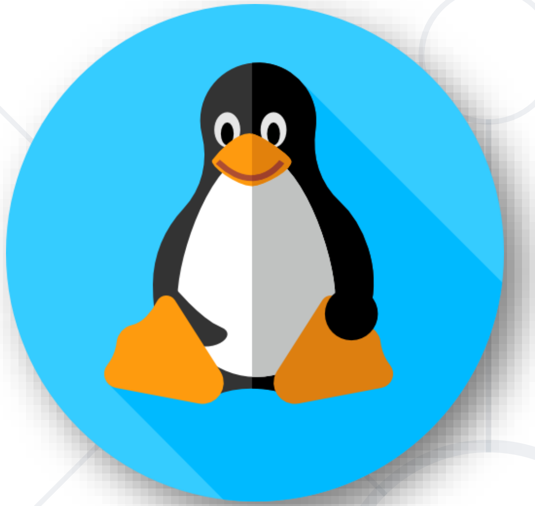
System Startup and Process Management

Boot Process. System Initialization. Process Monitoring and Management



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

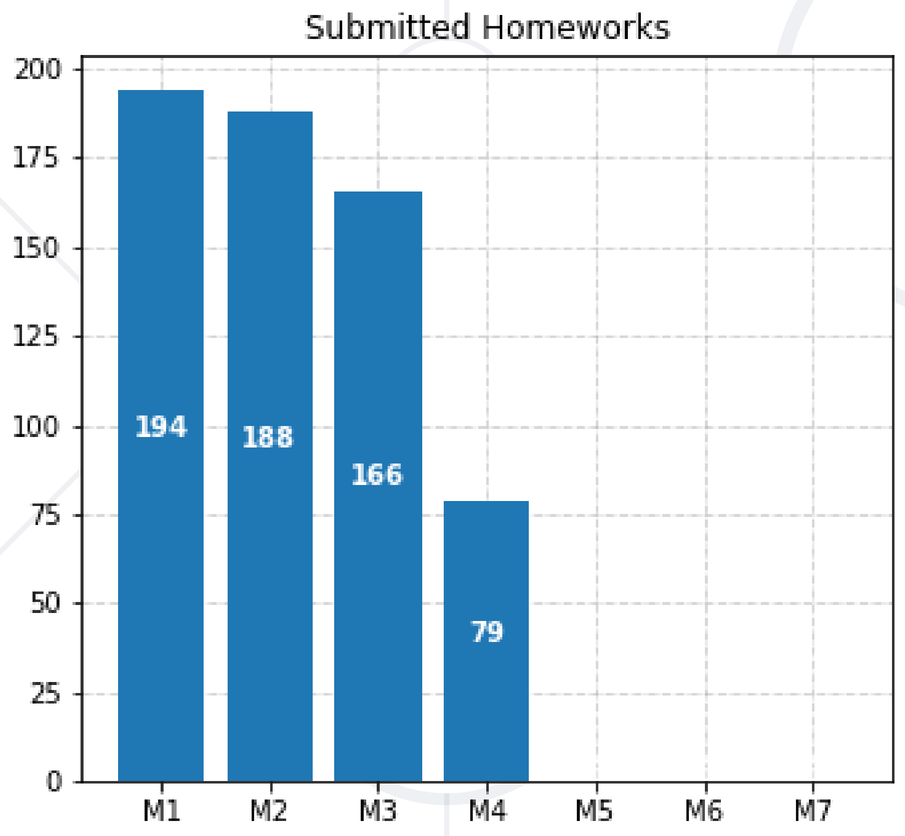
Have a Question?

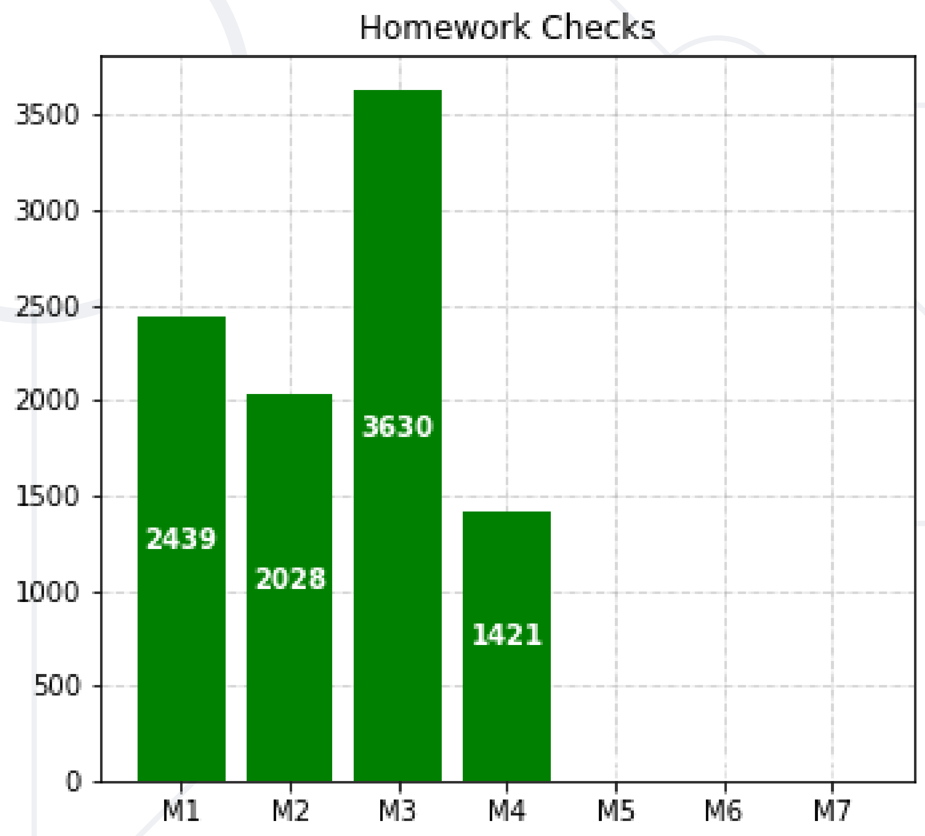
Software University

sli.do

#LSA

Homework Progress





Solutions for **M4** can be submitted until **23:59:59** on **03.04.2025**

Solutions for **M4** can be submitted until **23:59:59** on **10.04.2025**

Quick Overview

Previous Module (M4)

