

Hypervisors

A hypervisor is software that creates and runs virtual machines (VMs).

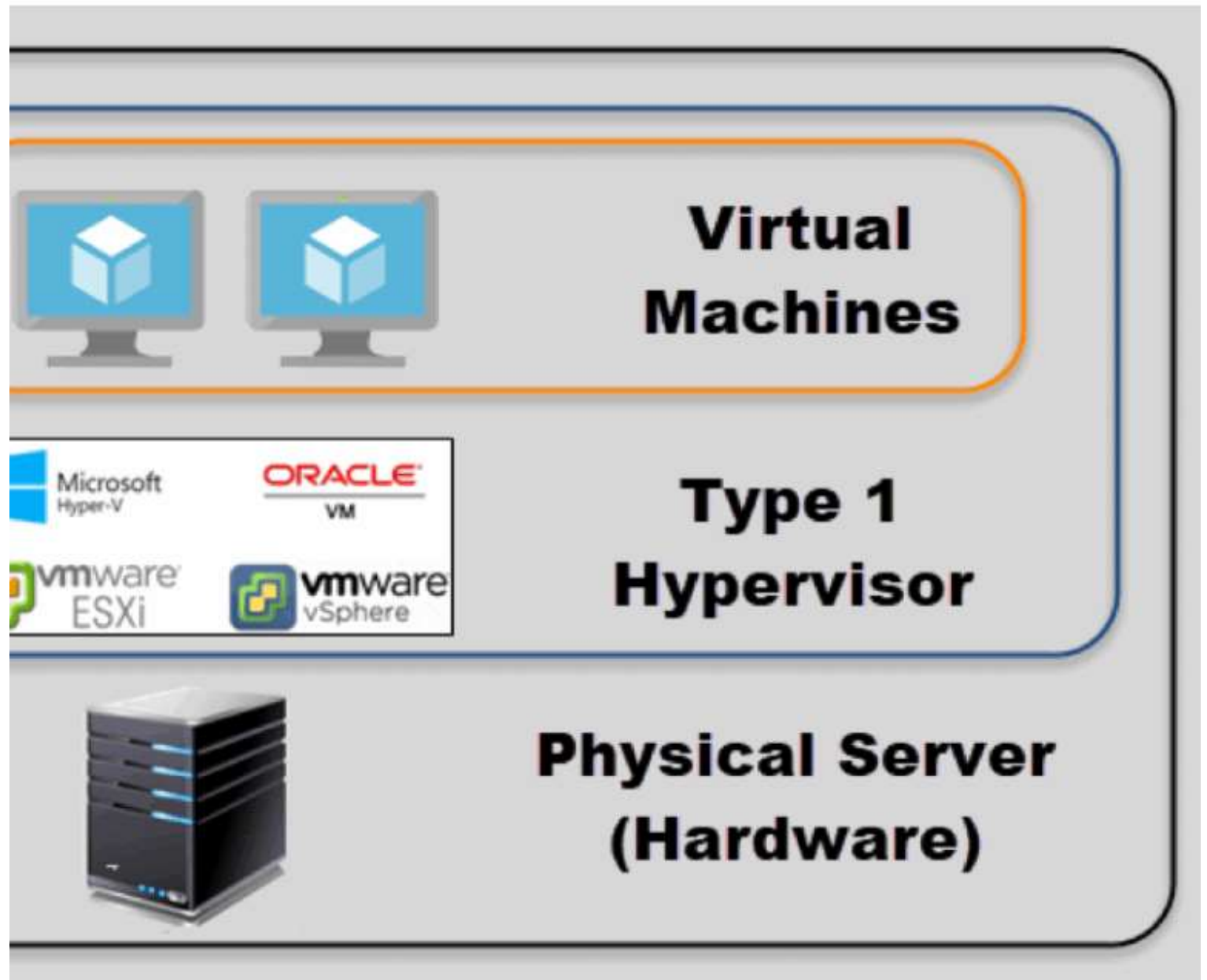
It is computer software, firmware or hardware that creates and runs virtual machines.

