Sunday, March 05, 2017

LIKE GEEKS

LINUX

How To Write Practical Shell Script

February 25, 2017 admin 0 Comments

In the last post, we've talked about **regex** and we see how to use them in sed and awk for text processing and we discussed before **Linux sed** command and **awk command**. During the series, we write small shell **scripts** but we didn't mix things up, I think we should take a small step and write a shell script that can be some useful.

The main reason for learning to write a shell script is to be able to create your own Linux utilities. Understanding how to write useful and practical scripts is important.

However, sometimes it helps to do something fun to learn a concept or skill. The scripts in this post they can be lots of fun! And they help empower your script writing concepts.

Our main points are:

Sending messages

Write command

Creating the send script

Monitoring Disk Space

Sending messages

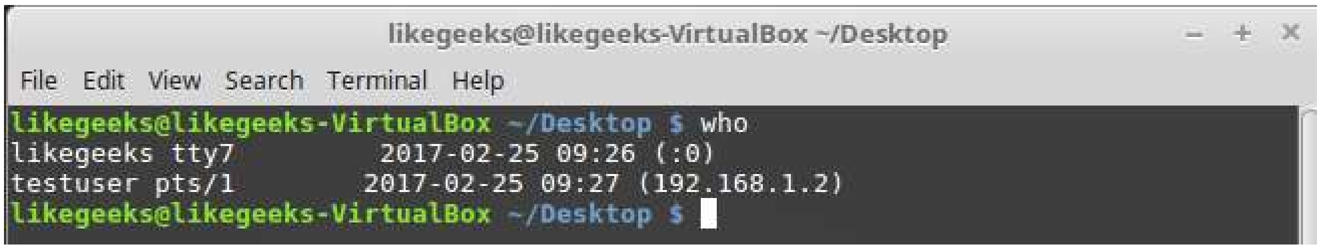
You can send messages to someone by phone or e-mail but one method, not commonly used anymore, is sending a message directly to a fellow system user's terminal. Because this technique is largely unknown

The shell script helps you to quickly and easily send a message to someone who is logged onto your Linux system.

For this simple shell script, only a few functions are required. Several of the commands are common and have been covered in our series of shell scripting; you can review the posts to understand what we are talking about

The first command needed is the who command. The who command allows you to see all the users currently logged into the system.

```
$ who
```

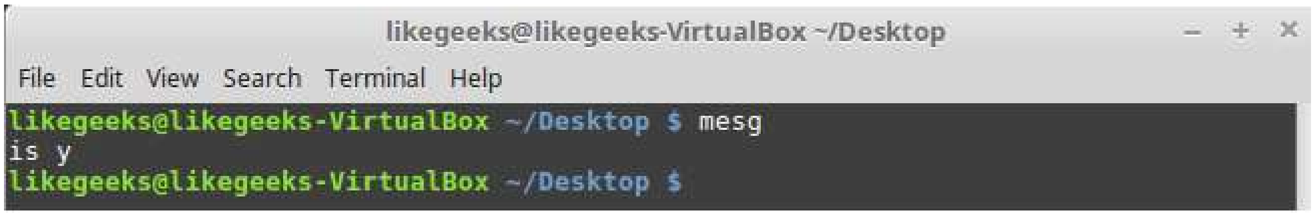

likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ who
likegeeks tty7         2017-02-25 09:26 (:0)
testuser pts/1        2017-02-25 09:27 (192.168.1.2)
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

To send a message, you only need the first two items. Both the username and the user's current terminal are necessary

Users can prevent anyone to send them messages via the mesg command. Therefore, before you start attempting to send messages, it's a good idea to check whether messages are allowed. For yourself, you can simply enter the mesg command like this

```
$ mesg
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ mesg
is y
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

The is n result shows that messaging is turned off. If the result showed is y, messages would be allowed.

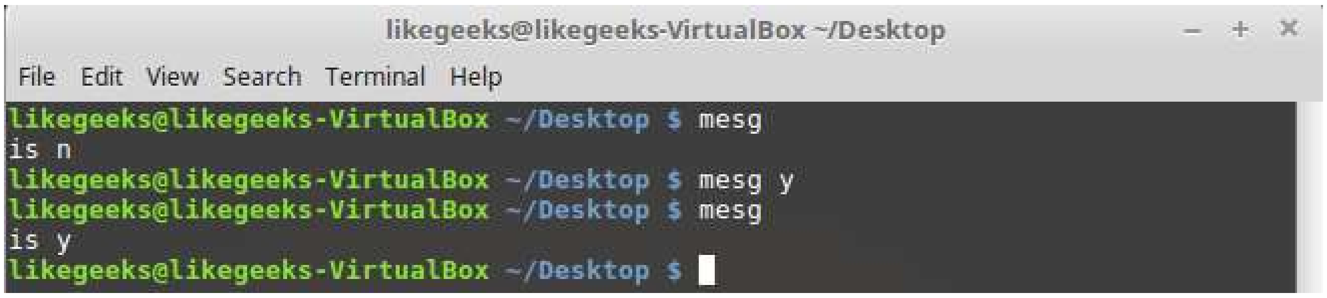
To check anyone else's message status, you can use the who command again. Keep in mind that this checks the message status only for those who are currently logged into the system. You use the -T option to check their message status

If you see dash (-) that means messages are turned off and if you see plus sign (+) that means messages are enabled

To allow messages to be sent to you, if it is turned off, you need to use the message command with the y option like this

```
$ mesg y
```

```
$ mesg
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ mesg
is n
likegeeks@likegeeks-VirtualBox ~/Desktop $ mesg y
likegeeks@likegeeks-VirtualBox ~/Desktop $ mesg
is y
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Sure enough, the command shows is y, which indicates messages are allowed to this user

Of course, we need another user to be able to communicate with him so in my case I'm going to connect to my PC using ssh and I'm already logged in with my user, so we have two users logged onto the system

Now we can send the message from one user to another.

