Computer Systems Architecture II

CM10195

Content
Introduction to Operating Systems: what they are and what they do, history, ownership and protection of resources.
Processes, scheduling, deadlock.
Interprocess communication (IPC).
Memory, virtual memory and memory management.
Files, file systems.
The networking problem: interoperability via models, standards, protocols and implementations.
The ISO 7-layer model.
The TCP/IP 4-layer implementation and outline of its standards (RFCs) and protocols.
Basic network addressing (MAC, IP, domain names) and the links between them (ARP, DNS).

Ronaldo Butrus

Table of Contents

This document was created with reference to lecture slides by Alan Hayes.
All content is taken directly from these slides and the author cannot guarantee the validity of such content.
Completed 11 May 2023.

Introduction to Operating Systems

- An **operating system** (OS) is a program (often called the kernel or monitor) to:
  - manage (limited) computer resources
  - provide applications programmer with a usable programming interface to access these resources
- The interface that the end user interacts with is a program that uses the operating system.

- Resources are:
  - **hardware:** CPU, memory, disk, network, sound, video, keyboard, mouse, printer, camera etc.
  - **software:** controls hardware

- Resource protection:
  - **security:** prevent one program from corrupting another program or data
  - **authorisation:** ensuring resources are only available to appropriate programs
  - **authentication:** ensuring a program is indeed authorised to use a resource
  - **validation:** ensuring inputs are legitimate and valid

- Operating systems criteria:
  - **responsiveness:** making a program respond snappily or processing packets as they arrive
  - **real time:** certain events must be dealt with in a small fixed period of time
  - **security:** prevent accidental or malicious access or modification of resources
  - **efficient** and **lightweight**
  - **flexible** and not a hindrance to the programmer

- Programming interface:
  - programmer does not have to know hardware details to use it
  - implemented once by hardware expert
  - GUI is not part of the OS (if it were, a bug in the GUI would crash the OS)

- Layer abstraction:
  - **applications:** browser, word processor, game
  - **UI:** command line, windowing, touch
  - **system libraries:** maths, graphics, sound
  - **OS:** Linux, Windows, Android, MacOS
  - **hardware:** PC, phone, PVR, SatNav

- A monitor is a program that is part of the operating system:
  - runs programs and places results appropriately
  - loads application into memory and starts executing it
  - jumps to the next program
  - jumps between programs while waiting for resources by loading multiple programs into memory

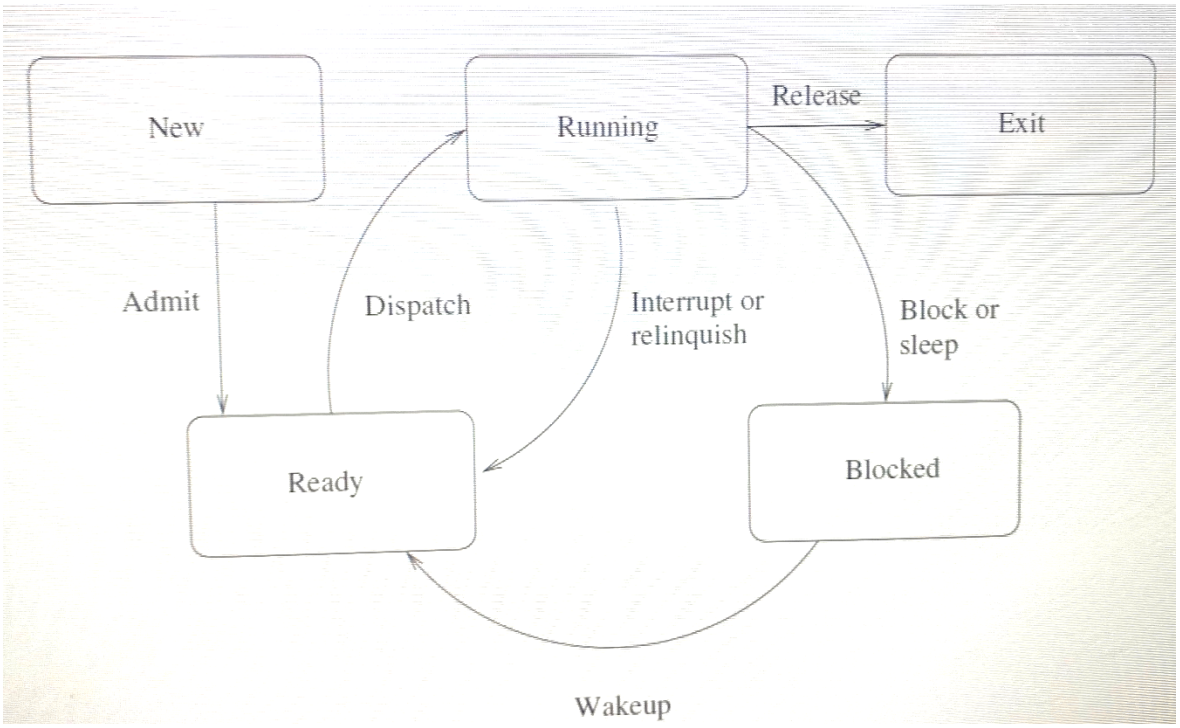
- Multitasking example:
  - monitor starts program 1
  - program 1 requires user input
  - monitor sets up hardware drive
  - monitor starts program 2
  - there is a single stream of control jumping between monitor and several programs