What We Covered

1. Network Basics
2. Services Control
3. Software Management
4. Network Services

A Note

Additional Information

About Virtual Networks

- **NAT** (Adapter | Network)
  - Isolates the VM/VMs from the **outside world** and at the same time **provides access to Internet**
- **Bridged** (External)
  - **Attached** to the real (physical) **network adapter** on the **host**
  - Exposes the VM to the real (physical) network
- **Internal Network** (Host Only)
  - **Virtual network** for interconnecting VMs
  - **Isolated** from the outside world

About Configuration Files and Changes

- A few facts about configuration files
  - Usually, they are **text files** and reside in **`/etc`** directly or indirectly
  - Contain many **comments** with **default** or **possible** values
  - Sample configuration files are (usually) available at **`/usr/share/doc/<package>/`**
  - May have **extension** like **.conf**, **.yaml**, **.sh**, and etc.

About Configuration Files and Changes

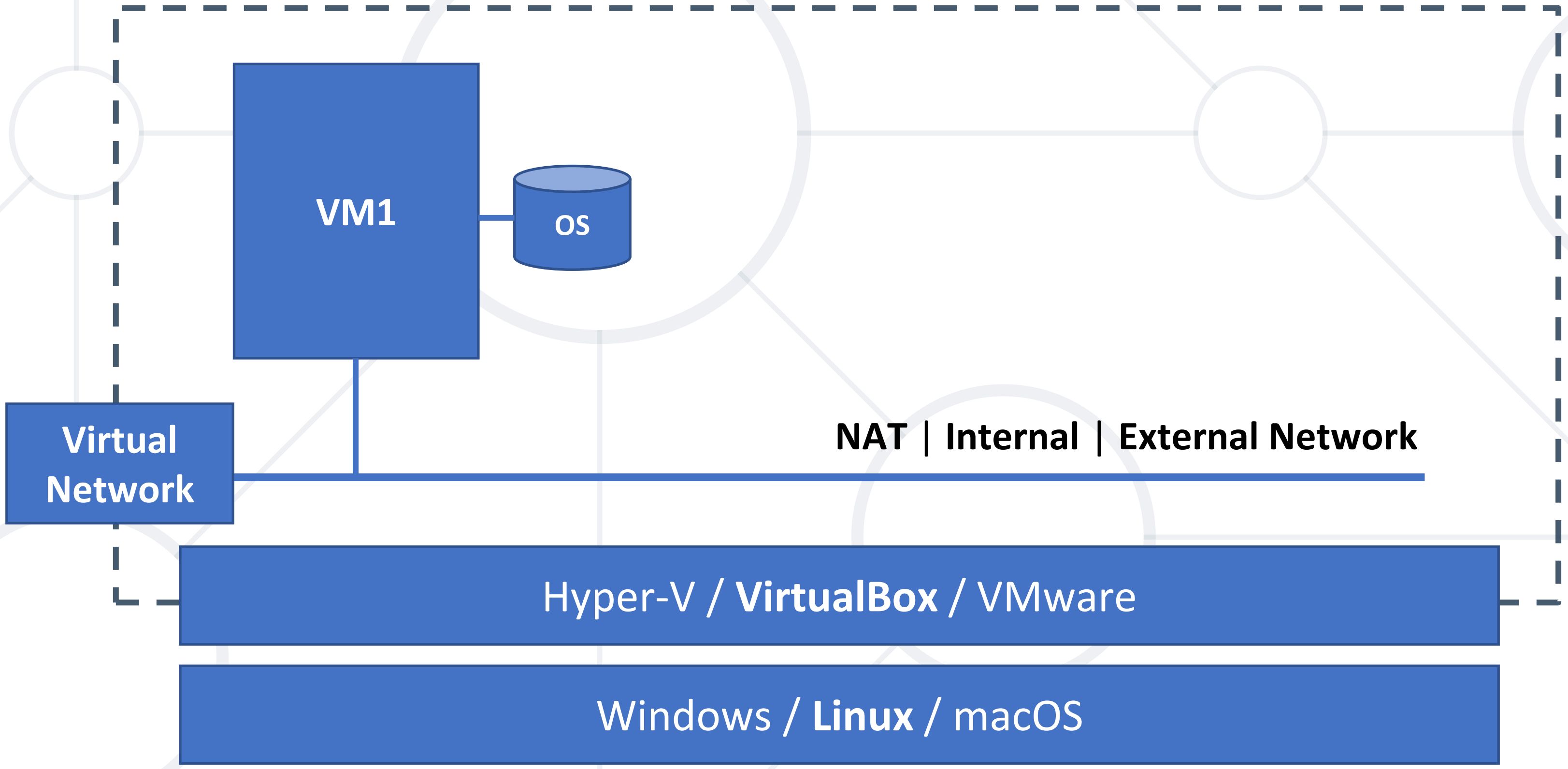
- Handle them with care
  - Before making any changes **make a copy** of the original file
  - Good practice is to add an **additional extension**: **`file.conf`** -> **`file.conf.bak`**
  - Compare **spaces**, **line ends**, and **structure** with the original
  - Before restarting the service, test the configuration (for example **dhcpd -t**)

This Module (M5)

Topics and Lab Infrastructure

Table of Contents

Lab Infrastructure


VM1
OS
Virtual Network
NAT | Internal | External Network
Hyper-V / **VirtualBox** / VMware
Windows / **Linux** / macOS