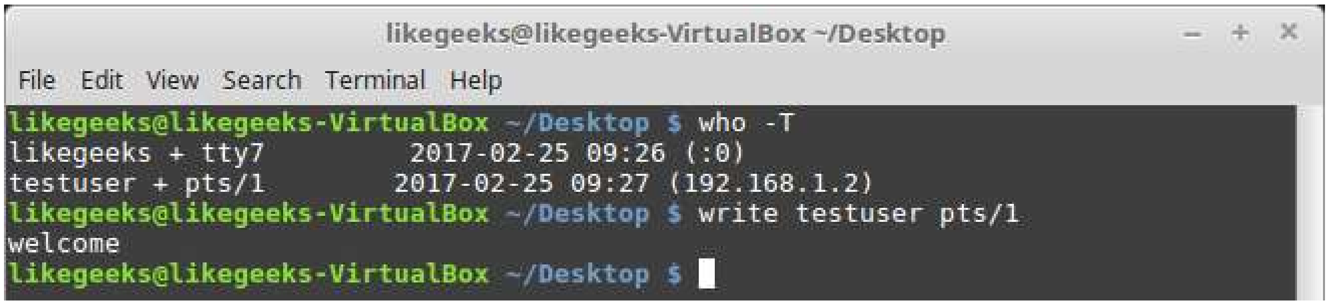
Write command

The primary tool for sending messages between logged in users is the write command. As long as messaging is allowed, the write command allows you to send a message to another logged-in user using his username and current terminal

Note: The write command only allows you to successfully send messages to users logged onto a virtual console terminal. A user logged into the graphical environment (KDE, Gnome, Cinnamon or any) will not be able to receive messages.

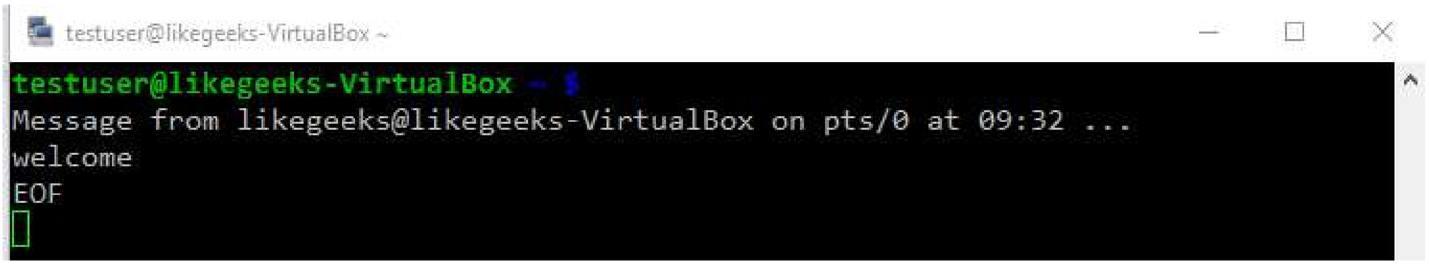
We will send a message to testuser user from my user likegeeks like this

```
$ write testuser pts/1
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ who -T
likegeeks + tty7         2017-02-25 09:26 (:0)
testuser + pts/1         2017-02-25 09:27 (192.168.1.2)
likegeeks@likegeeks-VirtualBox ~/Desktop $ write testuser pts/1
welcome
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

After the message is initiated by the write command, a blank line is shown for you to begin inputting the message text. It may be as many lines as you desire. When the Enter key is pressed, a new line is available for more message text. After you are finished entering message text, the whole message is sent by pressing the Ctrl+D key combination which is end of file signal I recommend you to review the post about **signals and jobs**.



```
testuser@likegeeks-VirtualBox ~ $
Message from likegeeks@likegeeks-VirtualBox on pts/0 at 09:32 ...
welcome
EOF
```

The receiver can see which user on which terminal sent the message. A timestamp is also included. Notice the EOF shown at the bottom. It indicates End Of File, which lets the message recipient know that this is the entire message.

I think now we have all part to build our shell script

Creating the send script

before we create our shell script, we need to determine whether the user we want to send a message to is logged on the system and this can be done using who command to determine that

```
logged_on=$(who | grep -i -m 1 $1 | awk '{print $1}')
```

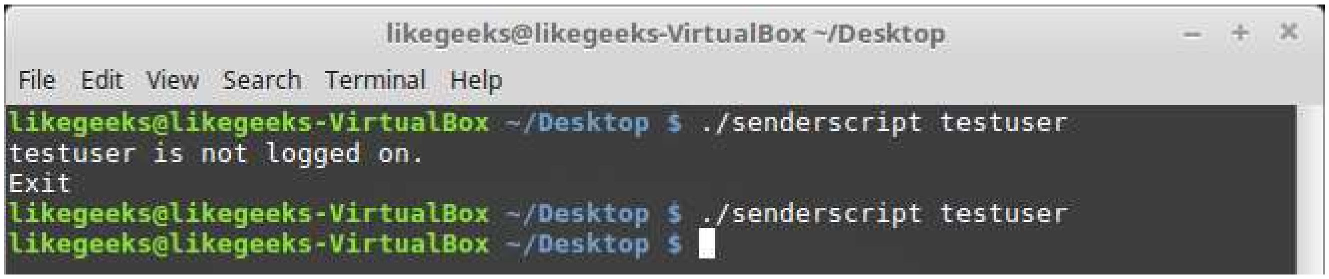
The results of the who command are piped into the grep command. The grep command uses the -i option to ignore case, so the username to be entered using uppercase or lowercase letters. The -m 1 option is included in the grep command, in case the user is logged into the system multiple times. The grep command produces either nothing, if the user is not logged on, or the username's first login information. This output is passed to the awk command. The awk command returns only the first item, either nothing or the username. This final output from the awk command is stored in the variable logged_on

Then we need to check the variable logged on if it contains something or not

