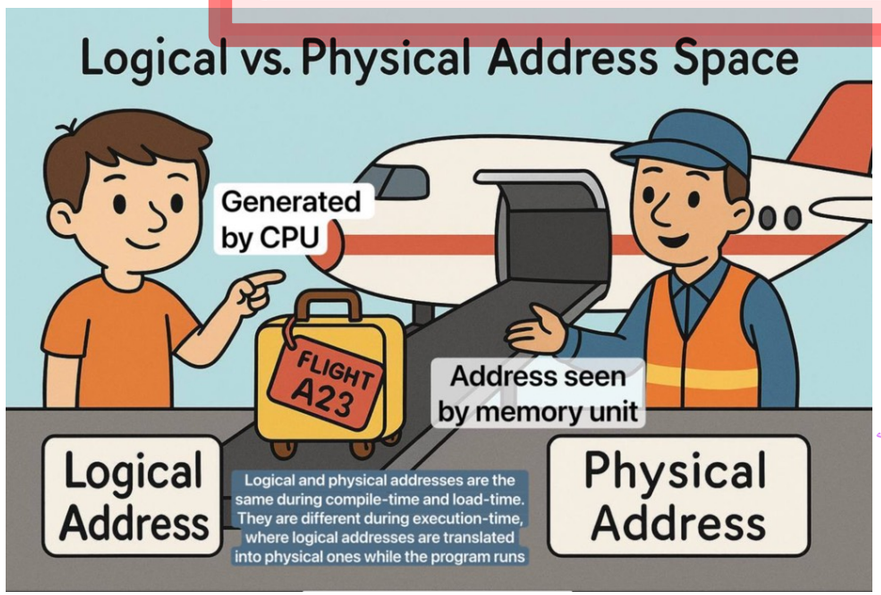


سؤال مقارنة او اختياري

Logical vs. Physical Address Space

مهم/مهم/مهم مقارنة مقالي

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - من اسمه عنوان منطقي "بالمخ" يعني خيالي "virtual"
    Logical address – generated by the CPU; also referred to as **virtual address**
  - من اسمه عنوان فيزيائي يعني "Physical" عنوان حقيقي
    Physical address – address seen by the memory unit
  - Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme



ملخص حلو From Tala

Physical address = logical address : load time & Compile time سلايد-1

Physical address ≠ logical address: Execution time سلايد-2





Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

قطعة هاردوير تحول العناوين المنطقيه الى فيزيائيه
والبرنامج ما يدري عن العناوين الفيزيائيه لأن الـ MMU
تحول العناوين تلقائيًا فهو يتعامل مع عناوين منطقيه



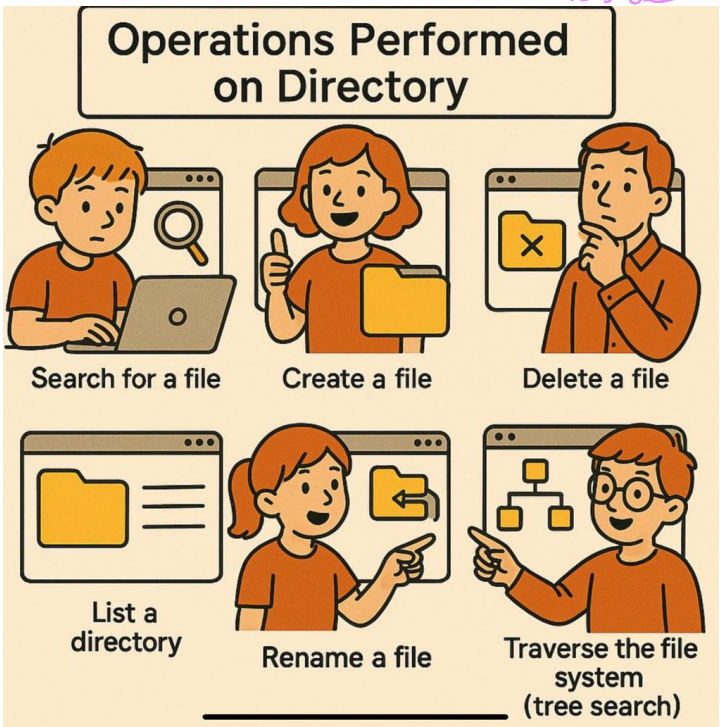


Operations Performed on Directory

list five معم / مهم / تعداد جبناهم

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system (tree search)

