FRONTIER
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY
Hewlett Packard Enterprise
AMD

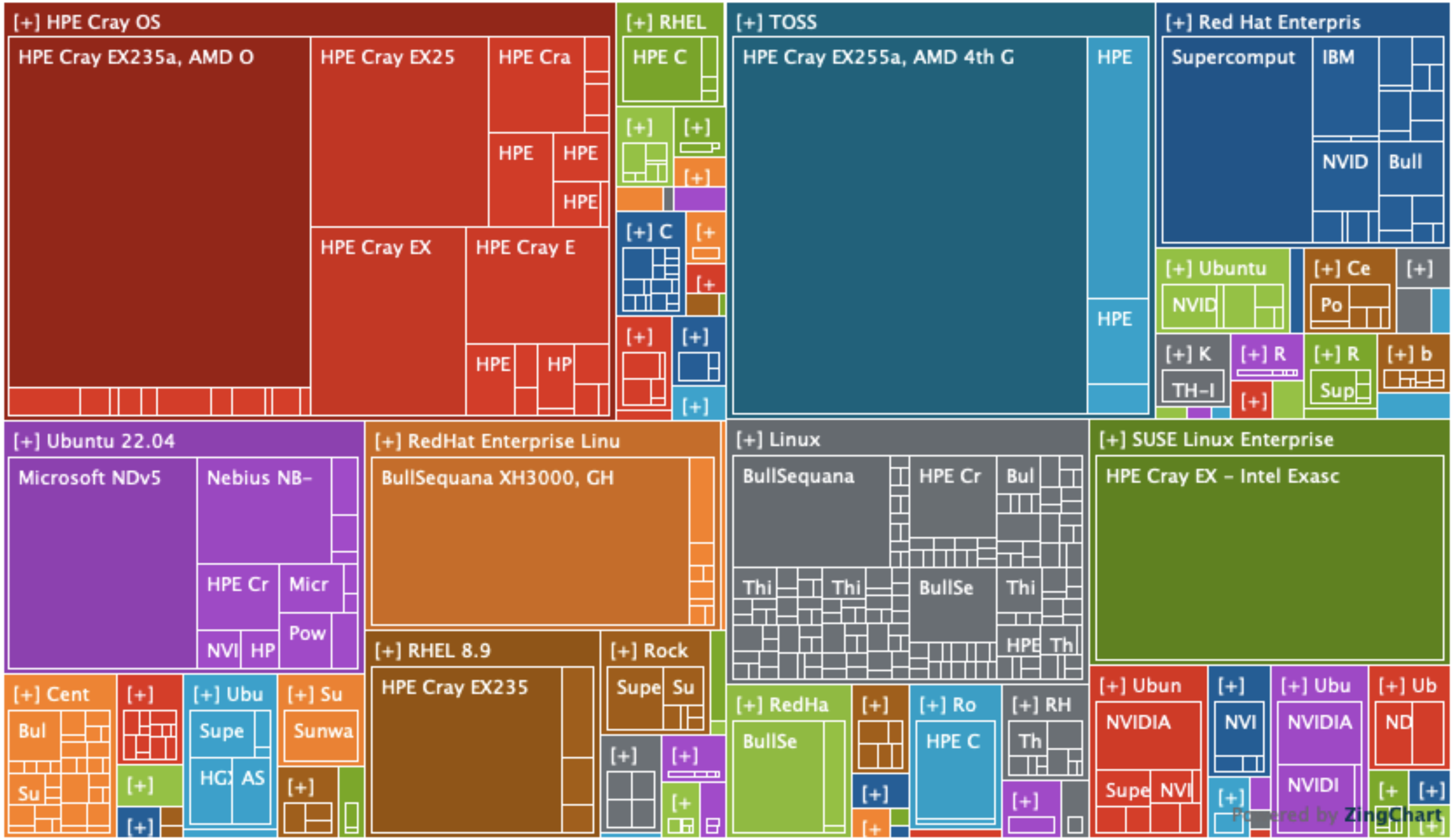
HPC series
working with Linux

Author:
Sami Ait Ali Oulahcen

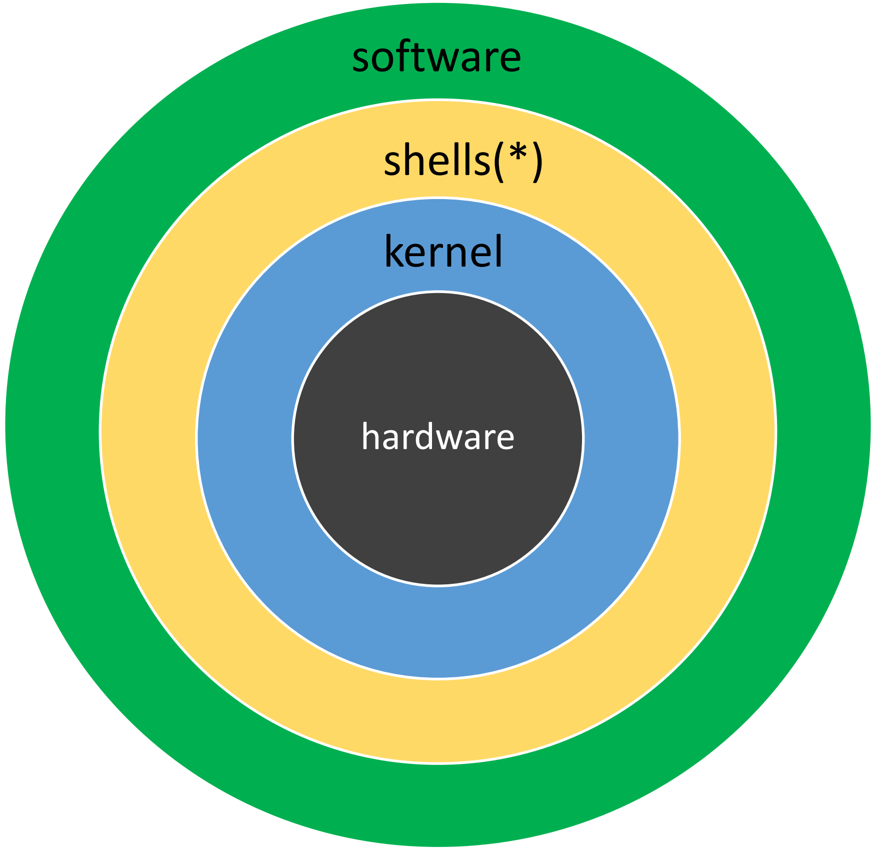
Rabat, Morocco
07 Aug 2023

Credit: Frontier Supercomputer https://www.flickr.com/photos/olcf/52117623843/

TOP500 HPC Operating Systems



UNIX/Linux architecture



(*) Here sits the user-space. It can be GUI launchers, systemd, CRON or any other user-space. Shell is relevant in the case of HPC.

Kernel

Contains device drivers

- Communicate with your hardware: CPU, RAM, GPU
- Block devices (physical media – hard drive, USB, CD)
- Character devices (keyboards, mice, terminals, modems) - Network devices (network cards)
- Pseudo devices (/dev/null, /dev/random)

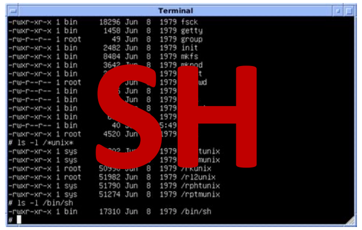
Filesystems

- Organize block devices into files and directories: ext2, ext3, ext4, xfs, beegfs, cephfs…

Examples:

OpenBSD 7.2:	OpenBSD kernel v7.2	UNIX
FreeBSD 13.1 :	FreeBSD kernel v13.1	
macOS 13:	Darwin/XNU v20	
debian 11:	Linux kernel v5.10	Linux
rhel 9 / rocky linux 9:	Linux kernel v5.14	
centOS 7:	Linux kernel v3.10	
fedora 37:	Linux kernel v6.0	

