UNIX LAB

CSC 2500

Department of Computer Science
Tennessee Tech University

LAB 7: More Bash Scripting with `while` Loops, Functions, `Case`, and `arrays`

Objective:

- Shell variables
- Simple `if` and `case` decision statements
- The `read` command
- File redirection
- Command substitution
- Looping with `while` and `for`

Home Work

For this lab, you will write a shell script called `nethelper.sh` that:

1. Reads in a list of hosts from a host file that is passed in as a parameter
2. Repeatedly, until the user selects quit (q), ask the user if the user wants to `ping` a host or look up a hosts DNS name
3. Once the user has selected an action, prompts the user for which host and then applies the action. Note that for the `ssh` action, the script must also ask the user for the user name that will be used to log into the host.

The following is an example execution of the script.

```
$ ./nethelper.sh hosts.txt
(P) for ping

(N) for nslookup

(Q) for quit

Select one of the above:  p

1) localhost

2) www.google.com

Enter a number to select a host: 2

ping -c 1 www.google.com
```

```
PING www.google.com (74.125.126.147) 56(84) bytes of data.

64 bytes from ik-in-f147.1e100.net (74.125.126.147): icmp_seq=1 ttl=52
time=1.72 ms


--- www.google.com ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 1.722/1.722/1.722/0.000 ms

(P) for ping

(N) for nslookup

(Q) for quit

Select one of the above:  n

1) localhost

2) www.google.com

Enter a number to select a host: 2

nslookup www.google.com

Server:         172.17.0.1

Address:        172.17.0.1#53


Non-authoritative answer:

Name:   www.google.com

Address: 74.125.124.103

Name:   www.google.com

Address: 74.125.124.105

Name:   www.google.com

Address: 74.125.124.147

Name:   www.google.com

Address: 74.125.124.104

Name:   www.google.com

Address: 74.125.124.99
```

```
Name:   www.google.com

Address: 74.125.124.106

(P) for ping

(N) for nslookup

Select one of the above:  q

$
```

Details

Step 1

To accomplish step 1 above (reading in a list of hosts from a file), you will write a function called `read_hosts`. Functions in bash scripts have the following syntax:

```
function name {
 # function body goes here
}
```

You can put any bash syntax inside the function as the function body. Also, note that the parameters to a function act like the parameters passed in at the command line. In other words, the parameters to the function are place in variable named $1, $2, and so on.

So, how do you read the hosts from a file? The file will be formatted as one host per line. Therefore, you can simply use `cat` to print the file, capture the output, and the iterator over the output using a `for` loop. However, you will have to put the hosts in an array called `hosts_array`. Bash array are simple. You simply use the brackets to indicate an index. So, the algorithm for your function will look like the following:

```
function read_hosts
```

1. set variable `hosts` to the results of calling `cat` on `$1` (the name of the file passed as a parameter)
2. initialize a `count` variable to 1 (`count` will be the index into the `array`)
3. `for` each `host` in `hosts` `do` the following:
    1. `hosts_array[$count]=$host`
    2. add 1 to `count`

To call the function, your script will simply execute the following code:

```
read_hosts $@
```

The `$@` passes the script parameter (the name of the file containing the hosts) to the function. In

the above algorithm, `hosts_array` is a global variable, so the function's caller will be able to use the global variable to access the host names.

Step 2 and 3

To implement step 2, you need a `while` loop. Before the loop, create a variable called `done` so the loop knows when to exit. In the body of the loop, print the menu (using `echo`, of course), then read the user's response. Next, use a `case` statement to determine which menu item the user chose (either a P, S, T, N, or Q). The `case` statement should execute the correct command based on the user's choice. For example, if the user chooses P, the script should run the ping command like so:

```
ping -c 1 ${hosts_array[$which_host]}
```

However, before you run `ping`, you should print the list of hosts, and ask the user to pick the host that the user wishes to include in the command. You will write a function called `pick_host` that allows the user to select a host. The algorithm for `pick_host` is as follows:

```
function pick_host
```

1. set variable `count` to 1
2. for each host in $1 (where $1 is the list of hosts passed it - not the array)
    1. `echo "$count) $host"`
    2. add 1 to `count`
3. prompt the user to enter a number to select a host
4. read the user's response into the `which_host` variable
5. make sure that `$which_host` is greater than or equal to 1 but less than `$count` (it is a valid host in the array). If it is not, exit with an error message.

The algorithm for the rest of steps 2 and 3 are as follows:

```
while done == 0
```

1. `echo` P for `ping` or N for nslookup
2. prompt user for response
3. read response into variable `cmd`
4. call `read_hosts` function
5.
```
case $cmd in
    P|p)
       pick_host "$hosts"
       echo "ping -c 1 ${hosts_array[$which_host]}"
       ping -c 1 ${hosts_array[$which_host]}
       ;;

    N|n)
       pick_host "$hosts"
       echo "nslookup ${hosts_array[$which_host]}"
       nslookup ${hosts_array[$which_host]}
```

```
            ;;
        Q|q)

            done=1;
            ;;
        *) echo "Bad choice";
            ;;
    esac

done
```

Submission

Submit your `nethelper.sh`

Submission Deadline:

Submission Site: iLearn (a Dropbox folder named “Lab 07”)