```
if [ -z $logged_on ]

then

echo "$1 is not logged on."

echo "Exit"
```

```
exit
fi
```

I recommend you to read the post about if statement and how to use it **Bash Script**


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser
testuser is not logged on.
Exit
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

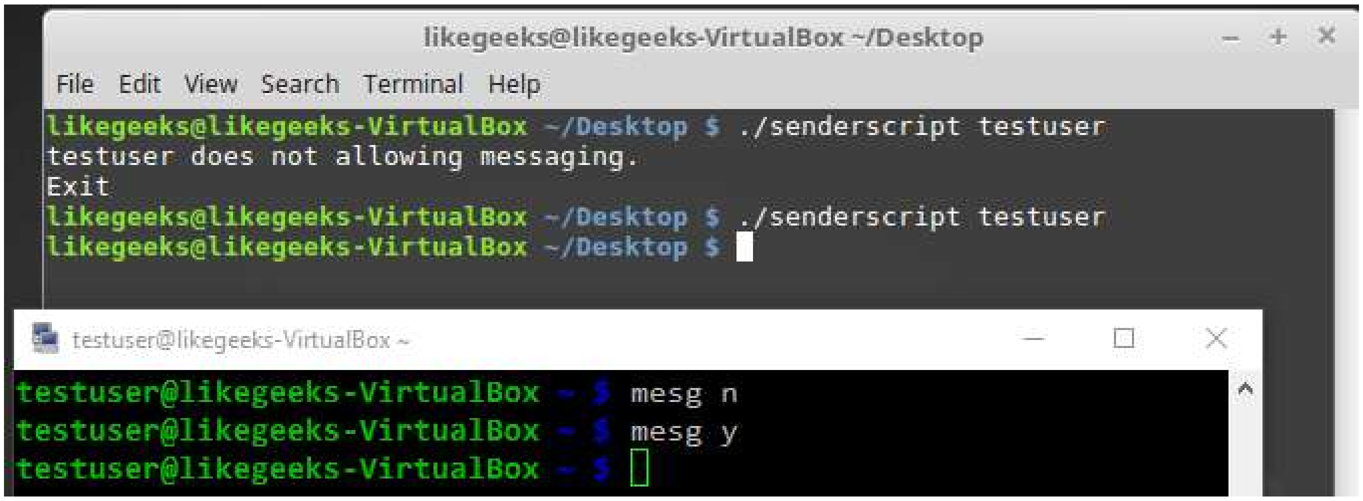
The logged_on variable is tested to determine if it is a zero-length variable.

If it is a zero-length variable, the script user is informed via echo commands that the user is not currently logged onto the system, and the script is exited via the exit command. If the user is logged onto the system, the logged_on variable contains the user's username, and the script continues

Checking if user accepts messages

The next step is to determine whether a logged on user accepts messages

```
allowed=$(who -T | grep -i -m 1 $1 | awk '{print $2}')
if [ $allowed != "+" ]
then
echo "$1 does not allowing messaging."
echo "Exit"
exit
fi
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
testuser@likegeeks-VirtualBox ~

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser
testuser does not allowing messaging.
Exit
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

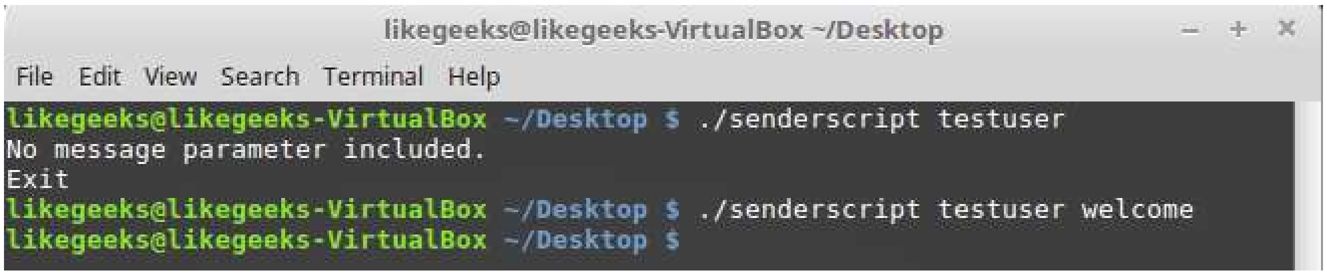
```
testuser@likegeeks-VirtualBox ~ $ mesg n
testuser@likegeeks-VirtualBox ~ $ mesg y
testuser@likegeeks-VirtualBox ~ $
```

Notice that we use the who command with -T. This displays a (+) next to the username if messaging is allowed. Otherwise, it displays a (-) next to the username, if messaging is not allowed. The results from the who command are then piped into grep and awk to pull out only the messaging indicator. The messaging indicator is stored in the allowed variable. Finally, an if statement is used to test for a messaging indicator not set to +. If the indicator is not set to +, the script user is informed and the script is exited. However, if the messaging indicator shows messaging is allowed, the script continues.