Shells



```
-rwxr-xr-x 1 bin      18296 Jun  8  1979 fsck
-rwxr-xr-x 1 bin       1458 Jun  8  1979 getty
-rw-r--r-- 1 root        49 Jun  8  1979 group
-rwxr-xr-x 1 bin       2482 Jun  8  1979 init
-rwxr-xr-x 1 bin       8484 Jun  8  1979 mkfs
-rwxr-xr-x 1 bin       3642 [illegible] 8  1979 mknod
-rwxr-xr-x 1 bin       [illegible]      1979 [illegible]t
-rw-r--r-- 1 root      [illegible] Jun  1979 [illegible]wd
-rw-r--r-- 1 bin       [illegible]6 Jun  8  1979
-rw-r--r-- 1 bin       [illegible] Jun  8  1979
-rwxr-xr-x 1 bin       [illegible]  8  1979
-rwxr-xr-x 1 bin       [illegible]      1979
-rw-r--r-- 1 bin         40 [illegible] 5:49
-rwxr-xr-x 1 root      4520 Jun [illegible] 1979
# ls -l /*unix*
-rwxr-xr-x 1 sys       [illegible]      1979 [illegible]tunix
-rwxr-xr-x 1 sys       [illegible]      1979 [illegible]munix
-rwxr-xr-x 1 root     509[illegible]  8  1979 /rkunix
-rwxr-xr-x 1 root     51982 Jun  8  1979 /rl2unix
-rwxr-xr-x 1 sys      51790 Jun  8  1979 /rphtunix
-rwxr-xr-x 1 sys      51274 Jun  8  1979 /rptmunix
# ls -l /bin/sh
-rwxr-xr-x 1 bin      17310 Jun  8  1979 /bin/sh
#
```

- All commands/scripts and software are run inside a shell (as a fork):

  The script: $ ./my-script.sh

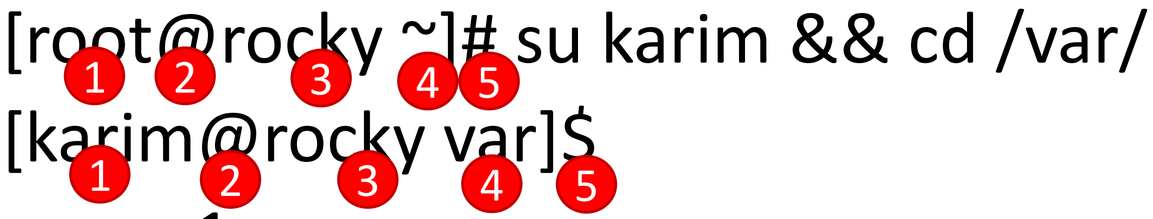
  is equivalent to: $ bash my-script.sh

- To determine which shell you are currently using, type:

```
$ echo $SHELL
```

The bash shell

- Most commonly used in HPC systems
- What it looks like:



```
[root@rocky ~]# su karim && cd /var/
[karim@rocky var]$
```

