

# Systemd

??

Linux ????????????? SysV Init Script? **Systemd** ?????????? CentOS 7 ??????????

???? SysV Init ??????????????????

?????

- [How to enable rc.local shell script on systemd while booting Linux system](#)
- [RHEL] [Overview of systemd for RHEL 7](#)
- [RHEL] [How to configure a command, script, or daemon to run after boot has finished in RHEL 7, 8](#)
- [How to Find Systemd or Any Other init System in Linux \(debugpoint.com\)](#)

??? Linux?

- CentOS 7+
- Ubuntu 16.04+

????????

- [Supervisor](#)  
??????? Ubuntu 9.10? Mac OS X (10.4/10.5/10.6)? Solaris (10 for Intel) ? FreeBSD 6.1????????? Python 2.4????? Python 3?

?????

- /etc/systemd/system ????????????
- /lib/systemd/system ??????????????

## How to determine

```
↪ ps --no-headers -o comm 1  
systemd
```

?????

## /etc/systemd/system/backup.service

```
[Unit]
Description=Backup daemon

[Service]
Type=simple
ExecStart=/path/to/backup

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

### TIP:

“ multi-user.target ??? Run Level 3

???????? http://0pointer.de/blog/projects/systemd-for-admins-3.html

??????????

- [How to create a systemd service in Linux \(linuxhandbook.com\)](http://linuxhandbook.com)
- [How to Create a Systemd Service Unit in Linux \(tecmin.com\)](http://tecmin.com)

## /etc/systemd/system/freepbx.service

```
[Unit]
Description=Freepbx
After=mariadb.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/sbin/fwconsole start
ExecStop=/usr/sbin/fwconsole stop

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

??????????

```
systemctl enable freepbx
```

systemctl ??

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl cat freepbx.service
```

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl show freepbx.service
```

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl edit freepbx.service
```

????

????

```
# Reload Systemd
```

```
systemctl daemon-reload
```

```
# [ ] [ ] [ ] [ ]
```

```
systemctl start <service-name>
```

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl status <service-name>
```

```
systemctl is-active <service-name>
```

```
systemctl is-enabled <service-name>
```

```
# [ ] [ ] [ ] [ ]
```

```
systemctl stop <service-name>
```

```
# [ ] [ ]:[ ] [ ] [ ] [ ]
```

```
systemctl enable <service-name>
```

```
# [ ] [ ]:[ ] [ ] [ ] [ ]
```

```
systemctl disable <service-name>
```

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl list-unit-files --type=service --state=enabled
```

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
systemctl cat <service-name>
```

??????

```
# View status of all services and units
```

```
systemctl
```

```
systemctl | grep ssh
```

```
# list active services
```

```
systemctl list-units --type=service
```

```
systemctl --type service
```

```
systemctl -t service
```

```
# List all the RUNNING systemd services
```

```
systemctl list-units --type=service --state=running
```

```
# List all LOADED systemd services including the inactive ones
```

```
systemctl list-units --type=service
```

```
systemctl --type service
```

```
# List all the INACTIVE systemd services
```

```
systemctl list-units --all --type=service --state=inactive
```

```
# List all the INSTALLED systemd services
```

```
systemctl list-unit-files --type=service
```

```
# List all systemd services that will be run at each boot automatically
```

```
systemctl list-unit-files --type=service --state=enabled
```

?????

```
# Halt the system
```

```
systemctl halt
```

```
# Poeroff the system
```

```
systemctl poweroff
```

```
# Reboot the system
```

```
systemctl reboot
```

```
# Reboot the system into UEFI settings
```

```
systemctl reboot --firmware-setup
```

????????

```
# Find which target unit is used by default
```

```
# GUI mode: graphical.target
```

```
# Text mode: multi-user.target
```

```
systemctl get-default
```

```
ls -l /etc/systemd/system/default.target
```

```
# To change boot target to the text mode
```

```
sudo systemctl set-default multi-user.target
```

```
# To change boot target to the GUI mode
```

```
sudo systemctl set-default graphical.target
```

```
# Optional: Listing all systemd targets
```

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

```
# To immediately switch to the GUI login
```

```
systemctl isolate graphical.target
```