A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine

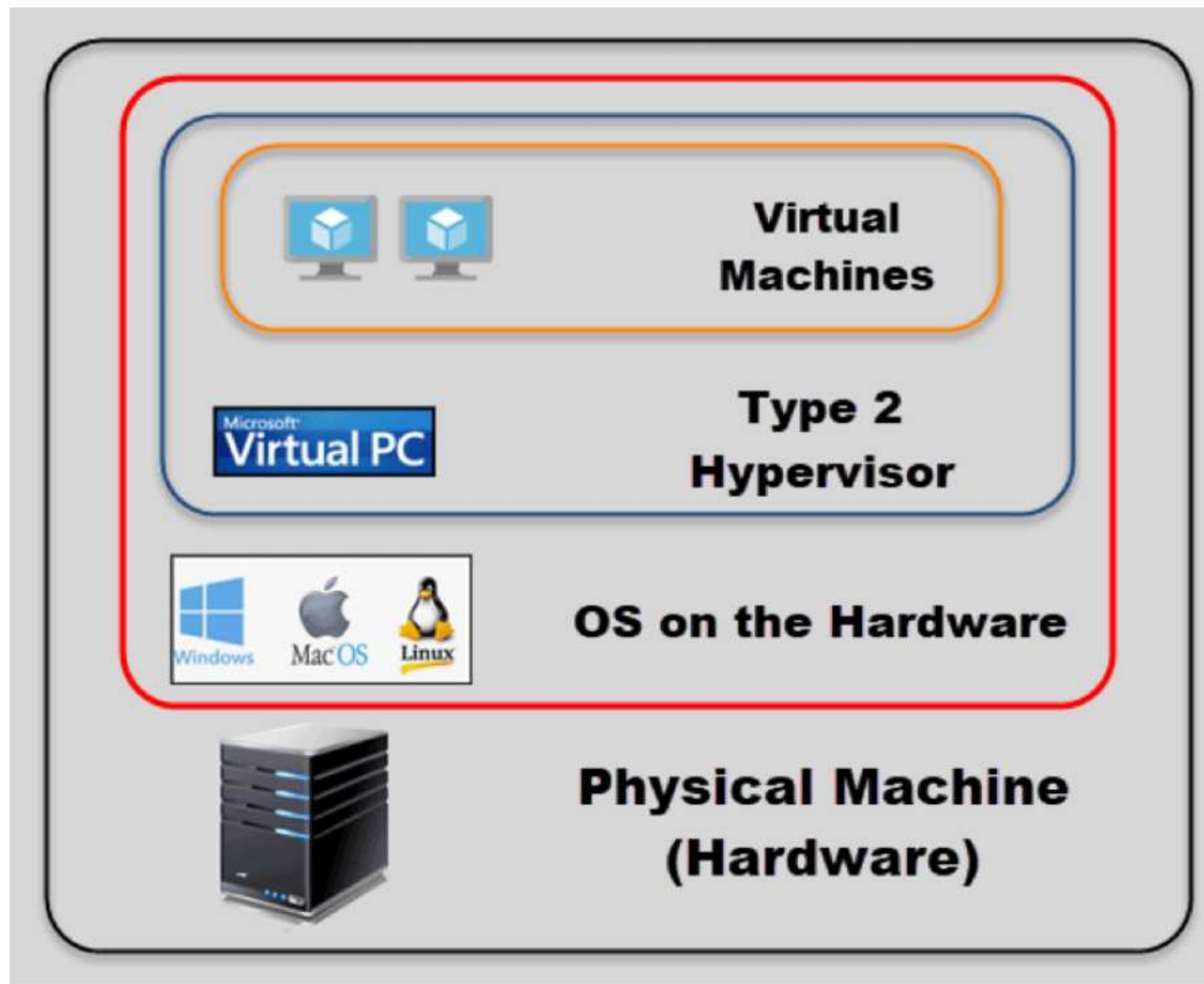
Type of Hypervisor

- Type 1 - is a layer of software we install directly on top of a physical server and its underlying hardware.
- Type 2 - This type of hypervisor runs inside of an operating system of a physical host machine.

Type 1 Hypervisor



Type 2 Hypervisor



Resources required for a Linux VM in LAB

RAM 2GB is enough

Hard disk - 20 GB

Authentic Link to download ISOs for CentOS machines is as follows:

<https://www.centos.org/download/>

What Is an ISO File?

