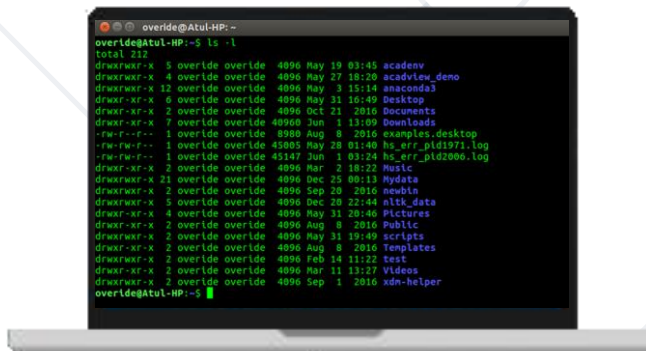


Linux and Linux Shell

Operation System, Linux OS, Linux Shell Commands



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://about.softuni.bg>

Have a Question?



sli.do

#Dev-Ops

Table of Contents

1. Operating System
2. Linux Operating System
3. Linux File System
4. Input / Output Streams
5. Command Sequences





Operating System

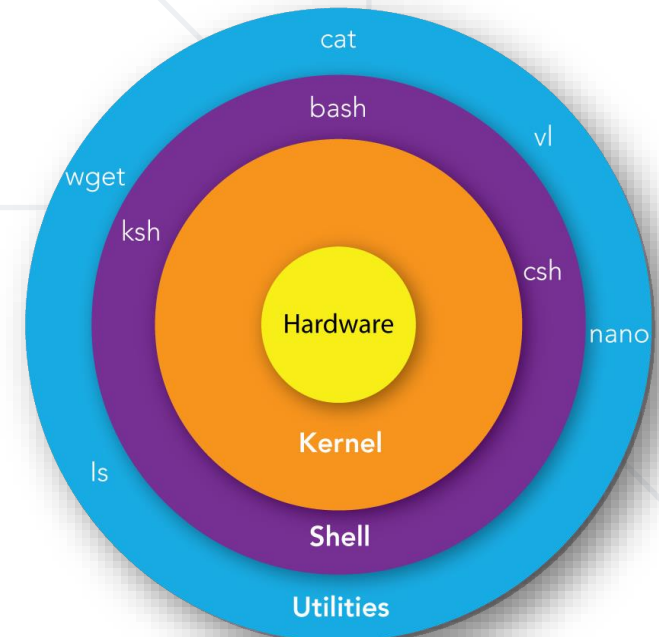
Definition, Functions, Components, Examples

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

- **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.

- **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 **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)

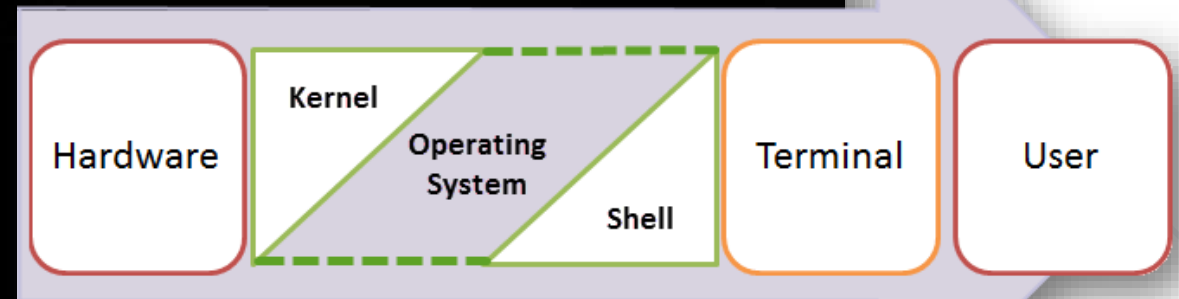


- **OS security** refers to providing a **protection system** for computer system resources and most importantly **data**
- Computers must be protected against unauthorized access, malicious access to system memory, viruses, worms, etc.
- **OS security** may be approached in many ways
 - **Isolation** 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 very popular free, **open-source** operating system
 - <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

