Bash Scripting Knowledge Check Solutions

Course Outline

Variables

```
#!/bin/bash
MY_STR="Hi there!"
echo 'My string is: $MY_STR'
echo "My string is: $MY_STR"
```

1. What will be the first line output by this script?
   `My string is: Hi there!`

2. What will be the second line output by this script?
   `My string is: $MY_STR`

`./example_script 0 1 2`

```
#!/bin/bash
echo $0
echo "Printing $3, $2, $1"
echo $#
```

3. What will be the first line output by this script?
   `./example_script`

4. What will be the second line output by this script?
   `Printing 2, 1, 0`

5. What will be the third line output by this script?
   `3`

Arithmetic

`./example_script 2 3`

```
let "num_1 = $1 * $2"
```

```
echo $num_1
expr $1 \* $2
expr $1 * $2
num_2=$(( num_1 / $1 ))
echo $num_2
```

1. What w ill be the first line output by this script?
   - ☐ 2
   - ☐ 3
   - ☑ 6
   - ☐ Error
2. What w ill be the second line output by this script?
   - ☐ 2
   - ☐ 3
   - ☑ 6
   - ☐ Error
3. What w ill be the third line output by this script?
   - ☐ 2
   - ☐ 3
   - ☐ 6
   - ☑ Error
4. What w ill be the fourth line output by this script?
   - ☑ 2
   - ☐ 3
   - ☐ 6
   - ☐ Error

```
#!/bin/bash
expr $RANDOM % 100
```

5. What w ill this script output?

   `A random number between 0 and 100`

6. Write a script to convert a fahrenheit value to celsius, to the nearest integer. (Hint: pass the fahrenheit value to the script as a command line argument.) For a challenge, output the answ er to 3 decimal places.

```
#!/bin/bash
# This script is one example solution to the problem
# It accepts a fahrenheit value and converts it to celsius
echo "$1 degrees fahrenheit is $(( ($1-32)*5/9 )) degrees celsius"
```

```
#!/bin/bash
# This script shows two solutions to the challenge problem
# It accepts a fahrenheit value and converts it to celsius
printf "$1 degrees fahrenheit is %.3f degrees celsius\n" $( awk "BEGIN {print ($1-32)*5/9}" )
printf "$1 degrees fahrenheit is %.3f degrees celsius\n" $( echo "($1-32)*5/9" | bc -l ) # bc must be installed
```

Comparisons and Operators

```
#!/bin/bash
[ -z $1 ]
echo $?
```

1. What is the output if the script is called like so?: `./example_script Hello`
   - [x] 0
   - [ ] 1
   - [ ] Hello
   - [ ] No output
2. What is the output if the script is called like so?: `./example_script`
   - [ ] 0
   - [x] 1
   - [ ] Hello
   - [ ] No output

If Statements

```
#!/etc/bash
num_a=1000
num_b=1001
if [ $num_a != $num_b ]
then
  echo "$num_a and $num_b are not the same"
fi
```

What is the output of this script?

- [ ] 1000 and 1001 are not the same
- [x] No output

If / Else If / Else Statements

```
#!/etc/bash
if [ $1 -gt $2 ]
then
  echo "$1"
elif [ $2 -gt $1 ]
then
  echo "$2"
else
  echo "--"
fi
```

What does this script do?
It prints the larger of the tw o numbers provided, or '--' if they are the same

For Loops

```
#!/bin/bash
for val in {1..10}
do
  echo $val
```

```
done
```

1. What w ill be the first line output by this script?
   `1`
2. What w ill be the fifth line output by this script?
   `5`

`./example_script /usr`

```
#!/bin/bash
for entry in $(ls $1)
do
  echo $entry
done
```

3. What w ill be the first line output by this script?
   It depends on the OS distribution but probably bin
4. Briefly describe w hat this script is doing
   It prints out each entry in the given directory

While Loops

```
#!/bin/bash
counter=5
while [ counter -gt 0]
do
  echo $counter
done
```

1. What w ill be the first line output by this script?
   `5`
2. What w ill be the fifth line output by this script?
   `5` ; there is no decrement changing the counter so this w ill loop forever.

```
#!/bin/bash
counter=1
while [ $counter -lt 20]
do
  echo $counter
  if [ $(($counter % 2)) -eq 0]; then
    echo "even"
  fi
  let "counter = counter + 1"
done
```

3. What w ill be the fourth line output by this script?
   `3`
4. What is the sixth line output by this script?
   `even`

Functions

```
#!/bin/bash
```

```
my_function() {
  if [ $# -ne 2 ]; then
    echo "Incorrect"
  else
    echo $(($1 + $2))
  fi
}

my_val=$(my_function 4 5)
echo "My first value is $my_val"
my_val=$(my_function 1 2 3)
echo "My second value is $my_val"
```

