

Security, Protection, and Virtualization

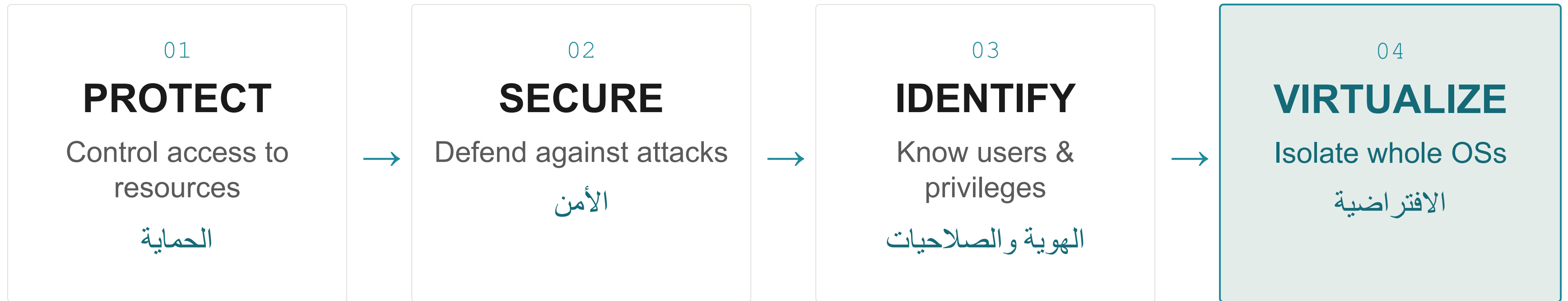
Sections 1.6–1.7 — Controlling access, defending against attack, abstracting the hardware

PROTECT
SECURE
IDENTIFY
VIRTUALIZE

Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

Four Steps



Control who uses resources → defend against attacks → identify users → create **isolated virtual computers** on one machine.

Why Access Must Be Regulated

If a computer system has **multiple users** and allows the **concurrent execution of multiple processes**, then **access to data must be regulated**.

Mechanisms ensure that **files, memory segments, the CPU, and other resources** can be operated on **only by those processes that have gained proper authorization** from the operating system.

Two distinct jobs follow from this: **protection** controls what processes and users **inside the system** may do; **security** defends the system against **external and internal attacks**.

PART 1

Protection

الحماية

Who is allowed to do what?

Why Do We Need Protection?

MANY USERS & PROCESSES SHARE

CPU

Memory

Files

Devices

WITHOUT RULES

Process A reads Process B's memory · a user modifies another's file · a program keeps the CPU forever

Protection = controlling **who** can access **which resource**, and **what operations** they may perform.

الحماية والتحكم في الوصول للموارد

Specify and Enforce

THE OS MUST

Specify the rules

Who is allowed?

Enforce the rules

Actually prevent unauthorized access

ANALOGY · UNIVERSITY BUILDING

Student	classroom ✓
	exam-control room ✗

Professor	more rooms
------------------	------------

Dean	broader permissions
-------------	---------------------

The building is **shared**, but access is **controlled** — exactly what the OS does.

Mechanisms We Already Know

Memory-addressing hardware

Each process stays in its own **address space**.

مساحة العناوين

Own apartment — can't walk into another's.

Timer

```
while(true) { }
```

Interrupt returns control to the OS.

المؤقت

Protects **CPU ownership**.

Device-control registers

A faulty program could damage devices or disturb other users.

Only via **privileged OS code**.

Three Purposes

01 Prevent **unauthorized use**

الاستخدام غير المصرح به

02 Improve **reliability** — catch errors early

الاعتمادية

03 Distinguish **authorized vs unauthorized**

غير مصرح له / مصرح له

Example: a subsystem produces invalid data → protected interfaces let the OS **stop it** before it damages a healthy part.

Protection improves **security** and **reliability**.

PART 2

Security

الأمن

How do we defend the whole system?

Adequate Protection, Still Vulnerable

A system can have **adequate protection** and still be **prone to failure and inappropriate access** .

Consider a user whose **authentication information** — her means of identifying herself to the system — is **stolen**. Her data could be **copied or deleted** even though file and memory protection are **working exactly as designed**.

This is why **security is a separate job** — protection from **external and internal attacks** that target **system vulnerabilities**.