Checking if message was included

To test for the message parameter, an if statement is used like this

```
if [ -z $2 ]

then

echo "No message parameter included."

echo "Exit"

exit

fi
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser
No message parameter included.
Exit
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser welcome
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Getting the current terminal

Before we send a message, we need to get the user current terminal and stored in a variable

```
terminal=$(who | grep -i -m 1 $1 | awk '{print $2}')
```

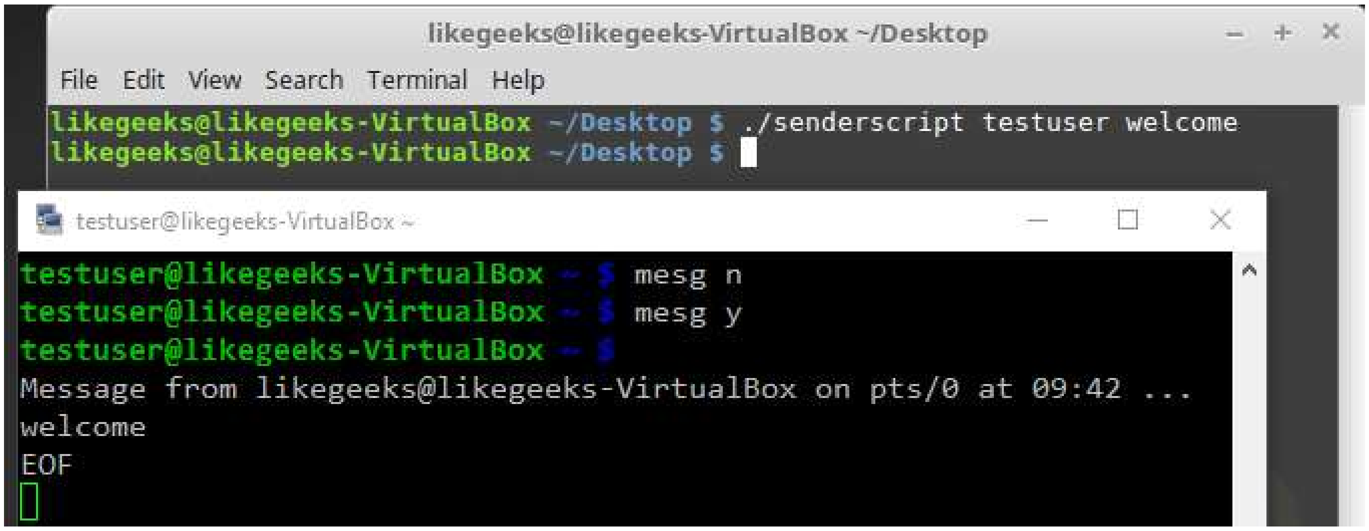
Then we can send the message

```
echo $2 | write $logged_on $terminal
```

Now we can test the whole shell script to see how it goes

```
$ ./senderscript likegeeks welcome
```

Let’s see the other shell window


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
testuser@likegeeks-VirtualBox ~

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser welcome
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

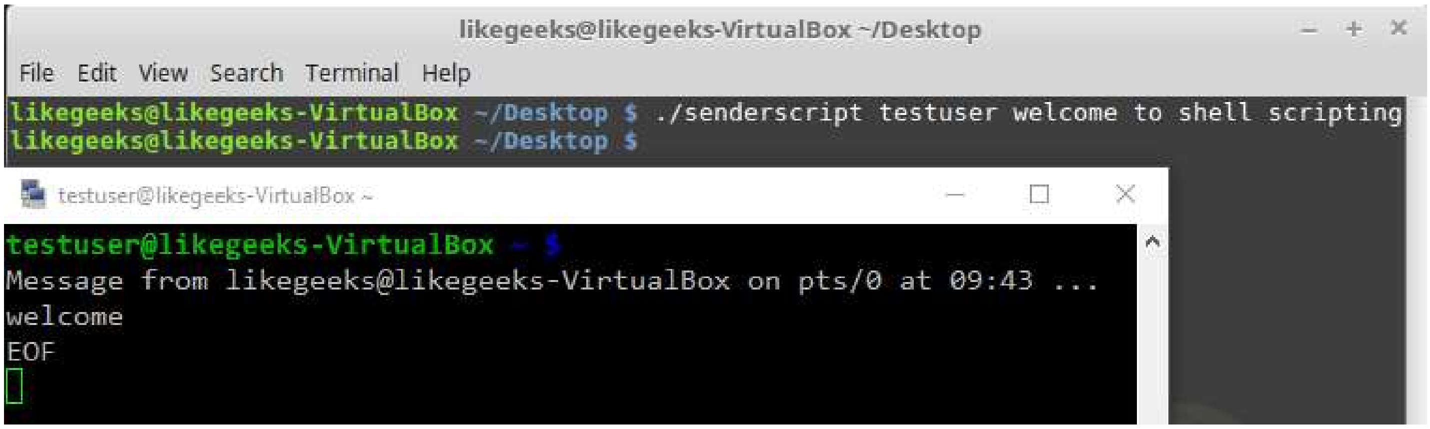
```
testuser@likegeeks-VirtualBox ~ $ mesg n
testuser@likegeeks-VirtualBox ~ $ mesg y
testuser@likegeeks-VirtualBox ~ $
Message from likegeeks@likegeeks-VirtualBox on pts/0 at 09:42 ...
welcome
EOF
```

Good! You can now send simple one-word messages

Sending a long message

Surely you want to send more than just a single word. If you try that with this shell script

```
$ ./senderscript likegeeks welcome to shell scripting
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
testuser@likegeeks-VirtualBox ~

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser welcome to shell scripting
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

```
testuser@likegeeks-VirtualBox ~ $
Message from likegeeks@likegeeks-VirtualBox on pts/0 at 09:43 ...
welcome
EOF
```

It didn't work. Only the first word of the message is sent. This is due to the script using parameters and each word is treated as a different parameter. To fix this problem we will use the shift command with the while loop you can review the shift command from our post **Bash Scripting**

```
shift
while [ -n "$1" ]
do
whole_message=$whole_message' '$1
shift
done
```

And now one thing need to be fixed the message parameter.