An ISO file, often called an ISO image, is a single file that's a perfect representation of an entire CD or DVD of any Operating System.

SHELL in Linux

SHELL is a program which provides the interface between the user and an operating system. When the user logs in OS starts a shell for user.

Command to check shell

```
echo $SHELL
```

Bash Prompt In Linux

We see the following default prompt in Linux environment: `[username@machinename <current dir>]$`

For example: `[tom@localhost Desktop]$`

Difference between \$ and

The dollar sign \$ means the current user is a regular user.

A root user would be identified with a hash sign #.

For example: `[root@localhost ~]#`

- requires given linux commands to be executed with root privileges either directly as a root user or by use of sudo command

\$ - requires given linux commands to be executed as a regular non-privileged user

User add on the Linux Box

```
[root@localhost ~]# useradd <username>
```

```
[root@localhost ~]# passwd <username>
```

To verify if the user has been added or not – check the contents of the file which contains the user information i.e /etc/passwd

```
[root@localhost ~]# cat /etc/passwd
```

This will display the contents of /etc/passwd file and it should contain the information about the newly added user

Note: cat is used to see the contents of a file

File System in Linux

A file system is a logical collection of files on a partition or disk.

Types of Linux File systems

- Ext Very old and no longer used due to limitations.
- Ext2 first Linux file system that allows two terabytes of data allowed
- Ext3 Ext2 + upgrades and backward compatibility. Issue with this -> The only problem with it that the servers don't use this kind of file system because this file system doesn't support file recovery or disk snapshots.
- Ext4 faster and allow large files with significant speed.
- swap The swap partition is an independent section of the hard disk used solely for swapping; no other files can reside there.

Linux File Hierarchy Structure

- / This is the root directory which should contain only the directories needed at the top level of the file structure
- /bin This is where the executable files are located. These files are available to all users
- /dev These are device drivers
- /etc Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages
- /lib Contains shared library files and sometimes other kernel-related files
- /boot Contains files for booting the system
- /home Contains the home directory for users and other accounts

Linux File Hierarchy Structure – Continued...

- /mnt Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppy diskette drive, respectively
- /proc Contains all processes marked as a file by process number or other information that is dynamic to the system
- /tmp Holds temporary files used between system boots
- /usr Used for miscellaneous purposes and can be used by many users. Includes administrative commands, shared files, library files, and others
- /var Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
- /sbin Contains binary (executable) files, usually for system administration. For example, fdisk and ifconfig utilities

Understanding the file permissions

```
---  ---  ---  
rwx  rwx  rwx  
user group other
```

- The first set of three characters (rwx) is for the owner permissions.
- The second set of three characters (rwx) is for the Group permissions.
- The third set of three characters (rwx) is for the All Users permissions.

Numeric mapping

```
r >> 4
```

```
w >> 2
```

```
x >> 1
```

Understanding the file permissions – Continued..

The 'r' means you can "read" the file's contents.

The 'w' means you can "write", or modify, the file's contents.

The 'x' means you can "execute" the file.

owner – The Owner permissions apply only the owner of the file or directory

group – The Group permissions apply only to the group that has been assigned to the file or directory

others – The other users permissions apply to all the other users on the system

Alphabet mapping

u for user

g for group

o for others

ACLs - Access control list in Linux

Access control list (ACL) provides an additional, more flexible permission mechanism for file systems

To set ACLs -> `setfacl -m "u:user:permissions" /path/to/file`

To check ACLs -> `getfacl <filepath>`

Note: `setfacl` and `getfacl` are used for setting up ACL and showing ACL respectively.

Process in Linux



Processes carry out tasks within the operating system.



A program is a set of machine code instructions and data stored in an executable image on disk and is, as such, a passive entity; a process can be thought of as a computer program in action.



States of Process

Running

The process is either running (it is the current process in the system) or it is ready to run (it is waiting to be assigned to one of the system's CPUs).

Waiting

The process is waiting for an event or for a resource. Linux differentiates between two types of waiting process; interruptible and uninterruptible. Interruptible waiting processes can be interrupted by signals whereas uninterruptible waiting processes are waiting directly on hardware conditions and cannot be interrupted under any circumstances.