From: Tala ملخص رهيب







Direct Memory Access

معهم/معهم جاء معاي

- Used to avoid **programmed I/O** (one byte at a time) for large data movement

يسرع عملية نقل البيانات ويخفف عل الـ CPU

- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
  - Source and destination addresses
  - Read or write mode
  - Count of bytes
  - Writes location of command block to DMA controller
  - Bus mastering of DMA controller – grabs bus from CPU
  - When done, interrupts to signal completion

↳ الـ DMA بـ تسرق الباص لأنه يعتبر طريقة وصول للبيانات بدال CPU يسمح للـ DMA تسرق الباص عشانها بـ تسرع عملية نقل البيانات





Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- New devices talking already-implemented protocols need no extra work
- Each OS has its own I/O subsystem structures and device driver frameworks
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential** or **random-access**
  - **Synchronous** or **asynchronous** (or both)
  - **Sharable** or **dedicated**
  - **Speed of operation**
  - **read-write**, **read only**, or **write only**



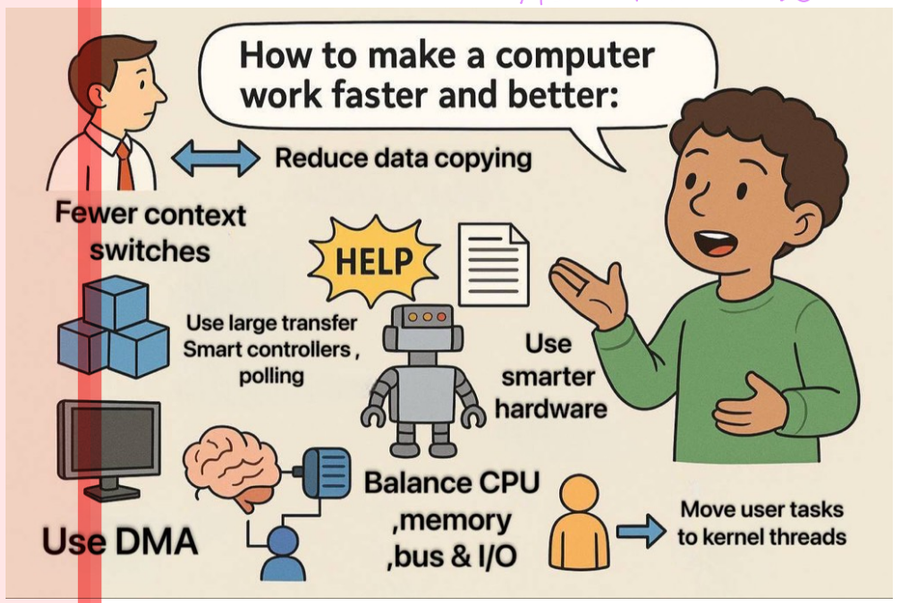


How to Improving Performance in I/O?

مهم جدًا تعداد/مهم/معم معاني

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Use smarter hardware devices
- Balance CPU, memory, bus, and I/O performance for highest throughput
- Move user-mode processes / daemons to kernel threads

From Tala ملخص حيث







Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** (increase) allows user to change to effective ID with more rights





Principles of Protection

مهم مقالي/مهم/مهم تعداد وشرح

مبدأ أقل امتياز

- Guiding principle – **principle of least privilege**
  - Programs, users and systems should be given just enough **privileges** to perform their tasks
  - Limits damage if entity has a bug, gets abused
  - Can be static (during life of system, during life of process)
  - Or dynamic (changed by process as needed) – **domain switching**, **privilege escalation**
  - "Need to know" a similar concept regarding access to data
- Must consider "grain" aspect
  - Rough-grained privilege management easier, simpler, but least privilege now done in large chunks
    - For example, traditional Unix processes either have abilities of the associated user, or of root
  - Fine-grained management more complex, more overhead, but more protective
    - File ACL lists, RBAC
- Domain can be user, process, procedure