Related, but Not the Same

PROTECTION ASKS

التحكم في الصلاحيات

Who is allowed to do what?

SECURITY ASKS

الدفاع ضد الهجمات

How do we defend against attacks & misuse?

EXAMPLE

Permissions perfectly configured. Attacker **steals a password** . OS sees valid login → allows access.

ANALOGY · DOOR LOCKS

Locks = **protection**.
Professor A's card is **stolen**.
Locks correctly open for the holder.

Protection **worked** — security **failed**: the identity was stolen.

Main Kinds of Attack

Viruses & Worms

دودة / فيروس

Malicious code that spreads. **Virus** attaches to a file; **worm** spreads by itself.

Denial of Service

حجب الخدمة

Flood of fake requests — **real users can't connect** .

Identity Theft

سرقة الهوية

Stolen username + password → attacker **appears as the real user** .

Theft of Service

سرقة الخدمة

Using system capacity **without permission**.

DoS = overload the resource so genuine users can't use it.

Prevention of some of these is considered an **operating-system function** on some systems, while others leave it to **policy or to additional software**. Because security incidents keep rising, **OS security features are a fast-growing area of research** .

PART 3

Identity & Privilege

الهوية والصلاحيات

First: who is this user?

Who Are You — and Prove It

Protection and security both require the system to **distinguish among all its users** . Every user is assigned a **user ID (UID)** — on Windows, a **security ID (SID)**. These numerical IDs are **unique: one per user** .

Identification

التعرف على الهوية

“I am Mostafa.”

Authentication

التحقق من الهوية

“Prove that you are Mostafa.”

UID = User ID · UNIX

SID = Security ID · Windows

IDs are **translated back to user names** when displayed

User logs in

→ **Authentication → UID**

→ **UID attached to processes & threads**

→ **Shown as username**

Why Attach the UID to Processes?

A PROCESS TRIES TO OPEN

`salary.txt`

THE OS ASKS

“Which user owns this process?”

User



UID



Process



Resource request



Permission check

The process **carries the user's UID** — so the OS can check permissions.

- تحمل كل عملية هوية المستخدم حتى يستطيع نظام التشغيل فحص صلاحياتها عند طلب أي مورد.

Groups

Permissions for a **collection of users** · `GID` = Group ID

المجموعات

PROF. AHMED BELONGS TO

EXAM.PDF PERMISSIONS

Faculty Research Committee
Exam Committee

Owner	Read + Write
Exam Committee	Read
Everyone else	No access

Much easier than setting permissions **user by user**.

Privilege Escalation & setuid

Temporary extra permission — **controlled + temporary**, never permanent. تصعيد الصلاحيات

- تخيل أن مستخدمًا عاديًا يحتاج إلى تنفيذ مهمة واحدة لا يسمح له النظام بتنفيذها مباشرة.

UNIX SETUID

Normally runs as UID of **who started it**
setuid runs as UID of the **file's owner**
 → **Effective UID** (معرّف المستخدم الفعّال)

ANALOGY · HOSPITAL

An employee can't enter the **medicine store** . For one job, the manager gives a **temporary card** — used only for that task.

- **Real UID** → من شغل البرنامج
- **Effective UID** → الصلاحيات المستخدمة أثناء التنفيذ



Protection vs Security

Protectionالحماية	Securityالأمن
Controls resource access	Defends against attacks
Who can do what?	Who is trying to break or misuse it?
Mostly internal rules	Internal + external threats
Memory protection, timer, file permissions	Authentication, credentials, policy, security software

Protection = **Permission**

Security = **Defense**

PART 4

Virtualization

الافتراضية

Protect OS A from OS B

One Machine, Many Computers

One physical computer behaves like **several independent computers**.

PHYSICALLY

1 CPU system 16 GB RAM 1 SSD



PRESENTED AS

VM1 → Windows

VM2 → Linux

VM3 → another Linux

Each **VM** thinks it owns its CPU, memory, disk, and network.