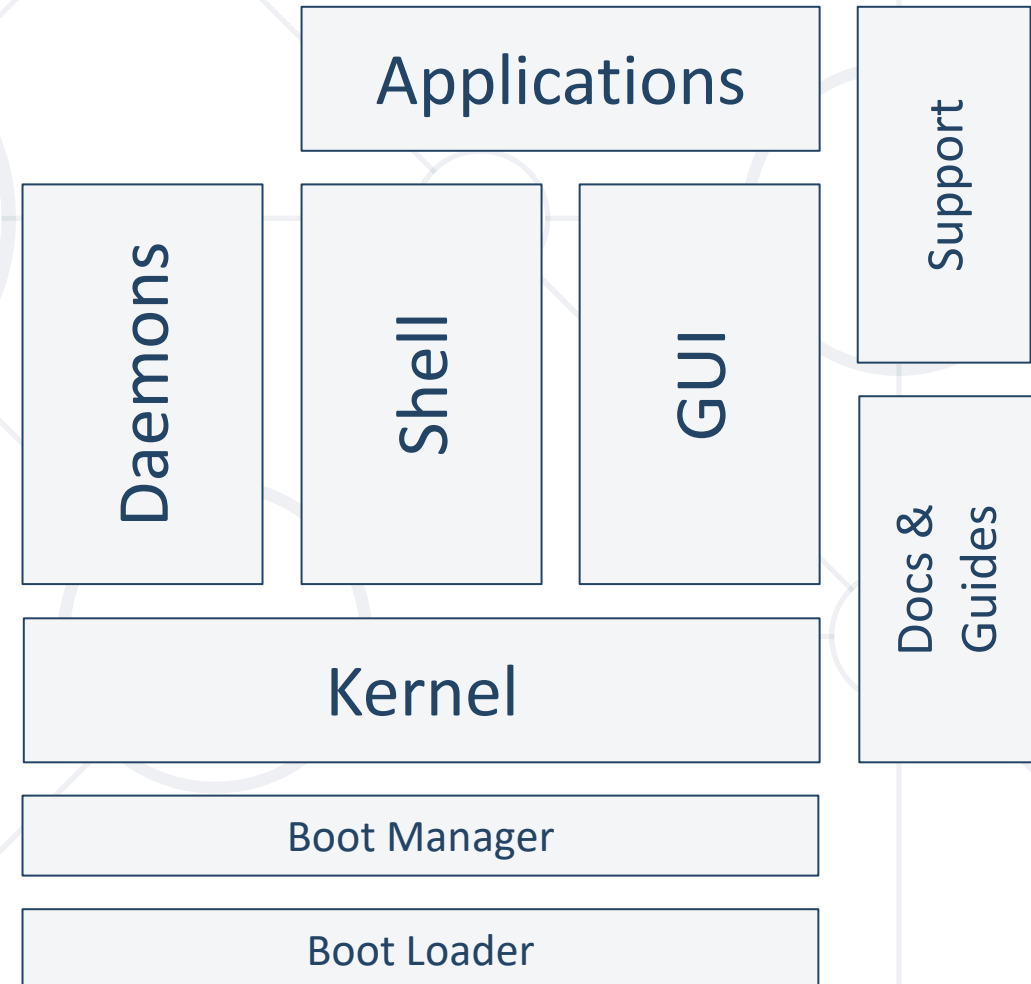
- Linux has many **distributions** (vendors)
- **Differences** in console commands, file locations, package management systems
- Most popular Linux distributions
 - **Ubuntu** – user-friendly, stable, popular
 - **Alpine** – minimal, secure, lightweight
 - **CentOS** – enterprise-grade, stable, secure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

