Lab - Linux Fedora BASH Scripting Part 2

In lab 2, we take what we have learned with writing a BASH script and expand upon it. When it comes to server administration especially we Linux, using a script can automate tasks ran on a regular basis.

Just a quick word about BASH scripting. The chances of someone coming up with an idea for a BASH script that has not already been written is slim to none. Any script you or I would like to have in our script arsenal can be easily found on the Internet. You may be saying, "that's great, but I want to learn how to write scripts." The best way to learn to do scripting is to investigate and examine good working script examples.

You can also study a shell script language such as sh which is the scripting language BASH uses. Dry stuff but if you dream in zeroes and ones and find binary jokes humorous, it could be a valid option.

A lot of scripts are posted in the wild that don't work but again if the script is one you want, you can investigate and research the issue, and in the process, you learn how to write BASH scripts.

BASH uses a scripting language, not a programming language (it does not compile) none the less, it can instruct the computer to perform some high-level functions much the same way a high-level programming language works.

Checking Server Utilization

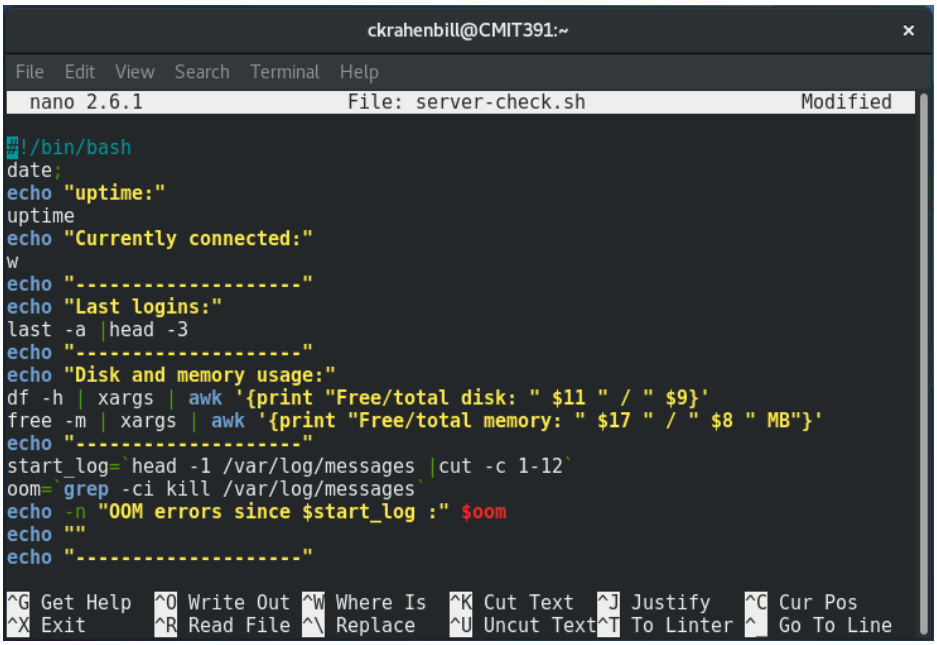
In Windows, we have the task manager and other GUI friendly tools to help monitor are unitization of resources. In Linux, we have a BASH file. This is a long one, so I would copy and paste the contents into the nana text file.

Open a nano text file and name it something clever (nano server-check.sh)