Stopped

The process has been stopped, usually by receiving a signal. A process that is being debugged can be in a stopped state.

Zombie

This is a halted process which, for some reason, still has a `task_struct` data structure in the task vector. It is what it sounds like, a dead process.



Initializing a process

A process can be run in two ways:

Foreground Process

Every process when started runs in foreground by default, receives input from the keyboard and sends output to the screen.

When a command/process is running in the foreground and is taking a lot of time, no other processes can be run or started because the prompt would not be available until the program finishes processing and comes out

Background Process

It runs in the background without keyboard input and waits till keyboard input is required. Thus, other processes can be done in parallel with the process running in background since they do not have to wait for the previous process to be completed.

Initializing a process – Continued...

`bg %<job_id>`

The `bg` command is used on Linux in order to send a process to the background

`fg %<job_id>`

The `fg` command is used in order to send a process to the foreground

NOHUP

A process may not continue to run when you log out or close your terminal.

Use nohup to handle this situation.

```
nohup <script>.sh &
```

Appending an ampersand (&) will send the process to the background and allow you to continue using the terminal



Nice value

A nice value of -20 represents highest priority, and a nice value of 19 represent least priority for a process.

Changing Process Priority

Increase the priority

```
nice -n -5 -p <PID>
```

Decrease the priority:

```
nice -n 5 -p <PID>
```

Change the Priority of All Processes Owned by User

`renice -n 5 -u <username>`

```
[root@centos7 ~]# renice -n 18 -u user5
1006 (user ID) old priority 0, new priority 18
[root@centos7 ~]#
```

To find the process ID and parent process ID of the current shell

echo \$\$

echo \$PPID

```
[root@centos7 ~]# echo $$  
15592  
[root@centos7 ~]# echo $PPID  
15574  
[root@centos7 ~]#
```

How to check pid of a process

pidof <process name>

pidof system

```
[root@centos7 ~]# pidof systemd  
1  
[root@centos7 ~]#
```

Listing Running Processes

ps

ps -f >> full listing

ps -elf >> every running process (-e) and a full listing (-f).

Daemons

These are special types of background processes that start at system startup and keep running forever as a service; they don't die

Parent and Child process

- Parent processes – these are processes that create other processes during run-time.
- Child processes – these processes are created by other processes during run-time.
- PID --> PID
- PPID --> Parent ID of the process



Scheduling

Preemptive Scheduling

The process can be interrupted, even before the completion.

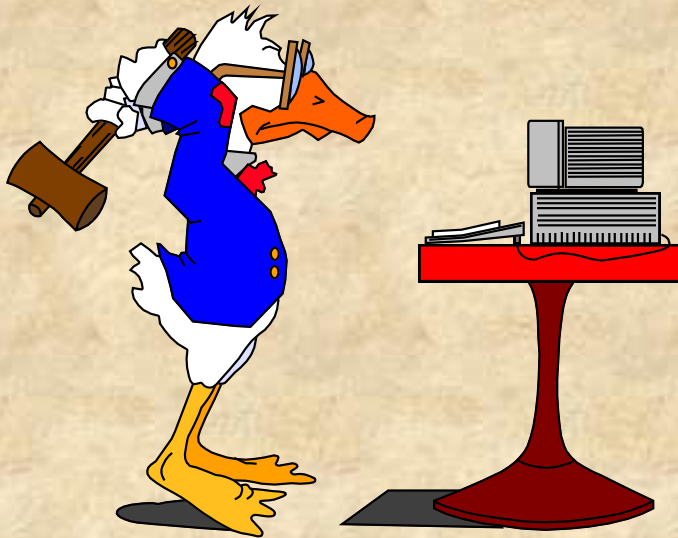
Non-Preemptive Scheduling

The process is not interrupted until its life cycle is complete.

