

5. Automating System Administration

IT5406 - Systems and Network Administration

Level III - Semester 5

Overview

5 Automating System Administration (4 hours) [Ref 1: Pg. (182-222)]

5.1. Scripting Philosophy

5.2. Shell Basics

5.3. Shell Scripting

5.4. Regular Expressions

5.5. Python Scripting

5.6. Revision Control with GIT [Ref 1: Pg. (235-240)]

5.1. Scripting Philosophy

- Write microscripts - keep small scripts to make the day to day work easy.
- Learn a few tools - Shell, Text editors, etc
- Automate all the things - there are not only shell scripts, but number of other applications and programming languages which can be used to automate certain tasks.
- Don't optimize prematurely
- Pick the right scripting language
- Follow best practices

5.2. Shell Basics

- Linux systems have different kinds of shell programs such as,
 - Bourne shell, sh
 - Almquist shell, ash
 - Bourne-again shell, bash
 - Korn shell, ksh
 - C shell, csh
- bash is pretty much the universal standard these days.

<https://www.gnu.org/software/bash/>



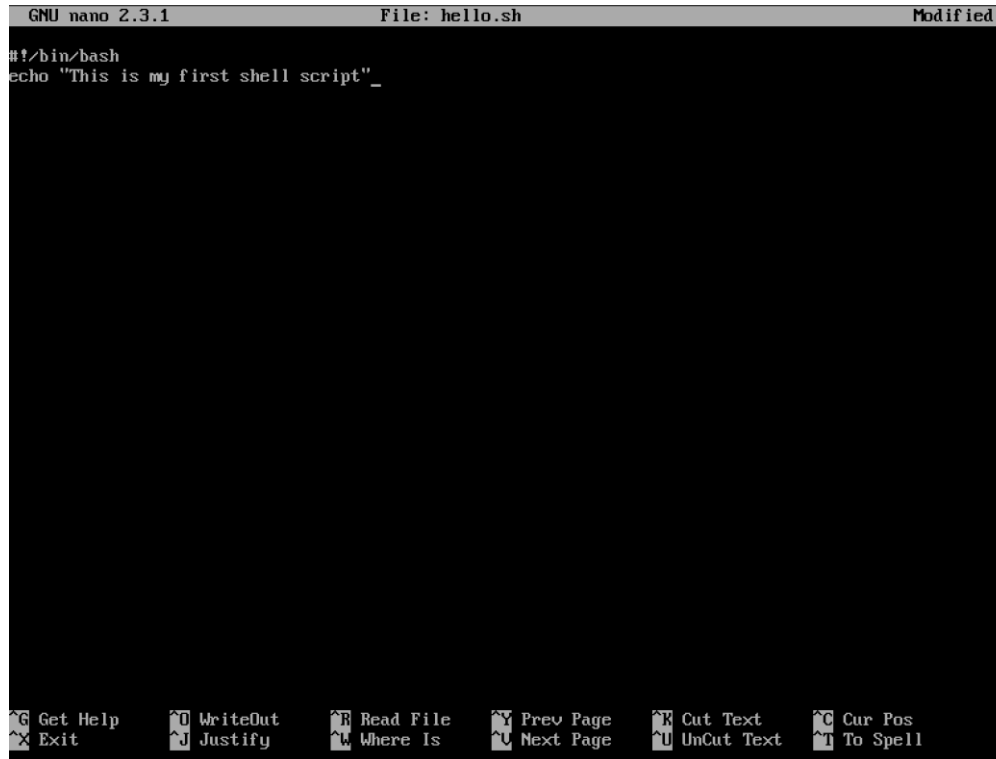
Ref 1: Pg (189)

5.2. Shell Basics...(2)

- Do you remember?
 - Basic shell commands - man, ls, mkdir, cp, mv ... etc
 - Pipes and redirection
 - Environment variables
 - Filter commands - grep, sort, head, tail, tee ... etc
- If not go through the basic shell commands.

5.3. Shell Scripting

- An shell script can consist of nothing but a series of command lines.



```
GNU nano 2.3.1      File: hello.sh      Modified
#!/bin/bash
echo "This is my first shell script" _
```

The screenshot shows a terminal window with the GNU nano 2.3.1 text editor. The editor is editing a file named 'hello.sh'. The first line of the script is the shebang '#!/bin/bash', and the second line is the command 'echo "This is my first shell script" _'. The bottom of the window displays various keyboard shortcuts for nano, such as ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Page, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where Is, ^U Next Page, ^U UnCut Text, and ^T To Spell.

