

System and Network Administration - Lab 9 - Systemd

Jaffar Totanji - j.totanji@innopolis.university

Questions to answer:

1. For that, we can use:

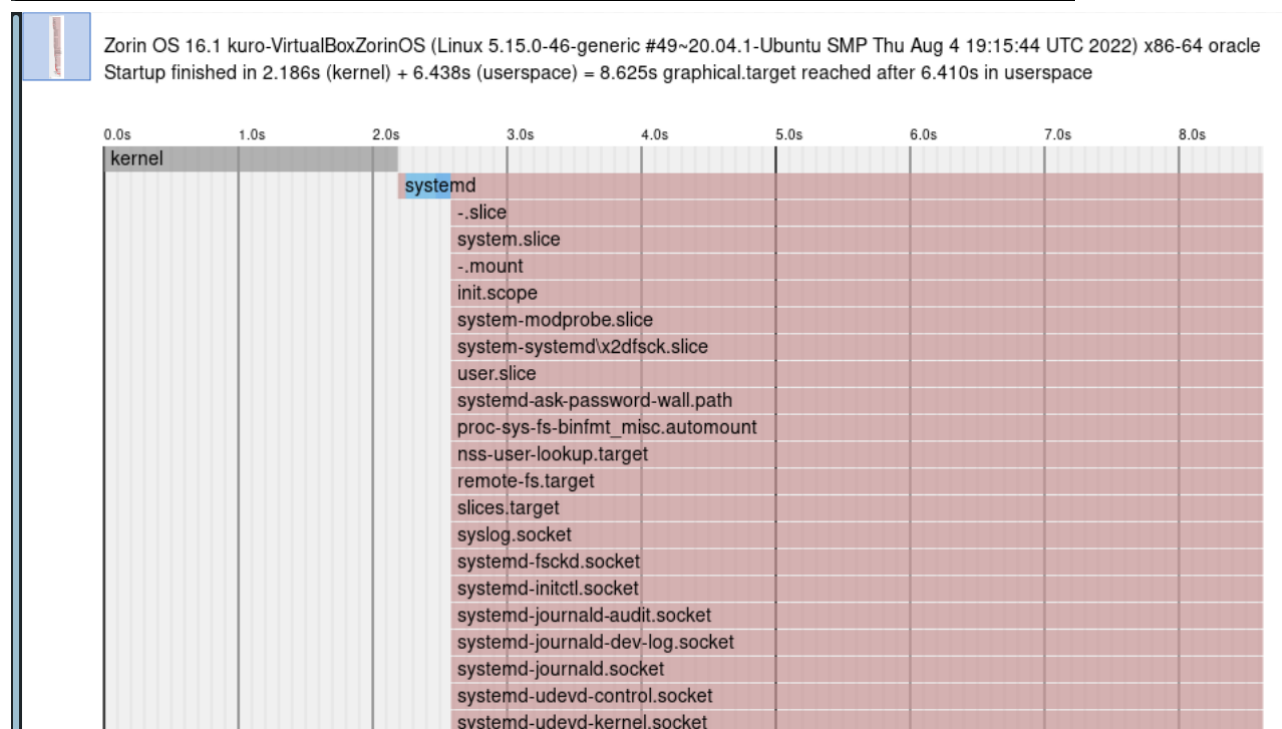
```
systemd-analyze
```

```
kuro@kuro-VirtualBoxZorinOS:~$ systemd-analyze
Startup finished in 2.186s (kernel) + 6.438s (userspace) = 8.625s
graphical.target reached after 6.410s in userspace
kuro@kuro-VirtualBoxZorinOS:~$
```

In order to view the services that have been started and how long it took each of them to initialize, we can use `systemd-analyze` again, but this time with its charting utility `plot`. The resulting chart is huge, so I have only included the top part of it here:

```
systemd-analyze plot
```

```
kuro@kuro-VirtualBoxZorinOS:~$ systemd-analyze plot > ./boot.svg
kuro@kuro-VirtualBoxZorinOS:~$
```



2. We can trace back the `Requires` variable of every service using `systemctl`. Our starting point looking like this:

```
systemctl show -p Requires graphical.target
```

and our chain of services looks like this:

```
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl show -p Requires graphical.target
Requires=multi-user.target
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl show -p Requires multi-user.target
Requires=basic.target
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl show -p Requires sysinit.target
Requires=
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl show -p Wants sysinit.target
Wants=systemd-udevd.service systemd-binfmt.service dev-hugepages.mount sys-kern>
kuro@kuro-VirtualBoxZorinOS:~$
```

