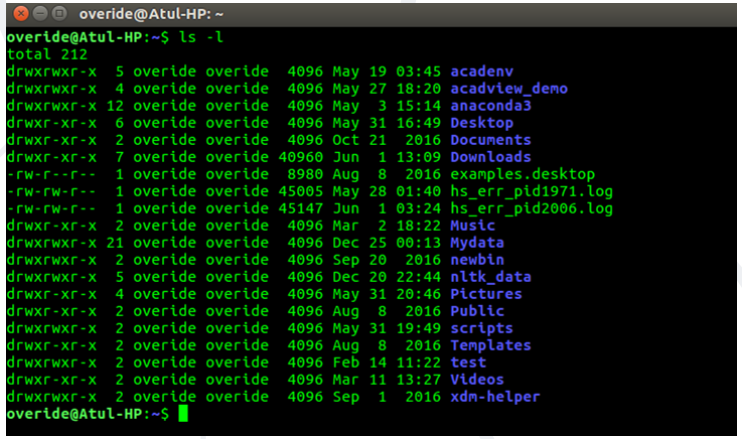
Linux and Linux Shell

Operation System, Linux OS, Linux Shell Commands, Environment Variables and SSH



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg

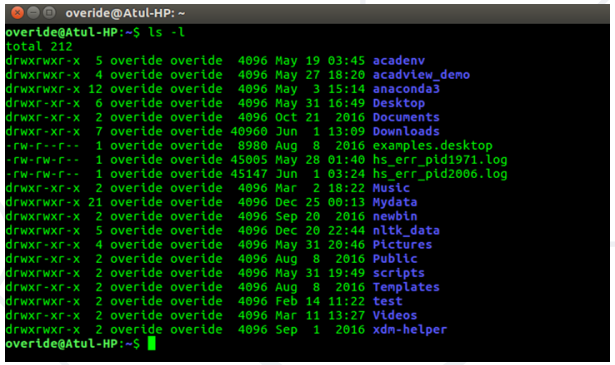
Have a Question?

Software University

sli.do

#Dev-Ops

Table of Contents



Operating System

Definition, Functions, Components, Examples

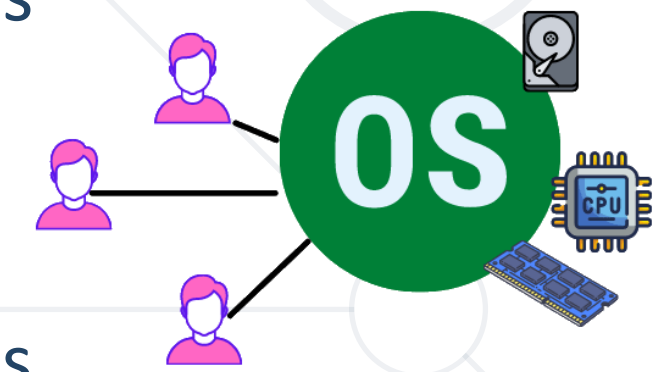
What is an Operating System?

- The **operating system** (OS) controls the computer (device)
  - Controls the hardware, processes (programs), resources, users
- Manages computer **hardware**, **software resources**, and provides **common services** for computer programs
  - It also coordinates all of this to make sure each program gets what it needs

- Allows users to **communicate with the computer** without knowing how to speak the computer's language

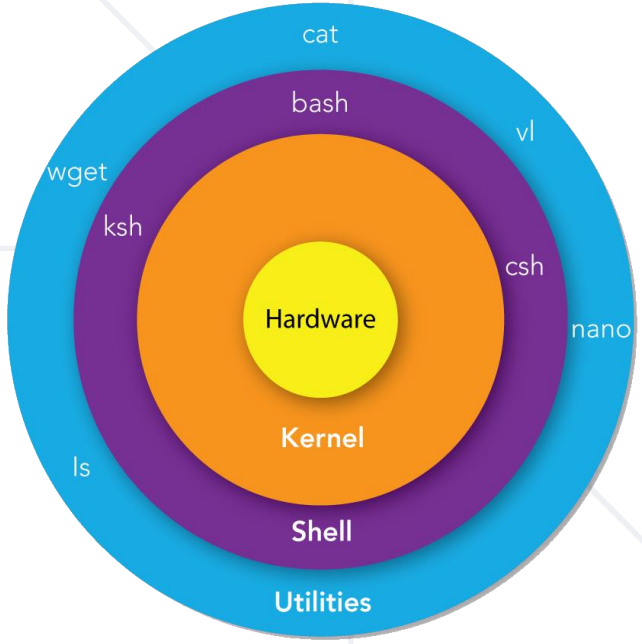
Important Functions of Operating Systems

- **Process** management (programs, which run in the OS)
  - **Process scheduling** – OS decides which process gets the processor, when and for how much time
  - Keeps tracks of **processor** and status of a process
- **Memory** management
  - Keeps tracks of **primary memory** (RAM), allocates / de-allocates memory for each process
- **Users** / **privileges** management
- **Device** management, **file** management, **security**, etc.



Operating Systems Components

- **Kernel**
  - **Essential OS component** that loads first and remains within the main memory
  - Provides the basic level of control of all the **computer peripherals**
- **Shell**
  - An **interface** between the **OS** and the **user**
  - Helps users to **access the services**, provided by the OS
  - It might be a **command-line interpreter** (CLI) or GUI app
- **Utilities** == small programs that provide additional capabilities to those, provided by the operating system
  - e.g., text editor, ZIP archiver, remote shell (SSH)



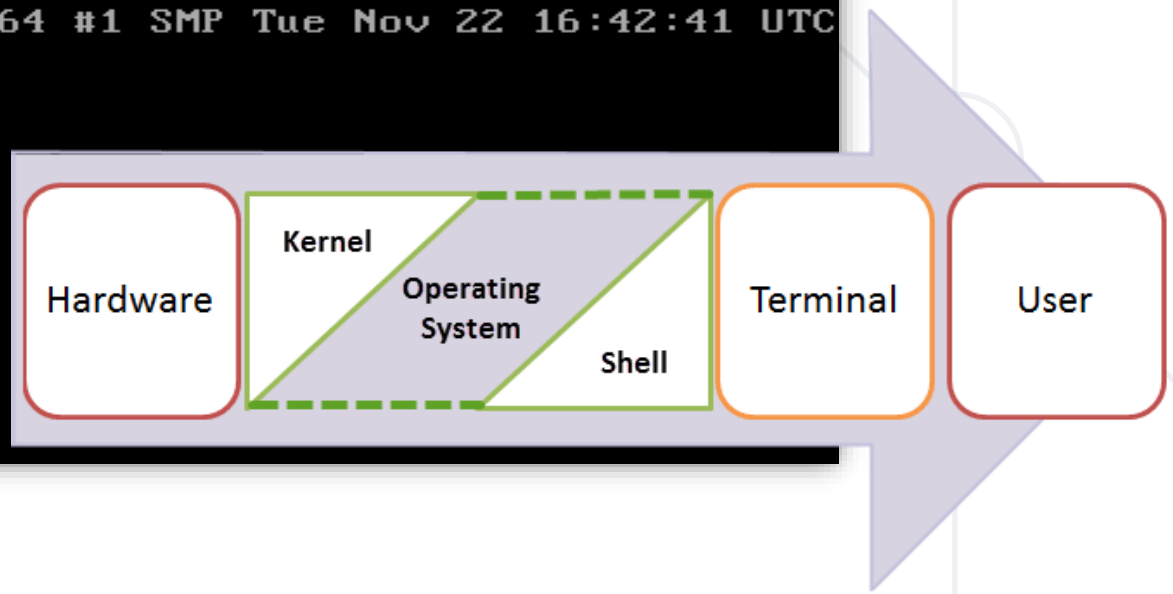
OS Security

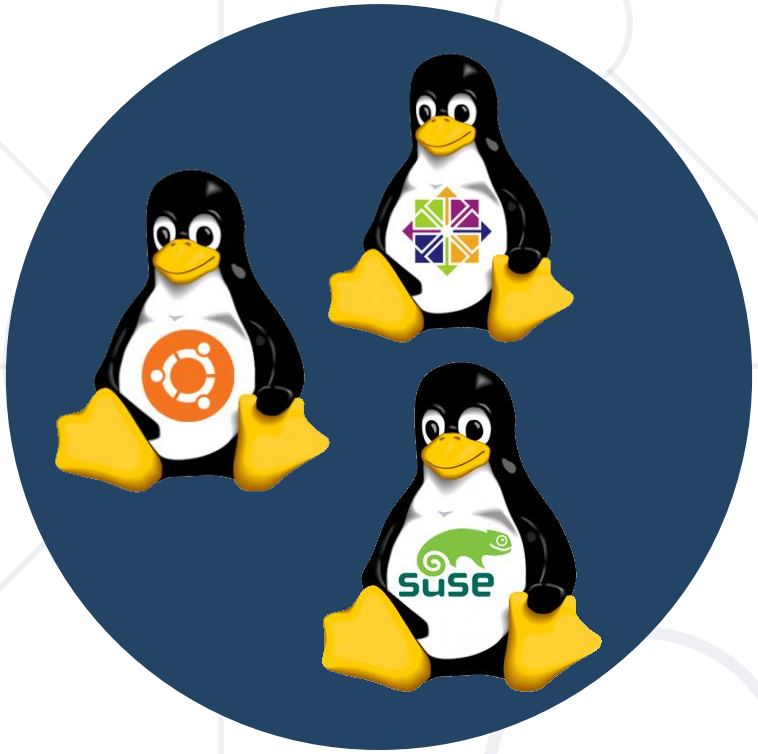
- **OS security** refers to providing a **protection system** for computer system resources and most importantly **data**
- Computers must be protected against **unauthorized access**, **malicious access to system memory**, **viruses**, **worms**, etc.
- **OS security** may be approached in many ways:
  - **Isolation** between processes (RAM, CPU, file system)
  - **Users**, **groups**, **permissions** (process, file system, others)
  - Filtering all incoming and outgoing **network traffic** through a **firewall**

