Main Memory:
ogical(V) Address: an address generated by software based on the program instr
Physical Address: an address seen by memory unit
Base and Limit registers define the legal address space for each process.
Multiple-partition allocation:
Hole – block of available memory
Where process arrives, it allocates to memory from a hole large enough
OS maintains info (a) allocation partitions (b) free partitions (hole)
Dynamic-storage allocation(first-fit, best-fit, worst fit)
External Frag: total memory space exists to satisfy a request but is not contiguous
nternal Frag: allocated memory may be slightly larger than requested memory
Compaction: reduce external frag, shuffle memory, place all free memory in 1 blok
Page #: index into a page table with base address of each page in physical mem.
Page offset: combined with base addr,define physical mem addr is sent to mem un
To run program of n pages, find n free frames and load program
- set up page table to translate logical to physical addresses

Frames: fixed-sized blocks
Pages: divide logical memory into blocks of same size
PTBR: points to the page table
PRLR: indicates size of the page table
TLB's: special fast-lookup hardware, caches the address translation
TLB must be consistent with page table
Segmentation: memory-management scheme that supports user view of memory
Seg Table: map 2D logical addresses to one-dimensional physical addresses

Virtual Memory:
Virtual Memory: only part of the program needs to be in mem for execution
- separation of user logical/physical memory
- more efficient process creation
- address spaces, shared by several processes
- implemented via (a)demand paging (b)demand segmentation

Demand Paging: bring a page into memory only when it is needed
- less I/O needed
- less memory needed
- faster response
- more users

Page Replacement: over-allocation of memory due to demand paging
- Sol: replace an existing not currently being used memory with the new one

Modify(dirty) bit: reduce overhead of page trans, only modified page write to disk
FIFO page algorithm: replace the oldest page
Optimal Page Algorithm: replace page that will not be used for long periods of tim
Least Recently Used: replace page tat has not been used for the longest period
Thrashing: a process is busy swapping pages in and out
- occurs when size of locality > total memory size

Slab Allocator:
- slab: one or more continuous pages
- cache: consists of 1 or more slabs, single cache for each unique kernel struc
- when a new object is needed, the slab allocator tries to find a "free" object; if found, it is marked as used.
- Benefits: no fragmentation, fast memory when object are frequently all&de

File System:
File: a named collection of related information that is recorded in secondary stor.
File Ops: create, read, write, reposition, delete, truncate
Open(Fi): search for F1 and move content to memory
Close(Fi): move content of entry Fi to directory
Disk Structure:
- partitions
- raw (unformatted) or formatted with a file system
- volume: any entity containing file system
- general purpose file systems
- special purpose file systems

Single Level Directory: single directory for all users, name problem.
Two-Level Directory: separate directory for each user
- path name defined by a user name and a file name

Tree-Structured Directories:
- absolute path name: begins at the root, follows a part down to specified file
- Relative Path Name: defines a path from the current directory

File-system Structure: file system resides on secondary storage
- logical file system
- file-organization module
- basic file system
- I/O control

In-memory File System Structures:
- mount table: info about each mounted volume
- directory-structure cache: directory info of recently accessed directories
- system-wide open-file table: copy of FCB of each open file
- Per-Process open-file Table: a poiter to an entry in sys-wide open-file table
- Buffers: file-system blocks when read from or written to disk

A file system must be mounted before it can be accessed

Virtual File System: abstraction of generic file system operations from specific file system implementations.
Four main object types:
- Inode: object represents an individual file
- File: an open file
- Superblock: entire file system
- Dentry: an individual directory entry

Directory Implementation
- Linear List: of file names with pointer to the data blocks
- Hash Table: linear list with hash data structure

Allocation Methods:
- Contiguous (sharing a common border) Allocation: each file occupies a set of contiguous blocks on the disk
- Simple: only starting location (block#) and length(num blocks) required
- Random access is easy
- Waste of space
- Files cannot grow

