send signals to running program

- Ctrl-C (Send SIGINT), On pressing Ctrl-C
  - If a programming is running in foreground, send SIGINT to the program (child process)
  - If no foreground program exists, re-display the msh prompt
- Ctrl+z (Send SIGSTP), On pressing Ctrl+z
  - The program running in foreground, should stop the program and parent will display pid of child

Built-in commands

- exit - This command will terminate the msh program
- cd - Change directory
- pwd - show the current working directory

Background Process / Job control

- Allow a program to run in background

  To run a program in background use ampersand (&) after the command. For eg: sleep 50 &
- Implement fg, bg and jobs commands
  - bg will will move a stopped process to background sleep 10 & is equivalent to sleep 10 then ctrl + z and bg.

    After this the msh prompt should be displayed indicating it is ready to accept further commands. After a bg process ends, cleanup the process using wait.

    NOTE: You may have to use SIGCHLD signal handler for this

    On termination of bg process, display its exit status. User should be able to run any number of background processes.
  - fg will bring a background process to foreground. Only fg bring last background process, or fg <pid> will bring given pid to foreground.
  - jobs will print all background process details.

Pipe functionality

- Allow multiple processes communication by using pipes.
- Create pipes and childs dynamically as per pipes passed from command-line

  Eg: ls | wc, ls -l /dev | grep tty | wc -l

3 Sample Output

```
user@emertxe] ./msh
msh] ls
commands.c  commands.h  error_messages.h  history.c  history.h  mini_shell.c  mini_shell.h  msh  prompt.c  prompt.h  sca
msh]
```

Fig 3 1: Usage

```
msh] echo $$
3036
msh] pwd
/home/user/ECEP/6-LinuxInternals/Project/MiniShell/
msh] echo $?
0
msh]
```

Fig 3 2: Shell Functions

```
msh] mkdir Test
msh] cd Test
msh] pwd
/home/user/ECEP/6-LinuxInternals/Project/MiniShell/Test
msh] cd ..
msh] pwd
/home/user/ECEP/6-LinuxInternals/Project/MiniShell
msh]
```

Fig 3 3: Basic Commands

```
msh] ^C
msh] ^C
msh] ^C
msh] cat
Hello^Z
[1]+  Stopped                 cat
msh] fg
cat
^C
msh]
```

Fig 3 4: Signal Handling

```
msh] firefox &
msh] fg
firefox
^C
msh]
```

Fig 3 5: Background and Foreground control

```
msh] ls -1
commands.c
commands.h
error_messages.h
history.c
history.h
mini_shell.c
mini_shell.h
msh
prompt.c
prompt.h
scan_codes.c
scan_codes.h
signal_handling.c
signal_handling.h
msh] ls -1 | wc -l
14
msh]
```

Fig 3 6: Pipe Function

```
msh] echo "Hello World" > file.txt
msh] cat file.txt
Hello World
msh] echo "Tata Bye bye" >> file.txt
msh] cat file.txt
Hello World
Tata Bye bye
msh]
```

Fig 3 7: Redirection

4 Artifacts

Skeleton Code

- www.emertxe.com/content/linux-internals/code/minishell_src.zip

References

- https://www.gnu.org/software/libc/manual/html_node/Implementing-a-Shell.html
- https://oskarth.com/unix01/
- http://www.tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html
- https://www.gnu.org/software/bash/manual/bash.pdf