Process Management in Linux

Dr.Hashamdar

Massachusetts Institute of Technology
(MIT)



The Linux Operating System

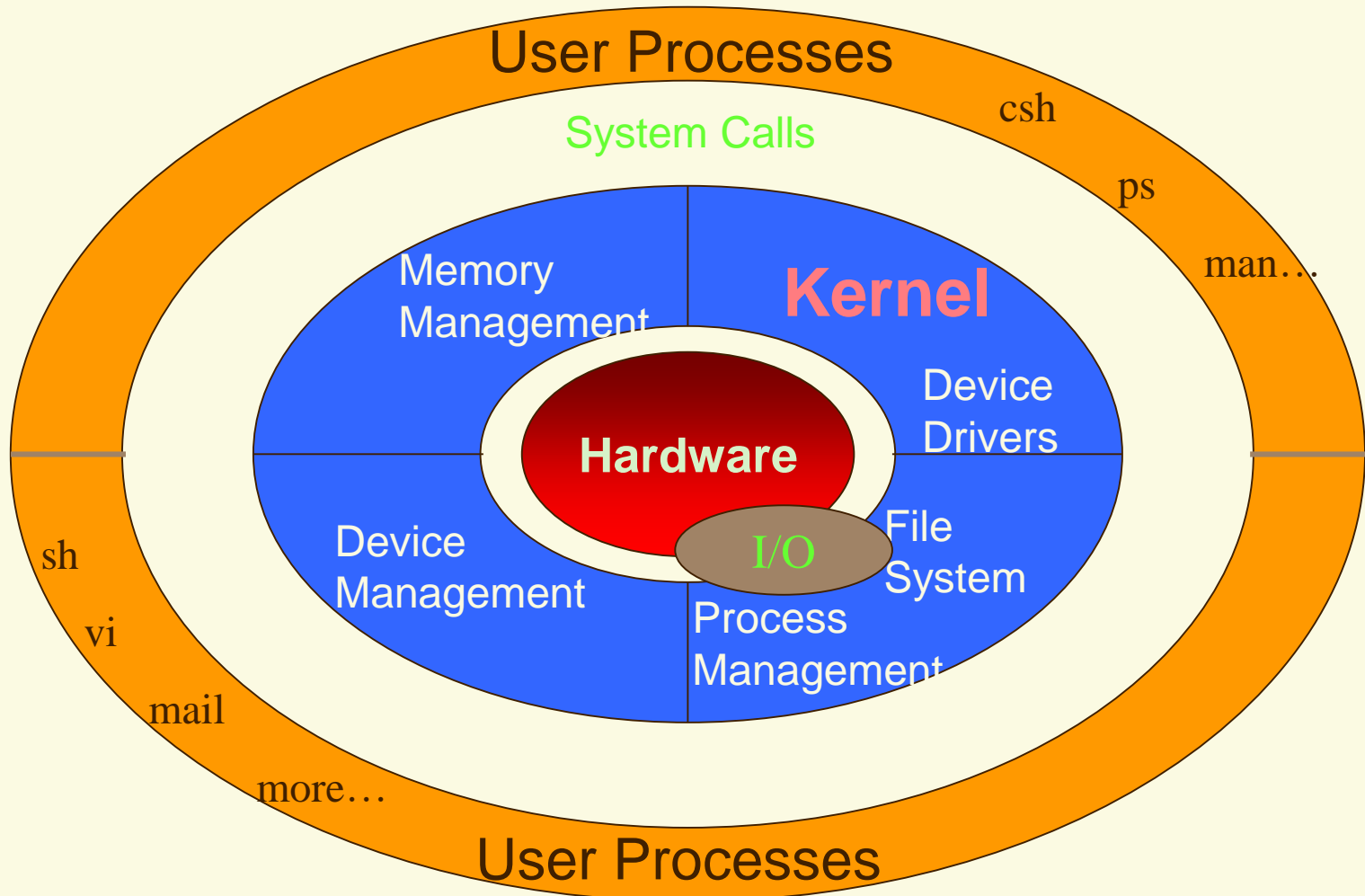


The Linux kernel tracks:

- The location of the process execution (context)
- Which files the process is accessing or has access to
- What users and groups the process belongs to (credentials)
- current directory for the process is
- memory space the process has access to and how it uses it.

system- management programs	user processes	user utility programs	compilers
system shared libraries			
Linux kernel			
loadable kernel modules			

What is a process?



What is a process?