البرامج واليوزرز والانظمه
يُعطون أقل الصلاحيات: اللي تخليهم يبون شغلهم

هذا الشي يقلل الاضرار لو صار خلل

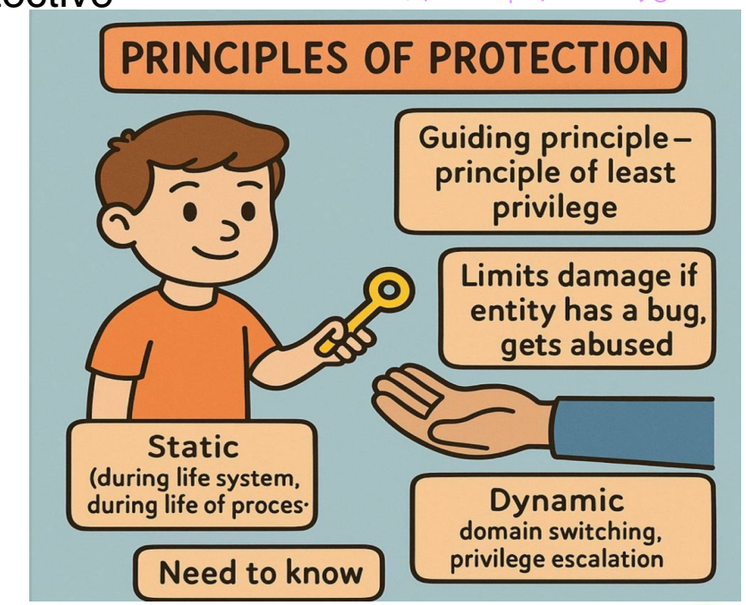
الصلاحيات تكون ثابته طوال فترة انظام أو متغيره حسب الحاجه مثل تبديل المجال أو تصعيد الصلاحيات
تبديل المجال
تصعيد الصلاحيات

نفس الفكره تطبق على كونسبت "الحاجه للمعرفه"

إدارة الصلاحيات الواسعه اسهل لكنها أقل دقه

يقولون جاد بفايل بس ماعرف كيف

From Tala ملخص رهيب





The Security Problem

- System **secure** if resources used and accessed as intended under all circumstances
  - Unachievable
- Intruders (crackers) attempt to breach security
- **Threat** is potential security violation احتمال حدوث اختراق او ضرر
- **Attack** is attempt to breach security محاوله تعليه لضرر الامان
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse





List five of Security Violation Categories?