Linked Allocation: each file is a linked list of disk blocks which my be scattered anywhere on the disk
- the directory contains a pointer to the first and last blocks of a file
- simple: need only starting address
- no wasted space
- no random access
- space overhead for storing the pointers
- reliability problem caused by lost/bad pointers

File Allocation Table: one entry for each block and is indexed by block number.
Each blocks table entry contains a block number of the next block in the file
- improved random access: disk head can find the location of any block by reading FAT. FAT can be cached to reduce disk head movement.
- Greater space overhead than linked allocaton

Indexed Allocation: handle large files, each file has its own index block, which is ar array of disk=block addresses.
- support random access
- greater space overhead than linked allocation

Combined Scheme in UNIX
- inode keeps 15 pointers
- first 12: point to direct blocks
- next 3 to indirect blocks (a)single indirect block(b)double(c)triple

Free-Space Management: keep track of free space for reuse

Linked Free Space List on Disk:
- cannot get contiguous space easily
- no waste of space

Mass Storage Structure:
- magnetic disks (secondary storage)
- drive attached to computer via I/O bus

Disk Drives: addressed as large 1-dimensional arrays of logical blocks
Low-level(Physical) Formatting: divide disk into sectors that the disk controller can read and write.
Partition: disk into one or more groups of cylinders
Logical Formatting: "creating a file system", store initial file system data to disk
Access Time:
- Seek time: time for the disk arm to move the heads to the cylinder containing the desired sector. (Seek time = seek distance)
- Rotational Latency: additional time waiting for the disk to rotate the desired sector to the disk head.

Disk Scheduling
- FCFS: fair but not fastest, total head movement 640 cylinders
- SSTF: select request with minimum seek time, may cause starvation, not optimal, 236 cylinders
- SCAN (elevator Algorithm): disk arm starts at one end and moves toward the other, servicing. Has an uneven wait time
- C-SCAN: when reaches the other end, it immediately returns to the beginning of the disk, without servicing any requests on the return trip

Swap-space Management: virtual memory uses disk space as an extension of main memory.
- kernel uses swap maps to track swap-space use

RAID Sturcture:
- redundant arrays of independent (inexpensive) disks
- improve reliability through redundancy, duplication disks
- improve performance through parallelism, can read multiple disks

RAID Levels:
- 0: non-redundant stripping
- 1: mirrored disks
- 2: memory-style error correction codes
- 3: bit-interleaved parity
- 4: block-interleaved parity
- 5: block-interleaved distributed parity

Solid-State Drive: storage device using memory to store persistent data
- higher reliability

- consistent read performance
- more power efficient

Disadvantages:

- more expensive
- less capacity
- less write longevity
- slower write speeds

/O Hardware:

- Port: device communication with computer
- Bus: set of wires, rigidly-defined messages that be sent on the wires
- Controller: electronic that can operate a port, a bus (host adapter),or device

CPU talks to device controllers through registers.
Polling: determines state of device

- command-ready
- busy
- error

nterupts:

- interrupt-request line: triggered by I/O device
- Interrupt Handler: receivers interrupts
- Maskable: ignore or delay some interrupts
- Interrupt Vector: dispatch interrupt to correct handler
- Also used for exceptions (divide by zero)

Direct Memory Access:

- used to avoid programmed I/O for large data movement
- requires DMA controller

Perform DMA Transfer:
1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address, dec C
6. When C = 0, DMA interrupts CPU to signal transfer completion

Application I/O Interface: IO calls encapsulate device behaviors in generic classes
Programmable Interval Timer: used for timings, periodic interrupts, always a copy
Kernal I/O Subsystem:

- caching: fast memory holding copy of data, always a copy
- spooling: hold for a device, if device can serve only 1 request at a time
- device reservation: exclusive access to a device, system calls for alloc & deal