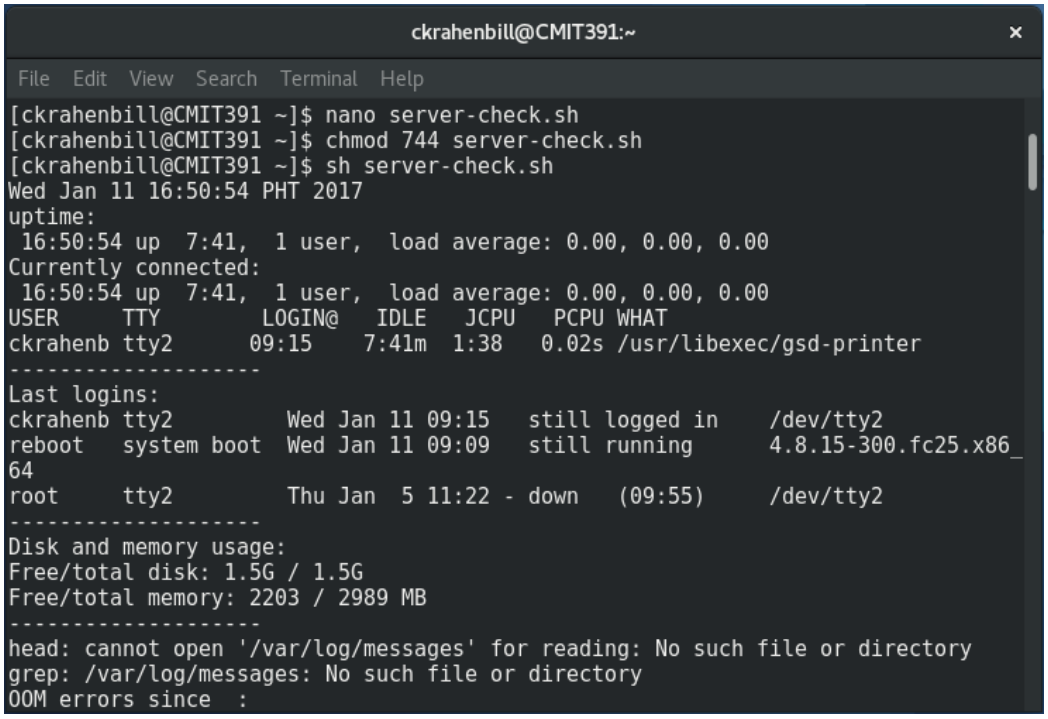
Copy and paste everything in grey into the text file.

```
#!/bin/bash
date;
echo "uptime:"
uptime
echo "Currently connected:"
w
echo "--------------------"
echo "Last logins:"
last -a |head -3
echo "--------------------"
echo "Disk and memory usage:"
df -h | xargs | awk '{print "Free/total disk: " $11 " / " $9}'
free -m | xargs | awk '{print "Free/total memory: " $17 " / " $8 " MB"}'
```

```
echo "--------------------"
start_log=`head -1 /var/log/messages |cut -c 1-12`
oom=`grep -ci kill /var/log/messages`
echo -n "OOM errors since $start_log :" $oom
echo ""
echo "--------------------"
echo "Utilization and most expensive processes:"
top -b |head -3
echo
top -b |head -10 |tail -4
echo "--------------------"
echo "Open TCP ports:"
nmap -p- -T4 127.0.0.1
echo "--------------------"
echo "Current connections:"
ss -s
echo "--------------------"
echo "processes:"
ps auxf --width=200
echo "--------------------"
echo "vmstat:"
vmstat 1 5
```



- Ctrl+x to exit, 'Y' to save, enter to close.
- Make the file executable.
- Run the script.



Great script! All we had to do was build a text file and copy and paste the contents; done like a real world, working server administrator. No need to waste a lot of time for no reason every time we need a script. Google has been my job saver for years, let it be yours.

Check Disk Space with an Email Alert

This next script is something of a job saver. It checks a disk partition and alerts you if the disk is getting close to maxing out.

```
MAX=95
EMAIL=USER@domain.com
PART=sda1
USE=`df -h |grep $PART | awk '{ print $5 }' | cut -d'%' -f1`
if [ $USE -gt $MAX ]; then
echo "Percent used: $USE" | mail -s "Running out of disk space"
$EMAIL
fi
```

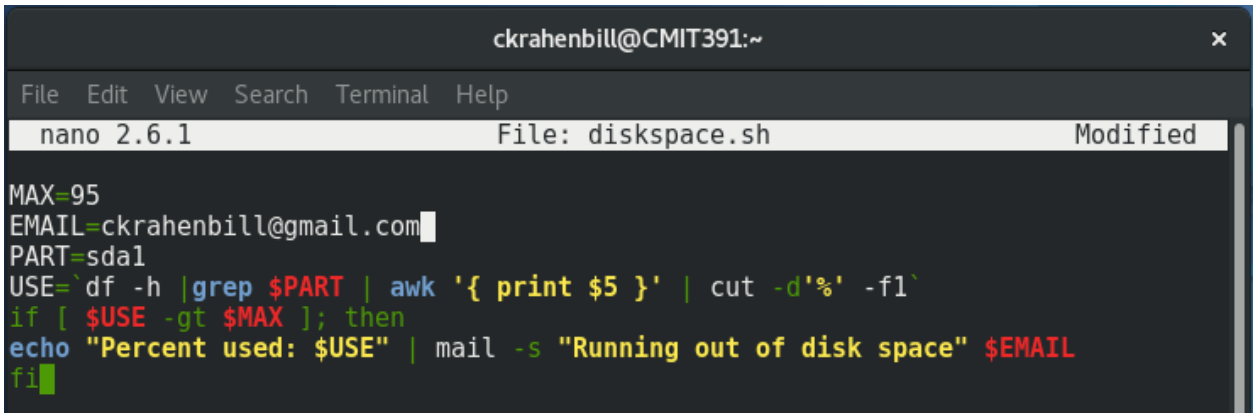
We can read the script and understand with it trying to accomplish. The MAX=95 is the amount of disk usage before we are altered. You can change the amount to anything you want. The next line is the email address. We need to change this to a live email address.