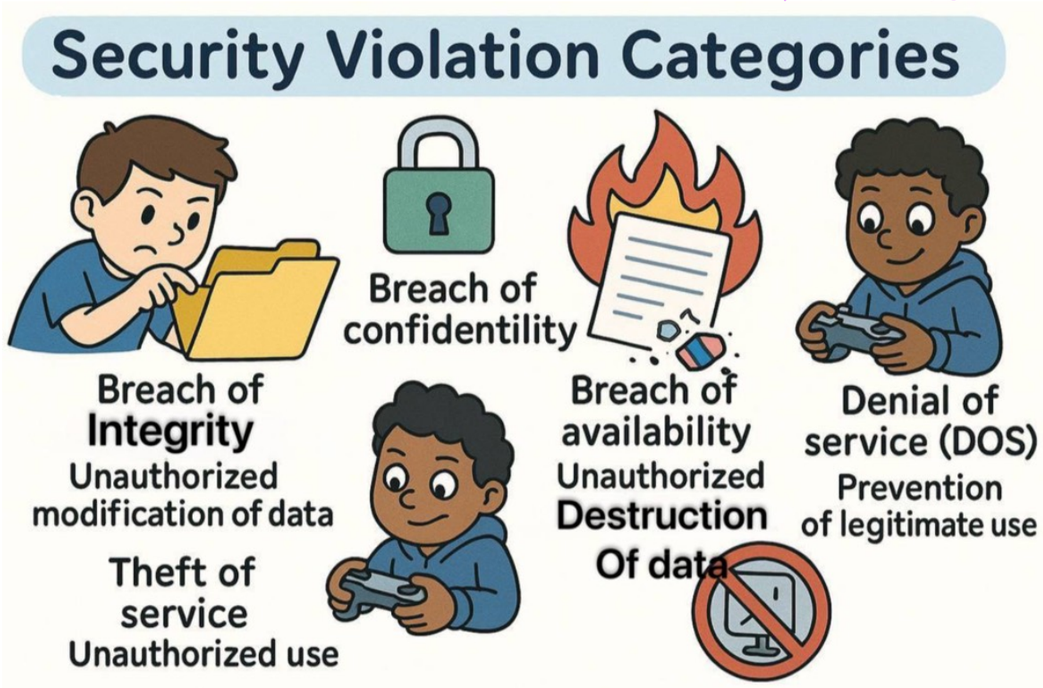
معهم جدًا تعداد / معم / معم

#if system has CIA it will Be Secure

- **Breach of confidentiality** C → قراءة البيانات بدون تصريح
  - Unauthorized reading of data
- **Breach of integrity** I → تعديل البيانات بدون تصريح
  - Unauthorized modification of data
- **Breach of availability** A → تدمير الوصول مثل تدمير البيانات
  - Unauthorized destruction of data
- **Theft of service** سرقة الموارد
  - Unauthorized use of resources
- **Denial of service (DOS)** منع المستخدمين الصريح لهم من استخدام الخدمات
  - Prevention of legitimate use

MCQ

From Tala ملخص رهيب



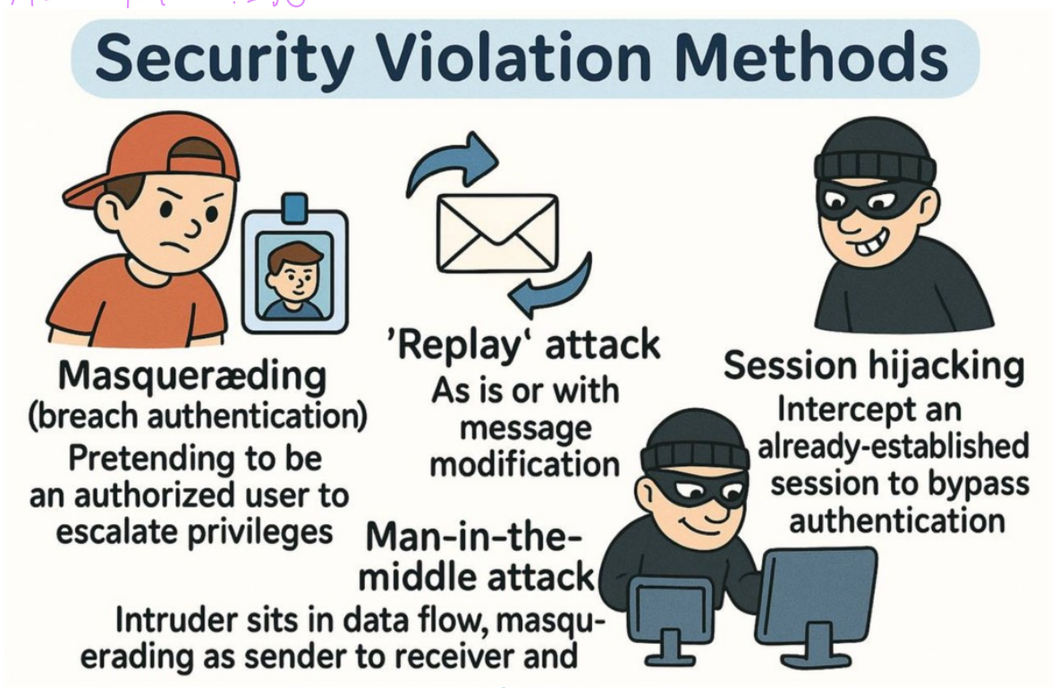




Security Violation Methods

معم / معم

- **Masquerading** (breach **authentication**)
  - Pretending to be an authorized user to escalate privileges
- **Replay attack**
  - As is or with message modification
- **Man-in-the-middle attack**
  - Intruder sits in data flow, masquerading as sender to receiver and vice versa
- **Session hijacking**
  - Intercept an already-established session to bypass authentication







Security Measure Levels

- Impossible to have absolute security, but make cost to perpetrator sufficiently high to deter most intruders
- Security must occur at four levels to be effective:
  - **Physical**
    - Data centers, servers, connected terminals
  - **Human**
    - Avoid **social engineering**, **phishing**, **dumpster diving**
  - **Operating System**
    - Protection mechanisms, debugging
  - **Network**
    - Intercepted communications, interruption, DOS
- Security is as weak as the weakest link in the chain
- But can too much security be a problem?