1- user

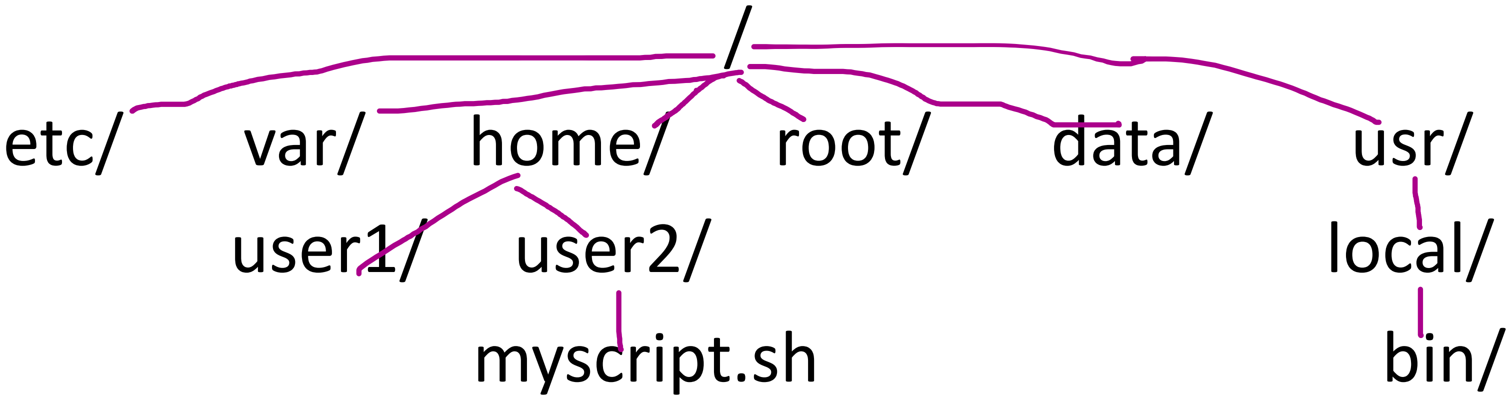
2-@

3- hostname

4- directory name (~ is special for home)

5- ‘#’ for root and ‘$’ for other users

Linux filesystem hierarchy



/

etc/ var/ home/ root/ data/ usr/

user1/ user2/ local/

myscript.sh bin/

- Unlike Windows where the root sits at a drive like C:/ or D:/, Linux filesystem is independent of drives
- Directories can sit on different partitions/drives, or can even be mounted on NFS shares

NFS shares

- Why can HPC systems run code from different nodes ?
- Because they all see the **same content** from the **user space** /home, /scratch, /data, and other directories containing shared **HPC software**
- This is thanks to NFS
- Network File System (NFS) is a distributed filesystem available through the network, mounted in as many hosts as desired in read & write (RW) or read only (RO)
- Users get access to the files over the network as if they were stored locally

Storage in HPC

- /home: **long term storage** for users. Each user gets a dedicated personal space with a defined quota under /home/$user
- /scratch: **temporary storage** for running jobs. It’s recommended to use for input/output data for jobs as it has a **faster filesystem** than /home or /data
- /data: additional long term storage offered to users/teams/projects

Users & groups

useradd/passwd/su/groupadd/usermod

Create users:

```
# useradd karim -p mypassword
```

To change the user password (you can omit the name for current user):

```
# passwd karim
```

To create a user for an application :

```
# useradd myapp -r -s /usr/sbin/nologin
```

To add a user to other groups:

```
# usermod myapp -aG docker
```

To run a command under a certain user:

```
# su karim -c ”command”
```

Permissions

ls -lh/chmod/chown/chgrp/chcon

Every file in linux has an owner, group, and rights (you can show this by running ls -lh /path/to/file)

chmod: change mode (rights) for file or directory

# chmod +x myscript.sh

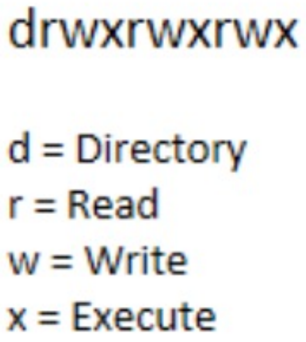
# chmod 600 rsa.key

chown: change owner (and group) for file or directory

# chown karim:docker /opt/myapp/

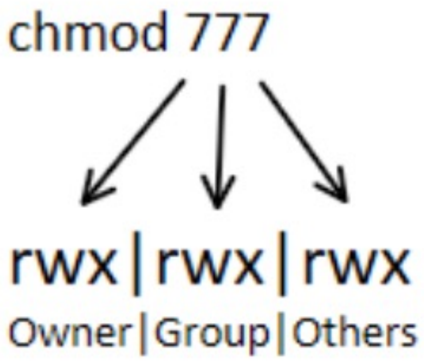
chgrp: change group for file or directory