Processes are not just **programs**

- Program instructions plus other components as needed (primarily data)

Sometimes called **jobs** or **tasks**

Process Management

- ❏ UNIX process management separates the creation of processes and the running of a new program into two distinct operations.
 - The fork system call creates a new process
 - A new program is run after a call to exec
- ❏ Under UNIX, a process encompasses all the information that the operating system must maintain to track the context of a single execution of a single program

Process Management



Under Linux, process properties fall into 3 groups

- (i) Process's Identity
- (ii) Process's Environment
- (iii) Process's Context

(i) Process Identity



Three different attributes for a *process identity*

a) Process ID (PID)

b) Credentials

c) Personality

a) Process ID (PID)

- 📄 PID is a unique integer
- 📄 PID cannot be changed by the process
- 📄 Kernel uses PID to track a process
 - Tracks exit status
 - Relationship to other processes

a) Process ID (PID)

 When process exits

- PID is kept until it can be safely discarded.
- PID that is kept in an active status is called a zombie.
- All child processes (now called orphan processes) become child processed of the init process.

(i) Process Identity



Three different attributes for a *process identity*

a) Process ID (PID)

b) Credentials

c) Personality

b) Credentials

☞ Are used to insure security

☞ Made up of

- users
- and groups

☞ User ID (UID) and Group ID (GID)

- either one can be set within a limit
- symbolic user and group names that are mapped to a unique integer value (usually positive).

(i) Process Identity



Three different attributes for a *process identity*

- a) Process ID (PID)
- b) Credentials
- c) Personality

c) Personality

- Personality identifiers allow slight modifications to the semantics of certain system calls.
- “Personalities are primarily used by emulation libraries to request that system calls can be compatible with certain specific flavors of unit.” (Galvin, 710)

(i) Process Identity



Three different attributes for a *process identity*

- a) Process ID (PID)
- b) Credentials
- c) Personality

Process Management



Under Linux, process properties fall into 3 groups

- (i) Process's Identity
- (ii) Process's Environment
- (iii) Process's Context

(ii) Process Environment



The process's environment is inherited from its parent, and is composed of **2 null-terminated vectors**:

- The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself
- The environment vector is a list of “**NAME=VALUE**” pairs that associates named environment variables with arbitrary textual values

(ii) Process Environment

- ❏ Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user-mode system software
- ❏ The environment-variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole

Process Management






Under Linux, process properties fall into 3 groups

- (i) Process's Identity
- (ii) Process's Environment
- (iii) Process's Context

(iii) Process Context

- ❏ The (constantly changing) state of a running program at any point in time
- ❏ The **scheduling context** is the most important part of the process context; it is the information that the scheduler needs to suspend and restart the process
- ❏ The kernel maintains **accounting** information about the resources currently being consumed by each process, and the total resources consumed by the process in its lifetime so far

(iii) Process Context

-  The **file table** is an array of pointers to kernel file structures
 - When making file I/O system calls, processes refer to files by their index into this table
-  The **signal-handler table** defines the routine in the process's address space to be called when specific signals arrive
-  The **virtual-memory context** of a process describes the full contents of the its private address space

Process Management



Under Linux, process properties fall into 3 groups

- (i) Process's Identity
- (ii) Process's Environment
- (iii) Process's Context

Processes and Threads

- Linux uses the same internal representation for processes and threads;
- a thread is simply a new process that happens to share the same address space as its parent


Processes and Threads

- 📄 A distinction is only made when a new thread is created by the **clone** system call
 - **Fork** creates a new process with its own entirely new process context
 - **clone** creates a new process with its own identity, but that is allowed to share the data structures of its parent
- 📄 Using **clone** gives an application fine-grained control over exactly what is shared between two threads

Organization of Table of Processes

- ❏ Each process is referenced by **descriptor**
 - Describes process attributes together with information needed to manage process
- ❏ Kernel dynamically allocates these descriptors when processes begin execution
- ❏ All process descriptors are organized in **doubly linked list**
- ❏ Scheduler used **Macro instructions** to manage and update process descriptor lists as needed

Process Synchronization

 To allow two processes to synchronize with each other, Linux provides:

- **Wait queue:** Linked circular list of process descriptors
- **Semaphores:** Used to solve problems of mutual exclusion and problems of producers and consumers
 - In Linux they contain 3 fields:
 - Semaphore counter
 - Number of waiting processes
 - List of processes waiting for semaphore