Instead of just sending parameter $2 to the write utility, the script is modified to send the variable, whole_message

```
echo $whole_message | write $logged_on $terminal
```

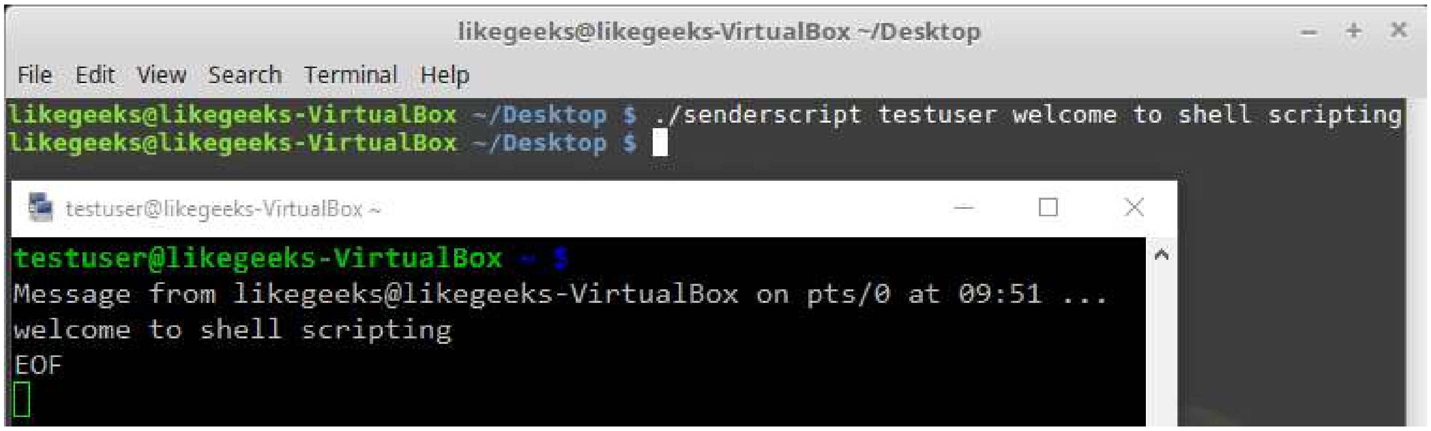
So now the whole script should be like this

```
#!/bin/bash
logged_on=$(who | grep -i -m 1 $1 | awk '{print $1}')
if [ -z $logged_on ]
then
```

```
echo "$1 is not logged on."

echo "Exit"

exit

fi

allowed=$(who -T | grep -i -m 1 $1 | awk '{print $2}')

if [ $allowed != "+" ]

then

echo "$1 does not allowing messaging."

echo "Exit"

exit

fi

if [ -z $2 ]

then

echo "No message parameter included."

echo "Exit"

exit

fi

terminal=$(who | grep -i -m 1 $1 | awk '{print $2}')

shift

while [ -n "$1" ]

do

whole_message=$whole_message' '$1

shift

done

echo $whole_message | write $logged_on $terminal
```

If you try now

```
$ ./senderscript likegeeks welcome to shell scripting
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./senderscript testuser welcome to shell scripting
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

```
testuser@likegeeks-VirtualBox ~ $
Message from likegeeks@likegeeks-VirtualBox on pts/0 at 09:51 ...
welcome to shell scripting
EOF
```

Awesome!! It worked. Again I'm not here to make a script to send the message to the user but the main reason to review our shell scripting knowledge and use all parts we've learned together and see how things work together.

Monitoring Disk Space

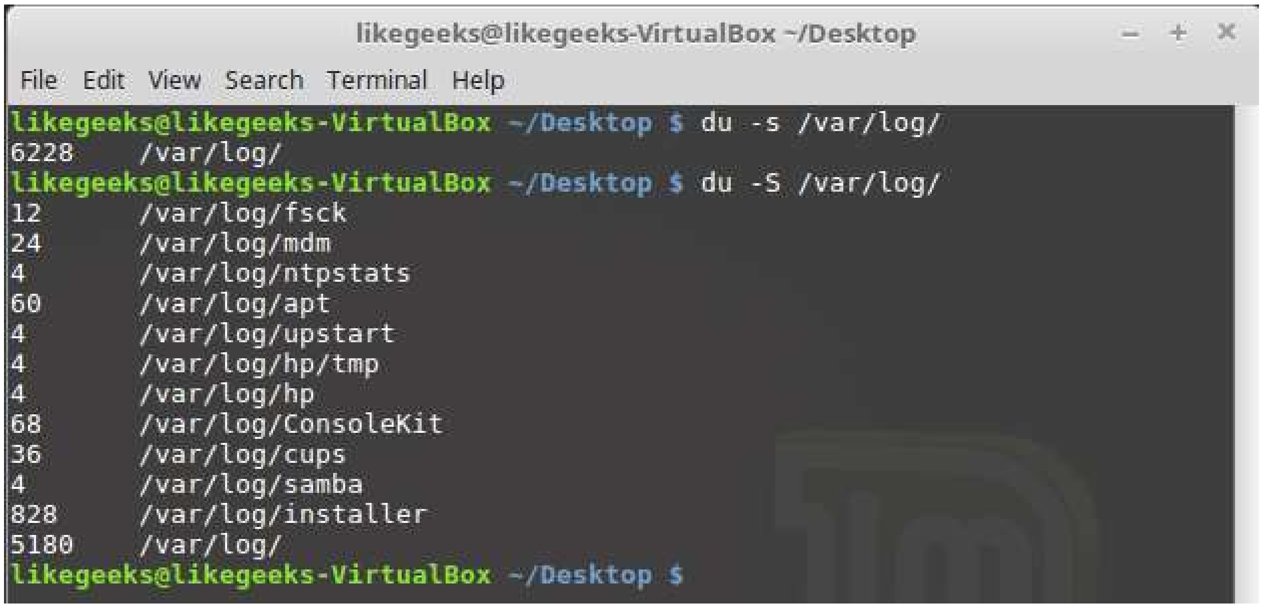
We are going to build a shell script utility that will help you determine the top ten disk space consumers for designated directories

The du command displays the disk usage for individual files and directories. The -s option lets you summarize totals at the directory level I recommend you to review my post about du command and all other **Main Linux Commands**. This comes in handy when calculating the total disk space used by an individual user

