

Module 3: Scripting with Linux

James Way & Venetia Furtado

Date: 11/17/2024

Aim: Creation and running an executable script in Linux to print useful information about the Linux Kernel.

1. Create an executable script

A bash script was written to perform the following functions:

- i. Display the current user and terminal.
- ii. Display the current running processes.
- iii. Display the current data and time, kernel version.
- iv. Display the kernel dump.

```
#!/bin/bash

echo "---- System Information Script ----"

# Display the current user and terminal
echo "Current User: $(whoami)"
echo "Terminal: $TERM"
echo "Shell: $SHELL"

# Display running processes
echo -e "\nCurrent Running Processes:"
ps aux

# Display current date, time, and kernel version
echo -e "\nCurrent Date and Time:"
date
echo -e "\nKernel Version:"
uname -r

# Display kernel dump
echo -e "\nKernel Dump (dmesg output):"
dmesg | tail -n 20 # Limiting output to the last 20 lines for brevity

echo -e "\n---- End of System Information ----"
```

Figure 1: Snapshot of the bash script.

The command used to make the bash script executable was: `chmod +x terminalInfo.sh`

The outputs have been shown in Figure 2 below.

```
pi@raspberrypi:~/PBK/Module3 $ ./terminalInfo.sh
--- System Information Script ---
Current User: pi
Terminal: xterm-256color
Shell: /bin/bash

Current Running Processes:
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.4  0.8 33720 7972 ?        Ss   19:21   0:03 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    19:21   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   19:21   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   19:21   0:00 [rcu_par_gp]
root         7  0.0  0.0      0     0 ?        I    19:21   0:00 [kworker/u8:0-events_unbound]
root         8  0.0  0.0      0     0 ?        I<   19:21   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    19:21   0:00 [ksoftirqd/0]
root        10  0.0  0.0      0     0 ?        I    19:21   0:00 [rcu_sched]
root        11  0.0  0.0      0     0 ?        I    19:21   0:00 [rcu_bh]
root        12  0.0  0.0      0     0 ?        S    19:21   0:00 [migration/0]
root        13  0.0  0.0      0     0 ?        S    19:21   0:00 [cpuhp/0]
root        14  0.0  0.0      0     0 ?        S    19:21   0:00 [cpuhp/1]
root        15  0.0  0.0      0     0 ?        S    19:21   0:00 [migration/1]
root        16  0.0  0.0      0     0 ?        S    19:21   0:00 [ksoftirqd/1]
root        19  0.0  0.0      0     0 ?        S    19:21   0:00 [cpuhp/2]
root        20  0.0  0.0      0     0 ?        S    19:21   0:00 [migration/2]
root        21  0.0  0.0      0     0 ?        S    19:21   0:00 [ksoftirqd/2]
root        24  0.0  0.0      0     0 ?        S    19:21   0:00 [cpuhp/3]

Current Date and Time:
Fri 15 Nov 2024 07:34:51 PM MST

Kernel Version:
4.19.75-v7+

Kernel Dump (dmesg output):
[ 8.854540] smsc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
[ 8.854821] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 9.721449] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 12.517537] Bluetooth: Core ver 2.22
[ 12.517659] NET: Registered protocol family 31
[ 12.517667] Bluetooth: HCI device and connection manager initialized
[ 12.517989] Bluetooth: HCI socket layer initialized
[ 12.518003] Bluetooth: L2CAP socket layer initialized
[ 12.518045] Bluetooth: SCO socket layer initialized
[ 12.535341] Bluetooth: HCI UART driver ver 2.3
[ 12.535351] Bluetooth: HCI UART protocol H4 registered
[ 12.535420] Bluetooth: HCI UART protocol Three-wire (H5) registered
[ 12.535549] Bluetooth: HCI UART protocol Broadcom registered
[ 12.922376] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[ 12.922383] Bluetooth: BNEP filters: protocol multicast
[ 12.922399] Bluetooth: BNEP socket layer initialized
[ 12.987307] Bluetooth: RFCOMM TTY layer initialized
[ 12.987335] Bluetooth: RFCOMM socket layer initialized
[ 12.987362] Bluetooth: RFCOMM ver 1.11
[ 13.196860] fuse init (API version 7.27)
```

Figure 2: Output of the bash script.

2. Make the bash script run in boot time

To make the script run in boot time, Raspberry Pi's system (the default init system) was used to create a service file that gets executed upon boot.

The systemd service was created: `sudo nano /etc/systemd/system/terminalInfo.service`

To enable the service to run at boot: `sudo systemctl enable terminalInfo.service`

To start the service run: `sudo systemctl start terminalInfo.service`

To check the status of the service: `sudo systemctl status terminalInfo.service` which shows "active".

To reboot system: `sudo reboot now`

To get the syslog run: `sudo cat /var/log/syslog`

The syslog shows that the script runs after the Raspberry Pi is reconnected with the network as specified in the service file.

A copy of the syslog file has been submitted for review.

```
[Unit]
Description=Run terminalInfo at boot
After=network.target

[Service]
ExecStart=/home/pi/PBX/Module3/terminalInfo.sh
Type=oneshot
RemainAfterExit=true

[Install]
WantedBy=multi-user.target
```

Figure 3: Systemd service file.

References:

- [1] [BASH Programming - Introduction HOW-TO](#)
- [2] <https://linuxconfig.org/how-to-autostart-bash-script-on-startup-on-raspberry-pi>