- 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

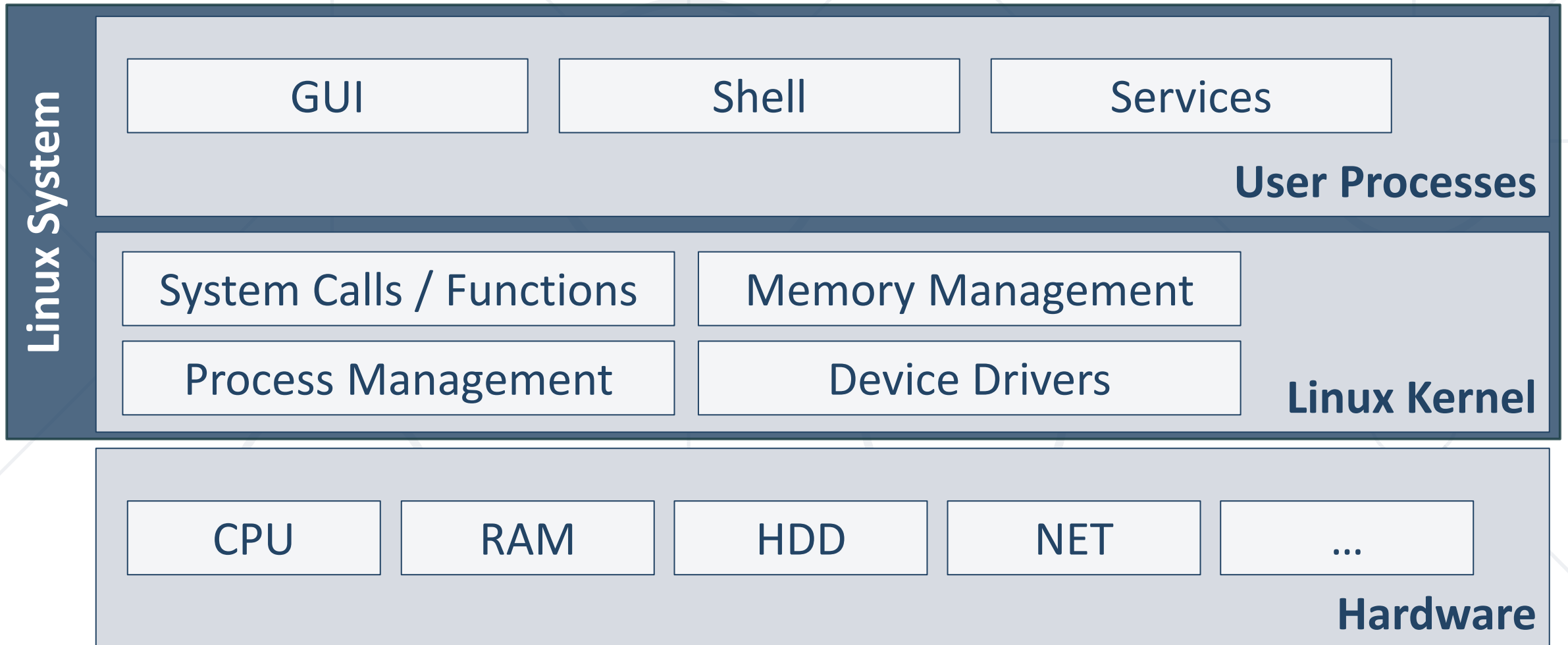
- Availability of apps: some **applications** that work on other OS **do not work in Linux**
- Other OS (like macOS, Windows) have better **usability** (UI and UX)
- **Learning curve**
 - It takes time and effort to master Linux
- **Lack** of standardization
 - Many distributions == many differences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

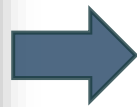
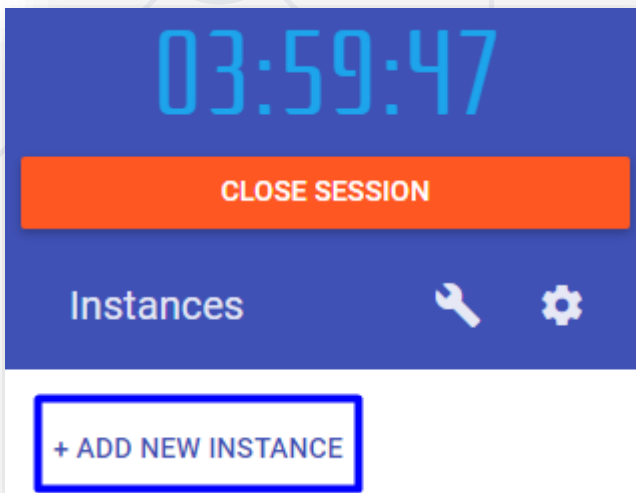




Linux Demo

Simple Commands on the Console

- **Docker Playground** 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 responsibilities.                                       #  
#                                                                              #  
# 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

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

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

Display user's processes with 2 sec delay 5 times

user@host:~\$ top -d 2 -n 5

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

- **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, ...

