Using Shell Scripting to Automate Linux System Maintenance Tasks

What is a shell script?

In few words, a shell script is nothing more and nothing less than a program that is executed step by step by a shell, which is another program that provides an interface layer between the Linux kernel and the end user.

By default, the shell used for user accounts in RHEL 7 is bash (/bin/bash). If you want a detailed description and some historical background, you can refer to [this Wikipedia article](https://en.wikipedia.org/wiki/Bash_%28Unix_shell%29).

To find out more about the enormous set of features provided by this shell, you may want to check out its man page, which is downloaded in in PDF format at ([Bash Commands](https://www.tecmint.com/wp-content/pdf/bash.pdf)). Other than that, it is assumed that you are familiar with Linux commands (if not, I strongly advise you to go through [A Guide from Newbies to SysAdmin](https://www.tecmint.com/60-commands-of-linux-a-guide-from-newbies-to-system-administrator/) article in Tecmint.com before proceeding). Now let’s get started.

Writing a script to display system information

For our convenience, let’s create a directory to store our shell scripts:

# mkdir scripts

# cd scripts

And open a new text file named system\_info.sh with your preferred text editor. We will begin by inserting a few comments at the top and some commands afterwards:

#!/bin/bash

# Sample script written for Part 4 of the RHCE series

# This script will return the following set of system information:

# -Hostname information:

echo -e "\e[31;43m\*\*\*\*\* HOSTNAME INFORMATION \*\*\*\*\*\e[0m"

hostnamectl

echo ""

# -File system disk space usage:

echo -e "\e[31;43m\*\*\*\*\* FILE SYSTEM DISK SPACE USAGE \*\*\*\*\*\e[0m"

df -h

echo ""

# -Free and used memory in the system:

echo -e "\e[31;43m \*\*\*\*\* FREE AND USED MEMORY \*\*\*\*\*\e[0m"

free

echo ""

# -System uptime and load:

echo -e "\e[31;43m\*\*\*\*\* SYSTEM UPTIME AND LOAD \*\*\*\*\*\e[0m"

uptime

echo ""

# -Logged-in users:

echo -e "\e[31;43m\*\*\*\*\* CURRENTLY LOGGED-IN USERS \*\*\*\*\*\e[0m"

who

echo ""

# -Top 5 processes as far as memory usage is concerned

echo -e "\e[31;43m\*\*\*\*\* TOP 5 MEMORY-CONSUMING PROCESSES \*\*\*\*\*\e[0m"

ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6

echo ""

echo -e "\e[1;32mDone.\e[0m"

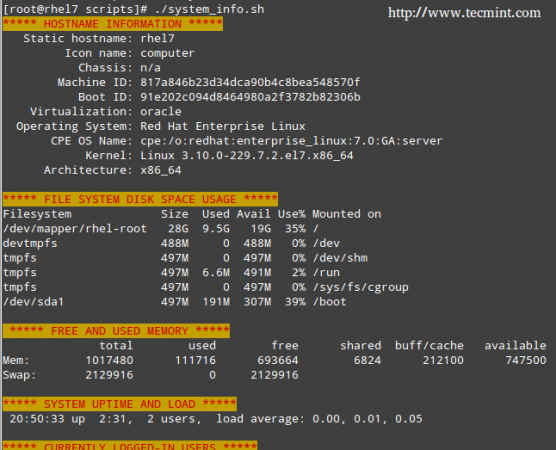
Next, give the script execute permissions:

# chmod +x system\_info.sh

and run it:

./system\_info.sh

Note that the headers of each section are shown in color for better visualization:

[[](https://www.tecmint.com/wp-content/uploads/2015/08/Server-Monitoring-Shell-Script.png)](https://www.tecmint.com/wp-content/uploads/2015/08/Server-Monitoring-Shell-Script.png)

Server Monitoring Shell Script

That functionality is provided by this command:

echo -e "\e[COLOR1;COLOR2m<YOUR TEXT HERE>\e[0m"

Where COLOR1 and COLOR2 are the foreground and background colors, respectively (more info and options are explained in this entry from the [Arch Linux Wiki](https://wiki.archlinux.org/index.php/Color_Bash_Prompt)) and <YOUR TEXT HERE> is the string that you want to show in color.

Automating Tasks

The tasks that you may need to automate may vary from case to case. Thus, we cannot possibly cover all of the possible scenarios in a single article, but we will present three classic tasks that can be automated using shell scripting:

1) update the local file database, 2) find (and alternatively delete) files with 777 permissions, and 3) alert when filesystem usage surpasses a defined limit.