1. What is the first line of output of this script?
   - [ ] `My first value is 4`
   - [ ] `My first value is 5`
   - [✓] `My first value is 9`
   - [ ] `My first value is Incorrect`
2. What is the second line of output of this script?
   - [ ] `My second value is 2`
   - [ ] `My second value is 3`
   - [ ] `My second value is 6`
   - [✓] `My second value is Incorrect`

Input

1. Write a simple script to collect a username and passw ord from the user. Make sure the passw ord is not visible as the user types it in.

```
#!/bin/bash
# This script is one example solution to the problem
echo Input your new credentials:
read -p "Username: " username
read -sp "Password: " userpass
echo # Return to the next line
echo Thank you, $username, your credentials have been input.
```

2. Now, develop that script to ask for the passw ord tw ice, confirming that the input w as the same both times. Keep asking until the passw ord has been input correctly. (Hint: make use of w hile loops, if statements, and user input.)

```
#!/bin/bash
# This script is one example solution to the problem
echo Input your new credentials:
read -p "Username: " username

passcorrect=0
while [ $passcorrect -eq 0]
do
  read -sp "Enter Password: " pass1
  echo
  read -sp "Confirm Password: " pass2
  echo
  if [ $pass1 == $pass2 ]
  then
    passcorrect=1
  else
    echo Those passwords did not match. Input again.
  fi
done
echo Thank you, $username, your credentials have been input.
```

File Operations

1. Write a simple script that w rites to a file in "/tmp/" named "bin_files". The first line of the file should have the text "These are the files located in /bin". The next lines should be all the files in your /bin directory. Finally, the last line should have the text "There are files in /bin" w here is the number of files.

```
#!/bin/bash
# This script is one example solution to the problem
file="/tmp/bin_files"
echo "These are the files located in /bin" > $file
ls /bin >> $file
num=$(ls /bin | wc -l)
echo "There are $num files in /bin" >> $file
```

0. Now, develop a script that w ill read in the "/tmp/bin_files" file you just created, echo out every filename that starts w ith a "cu", count the number of files that begin w ith "d", and output that number.
   (Hint: if statements using "==" behave differently in double brackets [[==]]. Recall you can use w ildcard matching i.e. `[[ test == tes* ]]`)

```
#!/bin/bash
# This script is one example solution to the problem
input="/tmp/bin_files"
count=0
while IFS= read -r line
do
  if [[ $line == cu* ]] ;
  then
    echo $line
  fi
  if [[ $line == d* ]] ;
  then
    let "count = $count + 1"
  fi
done < $input
echo "There are $count files that begin with \"d\""
```

Capstone

Write Your Own Scripts

Survey Script
Write a script to survey a target Linux box. You may assume you are root. Here are some ideas for w hat your script can do:

- Check w ho is logged in / how many users are logged in
- Check how long the system has been up
- Check system load averages
- Check the date of the systems
- Collect system information e.g. OS and conversion
- Enumerate the root directory's contents
- Enumerate the default logging directory's contents
- Gather the process list
- Capture the system's netw ork connections
- Capture its listening ports/services
- Check file system disk usage
- Check current remote mounts