Shell Definition

- Shell == **command line** interpreter
- It provides an interface that takes commands and passes them to the operating system
- When in GUI, we use **terminal emulators** to interact with the shell

```
[root@centosmin ~]# uname -a
Linux centosmin.softuni.lab 3.10.0-514.el7.x86_64 #1 SMP Tue Nov 22 16:42:41 UTC
 2016 x86_64 x86_64 x86_64 GNU/Linux
[root@centosmin ~]#
[root@centosmin ~]#
[root@centosmin ~]# cat /etc/hostname
centosmin.softuni.lab
[root@centosmin ~]#
[root@centosmin ~]# _
```



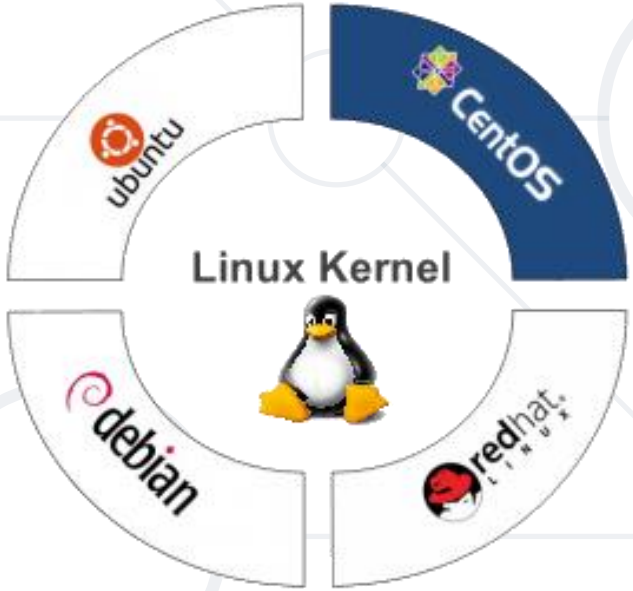


Linux Operating System

Architecture, Advantages and Disadvantages, Distribution

What is Linux?

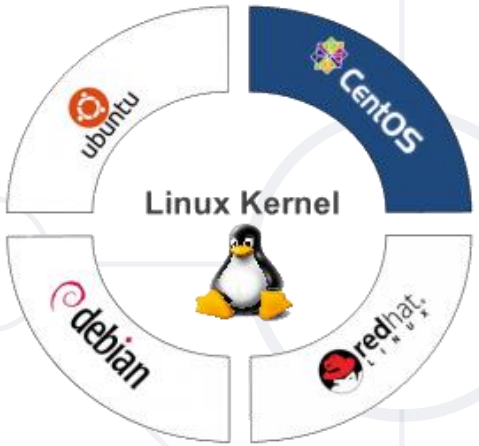
- **Linux OS** is a free, **open-source** operating system, very popular
  - https://github.com/torvalds/linux
  - Many **distributions** (variants), e. g. Ubuntu, Alpine, CentOS



- Linux is **NOT** the complete OS, it is just the **Linux Kernel**
  - Often the term is used to refer to the **whole OS** (Linux OS)
  - **Linux Kernel** is distributed along with all the necessary software and utilities, so that it can be used as an OS

Linux Distributions

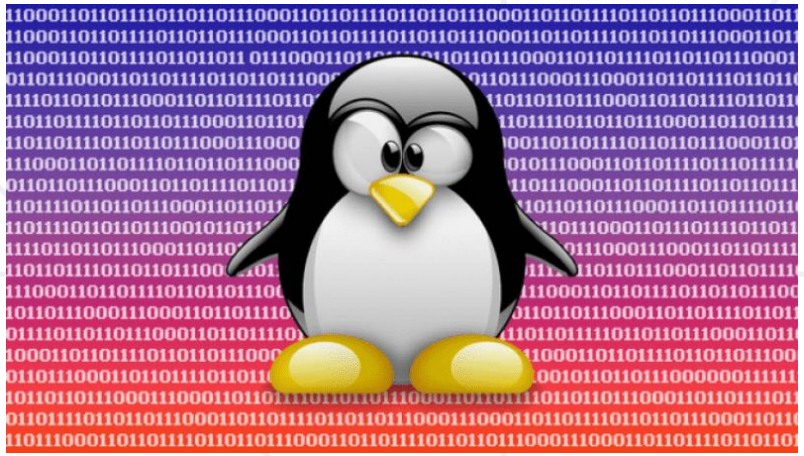
- Linux has many **distributions** (vendors)
- **Differences** in console commands, file locations, package management systems
- Most popular Linux distributions
  - **Ubuntu** – user-friendly, stable, popular – https://ubuntu.com
  - **Alpine** – minimal, secure, lightweight – https://alpinelinux.org
  - **CentOS** – enterprise-grade, stable, secure – https://centos.org
  - **Debian** – robust, reliable, versatile – https://debian.org
  - **Fedora** – community version of **Red Hat Enterprise Linux**



Linux Advantages

- Linux is **the most popular OS in the world**
  - You have many, many resources, available everywhere
  - Books, tutorials, videos, forums, questions / answers, certification programs, software, tools, etc.
- Linux is **open-source**, so anyone can contribute / enhance it
- Linux is **more secure** in comparison to other operating systems
- In Linux there is a larger number of **software updates**
- Linux provides **high performance** and efficiency