The third line is the Linux partition we what to monitor.

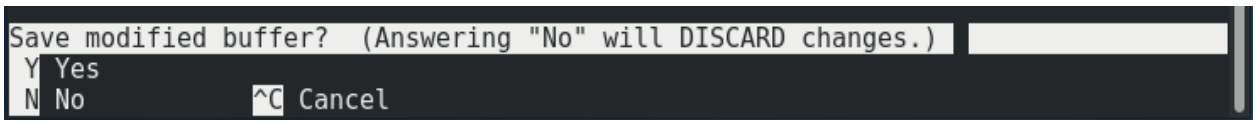
The fourth line uses the grep command to query the partition and return the results.

1. Create a text file using nano. Name the file something clever.
2. Copy and paste the contents of the script into your newly created text.
3. Save the file.
4. Make the file executable.
5. Run the script.

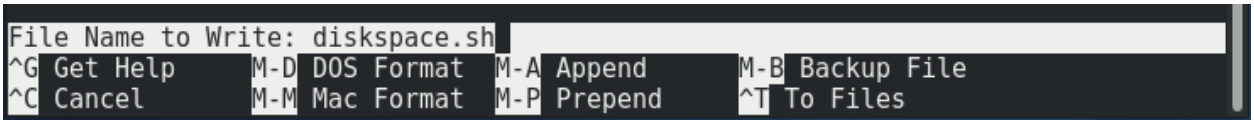
Build the file, paste the contents.



Save the script.



Write the contents to the file.



Run the script.

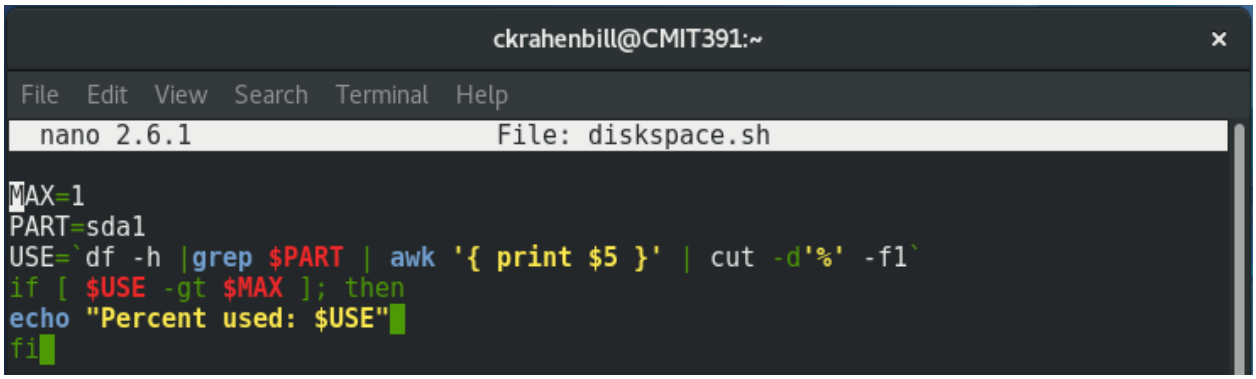
Unless the disk space for your partition is maxed out at 95%, there is nothing for the disk to do. Unless you have a mail server running SMTP, the script has no way to send an email when I run the script, the first time it does nothing because my disk is not maxed out at 95%.

I edit the script and set the max value to 1%. This time it wants to send me an email warning and echo the amount of disk space is available, but it can't without knowing the email server.

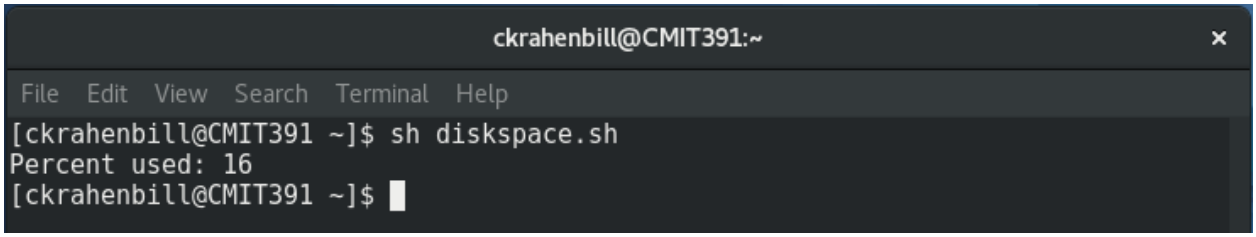
Solution:

Reduce the MAX= from 95 to 1 and remove EMAIL= line. Remove

| mail -s "Running out of disk space" $EMAIL variable at the end of the last line and you end up with this:



Now when I run the script, it tells me how much disk space usage there is.