List files and directories

- Syntax

```
ls [options]
```

- Examples

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

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


- Files and directories
 - Regular (-)
 - Directory (**d**)
- Special files
 - Symbolic link (**l**)
 - Block (**b**)
 - Character (**c**)
 - FIFO pipe (**p**)
 - Local socket (**s**)

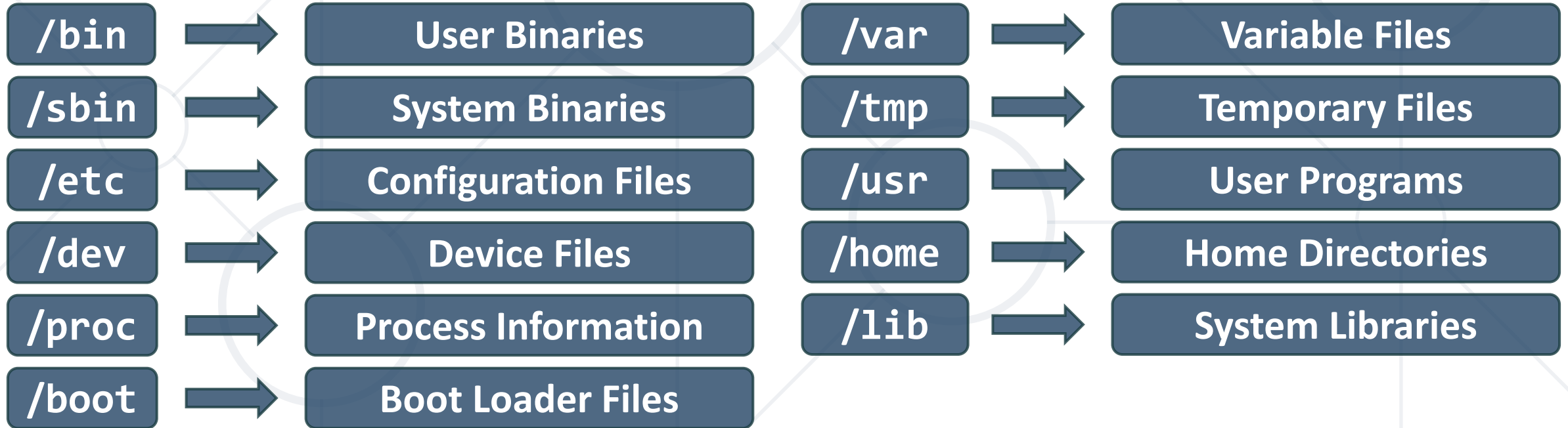
```
drwxr-xr-x 19 root root 4096 Mar 27 11:09 .
drwxr-xr-x 19 root root 4096 Mar 27 11:09 ..
lrwxrwxrwx 1 root root 7 Apr 23 2020 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 23 2020 boot
drwxr-xr-x 9 root root 3000 Mar 27 11:09 dev
drwxr-xr-x 94 root root 4096 Mar 27 12:10 etc
drwxr-xr-x 3 root root 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
```

Examine Root Directory files

■ Syntax

```
ls /
```



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 ../
```

- Create directories