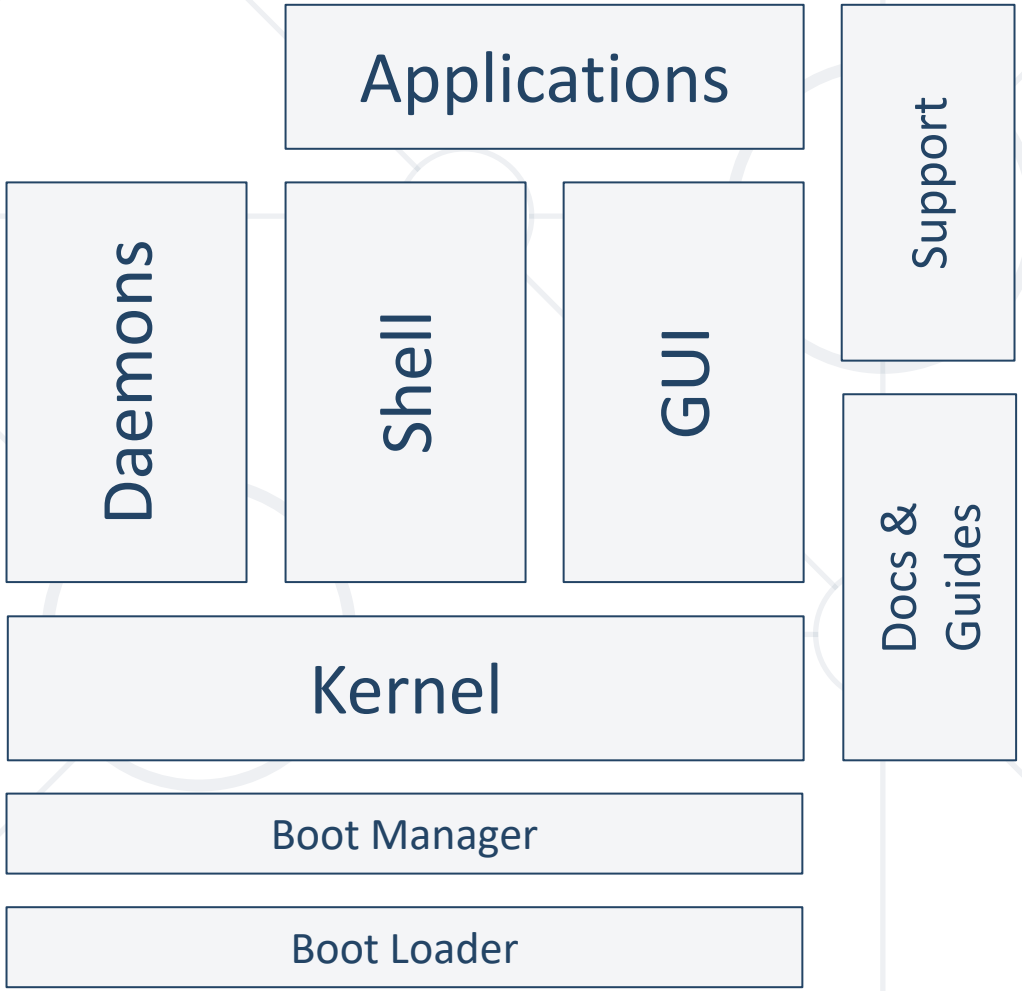
Linux Disadvantages



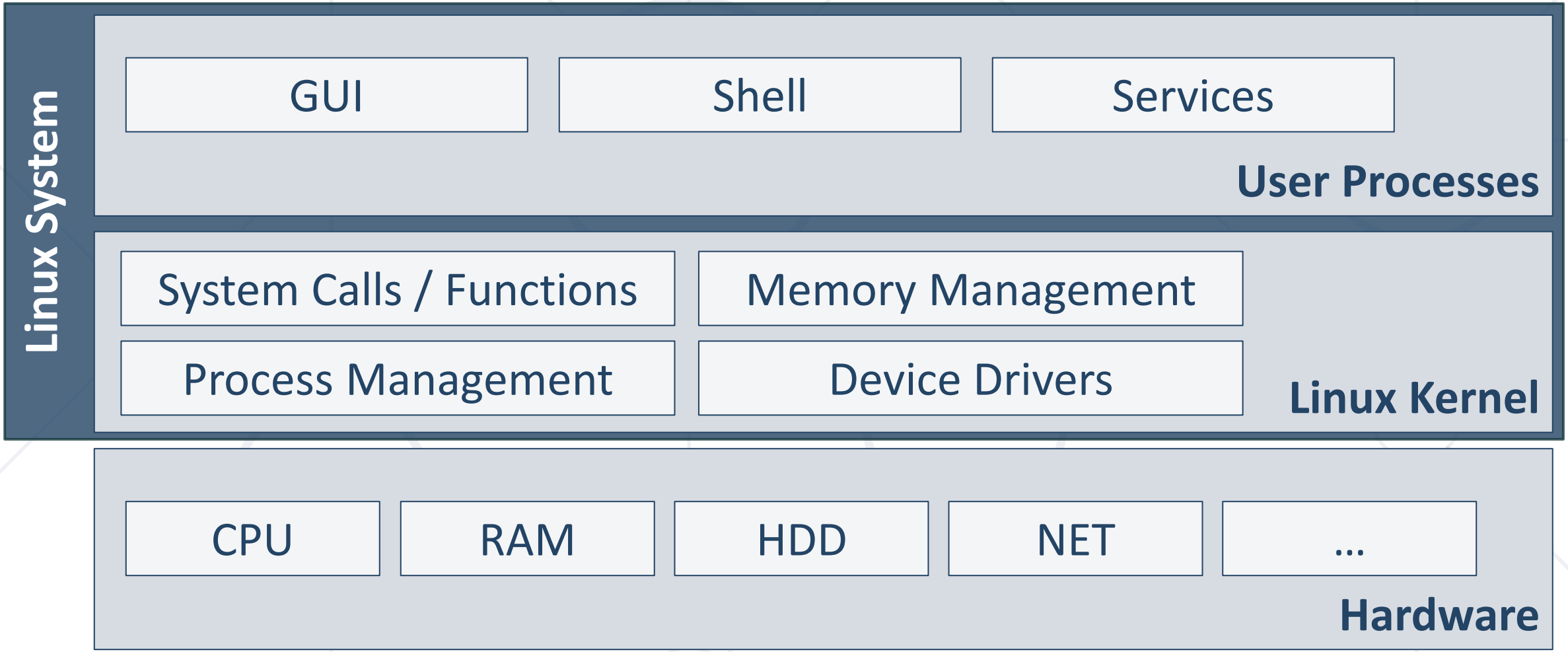
- Availability of apps: some **applications** that work on other OS **do not work in Linux**
- Other OS (like macOS, Windows) have better **usability** (UI and UX)
- **Learning curve**: it takes time and effort to master Linux
- Lack of standardization: many distributions, many differences
- Some **hardware drivers** are not available for Linux

Linux OS Components

- **System** components
  - Boot loader
  - Boot manager
  - Kernel
- **User** components
  - Daemons (services)
  - Shell (command line)
  - Graphical environments
  - User applications
- Documentation and Support



Linux System Architecture

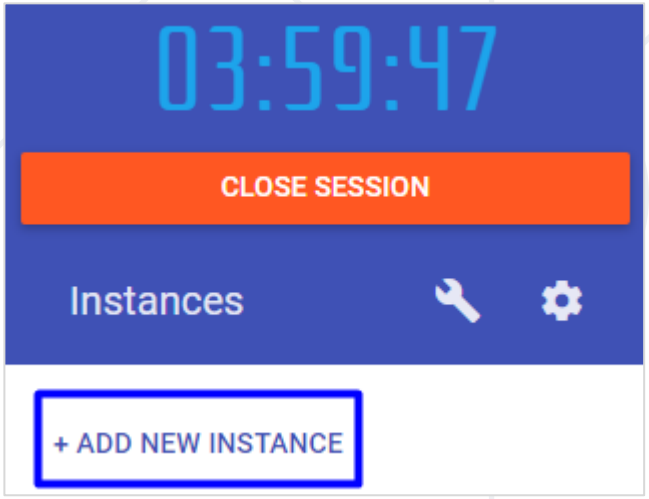


Linux Demo

Simple Commands on the Console

Docker Playground

- **Docker Playground** gives you an online **Linux virtual machine** to experiment with
  - Open Docker Playground and log in
  - Press **[Start]** and add a **new instance**
  - Now you have a **Linux environment** (Alpine Linux)




```
##################################################################
#                          WARNING!!!!                           #
# This is a sandbox environment. Using personal credentials      #
# is HIGHLY! discouraged. Any consequences of doing so are       #
# completely the user's responsibilites.                         #
#                                                                #
# The PWD team.                                                  #
##################################################################
[node1] (local) root@192.168.0.13 ~
$
```

Display the Current User

- The **whoami** command displays the currently logged-in user
- Example

```
user@host:~$ whoami
```

```
[node1] (local) root@192.168.0.28 ~
$ whoami
root
```

Check Linux System Info

- Type the **uname -a** command to print OS information

```
[node1] (local) root@192.168.0.13 ~
$ uname -a
Linux node1 4.4.0-210-generic #242-Ubuntu SMP Fri Apr 16 09:57:56 UTC 2021
x86_64 Linux
```

1. Kernel name
2. Network hostname
3. Kernel release information
4. Kernel version information
5. Machine hardware name

Display Linux processes

- **top [options]**
- Examples

```
top - 11:10:12 up 54 min,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 105 total,   1 running, 103 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1983.4 total,   1441.7 free,    167.8 used,    373.9 buff/cache
MiB Swap:   1965.0 total,   1965.0 free,      0.0 used.   1667.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    179 root     -51   0       0      0      0 S   0.3   0.0   0:01.09 irq/18-v
   1531 root      20   0       0      0      0 I   0.3   0.0   0:00.43 kworker/
   1537 root      20   0       0      0      0 I   0.3   0.0   0:01.81 kworker/
```

```
# Display all active processes in interactive mode
user@host:~$ top

# Display user's processes with 2 sec delay 5 times
user@host:~$ top -d 2 -n 5
```

```
lsauser@ubuntu:~$ top
```

```
lsauser@ubuntu:~$ top -d 2 -n 5 -u lsauser
```

```
PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
977 lsauser   20   0   18488   9828   8256 S   0.0   0.5   0:00.05 systemd
979 lsauser   20   0  103304   3368      4 S   0.0   0.2   0:00.00 (sd-pam)
```

File System in Linux

Files, Directories and Basic Commands

The File System in Linux

- **File system** == OS component, which organizes and manages **files** and **directories** on a storage device (e.g. SSD disk)
  - Popular file systems: **ext4**, **BTRFS**, **ZFS**, **NTFS**
- Most Linux distributions use **ext4** file system
  - Storage is organized in **directories**, which hold **files** and **other directories**
  - **Files** hold data (e.g. text data / binaries)
  - **Special files**: symlinks, pipes, sockets, …

```
│   ├── rules.d
│   │   ├── 60-fprint-autosuspend.rules
│   │   ├── 60-pcmcia.rules
│   │   ├── 60-raw.rules
│   │   ├── 70-persistent-cd.rules
│   │   ├── 70-persistent-net.rules
│   │   ├── 90-alsa.rules
│   │   ├── 90-hal.rules
│   │   ├── 91-drm-modeset.rules
│   │   └── 98-kexec.rules
│   └── udev.conf
├── updatedb.conf
├── uptrack
│   ├── trustdb.gpg
│   ├── uptrack.conf
│   └── uptrack.conf.rpmnew
├── vimrc
├── virc
├── vnstat.conf
├── warnquota.conf
├── webalizer.conf
├── wgetrc
├── X11
│   ├── applnk
│   ├── fontpath.d
│   └── prefdm
├── xdg
│   └── autostart
│       ├── gnome-keyring-daemon.desktop
│       ├── restorecond.desktop
│       └── sealertauto.desktop
├── xinetd.d
│   └── rsync
├── xml
│   └── catalog
├── yum
│   ├── pluginconf.d
│   │   ├── product-id.conf
│   │   ├── protectbase.conf
│   │   ├── rhnplugin.conf
│   │   └── subscription-manager.conf
│   ├── protected.d
│   ├── vars
│   └── version-groups.conf
```