BIOS
UEFI

System Startup & Boot Managers

Boot Process

BIOS vs UEFI

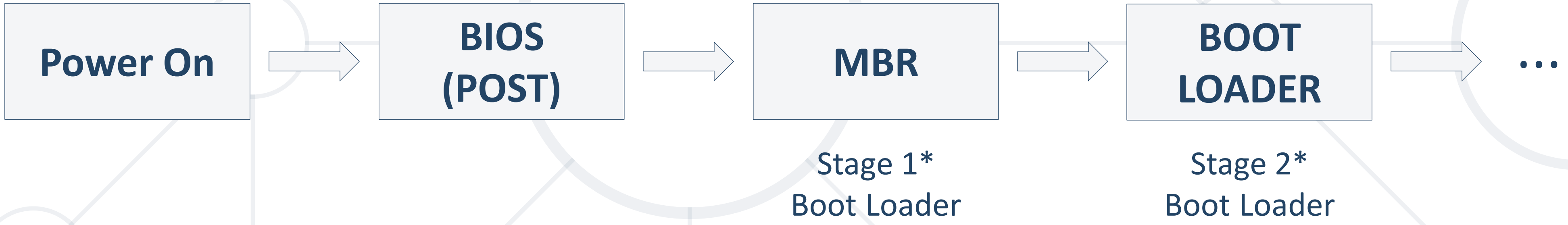
- Basic Input / Output System
- Dates to early 80s
- Operates in 16-bit mode
- Slower boot process
- Offers text user interface
- Settings stored in CMOS RAM chip
- Supports only MBR

- Unified Extensible Firmware Interface
- Dates to early 2000s
- Operates in 32-bit / 64-bit mode
- Faster boot process
- Offers graphical user interface
- Settings stored on ESP partition
- Supports both MBR and GPT

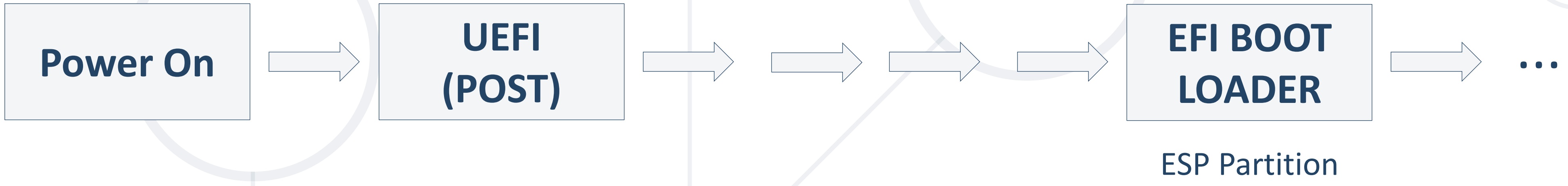
Boot Process (Generalized)

- BIOS based systems

POST = Power On Self Test



- UEFI based systems



** There is stage 1.5 between those two*

Boot Loaders

Overview. GRUB2

Boot Loaders

- **LILO** (Linux Loader)
  - Old, very rare these days
- **GRUB** (GNU Grand Unified Boot Loader)
  - Configurable, default on most distributions
- **SYSLINUX**
  - Used for installation, live, or rescue media
- **Loadlin** (Load Linux)
  - Used for booting from non-Linux OS

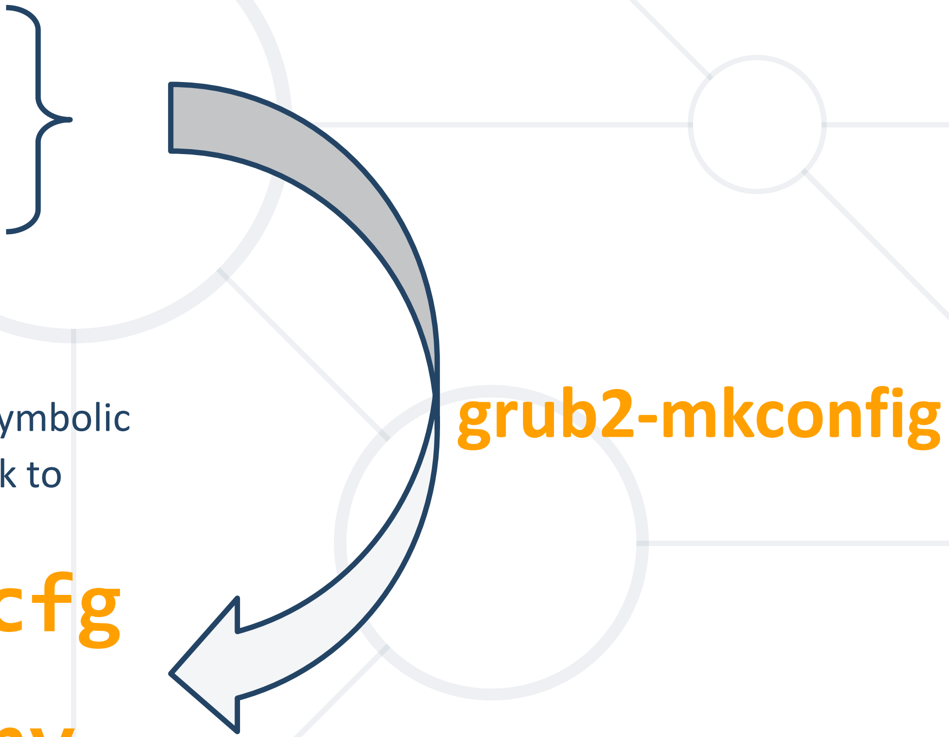
GRUB2

- **Grand Unified Boot Loader**
- Highly configurable
- Supports **many operating systems**
- **Interactive** and **default** OS selection
- **Temporal** configuration change
- **Integrated command prompt**

GRUB2 (BIOS)