## Journalctl

??????

```
# View the log of the specified service
```

```
journalctl -u <service-name>
```

```
journalctl -f -u <service-name>      # -f View live updates
```

```
journalctl -e -u <service-name>      # -e Jump to the end page of the log
```

```
journalctl -n 50 -u <service-name>    # -n Show the most recent n number of log lines
```

```
# [ ]/[ ]
```

```
# <[ ]> <[ ]>
```

```
# [ ]
```

```
journalctl --no-pager --since today \
```

```
--grep 'fail|error|fatal' --output json|jq '._EXE' | \
```

```
sort | uniq -c | sort --numeric --reverse --key 1
```

```
# view journal entries for time zones
```

```
journalctl --utc
```

```
# view only errors, warnings, etc in journal logs
```

```
# Error codes
```

```
# 0: emergency
```

```
# 1: alerts
```

```
# 2: critical
```

```
# 3: errors
```

```
# 4: warning
```

```
# 5: notice
```

```
# 6: info
```

```
# 7: debug
```

```
journalctl -p 0
```

```
# When you specify the error code, it shows all messages from that code and above.
```

```
# For example, if you specify the below command, it shows all messages with priority 2, 1 and 0
```

```
journalctl -p 2
```

```
# view journal logs for a specific boot
```

```
journalctl --list-boots
```

```
# To view a specific boot number you the first number or the boot ID as below.
```

```
journalctl -b -45
```

```
journalctl -b 8bab42c7e82440f886a3f041a7c95b98
```

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
# -p 3 [ ] [ ] error [ ] [ ] [ ] [ ]
```

```
journalctl -xb -p 3
```

```
# view journal logs for a specific time, date duration
```

```
journalctl --since "2020-12-04 06:00:00"
```

```
journalctl --since "2020-12-03" --until "2020-12-05 03:00:00"
```

```
journalctl --since yesterday
```

```
journalctl --since 09:00 --until "1 hour ago"
```

```
journalctl --since "1 hour ago"
```

```
journalctl --since "1 hour ago" -u cron
```

```
# see Kernel specific journal logs
journalctl -k

# see journal logs for a service name
journalctl -u NetworkManager.service

# By PID
journalctl _PID=1111
journalctl -o verbose _PID=1111

# If you do not know the service name, you can use the below
# command to list the systemd services in your system.
systemctl list-units --type=service

# view journal logs for a user, group
id -u debugpoint
journalctl _UID=1000 --since today

# view journal logs for an executable
journalctl /usr/bin/gnome-shell --since today

# Check the disk usage
journalctl --disk-usage

# Set the log clearance
sudo journalctl --vacuum-time=2d
sudo journalctl --vacuum-size=500M
```

## Application firewalls

An application firewall, unlike a gateway (router) or system level firewall, is meant to limit the networking of a single application. It can be used to prevent a compromised service from seeing into the local network, prevent programs from calling home, plug metadata leaks, or more tightly control a program's network access.

The `systemd` firewall directives is built on Linux kernel features. The required Kernel features might not be enabled in your specific environment (especially when using a custom kernel or container). Testing is key, as it is with any network filter and security solution. You should always test to verify that your firewall set up blocks and allows the traffic you specify.

- [systemd application firewalls by example](#)

# Run a custom script

## After mounting NFS

Listing mount systemd units

```
sudo systemctl list-units --type=mount
```

/root/bin/nfs-optimiation.sh:

```
#!/bin/bash
device_number=$(stat -c '%d' /cbz_efs/)
((major = (device_number & 0xFFF00) >> 8))
((minor = (device_number & 0xFF) | ((device_number >> 12) & 0xFFF00)))
_dev="/sys/class/bdi/$major:$minor/read_ahead_kb"
echo "DRVICE: $_dev"
echo "CURRENT VALUE: $(cat $_dev)"
echo "$0 called after mount /cbz_efs/"
echo 15000 > "$_dev"
```

Creating a new service unit

```
sudo chmod +x -v /root/bin/nfs-optimiation.sh

# Create a new systemd unit name after-cbz_efs-mount
sudo systemctl edit --force --full after-cbz_efs-mount
```

unit: after-cbz\_efs-mount

```
[Unit]
Description=Script to run after fstab mount for /cbz_efs/
Requires=cbz_efs.mount
After=cbz_efs.mount
RequiresMountsFor=/cbz_efs

[Service]
ExecStart=/root/bin/nfs-optimiation.sh

[Install]
WantedBy=cbz_efs.mount
```



## Activating the unit

```
sudo systemctl daemon-reload
sudo systemctl enable after-cbz_efs-mount
sudo systemctl start after-cbz_efs-mount
sudo systemctl status after-cbz_efs-mount
```

## After starting network

Create: `/etc/systemd/system/multi-user.target.wants/connection.service`

```
[Unit]
Description = making network connection up
After = network.target

[Service]
ExecStart = /root/scripts/conup.sh

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

Script: `/root/scripts/conup.sh`

```
#!/bin/bash
nmcli connection up enp0s3
```

## Activating the service

```
sudo systemctl daemon-reload
sudo systemctl enable connection.service
sudo systemctl start connection.service
sudo systemctl status connection.service
```

??????

## coredumpctl

```
# [ ] core dump
coredumpctl
```

```
# [program] core dump
coredumpctl dump <program-name>
```

```
# [PID]
coredumpctl dump _PID=XXX
```

```
# [core dump]
coredumpctl gdb <PID>
```

```
# [core dump files]
/var/lib/systemd/coredump
```

---

Revision #44

Created 13 August 2020 13:01:54 by Admin

Updated 23 May 2025 16:11:20 by Admin