Let’s create a file named auto\_tasks.sh in our scripts directory with the following content:

#!/bin/bash

# Sample script to automate tasks:

# -Update local file database:

echo -e "\e[4;32mUPDATING LOCAL FILE DATABASE\e[0m"

updatedb

if [ $? == 0 ]; then

echo "The local file database was updated correctly."

else

echo "The local file database was not updated correctly."

fi

echo ""

# -Find and / or delete files with 777 permissions.

echo -e "\e[4;32mLOOKING FOR FILES WITH 777 PERMISSIONS\e[0m"

# Enable either option (comment out the other line), but not both.

# Option 1: Delete files without prompting for confirmation. Assumes GNU version of find.

#find -type f -perm 0777 -delete

# Option 2: Ask for confirmation before deleting files. More portable across systems.

find -type f -perm 0777 -exec rm -i {} +;

echo ""

# -Alert when file system usage surpasses a defined limit

echo -e "\e[4;32mCHECKING FILE SYSTEM USAGE\e[0m"

THRESHOLD=30

while read line; do

# This variable stores the file system path as a string

FILESYSTEM=$(echo $line | awk '{print $1}')

# This variable stores the use percentage (XX%)

PERCENTAGE=$(echo $line | awk '{print $5}')

# Use percentage without the % sign.

USAGE=${PERCENTAGE%?}

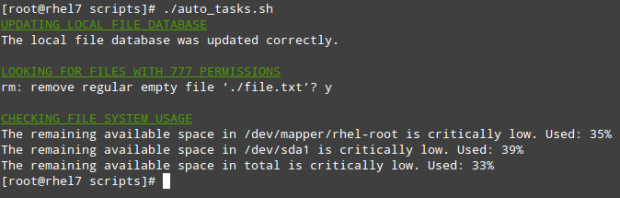
if [ $USAGE -gt $THRESHOLD ]; then

echo "The remaining available space in $FILESYSTEM is critically low. Used: $PERCENTAGE"

fi

done < <(df -h --total | grep -vi filesystem)

Please note that there is a space between the two < signs in the last line of the script.

[[](https://www.tecmint.com/wp-content/uploads/2015/08/Shell-Script-to-Find-777-Permissions.png)](https://www.tecmint.com/wp-content/uploads/2015/08/Shell-Script-to-Find-777-Permissions.png)

Shell Script to Find 777 Permissions

Using Cron

To take efficiency one step further, you will not want to sit in front of your computer and run those scripts manually. Rather, you will use cron to schedule those tasks to run on a periodic basis and sends the results to a predefined list of recipients via email or save them to a file that can be viewed using a web browser.

The following script (filesystem\_usage.sh) will run the well-known df -h command, format the output into a HTML table and save it in the report.html file:

#!/bin/bash

# Sample script to demonstrate the creation of an HTML report using shell scripting

# Web directory

WEB\_DIR=/var/www/html

# A little CSS and table layout to make the report look a little nicer

echo "<HTML>

<HEAD>

<style>

.titulo{font-size: 1em; color: white; background:#0863CE; padding: 0.1em 0.2em;}

table

{

border-collapse:collapse;

}

table, td, th

{

border:1px solid black;

}

</style>

<meta http-equiv='Content-Type' content='text/html; charset=UTF-8' />

</HEAD>

<BODY>" > $WEB\_DIR/report.html

# View hostname and insert it at the top of the html body

HOST=$(hostname)

echo "Filesystem usage for host <strong>$HOST</strong><br>

Last updated: <strong>$(date)</strong><br><br>

<table border='1'>

<tr><th class='titulo'>Filesystem</td>

<th class='titulo'>Size</td>

<th class='titulo'>Use %</td>

</tr>" >> $WEB\_DIR/report.html

# Read the output of df -h line by line

while read line; do

echo "<tr><td align='center'>" >> $WEB\_DIR/report.html

echo $line | awk '{print $1}' >> $WEB\_DIR/report.html

echo "</td><td align='center'>" >> $WEB\_DIR/report.html

echo $line | awk '{print $2}' >> $WEB\_DIR/report.html

echo "</td><td align='center'>" >> $WEB\_DIR/report.html

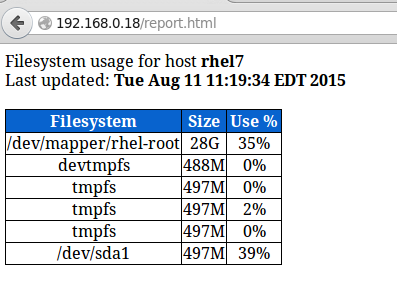
echo $line | awk '{print $5}' >> $WEB\_DIR/report.html

echo "</td></tr>" >> $WEB\_DIR/report.html

done < <(df -h | grep -vi filesystem)

echo "</table></BODY></HTML>" >> $WEB\_DIR/report.html

In our RHEL 7 server (192.168.0.18), this looks as follows:

[[](https://www.tecmint.com/wp-content/uploads/2015/08/Server-Monitoring-Report.png)](https://www.tecmint.com/wp-content/uploads/2015/08/Server-Monitoring-Report.png)

Server Monitoring Report

You can add to that report as much information as you want. To run the script every day at 1:30 pm, add the following crontab entry:

30 13 \* \* \* /root/scripts/filesystem\_usage.sh

Summary

You will most likely think of several other tasks that you want or need to automate; as you can see, using shell scripting will greatly simplify this effort. Feel free to let us know if you find this article helpful and don't hesitate to add your own ideas or comments via the form below.