We can then use

```
systemctl list-units --type target
```

to view some info about these services as well as others:

```
kuro@kuro-VirtualBoxZorinOS:~$ systemctl list-units --type target
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
basic.target	loaded	active	active	Basic System
cryptsetup.target	loaded	active	active	Local Encrypted Volumes
getty.target	loaded	active	active	Login Prompts
graphical.target	loaded	active	active	Graphical Interface
local-fs-pre.target	loaded	active	active	Local File Systems (Pre)
local-fs.target	loaded	active	active	Local File Systems
multi-user.target	loaded	active	active	Multi-User System
network-online.target	loaded	active	active	Network is Online
network.target	loaded	active	active	Network
nss-lookup.target	loaded	active	active	Host and Network Name Lookups
nss-user-lookup.target	loaded	active	active	User and Group Name Lookups
paths.target	loaded	active	active	Paths
remote-fs.target	loaded	active	active	Remote File Systems
slices.target	loaded	active	active	Slices
sockets.target	loaded	active	active	Sockets
sound.target	loaded	active	active	Sound Card
swap.target	loaded	active	active	Swap
sysinit.target	loaded	active	active	System Initialization
time-set.target	loaded	active	active	System Time Set
time-sync.target	loaded	active	active	System Time Synchronized

- `graphical.target`: `multi-user.target` with GUI.
- `multi-user.target`: All services running, but command-line interface (CLI) only.
- `sysinit.target`: System Initialization.

Let's first start with the meaning of "Wants" and "Requires" from the [man](#) page:

- **Requires:** Configures requirement dependencies on other units. If this unit gets activated, the units listed here will be activated as well. If one of the other units gets deactivated or its activation fails, this unit will be deactivated.
- **Wants:** A weaker version of **Requires**. Units listed in this option will be started if the configuring unit is. However, if the listed units fail to start or cannot be added to the transaction, this has no impact on the validity of the transaction as a whole. This is the recommended way to hook start-up of one unit to the start-up of another unit.

Focusing on the last 2 lines of the description of **Wants**, **sysinit.target** basically initializes our system aka starts it up, and in order for it to do that, it needs to start up lots of other services for the system to be considered running, and if one of them fails, the transaction is still valid as a whole.

3. The first step is to create a script that starts web server on a certain port and gives us the required statistical info. I did that in the form of 2 scripts, one that prints out the statistics **script_1.sh** and another that runs the previous script on a web server **script.sh**:



```

script.sh
1 #!/bin/bash
2
3 printf "System Uptime:\n"
4 uptime -p
5 echo "-----"
6
7 printf "Inode Usage:\n"
8 df -i 2> /dev/null
9 echo "-----|-----"
10
11 printf "Usage Statistics:\n"
12 memory_usage=$(free -m | awk 'NR==2{printf "Memory Usage: %s/%sMB (%.2f%%)\n", $3,$2,$3*100/$2 }')
13 disk_usage=$(df -h 2> /dev/null | awk 'NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n", $3,$2,$5}')
14 cpu_usage=$(top -bn1 | grep load | awk '{printf "CPU Usage: %.2f\n", $(NF-2)}')
15 cur_date=`date +%b_%d_%Y_%H_%M_%S`
16
17 echo "$cur_date: $memory_usage || $disk_usage || $cpu_usage"
18 echo "-----"
19
20 printf "Last 15 lines of /var/log/syslog:\n"
21 tail -15 /var/log/syslog
22 echo "-----"

script_1.sh
1 #!/bin/bash
2
3 while true;
4     do echo -e "HTTP/1.1 200 OK\n\n$(source script_1.sh)" \
5         | nc -l -k -p 8080 -q 1;
6 done
  
```

Then, we need to create a systemd service to accommodate the script at run it at start up and a slice to control resource usage. That is done exactly as described in the lab:

```
Terminal - kuro@kuro-VirtualBoxZorinOS: ~  
File Edit View Terminal Tabs Help  
[Unit]  
Description=Lab 9 SNA  
  