List files and directories

- Syntax `ls [options]`
- Examples

```
user@host:~$ ls
```

```
[node1] (local) root@192.168.0.8 ~/test
$ ls
example    file.txt
```

```
user@host:~$ ls -al
```

```
$ ls -la
total 0
drwxr-xr-x    3 root     root            37 Mar 13 11:47 .
drwx------    1 root     root            18 Mar 13 11:40 ..
drwxr-xr-x    2 root     root             6 Mar 13 11:40 example
-rw-r--r--    1 root     root             0 Mar 13 11:47 file.txt
```

File Types

- Files and directories
  - Regular (-)
  - Directory (d)
- Special files
  - Symbolic link (l)
  - Block device (b)
  - Character device (c)
  - Named pipe (p)
  - Socket (s)

```
drwxr-xr-x  19 root root    4096 Mar 27 11:09 .
drwxr-xr-x  19 root root    4096 Mar 27 11:09 ..
lrwxrwxrwx   1 root root       7 Apr 23  2020 bin -> usr/bin
drwxr-xr-x   2 root root    4096 Apr 23  2020 boot
drwxr-xr-x   9 root root    3000 Mar 27 11:09 dev
drwxr-xr-x  94 root root    4096 Mar 27 12:10 etc
drwxr-xr-x   3 root root    4096 Dec 11  2021 home
-rwxr-xr-x   3 root root 1440152 May  7  2022 init
lrwxrwxrwx   1 root root       7 Apr 23  2020 lib -> usr/lib
```

```
crw-------  1 root root   1,   1 Mar 27 11:09 mem
drwxr-xr-x  2 root root       60 Mar 27 11:09 net
crw-rw-rw-  1 root root   1,   3 Mar 27 11:09 null
crw-------  1 root root  10, 144 Mar 27 11:09 nvram
crw-------  1 root root 108,   0 Mar 27 11:09 ppp
crw-rw-rw-  1 root root   5,   2 Mar 27 12:33 ptmx
drwxr-xr-x  2 root root        0 Mar 27 11:09 pts
brw-------  1 root root   1,   0 Mar 27 11:09 ram0
brw-------  1 root root   1,   1 Mar 27 11:09 ram1
```

Explore the Linux File System

- Type the `ls /` command to examine **root directory** files

```
$ ls /
bin        docker.log  lib        opt        run        sys        var
certs      etc         media      proc       sbin       tmp
dev        home        mnt        root       srv        usr
```

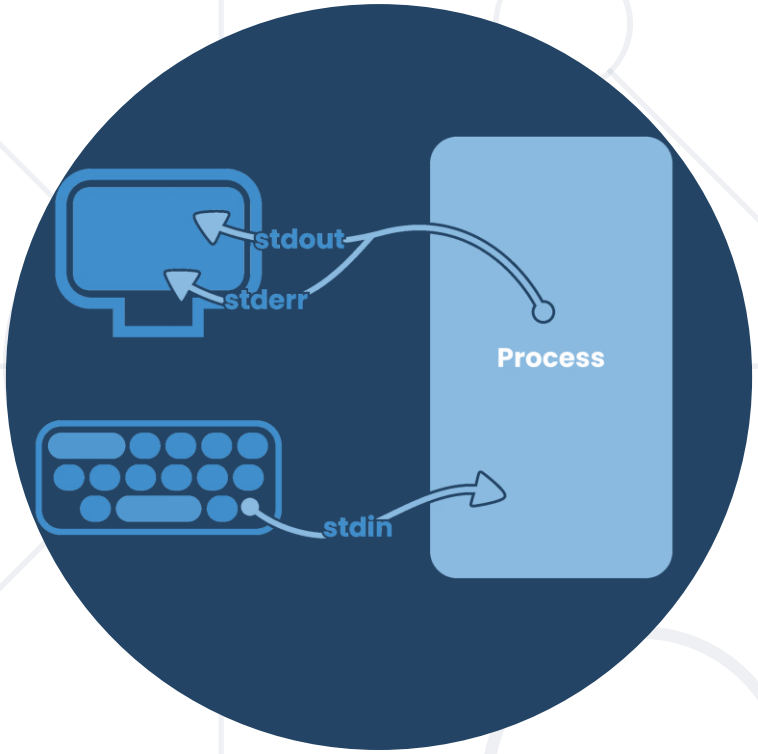
/bin	User Binaries	/var	Variable Files
/sbin	System Binaries	/tmp	Temporary Files
/etc	Configuration Files	/usr	User Programs
/dev	Device Files	/home	Home Directories
/proc	Process Information	/lib	System Libraries
/boot	Boot Loader Files		

Absolute vs Relative Path

- **Absolute path** (**starts** with **/**)
  - Calculated from the **root** of the **file system tree**, e.g. **`/dev/random`**
- **Relative path** (**no leading /**, uses **`.`** and **`..`**)
  - Calculated from the **current working directory**, e.g. **`../../bin/`**
- If we are in **/home/user** and we want to list folders

```
# Absolute notation
user@host:~$ ls -al /usr/bin

# Relative notation
user@host:~$ ls -al ../
```

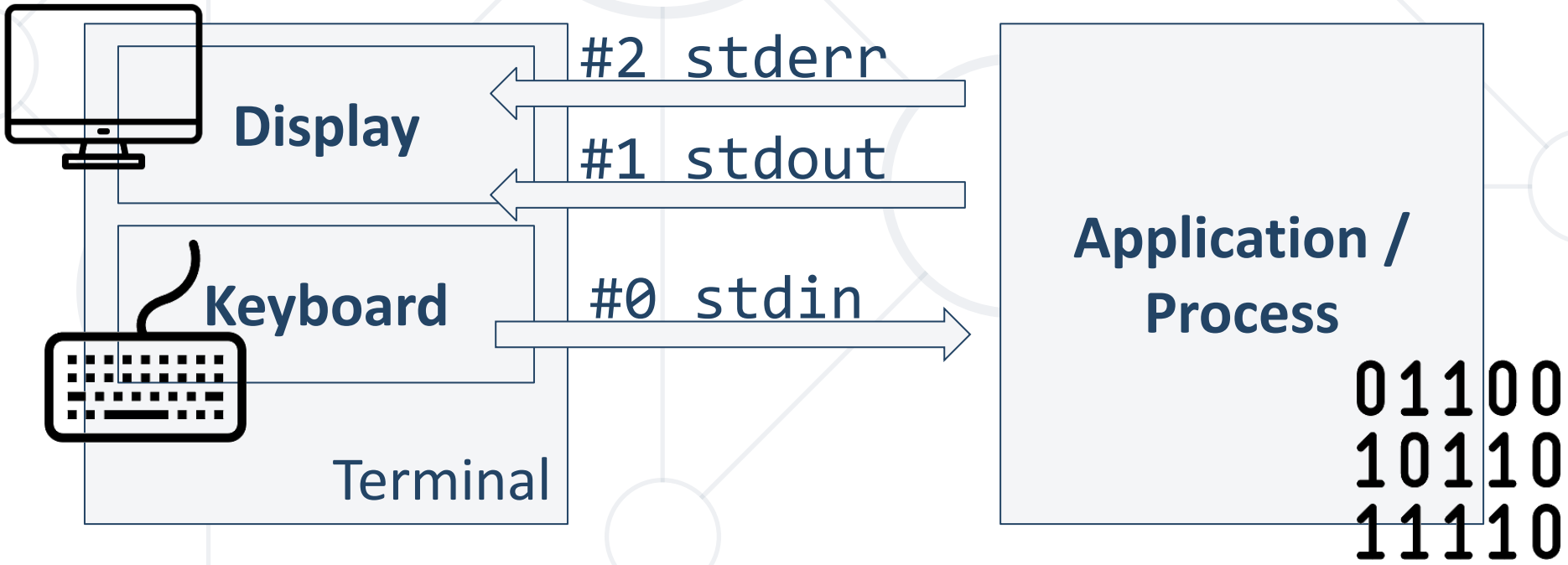


Input / Output Streams

Standard File Descriptors. Redirection

Standard File Descriptors

- `stdin` == standard **input** stream (N.0)
- `stdout` == standard **output** stream (N.1)
- `stderr` == standard **error output** stream (N.2)



Redirect Output (>)

- Redirect output streams (**stdout** or **stderr**) with **target overwrite**
- Examples

```
user@host:~$ echo 'Hello World!' > hello.txt

user@host:~$ echo 'Hello World!' 1> hello.txt
```

The same

1 == stdout

```
lsauser@ubuntu:~$ echo 'Hello World!' > hello.txt
```

```
lsauser@ubuntu:~$ echo 'Hello World!' 1> hello.txt
```

```
lsauser@ubuntu:~$ cat hello.txt
Hello World!
```

Redirect Output with Append (>>)

- Redirect output streams (**stdout** or **stderr**) with **target append**
- Example

```
user@host:~$ echo 'Line #2' >> file.txt
```