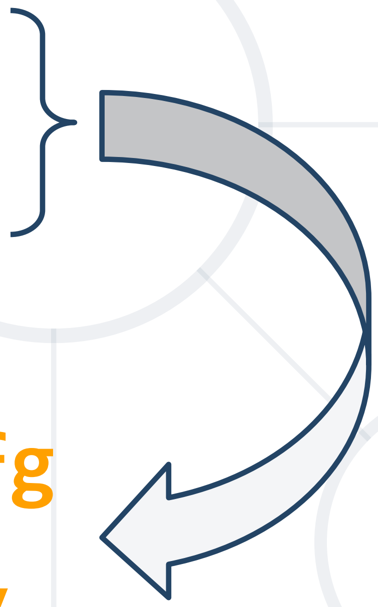
- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
  - `/etc/grub2.cfg`
- Resulting files
  - `/boot/grub2/grub.cfg`
  - `/boot/grub2/grubenv`

It is a symbolic link to



GRUB2 (BIOS)

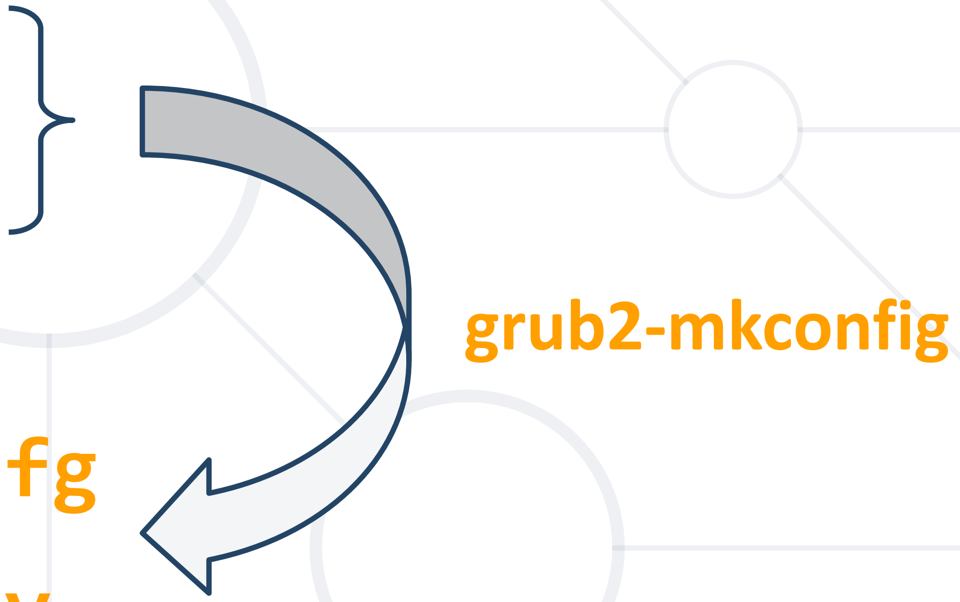
- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
- Resulting files
  - `/boot/grub/grub.cfg`
  - `/boot/grub/grubenv`



grub-mkconfig
update-grub

GRUB2 (BIOS)

- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
- Resulting files
  - `/boot/grub2/grub.cfg`
  - `/boot/grub2/grubenv`



GRUB2 (UEFI)

- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
  - `/etc/grub2.cfg` & `/etc/grub2-efi.cfg`

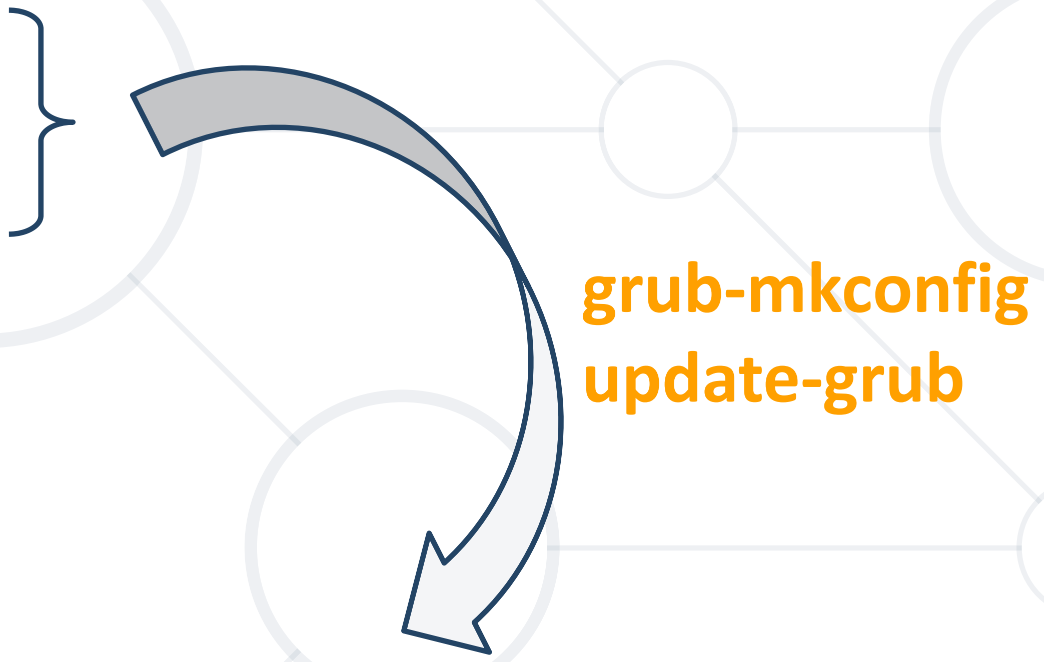
grub2-mkconfig

Are symbolic links to

- Resulting files
  - `/boot/grub2/grub.cfg`
  - `/boot/grub2/grubenv`
  - `/boot/efi/EFI/`*`<distribution-name>`*`/grub.cfg`