```
$ du -s /var/log/
```

The listing quickly becomes too detailed. The -S (capital S) option works better for our purposes here, providing a total for each directory and subdirectory individually.

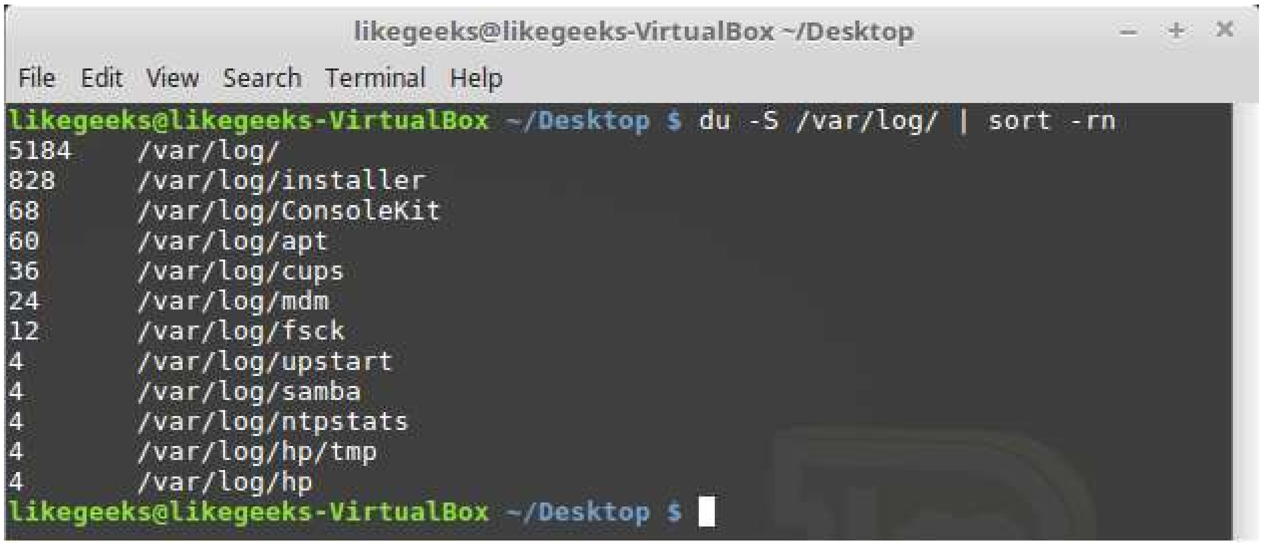
```
$ du -S /var/log/
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop $ du -s /var/log/
6228    /var/log/
likegeeks@likegeeks-VirtualBox ~/Desktop $ du -S /var/log/
12      /var/log/fsck
24      /var/log/mdm
4       /var/log/ntpstats
60      /var/log/apt
4       /var/log/upstart
4       /var/log/hp/tmp
4       /var/log/hp
68      /var/log/ConsoleKit
36      /var/log/cups
4       /var/log/samba
828     /var/log/installer
5180    /var/log/
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Because we are interested in the directories consuming the biggest chunks of disk space, the sort command is used on the listing produced by du

```
$ du -S /var/log/ | sort -rn
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop $ du -S /var/log/ | sort -rn
5184    /var/log/
828     /var/log/installer
68      /var/log/ConsoleKit
60      /var/log/apt
36      /var/log/cups
24      /var/log/mdm
12      /var/log/fsck
4       /var/log/upstart
4       /var/log/samba
4       /var/log/ntpstats
4       /var/log/hp/tmp
4       /var/log/hp
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

The -n option allows you to sort numerically. The -r option lists the largest numbers first (reverse order). This is perfect for finding the largest disk consumers

The sed command brings more clarity to this listing. To focus on the top ten disk space consumers, when line 11 is reached, sed is set to delete the rest of the listing. The next step is to add a line number for each line in the listing. To get those line numbers on the same line as the disk space text, combine the text lines using the N command. The sed commands will look like this