Protection: ensure that OS object is accessed correctly and only by those processes that are allowed to do so.
Domain Structure:
rights-set: subset of all valid operations that can be performed on the object
Domain is a set of access-rights associated with an subject (entity), can be switched via password, file system, command
Access Matrix:
View protection as a matrix (access matrix)
- rows represent domains
- columns represent objects
- access(i, j): set of operations that a process executing in domain i can invoke on object j
- Access control can be discretionary or mandatory
DAC: user who creates object can define access column for that object
MAC: sys admin determines the access matrix, user cannot modify it
Use of Access Matrix: can be expanded to dynamic protection
owner: return the owner of oi
copy: copy op from oi to oj
control: di can modify dj access rights
transfer: switch from domain di to dj
mechanism: ensures matrix can be manipulated by authorized entities
policy: user define the matrix: who can access what object in what mode
Implementation of Access Matrix: global table, per-object access list, per-domain capability list, lock-key.
For access matrix each column defines an access-control list for the object: domain, operation
each row defines a capability list (like a key): object, operation
Revocation of Access Rights:
Immediate vs. delayed, selective vs. general, partial vs. total, temp vs. permanent

Security:
Threat: potential security violation
Attack: attempt to breach security
Security Violation Categories:
confidentiality: unauthorized reading of data
integrity: unauthorized modification of data
availability: unauthorized destruction of data
Security Violation Methods:
Theft of service: unauthorized use of resources
Denial of service (DOS): prevention of legitimate use
Masquerading: pretending to be an authorized user
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:
physical: data centers, servers, connected terminals
host: operating system
network: intercepted communications, interruption, DOS human: social engineering, phishing, dumpster diving
Virus: code fragment embedded in legitimate program
Port Scanning:
Port scan: automated attempt to connect to ports on one or more hosts
goal: detection of answering service protocol
Denial of Service: overload target computer preventing it doing any useful work
Cryptography: symm encryption/asymm encryption (public/private key)
Firewall and Network Security:
A firewall is placed between trusted and untrusted hosts
A firewall limits network access between these two security domains

Main Memory:
Main memory and registers are only storage CPU can directly access
linker (or loader) binds relocatable addresses to absolute addresses
Address Binding: compile, load, or execution time
OS determines logical to physical address mapping using MMU.
MMU is hardware that maps logical address to physical address at run time
MMU converts virtual address to physical address
segmentation and paging are two typical types of MMU
Paging has no external fragmentation, but internal fragmentation
Associative memory: memory that supports parallel search
hit ratio – percentage of page translation that is found in the TLB
Page Sharing

- one copy of code shared by all processes of the same program
- shared memory can be used for inter-process communication

To reduce memory consumption of page tables:

- hierarchical page table, hashed page table, inverted page table

Hierarchial Page Tables: Break logical address space to multi-level of page tables
Two-Level Paging:
A logical address is divided into:
a page directory number (first level page table)
a page table number (2nd level page table)
a page offset
Hashed Page Tables: each element contains: page#, frame#, pointer next element
Inverted Page Table -entry has process id and the page# (virtual address)
Segmentation: supports user view of a program a program is a collection of segs
- main program
- function
- local variables, global variables
- stack
- each segment can be mapped to physical blocks
Segment table maps segments to physical memory
each segment table entry has:
base: the starting physical address where the segments reside in memory
limit: the maximum offset of the segment
memory protection bites: present/read/write/execution
Swapping extends physical memory with backing disks
a process can be swapped temporarily out of memory to a backing store
backing store is usually a (fast) disk
the process will be brought back into memory for continued execution

Threads:
A thread is an independent stream of instructions that can be scheduled to run as such by the kernel
Benefits: responsiveness, resource sharing, economy, scalability
Signals are used in UNIX systems to notify a process that a particular event has occurred.
synchronous signals are delivered to the same thread causing the signal
Dispatcher module gives control of the CPU to the process selected by the short-term scheduler
System call is a programming interface to access the OS services
Process is a program in execution, its execution must progress in sequential fashion. a program is static and passive, process is dynamic and active
Long-Term Scheduler (Job Scheduler)- selects which processes should be brought into the ready queue
Short-Term (CPU Scheduler)- selects process to be executed next & allocates CPU
Mid-term Scheduler- swap in/out partially executed process to relieve memory pressure