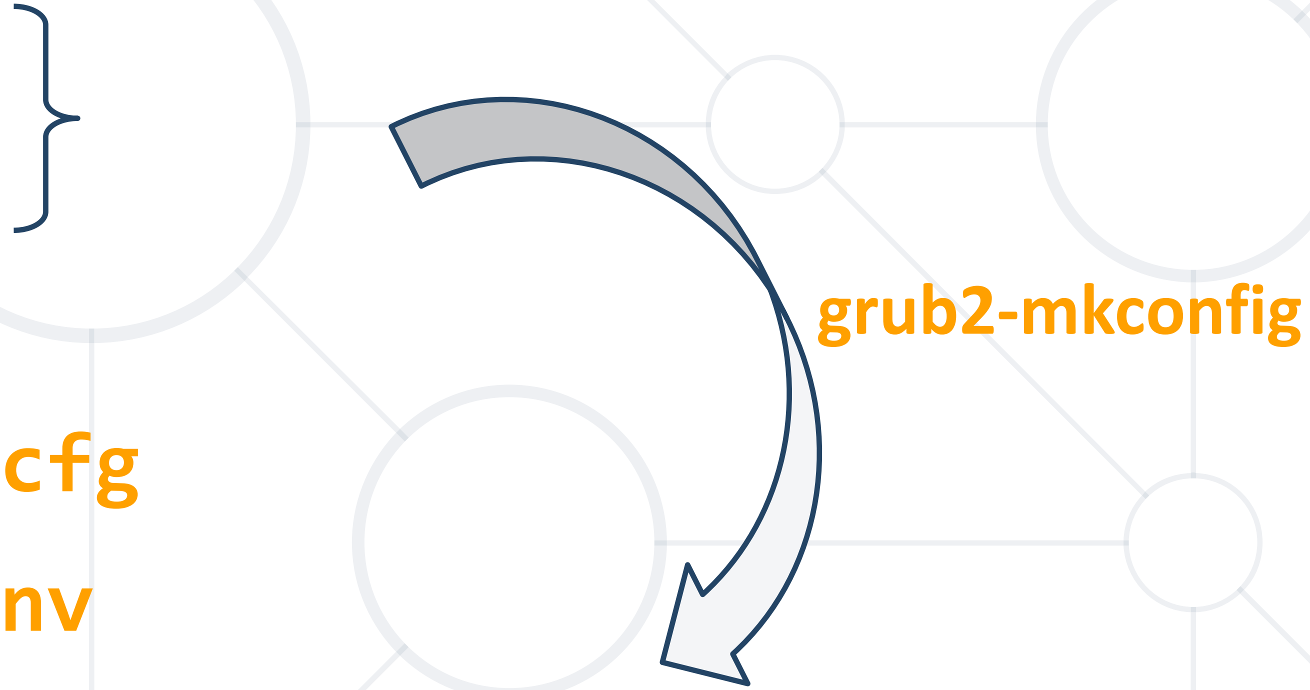
GRUB2 (UEFI)

- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
- Resulting files
  - `/boot/grub/grub.cfg`
  - `/boot/grub/grubenv`
  - `/boot/efi/EFI/`*`<distribution-name>`*`/grub.cfg`



GRUB2 (UEFI)

- Configuration files
  - `/etc/default/grub`
  - `/etc/grub.d/*`
- Resulting files
  - `/boot/grub2/grub.cfg`
  - `/boot/grub2/grubenv`
  - `/boot/efi/EFI/<distribution-name>/grub.cfg`



Default Settings

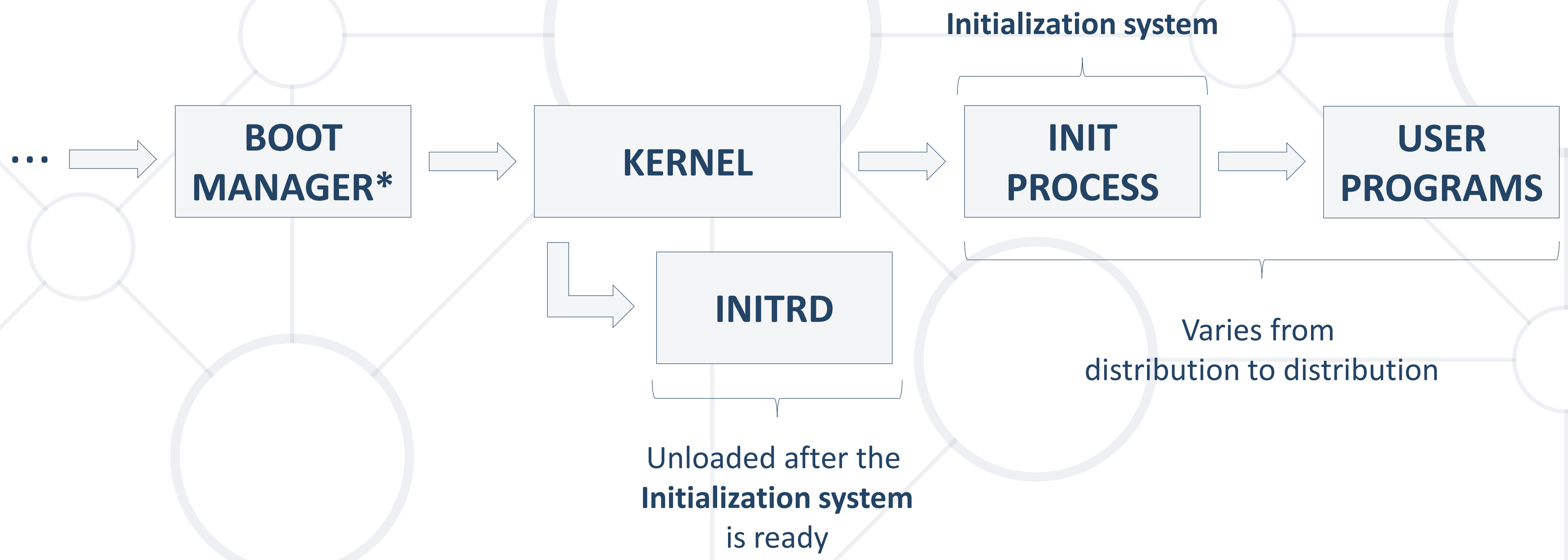
```
[user@host ~]$ cat -n /etc/default/grub
 1 GRUB_TIMEOUT=5
 2 GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g'
/etc/system-release)"
 3 GRUB_DEFAULT=saved
 4 GRUB_DISABLE_SUBMENU=true
 5 GRUB_TERMINAL_OUTPUT="console"
 6 GRUB_CMDLINE_LINUX="rd.lvm.lv=cl/root
rd.lvm.lv=cl/swap rhgb quiet"
 7 GRUB_DISABLE_RECOVERY="true"
[user@host ~]$
```