آلة افتراضية

VMM, Host and Guest

VMM = Virtual Machine Manager (Hypervisor)

مدير الآلات الافتراضية

Manages virtual CPUs · memory · storage · devices · isolation between guests

EXAMPLES

VMware

VirtualBox

Xen

Host

النظام المضيف

The real underlying machine

Guest

النظام الضيف

The OS running inside a VM

Windows host

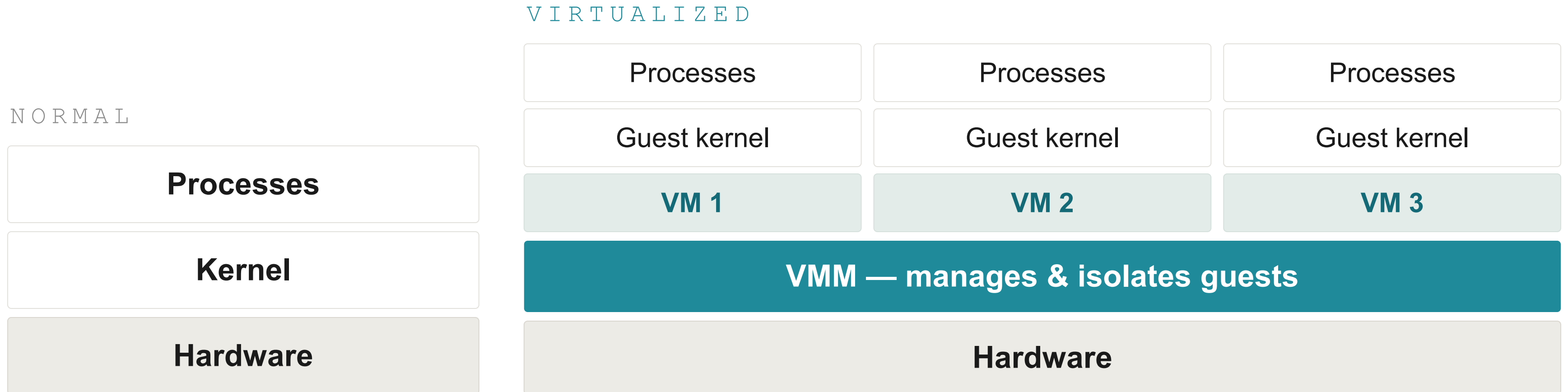


VirtualBox



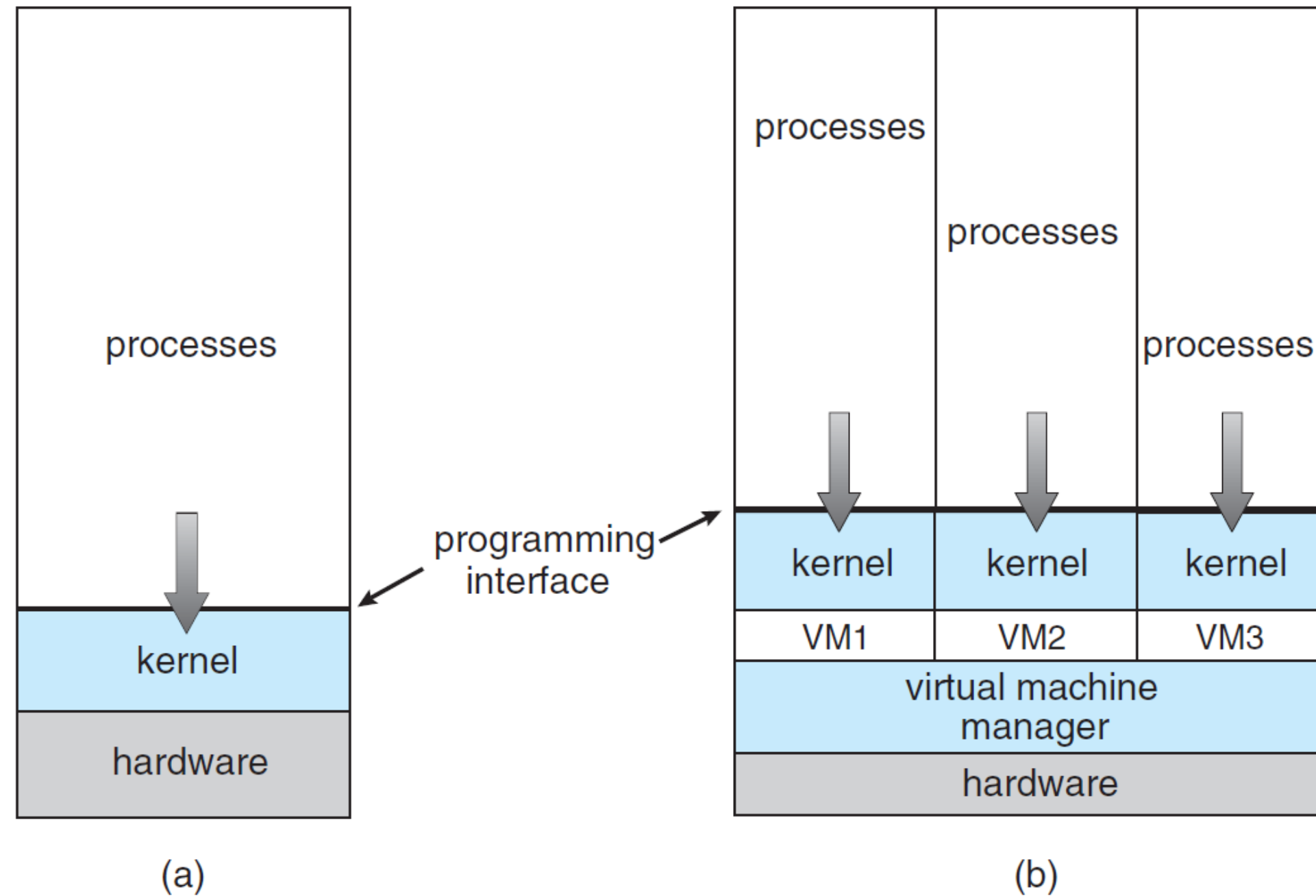
Ubuntu guest

Normal vs Virtualized



Many guest OSs share **one physical machine**; the VMM keeps them apart.

A Single Operating System, and Three Virtual Machines



The **VMM** runs the guest operating systems, **manages their resource use**, and **protects each guest from the others**.

Protection at Machine Scale

SMALL SCALE · EARLIER

Memory protection isolates **processes**
Process A X Process B's memory
Timer shares CPU among processes

MACHINE SCALE · NOW

Virtualization isolates **entire OSs**
Guest A X Guest B's memory/storage
VMM shares resources among VMs

RESOURCE CONTROL

VM1 wants 100% CPU



VMM limits it



VM2 & VM3 still run

Process isolation → **VM isolation**

Security Risk of the VMM

If the **VMM is compromised**, every guest may be affected.

ANALOGY · HOTEL

One room compromised

→ one guest affected

Master-key system compromised

→ **every room** affected

The VMM is a **highly privileged**, attractive target.

Virtualization vs Emulation

Virtualization

الافتراضية

Same CPU architecture → **near-native speed**

e.g. VMware on compatible hardware

Emulation

المحاكاة

Different architecture → instruction translation → **slower**