أعرفوا الفرق بينهم

مثال +





Program Threats (Cont.)

- **Logic Bomb**
  - Program that initiates a security incident under certain circumstances
- **Stack** and **Buffer Overflow**
  - Exploits a bug in a program (overflow either the stack or memory buffers)
  - Failure to check bounds on inputs, arguments
  - Write past arguments on the stack into the return address on stack
  - When routine returns from call, returns to hacked address
    - Pointed to code loaded onto stack that executes malicious code
  - Unauthorized user or privilege escalation





Encryption

- **Encryption** algorithm consists of
  - Set *K* of keys
  - Set *M* of Messages
  - Set *C* of ciphertexts (encrypted messages)
  - A function $E : K \rightarrow (M \rightarrow C)$. That is, for each $k \in K$, $E(k)$ is a function for generating ciphertexts from messages
    - Both E and $E(k)$ for any k should be efficiently computable functions
  - A function $D : K \rightarrow (C \rightarrow M)$. That is, for each $k \in K$, $D(k)$ is a function for generating messages from ciphertexts
    - Both D and $D(k)$ for any k should be efficiently computable functions
- An encryption algorithm must provide this essential property: Given a ciphertext c ∈ C, a computer can compute m such that E(k)(m) = c only if it possesses D(k)
  - Thus, a computer holding *D*(*k*) can decrypt ciphertexts to the plaintexts used to produce them, but a computer not holding *D*(*k*) cannot decrypt ciphertexts
  - Since ciphertexts are generally exposed (for example, sent on the network), it is important that it be infeasible to derive *D*(*k*) from the ciphertexts





Implementing Security Defenses

- **Defense in depth** is most common security theory – multiple layers of security
- Security policy describes what is being secured
- Vulnerability assessment compares real state of system / network compared to security policy
- Intrusion detection endeavors to detect attempted or successful intrusions
  - **Signature-based** detection spots known bad patterns
  - **Anomaly detection** spots differences from normal behavior
    - Can detect **zero-day** attacks
  - **False-positives** and **false-negatives** a problem
- Virus protection
- Auditing, accounting, and logging of all or specific system or network activities





Overview

تقسيم الجهاز الواحد الى عدة بيئات مختلفة مثلاً عندك جهاز واحد تقدر تشغل ويندوز و لينكس وماك على نفس الجهاز

MCQ

- Fundamental idea – abstract hardware of a single computer into several different execution environments → Virtual machine
  - Similar to layered approach
  - But layer creates virtual system (**virtual machine**, or **VM**) on which operation systems or applications can run

الطبقة تعطي الجهاز الواحد يشغل أنظمة أو برامج كأنها أجهزة منفصلة بدون ما تتداخل مع بعض

- Several components

الجهاز الحقيقي

  - **Host** – underlying hardware system
  - **Virtual machine manager** (**VMM**) or **hypervisor** – creates and runs virtual machines by providing interface that is ***identical*** to the host

برنامج يقدم واجهة عمل مطابقة للجهاز الحقيقي

    - (Except in the case of paravirtualization)
  - **Guest** – process provided with virtual copy of the host
    - Usually an operating system

النظام او العملية اللي تشتغل على نسخة من الhost
بالغالب يكون ال OS

- Single physical machine can run multiple operating systems concurrently, each in its own virtual machine





Implementation of VMMs

- Vary greatly, with options including:
  - **Type 0 hypervisors** - Hardware-based solutions that provide support for virtual machine creation and management via firmware
    - IBM LPARs and Oracle LDOMs are examples
  - **Type 1 hypervisors** - Operating-system-like software built to provide virtualization
    - Including VMware ESX, Joyent SmartOS, and Citrix XenServer
  - **Type 1 hypervisors** – Also includes general-purpose operating systems that provide standard functions as well as VMM functions
    - Including Microsoft Windows Server with HyperV and RedHat Linux with KVM
  - **Type 2 hypervisors** - Applications that run on standard operating systems but provide VMM features to guest operating systems
    - Includeing VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox





