Linux Shell Scripting

Creating a Local Linux Shell Scripting Enviroment (S2)

- Setting up a centos7 VM with VirtualBox and Vagrant

User and Account Creation (S3)

Getting Started with Shell Scripting: Naming, Permissions, Variables, Builtins

- Shebang(#!) → sets the interpreter for script
- See permissions with ls –l
- chmod 755 <file> → make it executable for all
    - chmod +x does the same
- Builtin: command that is executable with shell only
    - type <command> → to see if it's a builtin
    - use builtins when available
- help <command> and man <command> for help for commands
- Variables:
    - no spaces for variable assignment
    - no dashes in variables
    - all uppercase in convention
    - get variable value with $<variable> or ${<variable>}
        - single quotes prevent variable assignment

Special Variables, Pseudocode, Command Substitution, if Statement, Conditionals

- shell variables are predefined (see in manpage)
- whoami same as id –un → print username
- add command output to variable:

```
USER_NAME=$(id -un)
```

- if-then-else statement(bash specific):

```
if [[ "${UID}" -eq 0 ]]
then
    echo 'You are root'
else
    echo 'You are not root'
fi
```

    - help test → to see operators
- make sanity check for required stuff

Exit Statuses, Return Codes, String Test Conditionals, More Special Variables

- exit status:
    - number can be given to exit command
    - 0: succesfull
- **$? → special variable with exit status of last executed command**
- String comparison:
    - = → string comparison
    - == → pattern string comparison

Reading Standard Input, Creating Accounts, Username Conventions, More Quoting

- read builtin to get stdin
  - stdin(0), stdout(1), stderr(2)
- useradd to add user
  - usernames are case sensitive
  - – l → no login
  - – m → create home directory
- passwd to change password
  - –stdin → to pipe password in
    - Not in ubuntu (use chpasswd instead)
  - -e → user have to change password after first login
- su to change user

Password Generation and Shell Script Arguments (S4)

Random Data, Cryptographic Hash Functions, Text and String Manipulation

- RANDOM is bash builtin variable → generates random integer
- Checksum as password by piping data in hashfunctions
- shuf → randomize line output
- fold → make input into lines

Positional Parameters, Arguments, for Loops, Special Parameters

- parameter: variable used inside shell script
- argument: data passed into shell script → becomes value stored in parameter
    - positional parameter store arguments
    - $0 → command
    - after 0 storing the arguments
    - $# → to get number of supplied args
    - $@ to use in loops
    - $* all arguments getting merged to one argument
- for loop:
    - for NAME [in WORDS ...] ; do COMMANDS; done

The while Loop, Infinite Loops, Shifting, Sleeping

- while loop:
    - while COMMANDS; do COMMANDS; done
    -
- sleep → delay execution
- shift n→ removes n parameter from list and renames the following one

Linux Programming Conventions (S5)

Advanced Stdin, Stdout and Stderr

- redirection of stdout(1): >
    - append with >>
- redirection of stdin(0): <
- redirection of stderr(2): 2>
    - append with 2>>
- redirect stderr and stdout: &>
    - append with &>>
- pipe stderr and stdout: |&
- send output to stderr: >&2
- to discard output redirect to /dev/nullmueller_b3

Parsing Command Line Options (S6)

Case Statements

- Execute commands based on pattern matching.
    - case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;]... esac

Functions

- function:
    - function name { COMMANDS ; } or name () { COMMANDS ; }
    - Create a shell function named NAME. When invoked as a simple command, NAME runs COMMANDs in the calling shell's context. When NAME is invoked, the arguments are passed to the function as $1...$n, and the function's name is in $FUNCNAME.
    - Functions have to be defined before usage
- local <variable>: makes a variable local
- arguments given to functions can be accessed with positional parameters
- readonly <variable> → mark as readonly
- logger → to log in system log (/var/log/syslog)

Parsing Command Line Options with getopts

- Getopts is used by shell procedures to parse positional parameters as options.
    - while getopts vl:s OPTION
    - getopts optstring name [arg]
    - ${OPTARG} → to get the command argument value
- Arithmetic expansion:
    - with ((…)) :
        - NUM=$((2+3))
    - with let command
        - let NUM='2+3'
        - let NUM++
    - expr
        - NUM=$(expr 2+3)

Finding Files

- locate → using database
- find → up to date

Userdel command

- deletes user account and related files

Archives with tar

- tar command

Disabling Accounts

- chage → to change expiration date
- passwd –l → lock password(account), ssh-key login still possible

Transforming Data/ Data Processing/ Reporting (S7)

Cut and Awk

- Cut:
  - cut - remove sections from each line of files
- Awk:
  - Mightier for string operations than cut or grep

Sort and Uniq

- Sort:
  - To sort lines (see https://github.com/BennyTheSen/linux_command_line_course/blob/master/Linux%20Mastery.pdf)
- Uniq:
  - Lines have to be sorted before
  - Does the same as sort –u
  - Uniq –c → to show how often lines appared
- Wc:
  - Count lines, words and characters

Sed

- Stream editor for filtering and transforming text
- Good for replacing text

Networking Scripting & Automation of Distributed Systems (S8)

Configuring a Mini Network and Scripting for Remote Systems

- ssh-keygen → generate ssh key
- ssh-copy-id <server> → copy ssh key to a remote server