Guest expects ARM instructions
Real CPU is x86-64

ANALOGY

Two **English** speakers in separate offices

English ↔ Japanese
— every sentence translated

Translation adds **overhead**.

Where Is It Useful?

Laptops & desktops

Another OS without another machine **macOS host**
→ **Windows guest**

Software development

Test Windows, Linux, and different configurations on one system

Data centers & cloud

الحوسبة السحابية

One server → many virtual servers: sharing, management, isolation, utilization

Education

Same VM for every student **Ubuntu + GCC + Python + OS tools**

Virtualization is a **foundation of cloud computing.**

Memory Rules

Protection	Who may do what?
Security	Who may attack us?
Authentication	Are you really who you claim to be?
UID / GID	Which user / group are you?
Privilege escalation	Temporary extra permission
Virtualization	One physical machine behaves like many
VMM / Hypervisor	The manager of those VMs

Protection = control access · Security = defend · Virtualization = isolation for whole machines

What to Take from Sections 1.6–1.7

Protection

Regulating **who can access what within the system** — **specify the controls, then enforce them** .

Security

Defending the system against **attacks and misuse, external and internal alike** .

Virtualization

The **same isolation guarantee at machine scale** — one physical system behaving like many, at **near-native speed** .

All three rely on strong **user identification, authentication, and privilege management** to ensure a **trustworthy computing environment**.

Next: Section 1.8, Distributed Systems.