```
sed '{11,$D; =}' |
sed 'N; s/\n/ /' |
```

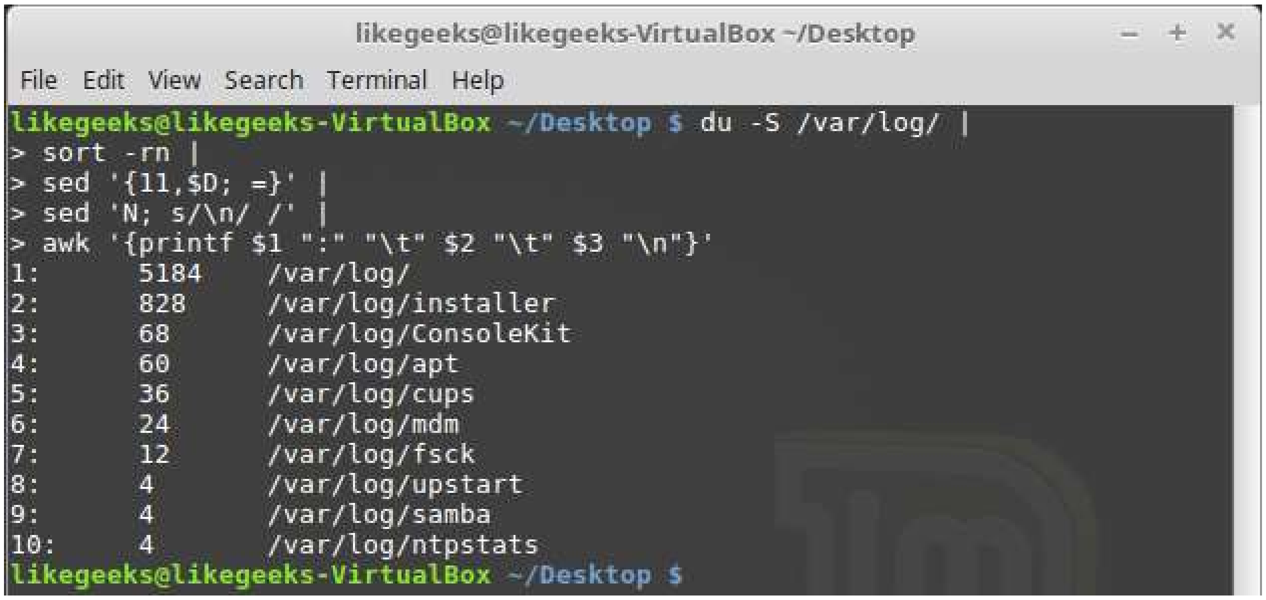
you can review my post about **sed command**

Then we can clean the output using the awk command. The output from the sed command is piped into the awk command and printed using the printf function

```
awk '{printf $1 ":" "\t" $2 "\t" $3 "\n"}'
```

After the line number, a colon (:) is added, and tab (\t) characters are put between the individual fields for each text line's output row.

```
$ du -S /var/log/ |
sort -rn |
sed '{11,$D; =}' |
sed 'N; s/\n/ /' |
awk '{printf $1 ":" "\t" $2 "\t" $3 "\n"}'
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ du -S /var/log/ |
> sort -rn |
> sed '{11,$D; =}' |
> sed 'N; s/\n/ /' |
> awk '{printf $1 ":" "\t" $2 "\t" $3 "\n"}'
1:      5184    /var/log/
2:      828     /var/log/installer
3:      68      /var/log/ConsoleKit
4:      60      /var/log/apt
5:      36      /var/log/cups
6:      24      /var/log/mdm
7:      12      /var/log/fsck
8:      4       /var/log/upstart
9:      4       /var/log/samba
10:     4       /var/log/ntpstats
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Now we understand how the command work let's write the script

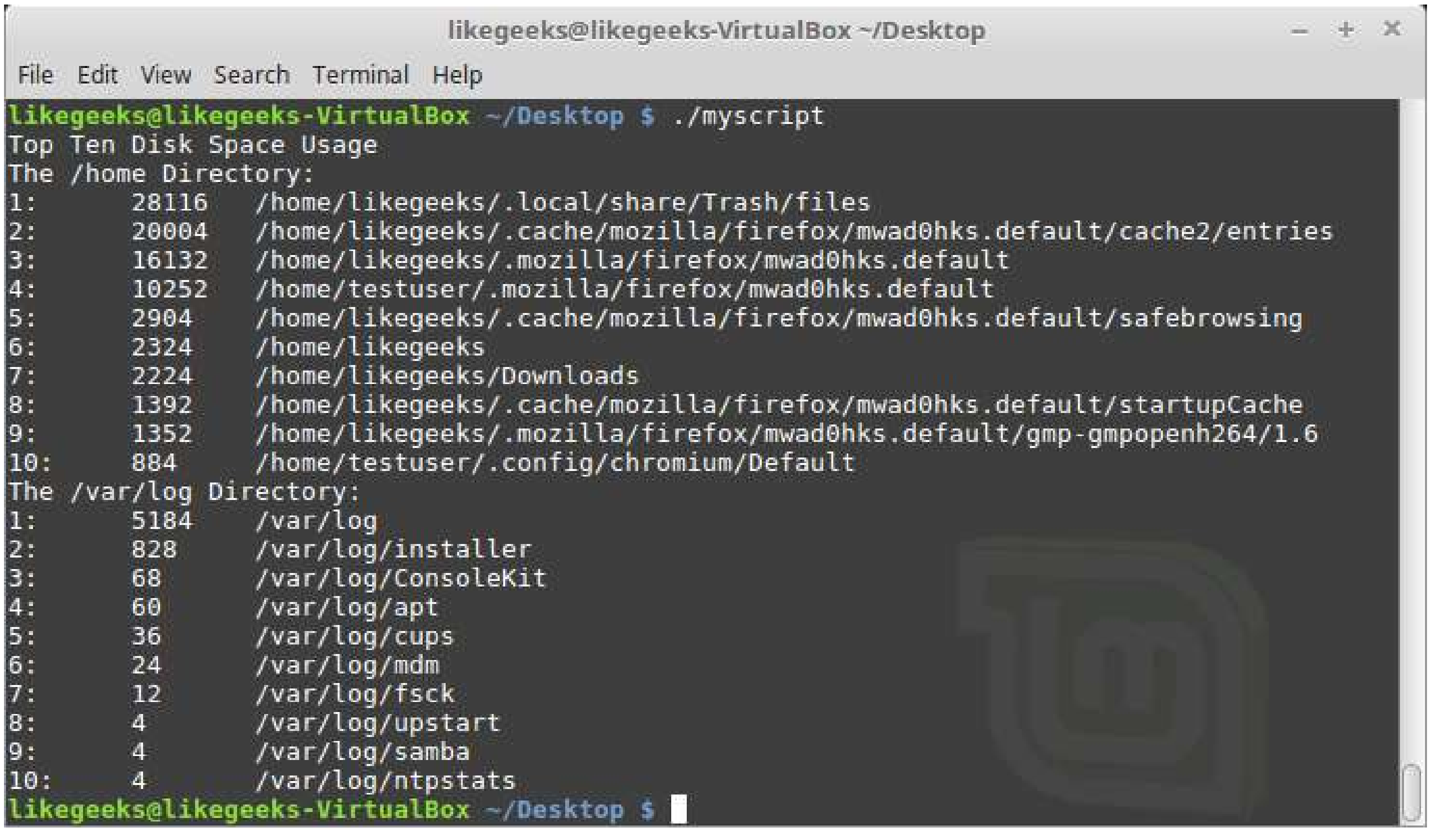
To be productive, the script creates a report for multiple directories. We will make a variable to accomplish this say MY_DIRECTORIES. For our purposes here, the variable is set to just two directories