Startup Sequence

From Boot to User Space

Next Steps

- From Boot to a Running System



Initialization system
...
BOOT MANAGER*
KERNEL
INIT PROCESS
USER PROGRAMS
INITRD
Varies from distribution to distribution
Unloaded after the **Initialization system** is ready

Initialization Methods

- **SysVinit** (System V init or just **SysV**)*
  - **Sequential**. Utilizes **runlevels**
  - Considered **obsolete in a way**
  - In use by **PCLinuxOS**, **Slackware Linux**, **Devuan GNU+Linux**, etc.
- **Upstart**
  - **Event driven**. Created by Canonical
  - Considered obsolete
  - In use by **UBports**. Used by Ubuntu 6.10 - 14.04, Fedora 9 - 15, …

** V stands for the number 5 and not for the letter v*

Initialization Methods

- OpenRC
  - Dependency-based initialization system
  - **Portable** between **Linux**, **FreeBSD**, and **NetBSD**
  - In use by **Gentoo Linux**, **Alpine Linux**, **GhostBSD**, etc.
- systemd
  - Dependency-based init system with **aggressive parallelization**
  - Offers some level of compatibility to SysVinit
  - In use by **CentOS**, **openSUSE**, **Ubuntu**, **Fedora**, **Debian**, **Arch**, etc.

systemd Components

- **systemd**
  - Systems and services manager
- **systemctl**
  - State inspection and state controlling utility
- **systemd-analyze**
  - Utility for performance statistics inspection

systemd Components

- journald
  - Logging component by default. Binary files. Replaceable
- consoled
  - User console daemon
- networkd
  - Provides networking support
- logind
  - Supports X display managers, user logins, and so on

dmesg

- Purpose
  - Print or control kernel ring buffer (RAM area)
- Syntax

```
dmesg [options]
```

- Example

```
# Display all messages in human readable format
[root@host ~]# dmesg -H
# Warning kernel related messages in readable form
[root@host ~]# dmesg -H -l warn -f kern
```

efibootmgr

- Purpose
  - Manipulate the EFI Boot Manager
- Syntax

```
efibootmgr [options]
```

- Example

```
# Display detailed configuration information
[root@host ~]# efibootmgr -v
# Change boot order
[root@host ~]# efibootmgr -o 0003,0001,0000,0002
```

Practice

systemd

systemd

Units. Targets. Dependencies

Units

- Units are the new **init scripts** *(sort of)*
- Unit is a file that **represents system component configuration**
- Naming convention
  - `"unit name"."unit type"`
- Locations
  - Installed by the distribution - `/usr/lib/systemd/system/*`
  - Runtime - `/run/systemd/system/*`
  - Custom - `/etc/systemd/system/*`

Unit Types

Type	Extension	Description
Service	.service	Describes how to manage a service or application
Socket	.socket	Describes a network or IPC socket, or a FIFO buffer
Target	.target	Provides synchronization points for other units when booting or changing states
Mount	.mount	Defines a mountpoint on the system to be managed by systemd
Automount	.automount	Configures a mountpoint that will be automatically mounted
Device	.device	Describes a device that is managed by systemd
Scope	.scope	Used to manage sets of externally created system processed
Timer	.timer	Timer managed by systemd. A matching unit will be started when the timer is reached
Path	.path	Defines a path that can be used for path-based activation
Slice	.slice	It is associated with Linux Control Group nodes
Snapshot	.snapshot	It allows to reconstruct the current state of the system. Used to roll back temporary states
Swap	.swap	Describes swap space on the system

Service Units

- /usr/lib/systemd/system/sshd.service

```
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8)
man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS

...
```

```
...

ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s
RestartPreventExitStatus=255

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

Target Units

- /usr/lib/systemd/system/multi-user.target

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

Target Units

- /usr/lib/systemd/system/rescue.target

```
[Unit]
Description=Rescue Mode
Documentation=man:systemd.special(7)
Requires=sysinit.target rescue.service
After=sysinit.target rescue.service
AllowIsolate=yes

[Install]
Alias=kbrequest.target
```

Unit Dependencies

- **Wants**
  - Stated with **Wants=unit2**
  - When unit1 is run, unit2 will be run as well
  - Whether unit2 starts successfully doesn't affect unit1

```
[Unit]
Description=unit1
Wants=unit2
```

- **Requires**
  - Stated with **Requires=unit2**
  - Both units will run simultaneously
  - If unit2 fails, unit1 will be deactivated

```
[Unit]
Description=unit1
Requires=unit2
```

Unit Execution Order

- **Before**
  - Defined with **Before=unit2**
  - **unit1** will be executed fully before **unit2** starts

```
[Unit]
Description=unit1
Before=unit2
```

- **After**
  - Defined with **After=unit2**
  - **unit2** will be executed fully before **unit1** starts

```
[Unit]
Description=unit1
After=unit2
```

Target vs Runlevel

- Correspondences between **target** and **runlevel**

Runlevel	Target	Action
0	poweroff.target	Shuts down and powers off the system
1	rescue.target	Configures a rescue shell session
2	multi-user.target	Nongraphical multiuser mode typically without network
3	multi-user.target	Nongraphical multiuser mode with network services
4	multi-user.target	Same as 3
5	graphical.target	Graphical multiuser mode with network services
6	reboot.target	Reboots the system

- There are symbolic links **runlevelX.target** as well

System Initialization

Control and Monitoring

Additional systemctl Scenarios

- Show default target

```
[root@host ~]# systemctl get-default
```

- Show active targets

```
[root@host ~]# systemctl list-units --type=target
```

- Change current target