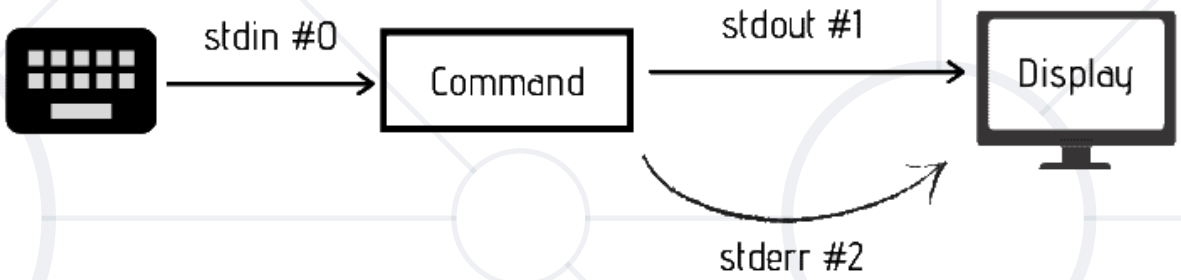
```
lsauser@ubuntu:~$ cat file.txt
Line #1
lsauser@ubuntu:~$ echo 'Line #2' >> file.txt
lsauser@ubuntu:~$ cat file.txt
Line #1
Line #2
```

Redirect Input (<)

- Redirect input stream (**stdin**)
  - Usually, it is omitted
- Examples

```
user@host:~$ cat < hello.txt

user@host:~$ cat hello.txt
```




```
lsauser@ubuntu:~$ cat < hello.txt
Hello!
```

```
lsauser@ubuntu:~$ cat hello.txt
Hello!
```

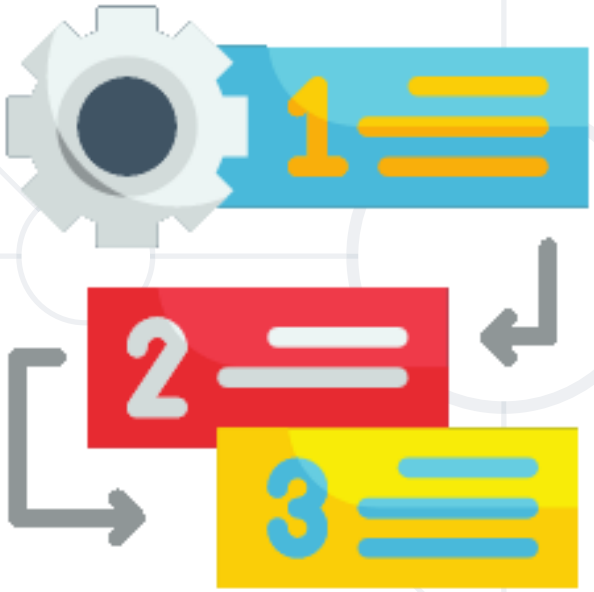
The same

Command Sequences

Execute Multiple Commands. Substitution

Commands Sequences

- Execute in order (disconnected)
  - **Sequence**: **command1 ; command2**
- Execute in order (connected)
  - **Pipe**: **command1 | command2**
- Execute conditionally
  - On **Success**: **command1 && command2**
  - On **Failure**: **command1 || command2**



Sequence (;)

- Always execute **next command**
- Example

```
user@host:~$ ls non-existing-file.txt ; echo Ok
```

```
lsauser@ubuntu:~$ ls non-existing-file.txt ; echo Ok
ls: cannot access 'non-existing-file.txt': No such file or directory
Ok
```

Pipe (|)

- **Chaining** two or more programs' **output together**
- Example

```
user@host:~$ ls | sort | head -n 3
```

```
lsauser@ubuntu:~$ ls | sort | head -n 3
copy-file.txt
dir1
dir2
```

On Success (&&)

- Next **command is executed** if previous one exited with a **status of 0** (success)
- Examples

```
user@host:~$ ls non-existing-file.txt && echo Ok
```

```
lsauser@ubuntu:~$ ls non-existing-file.txt && echo Ok
ls: cannot access 'non-existing-file.txt': No such file or directory
```

```
user@host:~$ ls existing-file.txt && echo Ok
```

```
lsauser@ubuntu:~$ ls file.txt && echo Ok
file.txt
Ok
```

On Failure (||)

- Next **command is NOT attempted** if previous one exited with **0**
- Examples

```
user@host:~$ ls existing-file.txt || echo Ok
```

```
lsauser@ubuntu:~$ ls file.txt || echo Ok
file.txt
```

```
user@host:~$ ls non-existing-file.txt || echo Ok
```

```
lsauser@ubuntu:~$ ls non-existing-file.txt || echo Ok
ls: cannot access 'non-existing-file.txt': No such file or directory
Ok
```

Users and Groups

Manage Users and Groups

Users in Linux

- Users file (**/etc/passwd**)

```
lsauser@ubuntu:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```
root:x:0:0:root:/root:/bin/bash
...
madmin:x:1000:1000:M.Admin:/home/madmin:/bin/bash
... (1) (2) (3) (4) (5) (6) (7)
```

1. Username (login)
2. Password placeholder
3. User ID
4. Group ID
5. Comment (full name, phone, etc.)
6. Home directory
7. User shell

Groups in Linux

- Groups file (/etc/group)

```
root:x:0:
...
wheel:x:10:madmin  (4)
...
madmin:x:1000:
...(1) (2) (3)
```

```
lsauser@ubuntu:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,lsauser
tty:x:5:syslog
```

1. Group name
2. Password placeholder
3. Group ID
4. Group members

Access Rights

Users, Groups and Permissions in the File System

Access Rights in the Linux File System

```
[root@vmi937769 softunisites]# ls -al
total 112
drwx--x--x 26 softunisites softunisites 4096 Feb 13 23:17 .
drwx--x--x 17 root         root         4096 Jan 23 14:36 ..
lrwxrwxrwx  1 softunisites softunisites   38 Jan  7 23:28 access-logs -> /etc/apache2/
drwxr-x---  7 softunisites nobody       4096 Mar 25 03:33 conf.softuni.bg
drwxr-xr-x  3 softunisites softunisites 4096 Feb 13 23:17 .cpaddons
drwx------  6 softunisites softunisites 4096 Mar 27 01:03 .cpanel
drwxr-x---  6 softunisites nobody       4096 Mar 25 03:33 fest.softuni.bg
```

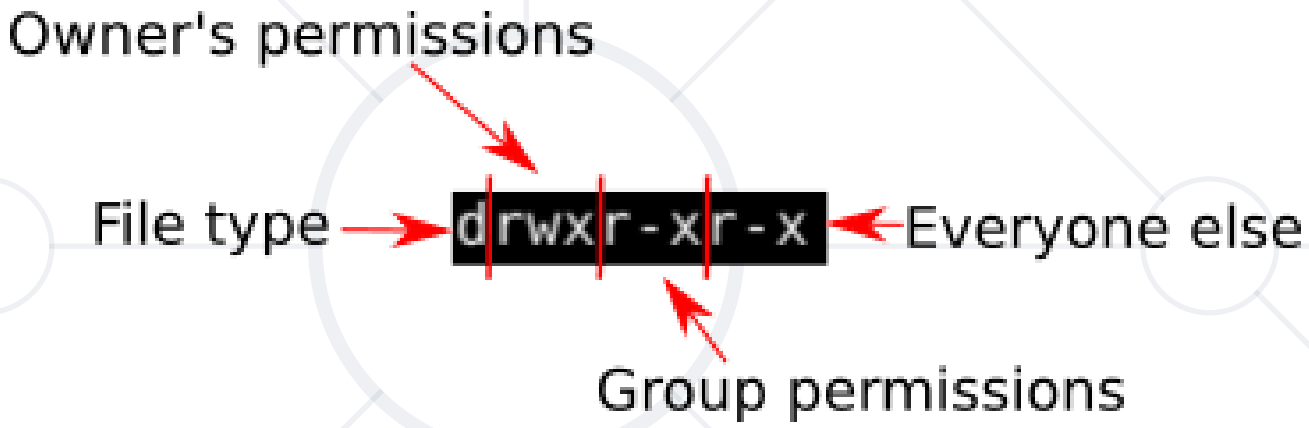
Group

Owner

Access Rights

read / write / execute

File Permissions and Octal Masks



Permissions	Octal Mask	Description
`---------`	000	No permissions
`rw-rw-rw-`	666	Everyone read + write
`rwxr-xr-x`	755	Owner full access, others read + execute
`rwxrwxrwx`	777	Everyone read, write, and execute

Access Rights (1)

- **Read**
  - **Files** – allows a user to view the contents of a file
  - **Directories** – allows a user to view the names of files in a directory

```
lsauser@ubuntu:~$ cat file.txt
test text
```

```
lsauser@ubuntu:~$ ls dir1
file2.txt  myscript.sh
```

```
lsauser@ubuntu:~$ echo 'new text in file' > file.txt
```

- **Write**
  - **Files** – allows a user to modify and delete the file
  - **Directories** – allows a user to delete the directory, modify its contents, and modify the contents of files that the user can read

```
lsauser@ubuntu:~$ cd dir1
lsauser@ubuntu:~/dir1$ rm file2.txt
```

