IS 340 – Operating Systems
HOP08 – BASH – Shell Functions

School of Technology & Computing (STC) @ City University of Seattle (CityU)

7/16/2019 Developed by Kevin Wang
6/18/2020 Reviewed by Kim Nguyen
02/14/2022 Reviewed by Ken Ling

Before You Start

- This exercise assumes that the user is working with the Ubuntu 18.04 distribution. If you are working with a different Linux distribution, the set of shell commands may vary from those available in Ubuntu 18.04.
- Students will use the EC2 Ubuntu virtual machine that they created in the module 1 exercise.
- All commands and code discussed in this exercise will run in the Ubuntu console.
- The directory path shown in screenshots may be different from yours.
- Some steps are not explained in the tutorial**.** If you are not sure what to do:
    1. Consult the resources listed below and experiment in the Ubuntu console and try to solve the problem yourself.
    2. If you cannot solve the problem after a few tries, ask a TA for help.

Learning Outcomes

Students will be able to:

- Define functions in the shell
- Define functions in the file
- Use functions in the file

Resources

- Linux command line: **bash + utilities**
  https://ss64.com/bash/

- Nano/Basics Guide
  https://wiki.gentoo.org/wiki/Nano/Basics_Guide

Preparation

1) Connect to your Ubuntu instance
   Open a command prompt
   Syntax: ssh -i LOCATION_OF_YOUR_KEY ubuntu@PULIC_DNS
   Example:
   >>>ssh -i key.pem ubuntu@ec2-33-222-101-222.us-west-2.compute.amazonaws.com

2) Navigate to your name directory under the IS340-Summer-2020
   >>> cd ~/ IS340-Summer-2020/YOURNAME
   Note: change YOURNAME to your real name

3) Create a Module8 directory under YOURNAME directory.
   Note: If this directory exists, skip this step.
   >>> mkdir Module8

4) Navigate to the Module8 directory.
>>> cd Module8

Define a function in the shell

1) The bash supports three different syntaxes for defining a function:
   - function name <compound command>
   - name() <compound command>
   - function name() <compound command>

   In this module, we will use the second syntax.

2) Define a function in the shell by typing the following command:
>>> circleArea() (area=`echo "3.1415926 * $1 * $1" | bc -l`; printf "The circle's area is %s\n" "$area")
Note: the script uses the bc command to calculate float numbers. The syntax is variable=`echo "formula" | bc -l`

3) Test the script by typing the following commands:
>>> circleArea 12

```
ubuntu@ip-172-31-20-124:~/CS120/Module6$ circleArea() ( area=`echo "3.1415926 * $1 * $1" | bc -l`; printf "The circle's area is %s\n" "$area" )
ubuntu@ip-172-31-20-124:~/CS120/Module6$ circleArea 12
The circle's area is 452.3893344
```

4) You can save the print result to a variable for future usage. Try it by typing the following commands:
>>> info=$(circleArea 5)
>>> echo $info

```
ubuntu@ip-172-31-20-124:~/CS120/Module6$ info=$( circleArea 5 )
ubuntu@ip-172-31-20-124:~/CS120/Module6$ echo $info
The circle's area is 78.5398150
```

Note: you can use this method to get a return value from a function

5) The function also can return an exit code (0 - 255). Test it by typing the following commands:
>>> getNumber() (return $(($RANDOM * 255 + 1)))
>>> getNumber
>>> echo $?
>>> getNumber
>>> echo $?
>>> getNumber
>>> echo $?

```
ubuntu@ip-172-31-20-124:~/CS120/Module6$ getNumber() ( return $(( $RANDOM * 255 + 1 )) )
ubuntu@ip-172-31-20-124:~/CS120/Module6$ getNumber
ubuntu@ip-172-31-20-124:~/CS120/Module6$ echo $?
70
ubuntu@ip-172-31-20-124:~/CS120/Module6$ getNumber
ubuntu@ip-172-31-20-124:~/CS120/Module6$ echo $?
140
ubuntu@ip-172-31-20-124:~/CS120/Module6$ getNumber
ubuntu@ip-172-31-20-124:~/CS120/Module6$ echo $?
143
```

Note: The $? represents the last exit code which we returned from the function. You are supposed to see different numbers since our function returns a random number every time it is executed.

Define a function in the file

1) Functions defined in the shell will lose after you close the shell. It is good to write your functions in the file for the reusability

2) Create a script file by typing the following command:
>>> nano SystemSnapshot.sh

```
#! /bin/bash

snapshot() {
  filename=$(date '+%Y-%m-%d%H:%M:%S').log
  top -n 1 | tee "$filename"
}
```

Note: we define a function called snapshot to take a system snapshot by running top command and save the information to a file with the name contains date and time that snapshot was taken.

3) Hit the control + x key to quit and save the file.

4) Before we can use the function that defines in a file, we have to source in the file in the current shell, making functions available to the shell. Run the following command to source in the file:
>>> . SystemSnapshot.sh

5) Try to run the function by typing the following command:
>>> snapshot
>>> clear

We run clear to clear the screen in order to see the file content later.

6) Run the following command to see file names in the folder:
>>> ls

```
ubuntu@ip-172-31-20-124:~/CS120/Module6$ ls
2019-07-1802:16:22.log  SystemSnapshot.sh
```

You will see a different file name because it was generated by the date and time you run the function.

```
top - 02:16:22 up 6 days, 20:55,  1 user,  load average: 0.00, 0.01, 0.00
Tasks:  86 total,   1 running,  52 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1007520 total,   167776 free,   110928 used,   728816 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   735816 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0  159848   9136   6708 S  0.0  0.9   0:06.57 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.47 ksoftirqd/0
    8 root      20   0       0      0      0 I  0.0  0.0   0:00.99 rcu_sched
    9 root      20   0       0      0      0 I  0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   11 root      rt   0       0      0      0 S  0.0  0.0   0:01.43 watchdog/0
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kdevtmpfs
   14 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 netns
   15 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_tasks_kthr
```

7) Check the file content by typing the following command:
>>> cat 2019-07-1802:16:22.log
Note: you should change the file name to your own file name.

Submit your Work to Brightspace

Please upload all your files for this hands-on practice to the HOP assignment on Brightspace.