```
[root@host ~]# systemctl isolate runlevel1.target
```

- systemd configuration search path

```
[root@host ~]# systemctl -p UnitPath show
```

systemd-analyze

- Purpose
  - Analyze system boot-up performance
- Syntax

```
systemd-analyze [options] [command]
```

- Examples

```
# Ordered list of all running units
[user@host ~]$ systemd-analyze blame
# Print tree of the time-critical chain of units
[user@host ~]$ systemd-analyze critical-chain
```

journalctl

- Purpose
  - Query the systemd journal
- Syntax

```
journalctl [options] [matches]
```

- Examples

```
# Display the journal in reverse
[user@host ~]$ journalctl --reverse
# Information only from system services and kernel
[root@host ~]# journalctl --system
```

Practice

Processes and Resources

Monitoring and Management

Processes and Jobs

- **Process**
  - Running program with its own address space
- **Job**
  - Interactive program that doesn't detach
  - It can be suspended with **Ctrl+Z**
  - It can execute in foreground or background mode

jobs

- Purpose
  - Display status of jobs
- Syntax

```
jobs [options] [jobspec]
```

- Examples

```
# List all jobs
[user@host ~]$ jobs
# Print all jobs with extended information
[user@host ~]$ jobs -l
```

fg

- Purpose
  - Move job to the foreground
- Syntax

```
fg [jobspec]
```

- Examples

```
# Moves the current job to the foreground
[user@host ~]$ fg
# Moves particular job to the foreground
[user@host ~]$ fg 2
```

bg

- Purpose
  - Move job to the background
- Syntax

```
bg [jobspec]
```

- Examples

```
# Move the current job to the background
[user@host ~]$ bg
# Move particular job to the background
[user@host ~]$ bg 2
```

ps

- Purpose
  - Report a snapshot of the current processes
- Syntax

```
ps [options]
```

- Examples

```
# List every process on the system
[user@host ~]$ ps aux
# Print a process tree
[user@host ~]$ ps axjf
```

pstree

- Purpose
  - Display a tree of processes
- Syntax

```
pstree [options] [pid, user]
```

- Examples

```
# Display a tree with all processes
[user@host ~]$ pstree
# Display a tree for particular process
[user@host ~]$ pstree 1000
```

pgrep

- Purpose
  - Lookup processes based on name and other attributes
- Syntax

```
pgrep [options] pattern
```

- Examples

```
# List all sshd processes owned by root user
[user@host ~]$ pgrep -u root sshd
# List all processes owned by root or admin users
[user@host ~]$ pgrep -u root, admin
```

top

- Purpose
  - Display Linux processes
- Syntax

```
top [options]
```

- Example

```
# Display all active processes in interactive mode
[user@host ~]$ top
# Display user2's processes with 2 sec delay 5 times
[user@host ~]$ top -d 2 -n 5 –u user2
```

htop

- Purpose
  - Interactive process viewer
- Syntax

```
htop [options]
```

- Example

```
# Display all active processes in interactive mode
[user@host ~]$ htop
# Start htop with refresh interval 10 seconds
[user@host ~]$ htop -d 100
```

(Some) Common Signals*

- Signals are a limited form of **inter-process communication** (IPC)
- A signal is an **asynchronous notification** sent to a process

Signal	Value	Action
SIGHUP	1	Hang up or shutdown and restart process
SIGINT	2	Interrupt a process (used by **Ctrl+c**)
SIGKILL	9	Kill the process (cannot be ignored or caught)
SIGTERM	15	Terminate a process (can be ignored or caught)
SIGTSTP	20	Stop the terminal (used by **Ctrl+z**)

* https://en.wikipedia.org/wiki/Signal_(IPC)

kill

- Purpose
  - Shell built-in and ext. command. Send a signal to a job or process
- Syntax

```
kill [options] pid | jobspec
```

- Examples

```
# Send SIGKILL to a process with PID=1302
[root@host ~]# kill -9 1302
# List all signals
[user@host ~]$ kill -l
```

killall

- Purpose
  - Kill processes by name
- Syntax

```
killall [options] process
```

- Examples

```
# Send SIGKILL to all bash processes
[user@host ~]$ killall -9 bash
# Send SIGTERM to all bash process with prompt
[user@host ~]$ killall -i bash
```

pkill

- Purpose
  - Signal (SIGTERM) processes based on name and other attributes
- Syntax

```
pkill [options] pattern
```

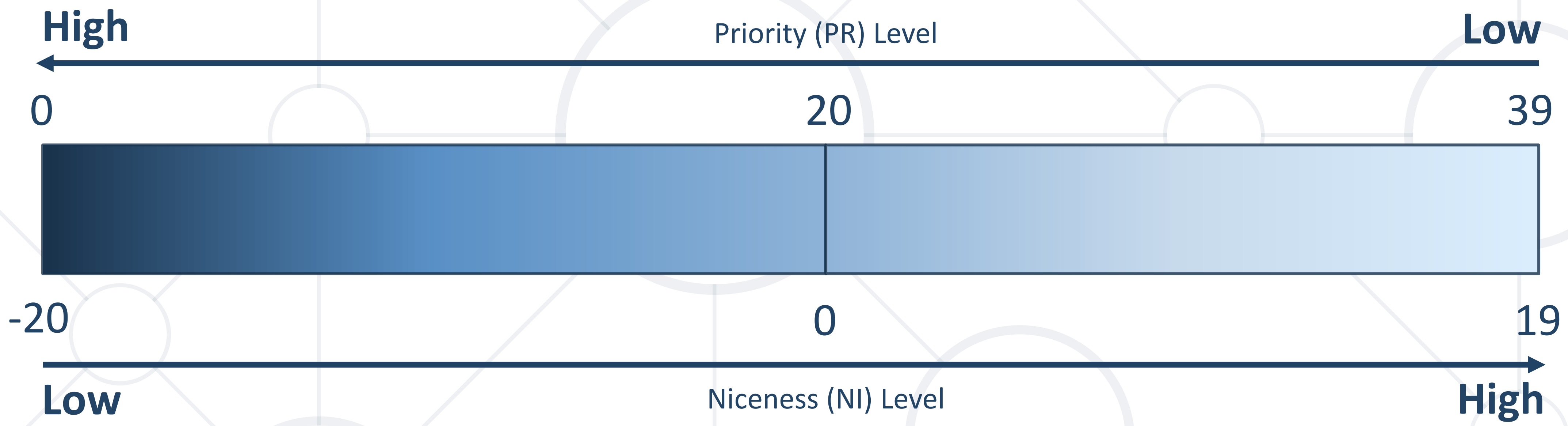
- Examples