[Service]  
ExecStart=/usr/bin/script.sh  
Slice=testslice.slice  
  
[Install]  
WantedBy=multi-user.target  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
"/lib/systemd/system/shellservice.service" 9L, 129C      6,21      All
```

```

Terminal - kuro@kuro-VirtualBoxZorinOS: ~
File Edit View Terminal Tabs Help

[Unit]
Description=Custom systemd slice for SNA lab 9
Before=slices.target

[Slice]
MemoryAccounting=true
CPUAccounting=true
MemoryMax=256M
CPUQuota=10%

"/etc/systemd/system/testslice.slice" 9L, 153C 9,12 All

```

Finally, the service in action:

```

← → ↺ localhost:8080 ☆

System Uptime:
up 1 hour, 57 minutes

Inode Usage:
-----
Filesystem      Inodes   IUsed   IFree  IUse% Mounted on
udev            235897   456    235351    1% /dev
tmpfs           251326   701    250625    1% /run
/dev/sda2       1605632 277968 1327664   18% /
tmpfs           251326   1    251325    1% /dev/shm
tmpfs           251326   2    251324    1% /run/lock
tmpfs           251326  19    251307    1% /sys/fs/cgroup
/dev/sda1        0        0        0 /boot/efi
tmpfs           251326   56    251270    1% /run/user/1000

Usage Statistics:
0%0000, 28_2022_15_59_17: Memory Usage: 814/1963MB (41,47%) || Disk Usage: 8/24GB (35%) || CPU Usage: 0,19

Last 15 lines of /var/log/syslog:
Oct 28 15:59:15 kuro-VirtualBoxZorinOS script.sh[5638]: Sec-Fetch-User: ?1
Oct 28 15:59:15 kuro-VirtualBoxZorinOS script.sh[5638]: #015
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: GET / HTTP/1.1
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Host: localhost:8080
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Accept-Language: en-US,en;q=0.5
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Accept-Encoding: gzip, deflate, br
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Connection: keep-alive
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Upgrade-Insecure-Requests: 1
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Sec-Fetch-Dest: document
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Sec-Fetch-Mode: navigate
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Sec-Fetch-Site: none
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: Sec-Fetch-User: ?1
Oct 28 15:59:16 kuro-VirtualBoxZorinOS script.sh[5655]: #015

```

4. For that, we first need to create a service that updates the package source list:

`update_list` script simply performs the update:

[illegible]

To then run the service periodically, we need to create a corresponding timer for the service that will trigger after boot up and on a daily thereafter:

[illegible]

Finally, we need to do `systemctl daemon-reload` to make `systemd` aware of our new files, and then we need to enable and start the timer which will trigger our service as scheduled:

```
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl daemon-reload
[sudo] password for kuro:
Sorry, try again.
[sudo] password for kuro:
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl enable updatescript.timer
kuro@kuro-VirtualBoxZorinOS:~$ sudo systemctl start updatescript.timer
kuro@kuro-VirtualBoxZorinOS:~$ systemctl status updatescript.timer updatescript.service
● updatescript.timer - Run 5 minutes after booting and on a daily basis thereafter
   Loaded: loaded (/lib/systemd/system/updatescript.timer; enabled; vendor pre>
   Active: active (waiting) since Fri 2022-10-28 18:03:25 MSK; 5min ago
   Trigger: Sat 2022-10-29 18:03:26 MSK; 23h left
   Triggers: ● updatescript.service

OCT 28 18:03:25 kuro-VirtualBoxZorinOS systemd[1]: Started Run 5 minutes after >

● updatescript.service - My custom updater
   Loaded: loaded (/lib/systemd/system/updatescript.service; enabled; vendor >
   Active: inactive (dead) since Fri 2022-10-28 18:03:58 MSK; 4min 47s ago
   TriggeredBy: ● updatescript.timer
   Main PID: 3831 (code=exited, status=0/SUCCESS)

OCT 28 18:03:57 kuro-VirtualBoxZorinOS update_list.sh[3833]: Reading package li>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: Building dependenc>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: Reading state info>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: 139 packages can b>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: W: Failed to fetch>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: W: Failed to fetch>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: W: Failed to fetch>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: W: Failed to fetch>
OCT 28 18:03:58 kuro-VirtualBoxZorinOS update_list.sh[3833]: W: Some index file>
kuro@kuro-VirtualBoxZorinOS:~$
```

End of Exercises