ITMO UNIVERSITY

National Research University ITMO

Faculty of Software Engineering and Computer Systems

SYSTEM SOFTWARE FUNDAMENTALS

Lab Work #1 (9)

Introduction to Shell Scripting

Timothy Labushev

Group P3302

Saint Petersburg

2019

Assignment

This report discusses the implementation of an interactive shell script that performs a fixed set of actions: *print the current working directory*, *list files in the current working directory*, *create a new directory*, *grant and revoke write access to a directory*.

For commands that require a directory name to be provided, the user may use TAB completion and `vi` or `emacs` keybindings. This is achieved by invoking the `read` utility with an `-e` switch, which uses GNU Readline to obtain a line from standard input. It is important to note that it is not mandated by POSIX and may be absent on uncommon and outdated systems.

Error handling is performed by checking the exit code of executed commands. In case of a non-zero code, a user-friendly error message printed. The error message as produced by the failed command is appened to a log file, which can be inspected later.

Code Listing

```
log_file="$HOME/lab1_err"

function print_cwd {
  echo $PWD
}

function list_cwd {
  echo $(ls)
}

function make_dir {
  echo Enter the path to the new directory \(TAB to autocomplete\):
  read -e path
  echo Creating $path
  mkdir -p "$path" 2>>$log_file \
    || echo_stderr Unable to create the specified directory \
        -- are you sure you have sufficient permissions?
}

function make_world_writeable {
  echo Enter the path to the directory you wish to make world-writeable:
  read -e path
  echo Altering permissions for $path
  chmod +w $path 2>>$log_file \
    || echo_stderr Unable to alter permissions for the specified directory \
        -- are you sure the directory exists and you have sufficient permissions?
}

function make_read_only {
  echo Enter the path to the directory you wish to make read only:
  read -e path
  echo Altering permissions for $path
  chmod -w $path 2>>$log_file \
    || echo_stderr Unable to alter permissions for the specified directory \
        -- are you sure the directory exists and you have sufficient permissions?
}

function run {
  echo; $1; echo
}

function echo_stderr {
  echo "$@" 1>&2
}

while true; do
  echo [1] Print current working directory
  echo [2] List files in current working directory
```

```
  echo [3] Create a new directory
  echo [4] Grant every user write access to a directory
  echo [5] Revoke write access to a directory from all users
  echo [6] Quit
  if read -p "Choose the action to perform: " action; then
    case $action in
    1) run print_cwd;;
    2) run list_cwd;;
    3) run make_dir;;
    4) run make_world_writeable;;
    5) run make_read_only;;
    6) break;;
    *) echo_stderr Unknown action [$action];;
    esac
  else
    break # handle EOF
  fi
done
```

Lessons Learned

While completing the assignment, I have familiarized myself with the basics of control flow in shell scripts: chaining commands with logical OR to handle error conditions (non-zero exit code), matching on value variants (`case`), executing repeating code in a loop and terminating it based on a condition.

In addition to that, I discovered that scripts running on systems with GNU utilities may provide the user with advanced input capabilities, such as filename autocompletion and configurable editing shortcuts, with practically zero cost to the script author.

At the same time, I have encountered certain limitations of shell scripting. The assignment features independent commands, with the user picking which one to execute by entering its index. Initially, I wanted to separate the command picking logic from the actual commands, making it easier to add or remove commands in the future. It turned out that while the language has function references and associative arrays, the syntax is arcane and the support is not universal among different shells. This leads me to a conclusion that, while shell scripts are quite versatile and fast to write for an experienced programmer, they are not the best choice for a system that needs to be actively maintained and adapted to changing requirements.