```
# Kill all sshd processed owned by root user
[root@host ~]# pkill $(pgrep -u root sshd)
...
```

Process Priorities

- Niceness is applicable to normal process only and not to the real-time ones
- Priority level corresponds to the CPU time granted to a process
- Only privileged account can lower a nice value

Process Priorities



- PR = 20 + NI

nice

- Purpose
  - Run a program with modified scheduling priority
- Syntax

```
nice [options] [command [arg]]
```

- Examples

```
# Start program with particular niceness
[user@host ~]$ nice -n 15 program
...
```

renice

- Purpose
  - Alter priority of running process
- Syntax

```
renice [-n] priority [options] identifier
```

- Examples

```
# Alter priority of a program
[root@host ~]# renice -n -5 program
...
```

watch

- Purpose
  - Executing a program periodically, showing output full-screen
- Syntax

```
watch [options] command
```

- Examples

```
# Monitor running processes. Refresh every 5 sec
[user@host ~]$ watch -n 5 ps -o pid,nice,cmd,user
...
```

nohup

- Purpose
  - Run command immune to SIGHUP with output to a file
- Syntax

```
nohup command [args]
```

- Examples

```
# Run command immune to SIGHUP in background
[user@host ~]$ nohup ping abv.bg &
...
```

screen

- Purpose
  - Screen manager with terminal emulation
- Syntax

```
screen [-r] [options]
```

- Examples

```
# List sessions
[user@host ~]$ screen -ls
# Re-connect a screen session
[user@host ~]$ screen -r 2815
```

tmux

- Purpose
  - Terminal multiplexer
- Syntax

```
tmux [options]
```

- Examples

```
# List sessions
[user@host ~]$ tmux ls
# Attach to session #2
[user@host ~]$ tmux attach-session -t 2
```

Resources

Monitoring

free

- Purpose
  - Display amount of free and used memory in the system
- Syntax

```
free [options]
```

- Example

```
# Display information in human readable format
[user@host ~]$ free -h
# Display information with 10 sec delay 5 times
[user@host ~]$ free -c 5 -s 10
```

df

- Purpose
  - Report file system disk space usage
- Syntax

```
df [options] [file]
```

- Example

```
# Display information in human readable format
[user@host ~]$ df -h
# Display information about particular file (drive)
[user@host ~]$ df -h /dev/sda2
```

du

- Purpose
  - Estimate disk space usage
- Syntax

```
du [options] [file]
```

- Example

```
# Display disk space usage in human readable format
[user@host ~]$ du -h
# Display disk space usage for first level folders
[root@host ~]# du -h -d 1 /
```

vmstat

- Purpose
  - Report virtual memory statistics
- Syntax

```
vmstat [options] [delay [count]]
```

- Example

```
# Display statistics for 5 times with 5 sec delay
[user@host ~]$ vmstat 5 5
# Display disk statistics
[user@host ~]$ vmstat -d
```

iostat

- Purpose
  - Report CPU and IO statistics
- Syntax

```
iostat [options]
```

- Example

```
# Statistics every two seconds
[root@host ~]# iostat -d 2
# Extended statistics
[root@host ~]# iostat -x
```

pidstat

- Purpose
  - Report statistics for Linux tasks
- Syntax

```
pidstat [options]
```

- Example

```
# Statistics about process with id 1001 every 2 sec
[root@host ~]# pidstat -p 1001 2
# Statistics about process with name mysql
[root@host ~]# pidstat -C "mysql"
```

sar

- Purpose
  - Display Linux processes
- Syntax

```
sar [options]
```

- Example

```
# CPU information 3 times with 1 sec interval
[root@host ~]# sar -u ALL 1 3
# Memory information 3 times with 1 sec interval
[root@host ~]# sar -r 1 3
```

iotop

- Purpose
  - Simple IO monitor
- Syntax

```
iotop [options]
```

- Example

```
# Prints only processes with IO
[root@host ~]# iotop -o
# Prints processes 3 times with 1 sec interval
[root@host ~]# iotop -b -n 3 -d 1
```

nmon

- Purpose
  - Performance monitor
- Syntax

```
nmon [options]
```

- Example

```
# Start nmon in interactive mode
[root@host ~]# nmon
# Start nmon in data capture mode
[root@host ~]# nmon -f -s 60 -c 120
```

lsof

- Purpose
  - List open files
- Syntax

```
lsof [options]
```

- Example

```
# List all open files under a directory
[root@host ~]# lsof +D /etc
# Check which processes are working with a file
[root@host ~]# lsof -t /var/log/file.log
```

Practice

Summary

- BIOS and UEFI based systems have different boot process
- After POST is done then the process is handled by the boot loader
- There are many boot loaders, but GRUB2 has huge install base
- Once boot loader is ready the process goes to kernel

Summary

- Kernel initializes the hardware and then loads the Initramfs
- Once kernel is done, it unloads initramfs and starts system initialization process
- Systemd uses units to control services offered by the system
- Units include service, target, mount, and etc. We can define our own units

Resources

- Overview of systemd for RHEL 7
  - https://access.redhat.com/articles/754933
- An introduction to the Linux boot and startup processes
  - https://opensource.com/article/17/2/linux-boot-and-startup
- Introduction to system basics
  - https://documentation.suse.com/sle-micro/6.0/html/Micro-systemd-basics/index.html

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License