Introduction to Processes

- A **process** (also task or job) describes:
  - executable **code**
  - **data** associated with the code
  - **process control block** containing information OS needs to run it such as:
    - program location
    - data location
    - memory permissions (MMU flags)
    - allocated time
    - used time
    - PC and registers contents

- Process states:
  - **new:** just created
  - **running:** currently executing
  - **ready:** ready to run but other processes are currently using CPU
  - **blocked:** waiting for event or resource to become available
  - **exit:** finished

- OS scheduling decision is making a choice of which process to move between which states.



- Typical transition:
  - OS decides to schedule a process on 'ready' list
  - process is dispatched (marked as running and executes)
  - process moves to 'ready' state due to one of:
    - voluntary suspension
    - interrupt
    - timer interrupt (time slice used)
    - unavailable resource (moves to 'blocked' until available, then back to 'ready')

Introduction to Scheduling

- **Scheduling** is choosing which process to run next:
  - try to give each process its fair share of CPU time
  - no starvation of any process
  - try to make interactive processes respond in human timescales
  - try to give as much computation time as possible to compute-heavy processes
  - ensure critical real-time processes are dealt with promptly

- Scheduling problems:
  - trying to service peripherals in a timely way
  - understanding various requirements (speeds) of hardware
  - trying to distribute work amongst multiple devices (e.g. CPUs and networks)
  - trying to use hardware as efficiently as possible
  - trying to make behaviour predictable
  - trying to degrade gracefully under heavy load

- Measurements:
  - **CPU cycles** used
  - **memory** used
  - **disk** used
  - **network** used

- Results:
  - **throughput:** number of jobs finished in given time
  - **turnaround:** response time
  - **real-time:** must deal with data now else car will crash
  - **money:** paid to prepare data in a time period

- **Cloud services** sell time on their machines based on disk storage, data input, data output and CPU time.

- Scheduling algorithms:
  - **Run until completion:**
    - first-in-first-out (FIFO)
    - good for large amounts of computation (no overheads of multitasking)
    - poor interaction with other hardware
    - not suitable for modern machines
    - basis for large supercomputers
  - **Shortest job first:**
    - no multitasking
    - good throughput
    - long jobs suffer and may be starved
    - difficult to estimate completion time
  - **Run until completion plus cooperative multitasking**
    - weak multitasking
    - uses round-robin or similar to choose another task
    - poor interaction with other hardware
    - can starve other processes
    - hard to write 'good citizen' programs
  - **Round robin:**
    - given each process, in turn, a fixed time slice
    - multitasking
    - equal priority
    - no starvation
    - better interactivity
    - not good for interactive or real-time (may have to wait long time for slice)
    - more suitable to systems of processes of equal importance
  - **Shortest remaining time:**
    - time slice, pick next process by estimating shortest time remaining
    - good for short jobs
    - good throughout
    - long jobs can be starved
    - still hard to estimate times
  - **Least completed next:**
    - pick process that has consumed least amount of CPU time next
    - all processes make equal progress in terms of CPU time
    - interactive processes get good attention as they use relatively little CPU
    - long jobs can be starved by many small jobs

- **Deadlock** is when a program A is blocked awaiting a resource that will not be available until a program waiting on another resource which is blocked by program A:
  - can happen on any shared resource that requires exclusive access
  - can be more than two processes involved
  - only possible if 4 Coffman Conditions are met:
    - **mutual exclusion:** only one process can use a resource at a time
    - **hold-and-wait:** a process continuous to hold a resource while waiting for other resources
    - **no pre-emption:** no resource can forcibly be removed from a process holding it
    - **circular wait:** circular chain of processes where each holds a resource that is needed by the next in the circle
  - **indefinite postponement** is when a process never manages to get all the resources it needs

Inter Process Communication

- **Synchronisation** involves communication between processes.
- **Inter-process communication (IPC)** must be supported by the OS.