Access Right (2)

- **Execute**

```
lsauser@ubuntu:~$ ./myscript.sh
Executed script
```

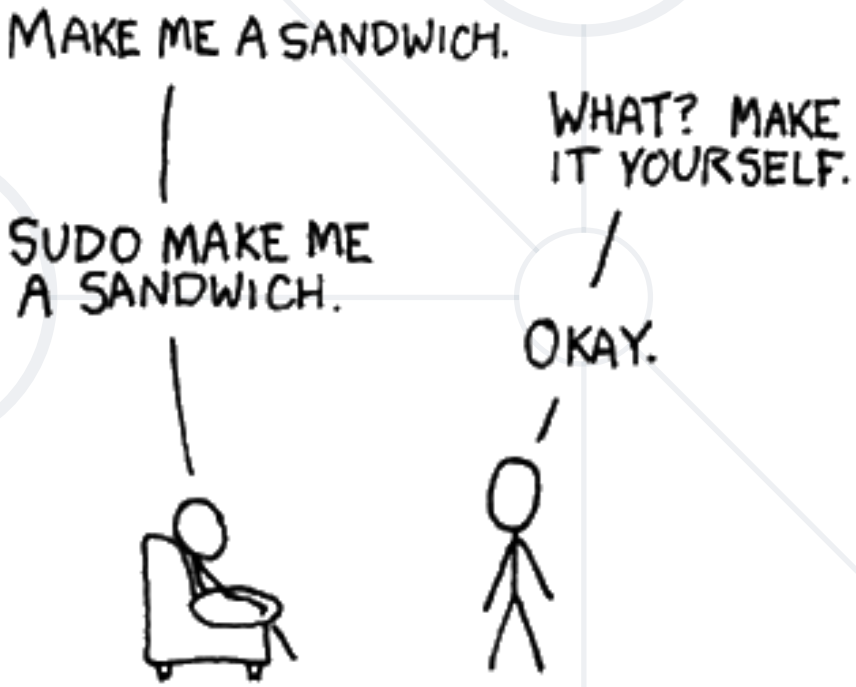
  - **Files** – allows a user to execute a file (the user must also have read permission)
  - **Directories** – allows a user to access, or traverse into, a directory and access metadata about files in it

```
lsauser@ubuntu:~$ stat file.txt
  File: file.txt
  Size: 17              Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 543729      Links: 1
Access: (0775/-rwxrwxr-x)  Uid: ( 1000/ lsauser)   Gid: ( 1000/ lsauser)
Access: 2022-03-27 08:55:42.168274494 +0000
Modify: 2022-03-27 09:06:44.848541413 +0000
Change: 2022-03-27 09:06:44.848541413 +0000
 Birth: -
```

SUDO (SuperUser DO) Configuration

- **sudo** is used to access restricted files and operations
  - Controls **who** can do **what** and from **where**
- Temporarily allows **ordinary users** to **perform administrative tasks**
  - Without logging in as the **root user**

```
sudo [command]
```



sudo

- Execute a command as **another user**

```
# Execute command as another user
user@host:~$ sudo -u testuser whoami

# Switch to a user
user@host:~$ sudo su testuser

# Switch to a user with a login shell
user@host:~$ su - testuser

# Execute a single command as root
user@host:~$ sudo chmod +x hello.txt
```

```
lsauser@ubuntu:~$ sudo -u testuser ls /home
helpdesk  lsauser
```

```
lsauser@ubuntu:~$ sudo su testuser
$ id
uid=1003(testuser) gid=2002(testuser) groups=2002(testuser)
```

```
lsauser@ubuntu:~$ sudo su - testuser
[sudo] password for lsauser:
$
```

Changing the File Permissions

- `cat > hello.sh`

```
echo "Current date:" && date
echo "Current user:" && whoami
[Ctrl-D]
```

- `chmod +x hello.sh`

```
nakov@Nakov-Laptop-HP:~$ cat > script.sh
echo "Current date:" && date
echo "Current user:" && whoami
nakov@Nakov-Laptop-HP:~$ ./script.sh
-bash: ./script.sh: Permission denied
nakov@Nakov-Laptop-HP:~$ ls -al script.sh
-rw-r--r-- 1 nakov nakov 60 Mar 27 13:55 script.sh
nakov@Nakov-Laptop-HP:~$ chmod +x script.sh
nakov@Nakov-Laptop-HP:~$ ls -al script.sh
-rwxr-xr-x 1 nakov nakov 60 Mar 27 13:55 script.sh
nakov@Nakov-Laptop-HP:~$ ./script.sh
Current date:
Mon Mar 27 13:56:17 EEST 2023
Current user:
nakov
nakov@Nakov-Laptop-HP:~$
```

The `chmod` command

```
nakov@Nakov-Laptop-HP:~$ chmod 000 script.sh
nakov@Nakov-Laptop-HP:~$ ls -al script.sh
---------- 1 nakov nakov 60 Mar 27 13:55 script.sh
nakov@Nakov-Laptop-HP:~$ chmod 755 script.sh
nakov@Nakov-Laptop-HP:~$ ls -al script.sh
-rwxr-xr-x 1 nakov nakov 60 Mar 27 13:55 script.sh
nakov@Nakov-Laptop-HP:~$ chmod 777 script.sh
nakov@Nakov-Laptop-HP:~$ ls -al script.sh
-rwxrwxrwx 1 nakov nakov 60 Mar 27 13:55 script.sh
nakov@Nakov-Laptop-HP:~$
```

chown

- Change **file owner** and **group**
- Syntax `chown [options] [owner][:[group]] file`

Can be replaced with "."

- Examples

```
# Change both owner and group of a file*
user@host:~$ chown user:users file.txt
```

```
lsauser@ubuntu:~$ sudo chown testuser:testuser file.txt
```

```
lsauser@ubuntu:~$ stat file.txt
  File: file.txt
  Size: 17              Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 543729      Links: 1
Access: (0775/-rwxrwxr-x)  Uid: ( 1003/testuser)   Gid: ( 2002/testuser)
```

* No **sudo** is needed when objects are owned by the current user

chgrp

- Change **group ownership**
- Syntax `chgrp [options] group file`
- Examples

```
# Change the group of files*
user@host:~$ chgrp developers file*
```

```
lsauser@ubuntu:~$ sudo chgrp developers file*
```

```
lsauser@ubuntu:~$ stat file.txt
  File: file.txt
  Size: 17              Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 543729      Links: 1
Access: (0775/-rwxrwxr-x)  Uid: ( 1003/testuser)   Gid: ( 2001/developers)
```

* No **sudo** is needed when objects are owned by the current user

Environment Variables

Linux Environment Variables

- **Environment variables** == dynamic **variables** used by the Linux shell
  - Provide config settings to Linux apps
  - They follow the **<NAME>=<VALUE>** formatting
  - They are case-sensitive
  - By convention environment variable names use CAPITAL_LETTERS

```
$ env
DOCKER_VERSION=20.10.17
CHARSET=UTF-8
HOSTNAME=node2
DOCKER_TLSENABLE=false
COMPOSE_VERSION=2.6.1
DOCKER_BUILDX_VERSION=0.8.2
PWD=/root
```

Environment Variables – Commands

- List all environment variables

```
env
printenv
```

```
nakov@Nakov-Laptop-HP:~$ env
SHELL=/bin/bash
WSL_DISTRO_NAME=Ubuntu
NAME=Nakov-Laptop-HP
PWD=/home/nakov
```

- Print a single environment variable

```
printenv HOME
echo $HOME
```

```
nakov@Nakov-Laptop-HP:~$ echo $HOME
/home/nakov
```

- Sets a new environment variable

```
export VAR=VALUE
```

```
nakov@Nakov-Laptop-HP:~$ export AUTHOR="Svetlin Nakov"
nakov@Nakov-Laptop-HP:~$ env | grep AUTHOR
AUTHOR=Svetlin Nakov
```

Secure Shell (SSH)

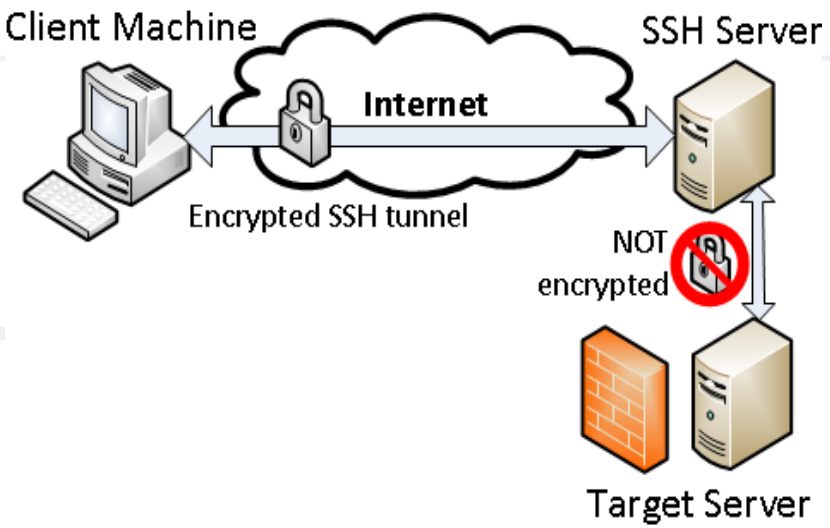
Connecting to Remote Linux Machine

SSH (Secure Shell)