```
mkdir [options] directory [directory ...]
```

- Copy files and directories

```
cp [options] source dest
```

- Move/Rename files

```
mv [options] source dest
```

- Remove files or directories

```
rm [options] file [file ...]
```

- Print the current working directory

```
pwd
```

- Output the first part (10 lines by default) of files

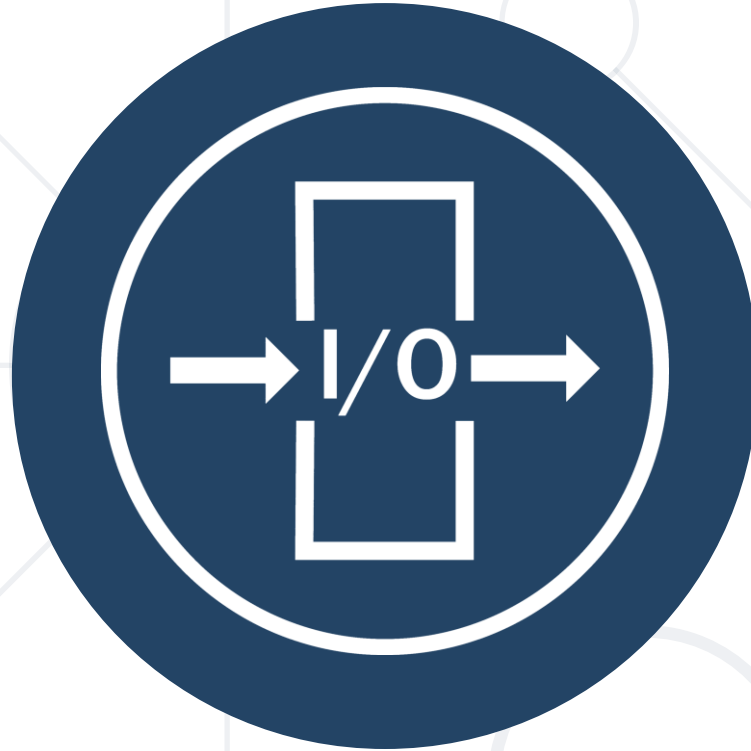
```
head [options] [files]
```

- Output the last part (10 lines by default) of files

```
tail [options] [files]
```

- Read data from the file and return the content as output

```
cat [filename]
```

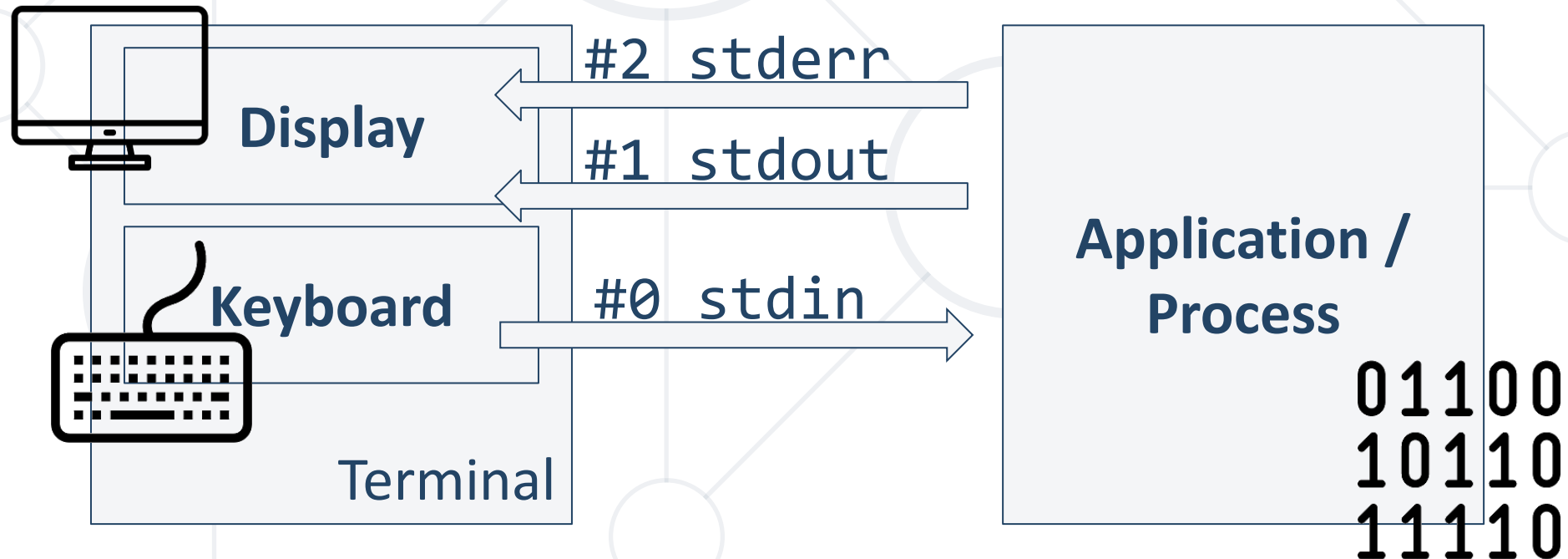


Input / Output Streams

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

The same

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

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

1 == stdout

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

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

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


Redirect Output with Append (>>)

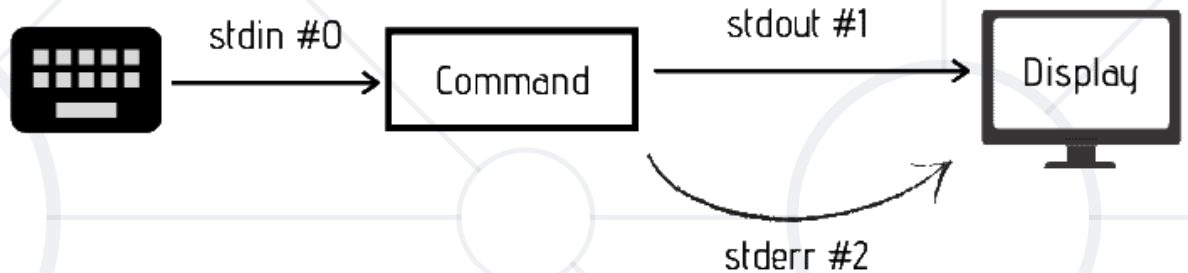
- Redirect output streams (**stdout** or **stderr**) with **target append**
- Example

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

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

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

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

The same



Command Sequences

Execute Multiple Commands. Substitution

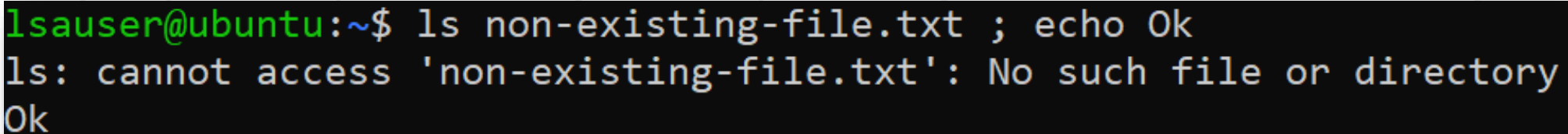
- Execute in order (disconnected)
 - **Sequence**: `command1 ; command2`
- Execute in order (connected)
 - **Pipe**: `command1 | command2`
- Execute conditionally
 - On **Success**: `command1 && command2`
 - On **Failure**: `command1 || command2`

Sequence (;)

- Always execute **next command**
- Example



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