- IPC mechanisms:
  - **files:** using a common file to share data between processes
    - must agree on file name
    - might have to poll a file until a change is detected
    - problems with simultaneous access
    - requires authorisation to read/write
    - doesn't scale well to large numbers of files or processes
    - considered for large amounts of data
    - an I/O intensive process is less likely to need to be scheduled soon but would like a fast response without a data read/write
    - a CPU intensive process would be scheduled soon but is not sensitive to delay due to data read/write
  - **pipes:** connects two processes, taking output from one as input to another
    - fixed size (typically 4096 bytes)
    - independent unidirectional reading/writing (FIFO)
    - provides synchronisation
    - too much or too little data can affect scheduling of the tasks
    - simple and efficient
    - powerful and commonly used
    - implemented as buffer held by kernel where read/write is done by a syscall
  - **sockets:** pipes between processes on different machines
    - basis of Internet
    - bidirectional between two processes
  - **shared memory:**
    - like files, using memory (faster)
    - fast if supported by mechanisms like semaphores to flag when data is ready
  - **semaphores:** a variable whose value can only be accessed and altered by two operations V and P
    - let S be a semaphore variable
    - S is set to 1 if attached resource is free
    - S is set to 0 if attached resource is busy
    - P(S) requests access to resource protected by S (and sets to busy)
    - V(S) releases resource protected by S
    - If multiple processes attempt a P(S) simultaneously only one will succeed
    - must work if process is rescheduled between a test and a change of value
    - must work if multiple parallel processors access semaphore simultaneously
    - each semaphore only requires a few bytes of shared memory
    - small and fast given hardware and software support
    - used in OSs and user programs to protect critical resources

Introduction to Memory

- **Memory swapping** is pre-empting a process and copying the contents of the memory it occupies to somewhere else (usually a disk) and then returning it later in the same state.
  - this is a slow operation with a large overhead, making it a tradeoff between speed and process size
  - when a process requests memory allocation that the OS cannot satisfy, it can try memory swapping a (preferably blocked) process
  - variants:
    - only one process in memory at one time (swapped entirely when scheduled)
    - swapping of processes (only marginally harder and fits well with scheduling)
    - swapping parts of a process (difficult to determine dependencies)
  - an I/O intensive process is less likely to need to be scheduled soon but would like a fast response without a memory swap
  - a CPU intensive process would be scheduled soon but is not sensitive to delay due to memory swap

- **Memory paging** is separating data into equally sized chunks, each page being a contiguous area of memory, and copying pages to and from disk.
  - memory fragmentation is caused by irregular sizes of processes/partitions
  - hardware is designed to make copying pages in/out of memory efficient
  - a virtual address is a per-process fictional address, only visible to the process
  - the OS has page tables to map each virtual address to its physical address
  - for each data transfer or instruction execution, we need to read the page table, calculate the physical address and access the physical address
  - the translation lookaside buffer (TLB) is a piece of hardware that maintains a copy of the virtual-physical mappings and can translate easily between them
  - if a memory access is for a page that has been ‘paged’ out, i.e. a TLB miss, the OS must choose which mapping to remove to make space for the new map:
    - **random:** pick random page
    - **FIFO:** poor, as pages that have been around for a long time tend to be needed
    - **LRU** (least recently used): good, but needs to keep track of time
    - **LFU** (least frequently used): increments counter on each page, not so good as pages just brought tend to have a low count
  - each page has permissions attached to it (read, write, execute)
  - an unauthorised access causes an interrupt and the OS sends a segmentation violation signal to the process

Introduction to Networks and Distributed Systems

- A **network** is a connection of interconnected computers together with the hardware and software used to connect them.
- Networks allow users to share data and peripheral devices.
- A **distributed system** is a system in which several autonomous processes interact by means of data transfer in order to cooperate to achieve a goal.