- **Secure Shell (ssh)** allows connecting to a remote machine's console

```
ssh 192.168.0.28 -l root
```

```
$ ssh 192.168.0.28 -l root
#################################################################
#                          WARNING!!!!                          #
# This is a sandbox environment. Using personal credentials    #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                               #
# The PWD team.                                                 #
#################################################################
[node1] (local) root@192.168.0.28 ~
$ exit
logout
Connection to 192.168.0.28 closed.
[node2] (local) root@192.168.0.27 ~
$
```



More Linux Commands

apt

- **apt** provides a high-level command line interface for the package management system
- Syntax `apt install <package>`
- Examples

```
# Install curl
user@host:~$ apt install curl
# Install wget
user@host:~$ apt install wget
```

apt (2)

- **update** is used to download package information from all configured sources
- Syntax

```
apt update
```

- Examples

```
# Update the local index of packages
user@host:~$ apt update
```

apt (3)

- **upgrade** is used to install available upgrades of all packages, currently installed on the system, from the configured sources
- Syntax

```
apt upgrade
```

- Examples

```
# Update the local index of packages
user@host:~$ apt upgrade
```

mkdir

- **Create directories**
- Syntax `mkdir [options] directory [directory …]`
- Examples

```
lsauser@ubuntu:~$ mkdir dir1 dir2
lsauser@ubuntu:~$ ls
dir1  dir2
```

```
# Create two directories
user@host:~$ mkdir dir1 dir1

# Create nested direktories
user@host:~$ mkdir -pv projects/project1
```

```
lsauser@ubuntu:~$ mkdir -pv projects/project1
mkdir: created directory 'projects'
mkdir: created directory 'projects/project1'
```

touch

- Change file **timestamp**
- Syntax `touch [options] file [file …]`
- Examples

```
# Change access time of a file
user@host:~$ touch -a .bash_history

# Create an empty file
user@host:~$ touch file1.txt
```

```
lsauser@ubuntu:~$ stat .bash_history
Access: 2022-04-09 07:29:18.259581408 +0000
lsauser@ubuntu:~$ touch -a .bash_history
lsauser@ubuntu:~$ stat .bash_history
Access: 2022-04-09 07:30:48.971622573 +0000
```

```
lsauser@ubuntu:~$ touch file1.txt
lsauser@ubuntu:~$ ls
dir1  dir2  file1.txt  projects
```

```
lsauser@ubuntu:~$ cd dir1
lsauser@ubuntu:~/dir1$ touch file2.txt
lsauser@ubuntu:~/dir1$ ls
file2.txt
```

File is created in the **current directory**

cp

- **Copy** files and directories
- Syntax `cp [options] source dest`
- Examples

```
# Copy single file
user@host:~$ cp file1.txt ~/dir1/file2.txt

# Copy multiple files to a folder
user@host:~$ cp /etc/*.conf ~/dir2
```

```
lsauser@ubuntu:~$ cp file1.txt ~/dir1/file2.txt
lsauser@ubuntu:~$ cat ~/dir1/file2.txt
sample text
```

```
lsauser@ubuntu:~$ ls ~/dir2
adduser.conf            hdparm.conf
ca-certificates.conf    host.conf
debconf.conf            ld.so.conf
```

mv

- **Move** (**rename**) files
- Syntax `mv [options] source dest`
- Examples

```
# Rename a file
user@host:~$ mv file1.txt first-file.txt

# Move multiple files
to a folder
user@host:~$ mv *.txt ~/TextFiles
```

```
lsauser@ubuntu:~$ mv file1.txt first-file.txt
lsauser@ubuntu:~$ cat first-file.txt
sample text
```

```
lsauser@ubuntu:~$ mkdir TextFiles
lsauser@ubuntu:~$ mv *.txt ~/TextFiles
lsauser@ubuntu:~$ ls TextFiles
first-file.txt
```

rm

- **Remove** files or directories
- Syntax `rm [options] file [file …]`
- Examples

```
# Remove file
user@host:~$ rm first-file.txt

# Remove folder and its contents
user@host:~$ rm -rf ~/TextFiles
```

```
lsauser@ubuntu:~$ cd ~/TextFiles
lsauser@ubuntu:~/TextFiles$ rm first-file.txt
lsauser@ubuntu:~/TextFiles$ ls
```

```
lsauser@ubuntu:~$ rm -rf ~/TextFiles
lsauser@ubuntu:~$ ls
dir1  dir2  projects
```

pwd

- **Print** the current working directory
- Syntax

```
pwd
```

- Examples

```
user@host:~$ pwd
```

```
[node1] (local) root@192.168.0.8 ~/test/example
$ pwd
/root/test/example
```

head

- **Output** the **first part** (10 lines by default) of files
- Syntax `head [options] [files]`
- Examples

```
# Show first ten lines of a file
user@host:~$ head /etc/passwd

# Show first three lines of a file
user@host:~$ head -n 3 /etc/passwd
```

```
lsauser@ubuntu:~$ head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

tail

- **Output** the **last part** (10 lines by default) of files
- Syntax `tail [options] [files]`
- Examples

```
# Show last ten lines of a file
user@host:~$ tail /etc/passwd

# Show last three lines of a file
user@host:~$ tail -n 3 /etc/passwd
```

```
lsauser@ubuntu:~$ tail -n 3 /etc/passwd
neweruser:x:1002:1002::/home/neweruser:/bin/sh
testuser:x:1003:2002::/home/testuser:/bin/sh
newuser:x:1004:2004::/users/newuser:/bin/bash
```

cat

- **Reads** data from the file and gives **the content** as output
- Syntax `cat [filename]`
- Example

```
# Show content of file "os-release"
user@host:~$ cat os-release
```

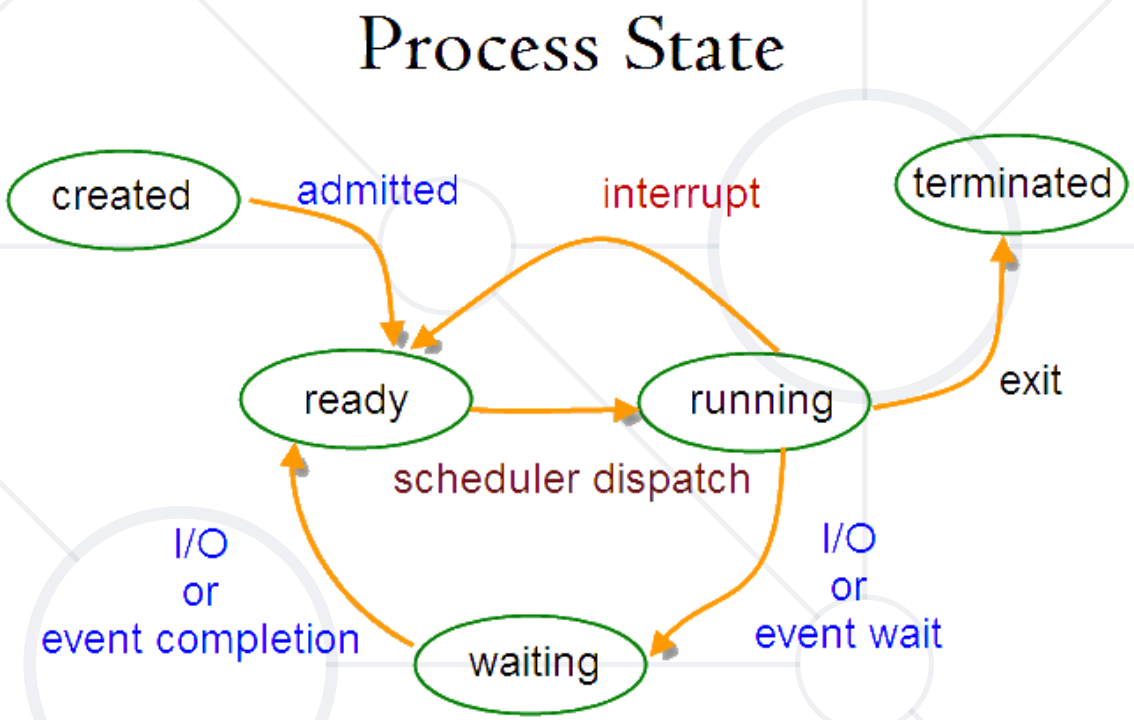
```
[node1] (local) root@192.168.0.18 /etc
$ cat os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.16.0
PRETTY_NAME="Alpine Linux v3.16"
HOME_URL="https://alpinelinux.org/"
```

Processes

Monitoring and Management

Processes and Jobs

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



jobs

- 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
```

```
lsauser@ubuntu:~$ jobs
[1]+  Stopped                 top
```

```
lsauser@ubuntu:~$ jobs -l
[1]+  1478 Stopped (signal)        top
```