# chgrp staff /home/shared/



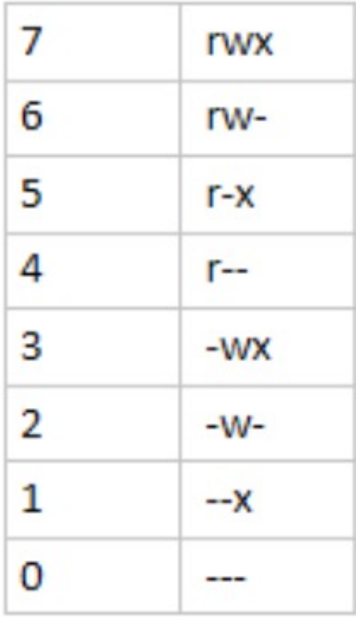
drwxrwxrwx

d = Directory
r = Read
w = Write
x = Execute



chmod 777

rwx|rwx|rwx
Owner|Group|Others



7	rwx
6	rw-
5	r-x
4	r--
3	-wx
2	-w-
1	--x
0	---

Filesystem navigation

pwd/cd/ls/mkdir/rmdir/rm/vi/cp/mv

To navigate to a directory, you can use:

The relative path: `~# cd folder1`

Or the full path: `~# cd /home/karim/folder1`

To list files: `~# ls -alh (/path/to/folder)`

To create a new directory: `# mkdir folder`

To remove an empty directory: `# rmdir folder`

To remove a directory with all its content (dangerous): `# rm -rf folder`

- Special directories:
  - . is current directory
  - .. is parent directory
  - ~ is home directory
- Hidden files or directories start with a dot (.)

Working with files

- To create a file without opening it, we use touch

```
# touch myfile.txt
```

- To modify a file, we use an editor like vi, vim, or nano

```
# vim myfile.txt
```

- Shortcuts in Linux are called Symbolic Links or Symlinks

```
# ln -s /path/to/file yourShortcut
```

- To remove a file, we use rm

```
# rm /path/to/file
```

Finding files

find/locate/grep/egrep

To find a file named my-code-1.py inside /home/karim/

```
# find /home/karim/ -name my-code-1.py
```

To find a file whose name contains ‘.py’

```
# find / -name ‘*.py’
```

You can also find a file by its content, for example let’s look for files that contain the word ’bwa-1.3’ inside /data/karim/scripts/

```
# grep -r ’bwa-1.3’ /data/karim/scripts/
```

You can use AI and online testers (regexer.com) to craft a regex (regular expression) for more complex searches

Linux resources

arch/free/df/top/ps

Print the architecture of the system (x86_64, arm64, riskV…)

```
# arch
```

Print the flavor/version of Linux

```
# cat /etc/os-release
```

Print information about the CPU

```
# cat /proc/cpuinfo | more
```

Check the total and free amount of RAM

```
# free -m
```

Check available partitions and disk space

```
# df -h
```

Linux processes

top/ps

Use # top OR # ps aux to check running processes

ps is a snapshot and customizable, top is real-time and interactive

Sort processes by CPU time:

```
# ps -eo pid,user,start,%mem,%cpu,cmd --sort=-%cpu
```

Sort processes by RAM usage:

```
# ps -eo pid,user,start,%mem,%cpu,cmd --sort=-%mem
```

While using top type ’P’ to sort by CPU, or ‘M’ to sort by RAM

Hot keys, Pipes & redirections

CTRL/|/ >/ >>

CTRL+C

Kill a running process on the terminal

CTRL+R

Search for previously issued commands.

CTRL+ALT+DELETE

Instantly start the reboot process.