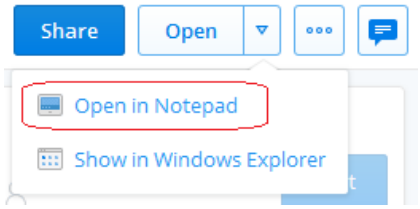
I took script I had found, modified it as needed and now I have a way to check my disk usage.

Here's are some useful admin scripts that can be run from a menu.

This is a very large script but to does 12 different but useful administrative task. Open the link to this text file, copy the text into a nano text named admin-scripts.sh, save the file and run the scripts.

Access the script file here: **Admin-script.txt**

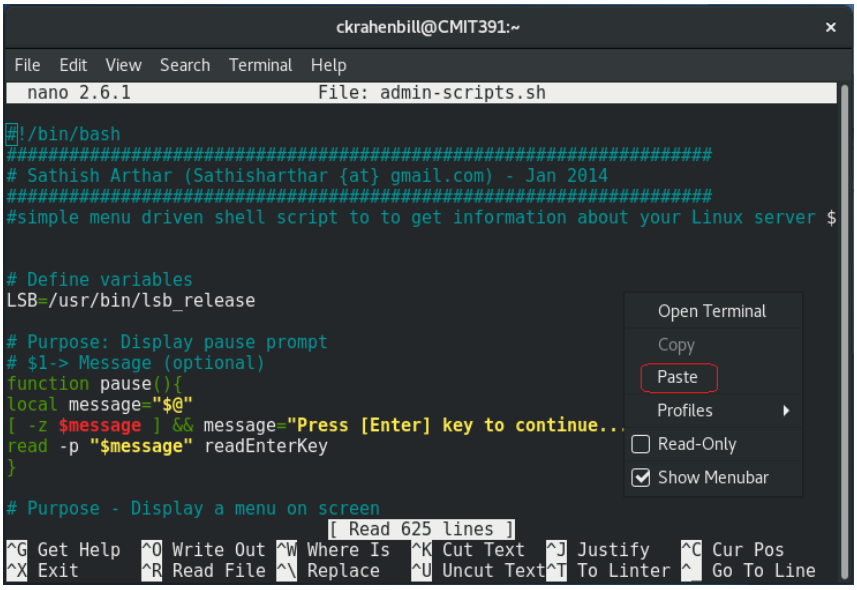
Use the open option and open using Notepad.



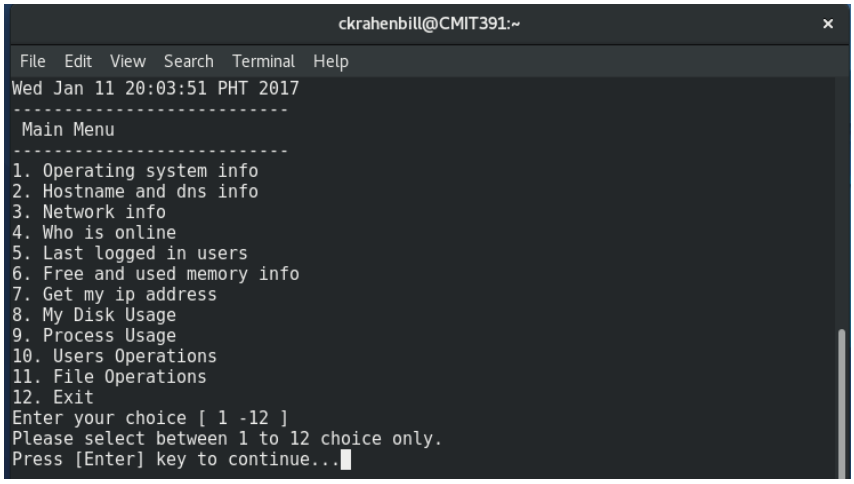
1. With the file open is notepad, place your cursor in the text pane, do a crtl+a and finally a ctrl+c.

2. Place your cursor in the nano text editor for the file you created named admin-scripts.sh.

3. Right-click and select paste from the context menu.



Run the script.



Runs through menu items and checks out your system.

End of the lab!