ps

- Report a **snapshot of the current processes**
- Syntax `ps [options]`
- Examples

```
lsauser@ubuntu:~$ ps aux
USER   PID %CPU %MEM    VSZ   RSS TTY STAT START   TIME COMMAND
root     2  0.0  0.0      0     0 ?   S    10:15   0:00 [kthreadd]
root     3  0.0  0.0      0     0 ?   I<   10:15   0:00 [rcu_gp]
```

```
# List every process on the system
user@host:~$ ps aux
```

```
lsauser@ubuntu:~$ ps axjf
PPID PID  PGID  SID TTY  TPGID STAT   UID   TIME COMMAND
   0   2     0    0 ?       -1 S        0   0:00 [kthreadd]
   2   3     0    0 ?       -1 I<       0   0:00  \_ [rcu_gp]
```

```
# Print a process tree
user@host:~$ ps axjf
```

kill

- **Send a signal** to a job or process
- Syntax `kill [options] pid | jobspec`
- Examples

```
# Send SIGKILL to a process with PID=1478
user@host:~$ sudo kill -9 1478

# List all signals
user@host:~$ sudo kill -l
```

```
lsauser@ubuntu:~$ sudo kill -9 1478
[1]+  Killed                  top
```

```
lsauser@ubuntu:~$ sudo kill -l
HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM STKFLT
CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH POLL PWR SYS
```

killall

- **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 processes with prompt
user@host:~$ killall -i bash
```

```
lsauser@ubuntu:~$ killall -9 bash
Connection to localhost closed.
```

```
lsauser@ubuntu:~$ killall -i bash
Kill bash(1469) ? (y/N) y
Kill bash(1718) ? (y/N) y
```

Data Fetching

curl

- **curl** is a tool for transferring data from or to a server
- Syntax `curl [options] URL`
- Examples

```
# Fetch only the HTTP headers of the specified URL
user@host:~$ curl -I https://www.gnu.org/
```

```
[node1] (local) root@192.168.0.8 ~/test
$ curl -I https://www.gnu.org/
HTTP/1.1 200 OK
Date: Mon, 13 Mar 2023 12:31:12 GMT
Server: Apache/2.4.29
Content-Location: home.html
Vary: negotiate,accept-language,Accept-Encoding
TCN: choice
Strict-Transport-Security: max-age=63072000
X-Frame-Options: sameorigin
X-Content-Type-Options: nosniff
Access-Control-Allow-Origin: (null)
Accept-Ranges: bytes
Cache-Control: max-age=0
Expires: Mon, 13 Mar 2023 12:31:12 GMT
Content-Type: text/html
Content-Language: en
```

wget

- **wget** is a free utility for non-interactive download of files from the Web
- Syntax `wget [options] URL`
- Examples

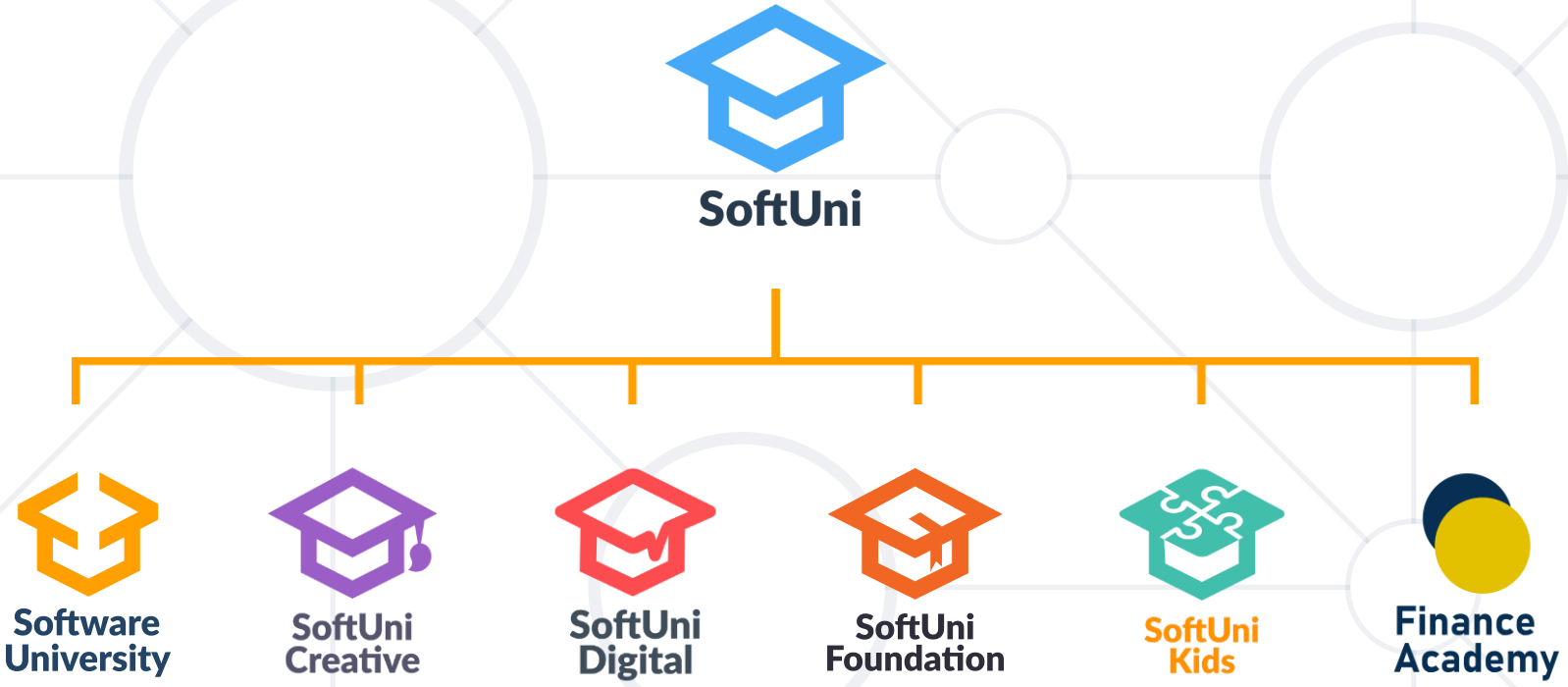
```
# Download file under different name
user@host:~$ wget -O latest-hugo.zip \
https://github.com/gohugoio/hugo/archive/master.zip
```

```
[node1] (local) root@192.168.0.8 ~/test
$ wget -O latest-hugo.zip https://github.com/gohugoio/hugo/archive/master.zip
Connecting to github.com (140.82.112.3:443)
Connecting to codeload.github.com (140.82.112.9:443)
saving to 'latest-hugo.zip'
latest-hugo.zip      100% |*********************************************************************************************************************| 28.8M  0:00:00 ETA
'latest-hugo.zip' saved
```

Summary

- **Operating systems** manage all of the software and hardware on the computer
- **Linux OS** distributions & file system
- **Shell** definition
- **Command sequences**
- **Environmental variables** – dynamic named variables
- Linux **commands** – used to interact with the system

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

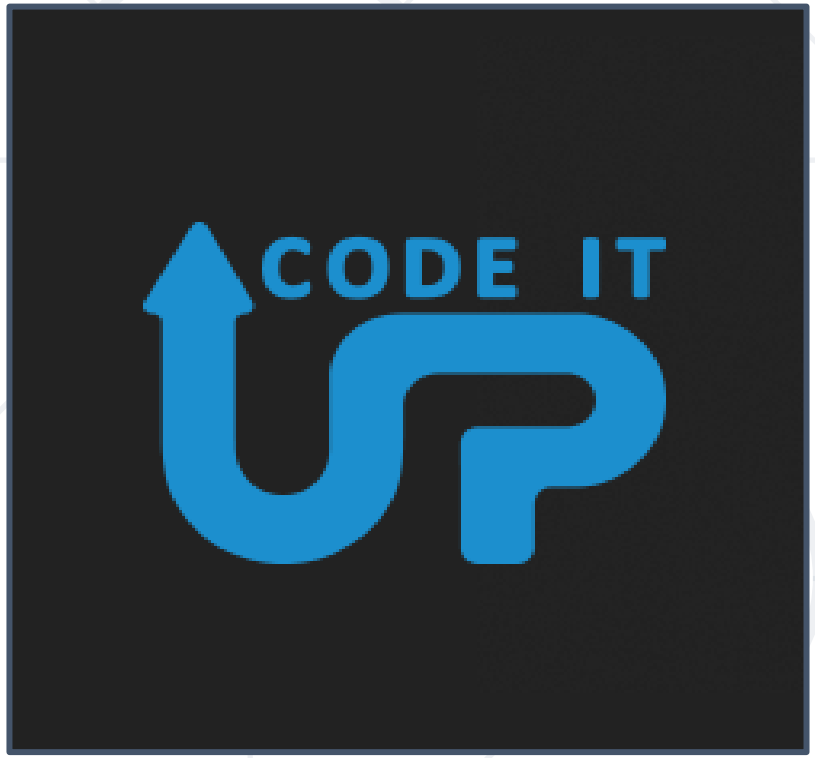
Postbank
Решения за твоето утре

BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

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
- Software University Forums
  - forum.softuni.bg