What is Bash Shell Scripting?

- A **shell script** is a script written for the shell, or command line interpreter, of an operating system.
- The shell is often considered a simple **domain-specific programming language**.
- Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

What is Bash Shell Scripting?

- Bash Shell Script is an **interpreted language**. This means that the shell analyzes each statement in the language one line at a time, then executes it. This differs from languages such as C, in which programs are compiled into executable files.

- Interpreted languages are generally easier to debug and modify; however, they usually take much longer to execute than compiled programs.

About My Shell Script

- My program is a simple network monitoring script that acts as wrapper for the ping command.
- It takes and IP address or multiple IP addresses as arguments, creates a log file of ping statistics, and outputs the connection status of the host.
- I intended to use it for lab 4, to monitor the time it took for dynamic routing to reroute network traffic when a node is taken down.

Special Shell Variables

$0	The name of the program is assigned here.
$1 - $9	The arguments typed on the command line are assigned here.
${10}	Any argument after $9 must be accessed using curly braces.

Special Shell Variables

- $# Number of arguments passed to the program or number of parameters set by executing the set statement.
- $* Collectively references all positional parameters as $1, $2, …
- $@ Same as $* except when double quoted; collectively references all positional parameters as "$1", "$2", …

Special Shell Variables

- $? Exit status of the last command not executed in the background.
- $! The process ID number of the last program sent to the background for execution.
- $$ The process ID number of the program being executed.

Special Shell Variables

- (( )) Arithmetic operator; parses faster, only accepts numeric input.
- [ ] Idiomatic operator; shell built in, older and slower, accepts alpha-numeric input.

Version 1.0

Let’s have a look at my original code.

Improvements

Version 1.0

```
#!/bin/bash
#
# Monitor: a script to monitor the connection
# status of one or more IP addresses
# Author Sean Callahan & Solomon Bundy

IP=
COUNT="-c 1"
INTERVAL="-i 1"
EMSG="[-i interval] [-c count] [-b run in bg] [--help] <IPaddress> <IPaddress>"

if [ "$#" -eq 0 ] #Test for no args
then
        echo "$EMSG"
        exit 0
fi

TEST=$(echo "$@" | grep ^--help$) #Test for --help
if [ "$?" -eq 0 ]
        then
        echo "$EMSG"
        exit 1
fi
```

Version 1.1

```
#!/bin/bash
#
# Monitor: a script to monitor the connection
# status of one or more IP addresses
# Author Sean Callahan & Solomon Bundy

_ip=
_count="-c 1"
_interval="-i 1"
_emsg="[-i interval] [-c count] [--help] <IPaddress> <IPaddress>"

if (("$#"==0))      #Test for no args
then
        echo "$_emsg"
        exit 0
fi

echo "$@" | grep -q '^--help$'      #Test for --help
if (("$?"==0))
        then
        echo "$_emsg"
        exit 1
fi
```

More Improvements

Version 1.0

```
while [ "$#" -gt 0 ] #Start main loop
do
     TEST2=$(echo "$1" | grep ^-c$) #grep for option -c
     if [ "$?" -eq 0 ]
     then
          COUNT="$1 $2" #Assign positional parameter $1 and $2 to COUNT
          shift
          shift
     else
          TEST3=$(echo "$1" | grep ^-c[0-9]) #grep for option -c w/ space
          if [ "$?" -eq 0 ]
          then
               COUNT="$1" #Assign positional parameter $1 to COUNT
               shift
          fi
     fi
```

Version 1.1

```
while (("$#">0))                          #Start main loop
do
     echo "$1" | grep -q '^-c$'            #grep for option -c
     if (("$?"==0))
     then
          _count="$1 $2"                   #Assign positional parameter $1 and $2 to _count
          shift
          shift
     else
          echo "$1" | grep -q '^-c[0-9]' #grep for option -c w/ space
          if (("$?"==0))
          then
               _count="$1"                 #Assign positional parameter $1 to _count
               shift
          fi
     fi
```

More Improvements

Version 1.0

```
set "$@" #Set all args (only IP addresses should be left at this point)

echo "$1" | grep -E -o -q '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-
        5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-
        9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)‘

        #Above line greps for a valid IP address

        if [ "$?" -ne 0 ]
        then
                echo "INVALID IP SKIPPING..."
                shift
                continue
        else
                IP="$1"
                shift
        fi
```

Version 1.1

```
set "$@” #Set all args (only IP addresses should be left at this point)

echo "$1" | grep -E -o -q '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-
        5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-
        9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)‘

        #Above line greps for a valid IP address

        if (("$?"!=0))
        then
                echo "INVALID IP SKIPPING..."
                shift
                continue
        else
                _ip="$1"
                shift
        fi
```