- Supermarket network:
  - each checkout terminal has a mini computer with a laser barcode reader
  - each mini computer is identified by an address
  - these are connected via a network
  - a computer with access to a store-specific database containing the price and stock of each product is connected to the network
  - this is connected to the company's central computer, which is connected to every supermarket in the region
  - changes in the price can be controlled from the computer in the head office
  - each store can order new stock by placing an order to the head office from their computer

- **Client-server model:**
  - data is stored on servers (powerful computers)
  - typically a server has access to data in a database
  - clients are users who access this data through the server
  - clients access data on their client machine via a network
  - communication takes the form of a client request and a server response

- **SWERN** (The South West England Regional Network):
  - one of the regional networks that make up JANET, the UK's research and education network
  - owned and operated by education institutions in the West of England
  - handed over to JANET in 2023

- **JANET** (Joint Academic Network):
  - high-speed network for UK research and education
  - provided by Jisc, a not-for-profit company
  - serves 18 million users
  - busiest National Research and Education Network in Europe by data volume
  - all further and higher education organisations in the UK are connected to JANET

- **Traceroute** is a tool that provides a map of where data packets travel from source to destination.

- The **Internet** is the world-wide collection of networks.
- An **internet** (internetwork) is a collection of networks.
- An **intranet** is a collection of networks belonging to a single organisation.
- The **world wide web** is the billions of pages that reside on the Internet and can be accessed through it.

OSI Model

- The **Open Systems Interconnection (OSI)** Reference Model deals with systems that are open for communication with other systems.
- It has seven layers developed with the following principles:
  - a layer should be created where a different layer of abstraction is needed
  - each layer performs a well-defined function
  - the function of each layer should contribute towards defining internationally standardised protocols, its boundaries chosen to minimise the information flow across the interface
  - the number of layers should be large enough that distinct functions should not be in the same layer and small enough that the architecture does not become disorganised

- Layers:
  - **physical** layer:
    - concerned with transmitting raw bits over communication channel
    - design issues are making sure that when one side sends a bit with value 1 it is received as such, i.e. value of high voltage, length of time, direction
  - **data link** layer:
    - take raw transmission facility and transform it into a line that appears free of transmission errors to the network layer above
    - breaks data into data frames, transmits the frames and processes acknowledgment frames sent back by the receiver
  - **network** layer:
    - concerned with controlling operation of subnet
    - determines how data packets are routed from source to destination
    - controls congestion
  - **transport** layer:
    - splits data from session layer into smaller units and pass into network layer
    - true source-to-destination (end-to-end), i.e. a program on source machine carries on conversation with similar program on destination machine
  - **session** layer:
    - allows users on different machines to establish sessions
    - manages dialogue between sessions
    - sessions allow unidirectional and bidirectional traffic, the session layer controls direction if unidirectional
  - **presentation** layer:
    - concerned with syntax and semantics of information transmitted
    - manages different representations (encodings)
    - manages data compression and cryptography
  - **application** layer:
    - contains protocols that are commonly needed, e.g. file transfer
    - different file systems have different file naming conventions
    - transferring files between two different systems requires handling these incompatibilities
    - e.g. web browser, email browser

- Each layer attaches that layer's header to the front of the data and passes it onto the next layer, from the application layer to the data link layer.
- These headers are stripped one by one by corresponding layers at the destination.
- Though actual data transmission is vertical, each layer is programmed as if it were horizontal (e.g. transport layer acts as though it communicates directly with transport layer on destination machine).

- OSI terminology:
  - **entities:** active elements in each layer (hardware or software)
  - **peer entities:** entities on the same layer on different machines
  - **service providers:** entities in layer N+1 implement service used by layer N
  - **service user:** layer which uses a service provider's service
  - **service access points (SAPs):** layer N SAPs are the places where layer N+1 can access the services offered

- Interface Data Unit:
  - in order for two layers to exchange information there has to be an agreed set of rules about the interface
  - at a typical interface the layer N+1 entity passes an **Interface Data Unit (IDU)** to the layer N entity through the SAP
  - The IDU consists of a **Service Data Unit (SDU)** and some control information.
  - The SDU is the information passed to the peer entity and then up to layer N+1, needed to help the lower layer to its job, e.g. the number of bytes in the data.

- **Connection-oriented service** (e.g. telephone):
  - service user first establishes a connection
  - then uses the connection
  - then terminates the connection
  - first-in-first-out (FIFO) connection

- **Connectionless-oriented service** (e.g. post):
  - each message carried full destination address and is routed through system independent of others
  - not necessarily FIFO

- A service is specified by a set of **primitives** (operations) available to a user or other entity to access the service, which tell the service to perform some action or report on an action taken by a peer entity.
- Classes of service primitives:
  - **indication:** an entity is to be informed about an event
  - **response:** an entity wants to respond to an event
  - **confirm:** an entity is to be informed about its request
  - **request:** an entity wants the service to do some work