- The first line is known as the “shebang” statement and declares the text file to be a script for interpretation by /bin/bash

Ref 1: Pg (198-209)

5.3. Shell Scripting...(2)

- Need execution and read permission to the script

```
[tharindu@localhost scripts]$ ls -l
total 4
-rw-rw-r--. 1 tharindu tharindu 49 Jan  3 00:43 hello.sh
[tharindu@localhost scripts]$ sudo chmod +x hello.sh
[sudo] password for tharindu:
[tharindu@localhost scripts]$ ls -l
total 4
-rwxrwxr-x. 1 tharindu tharindu 49 Jan  3 00:43 hello.sh
[tharindu@localhost scripts]$ _
```

- Execute the script.

```
[tharindu@localhost scripts]$ ./hello.sh
This is my first shell script
[tharindu@localhost scripts]$
```

5.3. Shell Scripting...(3)

- Read the user input.

```
#!/bin/bash  
  
echo "Enter your name: "  
read name  
  
echo "Hello $name"
```

```
[tharindu@localhost scripts]$ ./userinput.sh  
Enter your name:  
Kenneth  
Hello Kenneth  
[tharindu@localhost scripts]$
```


5.3. Shell Scripting...(4)

- Conditions

```
#!/bin/bash

echo "Enter a number: "
read number

if [ $number -gt 10 ]; then
    echo "Number is greater than 10"
else
    echo "Number is less than 10"
fi
```

```
[tharindu@localhost scripts]$ ./control.sh
Enter a number:
8
Number is less than 10
[tharindu@localhost scripts]$ ./control.sh
Enter a number:
12
Number is greater than 10
```

Ref 1: Pg (198-209)

5.3. Shell Scripting...(5)

- Loops - for

```
#!/bin/bash

for value in {1..5}
do
    echo $value
done
```

```
[tharindu@localhost scripts]$ ./loops.sh
1
2
3
4
5
```

5.3. Shell Scripting...(6)

- Loops - while

```
#!/bin/bash

x=1
while [ $x -le 10 ]
do
    echo $x
    ((x++))
done
```

```
[tharindu@localhost scripts]$ ./loops.sh
1
2
3
4
5
6
7
8
9
10
```

Ref 1: Pg (198-209)

5.3. Shell Scripting...(7)

- Arithmetic

```
#!/bin/bash

a=10
b=$((20))

c=$a+$b
d=$((a+b))

echo "$a + $b = $c \t(plus sign as string literal)"
echo "$a + $b = $d \t(plus sign as arithmetic addition)"
```

```
[tharindu@localhost scripts]$ ./maths.sh
10 + 20 = 10+20 \t(plus sign as string literal)
10 + 20 = 30 \t(plus sign as arithmetic addition)
[tharindu@localhost scripts]$ _
```

5.4. Regular Expressions

- Regular expressions are standardized patterns that parse and manipulate text.
- Regular expression comparison operator: `=~`
- Any single digit: `[0-9]`
- Any capital letter: `[A-Z]`
- Sequence of capital letters: `[A-Z] +`
- Beginning of a string: `^`
- End of a string: `$`

5.4. Regular Expressions...(2)

```
#!/bin/bash

echo "Enter a digit: "
read digit

if [[ $digit =~ [0-9] ]]; then
    echo "$digit is a digit"
else
    echo "$digit is not a digit"
fi
```

```
[tharindu@localhost scripts]$ ./regx.sh
Enter a digit:
2
2 is a digit
[tharindu@localhost scripts]$ ./regx.sh
Enter a digit:
s
s is not a digit
```

Ref 1: Pg (209-215)

5.5. Python Scripting

- Python is an interpreted language.
- It is a general-purpose scripting language and have extensive libraries of third party modules.
- It's the modern era's go-to language for both system administration and general-purpose scripting.
- Python 2 and Python 3 are two versions commonly used.
- Most of the operating systems are preinstalled with Python 2 which is by default called as python and Python 3 is called as python3.
- It is better to have experience in Python language.

5.5. Python Scripting...(2)

- Simple python script

```
import sys

number = input("Enter a number: ")

if int(number) > 10:
    print(number + " is greater than 10")
else:
    print(number + " is less than 10")
```

```
[tharindu@localhost scripts]$ python3 pythonScript.py
Enter a number: 3
3 is less than 10
```


5.6. Version Control with GIT

- It's important to keep track of configuration and code changes so that when these changes cause problems, you can easily revert to a known-good state.
- Revision control systems,
 - Define an organized way to trace the history of modifications to a file such that changes can be understood in context and so that earlier versions can be recovered.
 - They extend the concept of versioning beyond the level of individual files. Related groups of files can be versioned together, taking into account their interdependencies.
 - Coordinate the activities of multiple editors so that race conditions cannot cause anyone's changes to be permanently lost. So that incompatible changes from multiple editors do not become active simultaneously.
- One of the popular system is Git.

Ref 1: Pg (235)

5.6. Revision Control with GIT...(2)

- Git Commands
 - git init - creates the repository's infrastructure by creating a .git
 - git add - copies the changes from the working directory to staging area for the upcoming commit.
 - git commit - enters the contents of the index into the repository.

Reference

- Ref 1. Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley and Dan Mackin "UNIX and Linux System Administration Handbook " (5th Edition), Pearson Education, Inc., 2018.