- Collect user-, password-, and group-related files
- Gather the system's modules
- Gather the system's services
- Collect every user's cron jobs
- Compress all this data so it can be exfil'd

```
#!/bin/bash
# A Linux survey script
# You are assumed to be root
# The end result is a tarred set of files: tmp.txt and a collection of "tmp.*" randomly named files.
# tmp.txt is the "guide" or "table of contents" mapping the random filenames to their contents.

MYFILE=/tmp/tmp.txt
touch $MYFILE

#Who is logged in / how many users are logged in
USERS=$(mktemp)
echo "USERS: $USERS" >> $MYFILE
w >> $USERS
who >> $USERS
users >> $USERS

#How long the system has been up
SYSTEM=$(mktemp)
echo "SYSTEM: $SYSTEM" >> $MYFILE
uptime -p >> $SYSTEM
who -b >> $SYSTEM

#System load averages
uptime >> $SYSTEM
cat /proc/loadavg >> $SYSTEM

#Date
date >> $SYSTEM

#System information (kernel version)
#Operating system and version
echo "Kernel Name, Network Node Hostname, Kernel Release, Kernel Version, Machine Hardware Name, Processor Type, Hardware
uname -a >> $SYSTEM

#A directory listing of several important directories
FILES=$(mktemp)
echo "FILES: $FILES" >> $MYFILE
ls -al / >> $FILES

#Process list
PROCESSES=$(mktemp)
echo "PROCESSES: $PROCESSES" >> $MYFILE
ps -eo euser,ruser,suser,fuser,f,comm,label >> $PROCESSES
ps aux >> $PROCESSES

#Network connections and listening ports / services
NTWKING=$(mktemp)
echo "NTWKING: $NTWKING" >> $MYFILE
netstat -pantu >> $NTWKING

#File system disk usage and any current remote mounts
df >> $FILES
mount >> $FILES

#Directory listing of the logging directory
LOGS=$(mktemp)
echo "LOGS: $LOGS" >> $MYFILE
ls -al /var/log >> $LOGS

#Collect user and password files
PASSWD=$(mktemp)
SHADOW=$(mktemp)
GROUPS=$(mktemp)
cat /etc/passwd > $PASSWD
cat /etc/shadow > $SHADOW
cat /etc/group > $GROUPS

#Modules
MODULES=$(mktemp)
echo "MODULES: $MODULES" >> $MYFILE
lsmod >> $MODULES

#Services (systems using init will be in /etc/init.d)
service --status-all
SERVICES1=$(mktemp)
systemctl list-units --type service > $SERVICES1
SERVICES2=$(mktemp)
systemctl list-unit-files > $SERVICES2

#Anything else you feel would be beneficial information
### for each user in users, crontab -u user -l
CRONS=$(mktemp)
echo "CRONS: $CRONS" >> $MYFILE
for i in $(users); do echo $i >> $CRONS && crontab -u $i -l &>> $CRONS; done

### tar up and scp tmp files somewhere, or webserve so it can be downloaded, or just download via meterpreter, etc.
tar -cf srvy.tar tmp.*
#tar -tvf tmp.tar # to list tarred files

# Remove generated tmp files
rm tmp.*
```

Log Cleaning Script

Write a script to clean logs. Apply it to your survey script so you clean up your activity on the system. Here are some ideas for w hat your script can do:

- Erase all or a particular number of recent commands from bash history
- Clean any line w ith a particular IP address from every log in /var/log/
- For all logs in /var/log/, remove any lines w ith a timestamp from the past 2 minutes
  (Hint: use the `date` command and `egrep`)

```
#!/bin/bash
# A Linux cleanup script

# Remove the last 50 lines from bash history
# i.e. Preserve all but the last 50 lines
len_of_bash_hist=$(cat $HISTFILE | wc -l)
echo "Cleaning bash history..."
if [ $len_of_bash_hist -gt 50 ]; then
  head --lines=-50 $HISTFILE > new_bash_hist
  cat new_bash_hist > $HISTFILE
  rm new_bash_hist # cleanup
fi

# Remove all instances of a particular IP from all /var/log/ logs
bad_ip="10.10.10.10"
echo "Cleaning bad IP $bad_ip from logs..."
for file in $(grep -r -l $bad_ip /var/log)
do
  grep -v $bad_ip $file > cleanlog
  cat cleanlog > $file
  rm cleanlop # cleanup
done

# Remove any entries from the last 2 minutes from all /var/log/ logs
# Don't run this at the split nanosecond of noon or midnight; may have unexpected behavior
# Construct regular expressions of the current time, minus 1 minute, and minus 2 minutes
#   for both 12-hour and 24-hour formats
expr0_12hr=$(date "+%b %d %I:%M")":[0-9]{2} "$(date "+%p")
expr1_12hr=$(date -d "$(date)-1mins" "+%b %d %I:%M")":[0-9]{2} "$(date -d "$(date)-1mins" "+%p")
expr2_12hr=$(date -d "$(date)-2mins" "+%b %d %I:%M")":[0-9]{2} "$(date -d "$(date)-2mins" "+%p")
expr0_24hr=$(date "+%b %d %H:%M")
expr1_24hr=$(date -d "$(date)-1mins" "+%b %d %H:%M")
```

```
expr1_24hr=$(date -d "$(date)-2mins" "+%b %d %H:%M")
echo "Cleaning log entries timestamped in the past two minutes..."
for file in $(egrep -a -r -l "$expr0_12hr|$expr1_12hr|$expr2_12hr" /var/log)
do
  egrep -a -v "$expr0_12hr|$expr1_12hr|$expr2_12hr" $file > cleanlog
  cat cleanlog > $file
  rm cleanlog # cleanup
done
for file in $(egrep -a -r -l "$expr0_24hr|$expr1_24hr|$expr2_24hr" /var/log)
do
  egrep -a -v "$expr0_24hr|$expr1_24hr|$expr2_24hr" $file > cleanlog
  cat cleanlog > $file
  rm cleanlog # cleanup
done

echo Done!
```