MY_DIRECTORIES="/home /var/log"

We will make a for loop to perform the du command on each directory listed in the variable. Each time the for loop iterates through the list of values in the variable MY_DIRECTORIES

So the shell script will look like this

```
#!/bin/bash
MY_DIRECTORIES="/home /var/log"
echo "Top Ten Disk Space Usage"
for DIR in $MY_DIRECTORIES
do
echo "The $DIR Directory:"
du -S $DIR 2>/dev/null |
sort -rn |
sed '{11,$D; =}' |
sed 'N; s/\n/ /' |
awk '{printf $1 ":" "\t" $2 "\t" $3 "\n"}'
done
exit
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Top Ten Disk Space Usage
The /home Directory:
1:      28116   /home/likegeeks/.local/share/Trash/files
2:      20004   /home/likegeeks/.cache/mozilla/firefox/mwad0hks.default/cache2/entries
3:      16132   /home/likegeeks/.mozilla/firefox/mwad0hks.default
4:      10252   /home/testuser/.mozilla/firefox/mwad0hks.default
5:      2904    /home/likegeeks/.cache/mozilla/firefox/mwad0hks.default/safebrowsing
6:      2324    /home/likegeeks
7:      2224    /home/likegeeks/Downloads
8:      1392    /home/likegeeks/.cache/mozilla/firefox/mwad0hks.default/startupCache
9:      1352    /home/likegeeks/.mozilla/firefox/mwad0hks.default/gmp-gmpopenh264/1.6
10:     884     /home/testuser/.config/chromium/Default
The /var/log Directory:
1:      5184    /var/log
2:      828     /var/log/installer
3:      68      /var/log/ConsoleKit
4:      60      /var/log/apt
5:      36      /var/log/cups
6:      24      /var/log/mdm
7:      12      /var/log/fsck
8:      4       /var/log/upstart
9:      4       /var/log/samba
10:     4       /var/log/ntpstats
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Good!! The report shows the disk consumption for both directories in a beautifully formatted report

You can filter files so instead of calculating the consumption of all files you can calculate specific extension *.log or whatever, just change the file globing

One thing I have to mention here in production systems you can't rely on disk space report instead, consider setting disk quotas. If the quota package is installed

Quota package is specialized for that but here we are learning how sell scripts work when things mix together.

I hope you understand how shell scripts work and how commands work together

Again the shell scripts we've introduced here is for showing how shell scripting work, there are a ton of ways to implement any task in Linux

My post is finished for now. I tried to reduce the code length and make everything simple as possible hope you like it

Thank you.

41

Admin

https://likegeeks.com

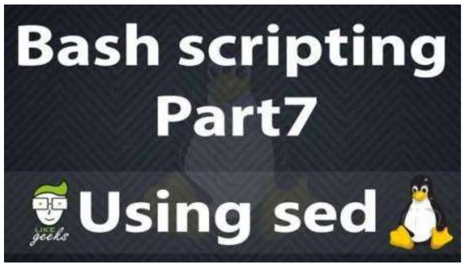
RELATED ARTICLES

LINUX

Regex tutorial for Linux

February 23, 2017 admin

In order to successfully working with the Linux sed editor and the awk command in your shell scripts you has to understand regular expressions or in short regex and to be accurate in our case it is bash regex, since there are many engines for regex you can use and we here in this regex […]



LINUX

31+ Examples for sed Linux command in text manipulation

February 19, 2017 admin

On the previous post we've talked about bash functions and how to use it from the command line and we've seen some other cool stuff I recommend you to review it, Today we will talk about a very useful tool for string manipulation called sed, sed

LINUX

Bash Script Step By Step, You will love it

February 7, 2017 admin

Today we are going to talk about bash script or shell scripting actually, they are called shell scripts in general but we are going to call them bash scripts because we are going to use bash among the other Linux shells. There are zsh, tcsh , ksh and other shells, you can review the basic […]