OS Component – CPU Scheduling

- Even single-CPU systems act like multiprocessor ones when virtualized
  - One or more virtual CPUs per guest
- Generally VMM has one or more physical CPUs and number of threads to run on them
  - Guests configured with certain number of VCPUs
    - Can be adjusted throughout life of VM
  - When enough CPUs for all guests -> VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
  - Usually not enough CPUs -> CPU **overcommitment**
    - VMM can use standard scheduling algorithms to put threads on CPUs
    - Some add fairness aspect

لما يكون عدد الـ CPU غير كافي يجي الـ VMM
ويرتب الوقت بين العمليات عشان يكون فيه توازن





Distributed File System

Discuss Distributed Systym معهم جاء / معهم الدكتور قاله لدفعة الـ 17

- **Distributed file system** (**DFS**) – a distributed implementation of the classical time-sharing model of a file system, where multiple users share files and storage resources

DFS يدير اجهزة التخزين المبعثرة

- A DFS manages set of dispersed storage devices
- Overall storage space managed by a DFS is composed of different, remotely located, smaller storage spaces
- There is usually a correspondence between constituent storage spaces and sets of files
- Challenges include:
  - Naming and Transparency
  - Remote File Access

هو نظام ملفات موزع مبني على فكرة المشاركة الزمنية بحيث يشارك عدة المستخدمين الملفات ومساحات التخزين





Example: TCP/IP

- The transmission of a network packet between hosts on an Ethernet network
- Every host has a unique IP address and a corresponding Ethernet **Media Access Control** (**MAC**) address
- Communication requires both addresses
- **Domain Name Service** (**DNS**) can be used to acquire IP addresses
- **Address Resolution Protocol** (**ARP**) is used to map MAC addresses to IP addresses
  - **Broadcast** to all other systems on the Ethernet network
- If the hosts are on the same network, ARP can be used
  - If the hosts are on different networks, the sending host will send the packet to a router which routes the packet to the destination network





System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in *registers*
    - In some cases, may be more parameters than registers
  - Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system
  - Block and stack methods do not limit the number or length of parameters being passed





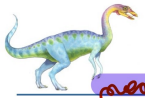
Distributed Computing

- Collection of separate, possibly heterogeneous, systems networked together
  - Network is a communications path
    - Local Area Network (**LAN**). Room, floor, building
    - Wide Area Network (**WAN**). Buildings, cities, countries
    - Metropolitan Area Network (**MAN**). Link building within a city

LAN < MAN < WAN

- Network Operating System provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system (Failure)



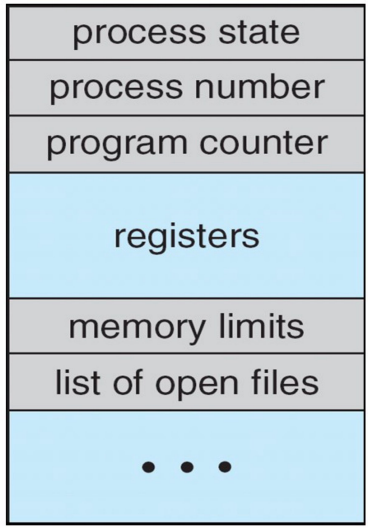


Process Control Block (PCB)

Information associated with each process

(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files





Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

(max) نبي نزيد الشغل والانتاجية

(min) وما نضيع الوقت

Question 2, 1x6=6 Marks

Mark the below Statements True or False:

1. One of the goals of scheduling techniques is to maximize turnaround time.? F [✗ خ]
2. CPU utilization means keeping the CPU as busy as possible. [✓]

السؤال السابق





Semaphore Usage

- **Counting semaphore** – integer value can range over an unrestricted domain
- **Binary semaphore** – integer value can range only between 0 and 1
  - Then a **mutex lock**
- Can implement a counting semaphore ***S*** as a binary semaphore
- Can solve various synchronization problems
- Consider P_1 and P_2 that require S_1 to happen before S_2

```
P1:
    S1;
    signal(synch);
P2:
    wait(synch);
    S2;
```