```
lsuser@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
```

```
lsuser@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
```

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

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



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

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



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

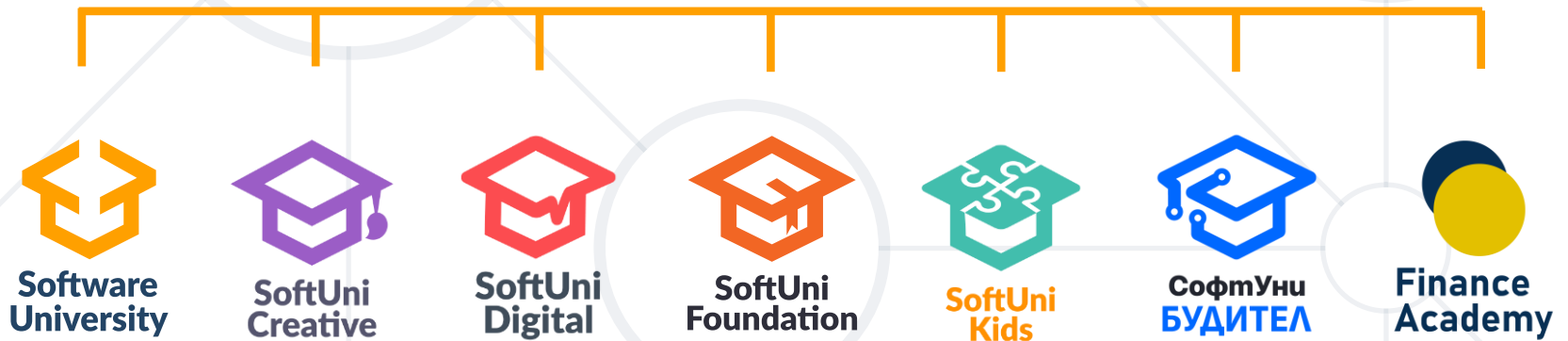

- **Operating systems** manage all of the software and hardware on the computer
- **Linux OS** distributions & file system
- **Shell** definition
- **Command sequences**



Questions?



SoftUni



SoftUni Diamond Partners



- 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



- 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>