GREP Options

```
echo "$1" | grep -E -o -q '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.
```

-E	Extend regular expression.
-o	Only matching
-q	Quiet mode

One Regular Expression To Rule Them All

- echo "$1" | grep -E -o -q '**(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.**
- The **|** is the **alternation operator**. Since the alternation operator has the **lowest precedence** of all, we use the round brackets to group the alternatives together. The **?** makes the preceding item optional. The \ is an escape character.
- The expression will first test 250 -255.
- If this fails, it will look for the next set of numbers, 200 -249. If this fails, it will look for 100 – 199, then 0-99.
- If successful, it will return 0, and test the next set of numbers in the expression. If nothing is found, it will return 1, and stop.

More Improvements

Version 1.0

```
ping "$COUNT" "$INTERVAL" "$IP" 1> $IP.log        #Create a log file of ping output
echo "Monitoring "$IP" "
PING=$(ping "$COUNT" "$INTERVAL" "$IP" | grep 'received' | awk -F',' '{ print $2 }' |
awk '{ print $1 }') #Check ping status

if [ "$PING" -eq 0 ]
then
        echo "Host : $IP is down (ping failed) at $(date)"
else
        echo "Host : $IP is up (ping succeeded at $(date)"
fi

done

exit 0
```

Version 1.1

```
echo $(date) >> "$_ip".log
ping "-W 3" "$_count" "$_interval" "$_ip" >> "$_ip".log    #Create a log file of ping output
echo " " >> "$_ip".log
echo "Monitoring "$_ip" “
_result=$(ping "-W 3" "$_count" "$_interval" "$_ip" | grep 'received' | awk -F',' '{ print $2
}' | awk '{ print $1 }')                        #Check ping status
        if (("$_result"==0))
        then
              echo "Host : "$_ip" is down (ping failed) at $(date)"
        else
              echo "Host : "$_ip" is up (ping succeeded) at $(date)"
        fi

done

exit 0
```

What’s Next?

- Utilize the /etc/hosts file to allow users to type in host names as well as IP addresses.
- Separate the regular expression and ping code into their own loops, so that the program won’t scan for all of the options every time it loops.
- Include an option for the program to run silently in the background, and only bring itself into the foreground when a ping is successful.

```
#!/bin/bash
#
# Monitor: a script to monitor the connection
# status of one or more IPaddresses
# Author Sean Callahan & Solomon Bundy

_ip=
_count="-c 1"
_interval="-i 1"
_emsg="[-i interval] [-c count] [--help] <IPaddress> <IPaddress>"

if (("$#"==0))                          # Test for no args
then
    echo "$_emsg"
    exit 0
fi

echo "$@" | grep -q '^--help$'          # Test for --help
if (("$?"==0))
then
    echo "$_emsg"
    exit 1
fi

while (("$#">0))                        # Start main loop
do
    echo "$1" | grep -q '^-c$'          # grep for option -c
    if (("$?"==0))
    then
        _count="$1 $2"                  # Assign positional parameter $1 and $2 to _count
        shift
        shift
    else
        echo "$1" | grep -q '^-c[0-9]' # grep for option -c w/ space
        if (("$?"==0))
        then
            _count="$1"                 # Assign positional parameter $1 to _count
            shift
        fi
    fi

    echo "$1" | grep -q '^-i$'          # Grep for option -i
    if (("$?"==0))
    then
        _interval="$1 $2"               # Assign positional parameter $1 and $2 to _interval
        shift
        shift
    else
        echo "$1" | grep -q '^-i[0-9]' # grep for option -i w/ space
        if (("$?"==0))
        then
            _interval="$1"              # Assign positional parameter $1 and $2 to _interval
            shift
        fi
    fi

    set "$@"                           # Set all args (only IPaddresses should be left at this
    point)

    echo "$1" | grep -E -o -q
    '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0
    -4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)'
                                       #Above line greps for a valid IPaddress
    if (("$?"!=0))
    then
        echo "INVALID IP SKIPPING..."
        shift


        continue
    else
        _ip="$1"
        shift
    fi
    echo $(date) >> "$_ip".log
    ping "-W 3" "$_count" "$_interval" "$_ip" >> "$_ip".log     #Create a log file of ping
    output
    echo " " >> "$_ip".log
    echo "Monitoring "$_ip" "
    _result=$(ping "-W 3" "$_ip" | grep 'received' | awk -F',' '{ print $2 }' | awk '{ print $1
    }') #Check ping status
    if (("$_result"==0))
    then
        echo "Host : "$_ip" is down (ping failed) at $(date)"
    else
        echo "Host : "$_ip" is up (ping succeeded at $(date)"
    fi

done

exit 0
```