The Internet Model

- Layers:
  - **host-to-network** layer:
    - capable of sending and receiving **Internet Protocol (IP)** packets
    - *corresponds to OSI physical and data link layers*
  - **Internet** layer:
    - IP handles movement and routing of data packets
    - IP is an unreliable protocol (doesn't guarantee delivery)
    - *corresponds to OSI network layer*
  - **transport** layer:
    - connection-oriented protocol is **Transmission Control Protocol (TCP)**
    - connectionless protocol is **User Dataram Protocol (UDP)**
    - *corresponds to OSI transport layer*
  - **application** layer:
    - Internet applications must deal with presentation issues
    - typically uses **Hypertext Transfer Protocol (HTTP)** and **Simple Mail Transfer Protocol (SMTP)**
    - *corresponds to OSI session, presentation and application layers*

- Reliable vs unreliable protocols:
  - sometimes reliable communications may not be available, e.g. Ethernet
  - sometimes better to lost packet than re-request and resend packets, e.g. videos
  - wireless communications are unreliable

- TCP:
  - provides reliable end-to-end byte stream over unreliable network
  - UDP is unreliable equivalent

- TCP service model:
  - TCP service obtained by both sender and receiver creating **sockets** (endpoints)
  - each socket has an **address** (consists of host IP address and its local 16-bit **port**)
  - connection is established between sockets on sending and receiving machines
  - port numbers below 1024 are reserved for standard services (**well-known** ports)

- Email transmission example:
  - start with text from email
  - email application encodes text
  - email application adds envelope header
  - TCP adds its header (reliability)
  - IP adds its header (routing)
  - Ethernet adds header (local routing) and trailer (checksum)
  - bits are transformed using a 4B/5B encoding to smooth bit patterns and sent using three-level electrical coding MLT-3 (physical)

- OSI vs TCP/IP:
  - different **number of layers**
  - OSI network layer supports **connection-oriented** and **connectionless** services; TCP/IP network layer only supports **connectionless** services
  - OSI model **built on concept of services, interfaces and protocols**; TCP/IP wasn't
  - OSI model first devised **before protocols**; TCP/IP devised **after protocols**
  - both based upon **layers of independent protocols**
  - **layer functionality** is similar

Network Security Fundamentals

- **Network security** is the control of unwanted intrusion into, use of or damage to communications on a network.
- Network administrators must:
  - protect against compromise
  - maintain high performance
  - keep costs to a minimum

- **Trust** is confidence users will act in accordance with the organisation's security rules.
- Users trust that other users will not attempt to violate the stability, privacy or integrity of the network and its resources.

- **Permission** is the authorisation to access an asset on a network.
- **Privilege** is the ability given to access the asset.
- If someone violates this trust then their access is revoked.

- **Third-party trust system** example:
  - user communicates with e-commerce server
  - user and server have common trustworthy **certificate authority**
  - user examines digital certificate issued to web server from a certificate authority
  - user can trust that identity of server is valid

- Primary security objectives:
  - **confidentiality:** ensures data is not intentionally or unintentionally disclosed to anyone without valid access
  - **privacy:** protects confidentiality, integrity and availability of personally identifiable data
  - **integrity:** data remains consistent both internally and externally
  - **nonrepudiation:** prevents user from being able to deny having performed an action (commonly using public-key cryptography)
  - **availability:** protection against downtime, loss of data and blocked access

- Network security goals:
  - ensuring confidentiality of resources
  - protecting integrity of data
  - maintaining availability of IT infrastructure
  - ensuring privacy of personally identifiable data
  - enforcing access control
  - monitoring IT environment for violations of policy
  - supporting business tasks and overall organisation missions

- Network security components:
  - **firewall:** a hardware device or software product designed to filter network traffic for harmful exploits, incursions, data messages or other events
    - situated at edge of network
    - protects against threats from Internet
    - protect against rogue users or applications
    - typically configured on a deny-by-default basis (examines all traffic)
    - network administrators choose which traffic is allowed through
    - ingress filtering takes place on inbound traffic
    - egress filtering takes place on outbound traffic
  - **virtual private network (VPN):** mechanism to establish a remote access connection across an intermediary network, often over the Internet
    - allows for cheap long-distance connections over the Internet
    - uses tunnelling protocols which use encryption so intercepted data is unintelligible
    - used for remote access, remote control or highly secure communication within an untrusted network
  - **proxy server:** a firewall variation which acts as a middleman between a client and external server
    - hides identity of original requester from server using network address translation (NAT)
    - can be used to filter content using domain names or keywords
    - can be used to block access to irrelevant resources