Redirect (>): Redirect the input to another file.

```
# echo “This is a test” > myfile.txt
```

Append (>>): Adds text input at the end of the file.

```
# echo “This is a new line” >>
myfile.txt
```

Pipe (|): let the output of one command serve as input to the next.

```
# cat myfile.txt | grep “hello”
```

Logs

tail/journalctl

- One of the most important aspects of Linux administration
- Logs give you a comprehensive idea of what is going on in a system
- System events, performance & security issues, failures, audit trails
- Common system logs:

Log file	Purpose
/var/log/kern.log	Kernel messages (drivers, hardware errors, panics)
/var/log/secure and /var/log/audit/audit.log	Authentication & privilege escalation activity, SELinux
/var/log/syslog or /var/log/messages	General system events, service startups/shutdowns
/var/log/mysql/ or /var/log/nginx/ or /var/log/php/	Application-specific logs
/var/log/ufw.log or /var/log/firewalld	Firewall accept/deny events

Modern Linux systems include a lot of the above logs into **systemd-journal** which can be access via # journalctl -xe

Logs in HPC

- Important peace of information for any running jobs: status, what the job is currently doing (use tail -f), warnings/errors …
- Log location as configurable option in many HPC software, by default in working directory
- Can use standard streams for programs that don’t offer logging

```
# command > output.txt 2> error.txt
# command &> log.txt (bash & zsh only)
# command > log.txt 2>&1 (all shells)
```

equivalent to 1&2>

Standard Streams: stdin, stdout, and stderr

- **Standard Input (stdin)** - File Descriptor 0
- **Standard Output (stdout) -** File Descriptor 1
- **Standard Error (stderr) -** File Descriptor 2

Network tools

ping/scp/nc/netstat/dig/curl/ip/tcpdump

scp (ssh copy): copy files/directories between servers

```
# scp user@server1:/home/file /opt/project/
```

netstat: can be used to display listening ports on the system

```
# netstat -anput
```

dig: check DNS resolution for a domain

```
# dig mysite.ma
```

curl: transfer data to or from a server, can be used to check local server

```
# curl -vvvv http://localhost:8080
```

```
# ip a (ip address): display all interfaces states and ip addresses
```

tcpdump: network capture tool to display all traffic on an interface

```
# tcpdump -n -i  ens18 port 8080
```

Tip: use “# man command” to get help on how to use it

Environment variables & sourcing

Environmental variables are a set of dynamic values that are defined for the current shell and are inherited by any child shells or processes.

```
# echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

```
# export PATH=$PATH:/opt/myapp
```

To make it “stick”, add to .bashrc (or /etc/bashrc for all users):

```
# echo ‘export PATH=$PATH:/opt/myapp’ >> .bashrc
```

Then source it:

```
# . .bashrc        OR        # source .bashrc
```

Package management

yum/apt/dnf/rpm/dpkg

apt (Debian/Ubuntu/Linux Mint/Deepin):

Install a package:	#apt install package
Uninstall:	#apt remove package
Uninstall & remove config files:	#apt purge package
Search for package:	#apt search package
Update:	#apt update & apt upgrade

Manually install a DEB package

#dpkg -i package.deb OR #apt install ./package.deb

To uninstall:

#dpkg -r package.deb

dnf (successor of yum) RHEL/Rocky Linux/Alma Linux/Fedora

To install a package:	#dnf install package
To unistall:	#dnf remove package
To search for package:	#dnf search package
What provides a command:	#dnf provides command
update:	#dnf update

Manually install a RPM package (*)

#rpm -i package.rpm OR #dnf localinstall package.rpm

To uninstall:

#rpm -e package.rpm

(*) RPM is also used by SUSE Linux and derivatives

Service management

systemctl

Most Linux distributions use systemd replacing its predecessor “system V init” to allow parallelism during boot as well as centralized management of processes, daemons, services and mount points.

To start a daemon:

```
# systemctl start nginx
```

To check its status:

```
# systemctl status nginx
```

To enable a daemon at system startup:

```
# systemctl enable nginx
```

To reload daemons (after you modify/create a daemon):

```
# systemctl daemon-reload
```

90zKids.com
You got it dude.