Binding Times

Compile time: If you know at compile time where the process will reside in memory, then absolute code can be generated. **Load time:** Must generate relocatable code if memory location is not known at compile time. **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another.

Address

Logical address generated by the CPU, also referred as virtual address **Physical address** address perceived by the memory unit.

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 only deals with logical addresses and never sees real addresses.

Swapping

A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution

Backing store: fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images

Roll out, roll in: swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.

Major part of swap time is **transfer time**.

Contiguous Allocation

Base register + Limit register

Multiple-partition allocation

Dynamic Storage-Allocation

First-fit Allocate the first hole that is big enough. **Best-fit** Allocate the smallest hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole. **Worst-fit** Allocate the largest hole; must also search entire list. Produces the largest leftover hole.

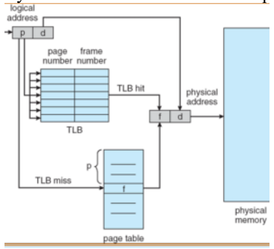
First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

Fragmentation

External Fragmentation: total memory space exists to satisfy a request, but it is not contiguous**.** **Internal Fragmentation:** allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

Paging (Internal fragmentation)

Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512B and 8,192B). Divide logical memory into blocks of same size called pages.



Structure of the Page Table

Hierarchical Page Tables: Break up the logical address space into multiple page tables. **Hashed Page Tables:** *Common in address spaces > 32 bits.* The virtual page number is hashed into a page table. This page table contains a chain of elements hashing to the same location. **Inverted Page Table:** One entry for each real page of memory Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page.

Segmentation (external fragmentation)

Logical address consists of a two tuple: <segment-number, offset>,

Segment table:maps two-dimensional physical addresses; each table entry has **base** and **limit**. **Allocation:** Since segments vary in length, memory allocation is a dynamic storage-allocation problem: first fit/best fit **Relocation:** dynamic**,** by segment table. **Sharing:** shared segments**,** same segment number.

Virtual memory

Concept: separation of user logical memory from physical memory. Logical address space can therefore be much larger than physical address space.

Demand Paging

Bring a page into memory only when it is needed. Less I/O needed. Less memory needed Faster response. More users

Lazy swapper: Never swaps a page into memory unless page will be needed.

Page Fault

1. Get empty frame
2. Swap page into frame
3. Reset tables
4. Set validation bit = **v**
5. Restart the instruction that caused the page fault.

Effective Access Time (EAT)

Page replacement

Use **modify (dirty)** bit to reduce overhead of page transfers – only modified pages are written to disk.

First-In-First-Out (FIFO) Belady's Anomaly: More frames, more page faults.

Optimal Algorithm

Replace page that will not be used for longest period of time.

Least Recently Used (LRU) Algorithm

Counter implementation: Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter. **Stack implementation** – keep a stack of page numbers in a double link form.

LRU Approximation

By Reference bit: With each page associate a bit, initially = 0. When page is referenced bit set to 1.

Second chance: Need reference bit. Clock replacement. If page to be replaced (in clock order) has reference bit = 1 then: set reference bit 0, leave page in memory, replace next page (in clock order), subject to same rules

LFU Algorithm:

replaces page with smallest count

MFU Algorithm:

based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Global replacement: one process can take a frame from another. **Local** replacement:selects from only its own set of allocated frames.

Thrashing: a process is busy swapping pages in and out.

Working-Set Model

a fixed number of page references.

WSSi (working set of Process Pi) = total number of pages referenced in the most recent

if D > WSS → thrashing

Allocation of Frames

Fixed Allocation

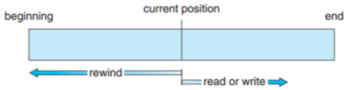
Equal allocation or Proportional allocation.

Priority Allocation

Use a proportional allocation scheme using priorities rather than size. select for replacement one of its frames or a process with lower priority number.s

File Access Methods

Sequential Access (based on a tape model). Information in the file is processed in order, one record after the other. Such a file can be reset to the beginning.



Direct Access (or **relative access**, based on a disk model). File is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order, file is viewed as a numbered sequence of blocks or records, which allow random access.

sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;

write_next	write cp;
	cp = cp + 1;

Simulation of sequential access on a direct-access file.

Directory

Single-Level Directory: all files are contained in the same directory. **Two-Level Directory:** 1. master file directory (MFD). 2. each user has its own user file directory (UFD).

Tree-Structured Directories: A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name.

Acyclic-Graph Directories: allows directories to share subdirectories and files. In a system where sharing is implemented by symbolic links, space for the file when deletion is deallocated, leaving the link dangling. Another approach to deletion is to preserve the file until all references to it are deleted.

Protection: Owner, Group, Universe

File Control Block

file permissions

file dates (create, access, write)

file owner, group, ACL

file size

file data blocks or pointers to file data blocks

Virtual File Systems

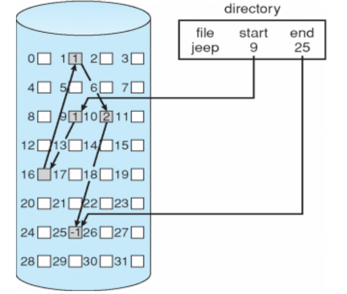
Virtual File Systems (VFS) provide an object-oriented way of implementing file systems. VFS allows same system call interface (API) to be used for different types of file systems.

File Allocation Methods

Contiguous Allocation Each file occupies a set of contiguous blocks on the disk. Random access. Simple - only starting location (block #) and length (number of blocks) are required. Wasteful of space (dynamic storage-allocation problem). Files cannot grow.

Extent-Based: Extent is a contiguous block of disks. Extent-based file systems allocate disk blocks in extents. Extents are allocated for file allocation. A file consists of several extents.

Linked Allocation: Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk. Simple – need only starting address. Free-space management system – no waste of space. Mapping. No random access.

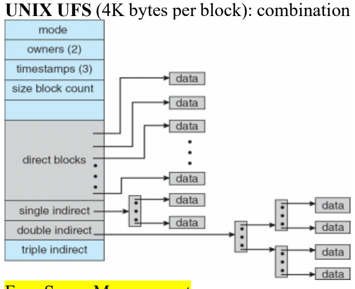


The space required for the pointers, so collect blocks into multiples, called **clusters**, and to allocate clusters rather than blocks.

File-Allocation Table: The directory entry contains the block number of the first block of the file. The table entry indexed by that block number contains the block number of the next block in the file.

Indexed Allocation: Brings all pointers together into the index block. Dynamic access without external fragmentation, but have overhead of index block.

UNIX UFS (4K bytes per block): combination



Free-Space Management

Frequently, the free-space list is implemented as a **bit map** or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0. Easy to get contiguous files. Must be kept on disk and be consistent with that in cache.

Linked list (free list): No waste of space but Cannot get contiguous space easily.

Log structured file systems

Log structured file systems record each update to the file system as a transaction.

Access a Disk block

Magnetic disks provide bulk of secondary storage of modern computers. Drives rotate at 60 to 200 times per second. **Transfer rate** is rate at which data flow between drive and computer. **Positioning time** (random-access time) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**). **Disk bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling

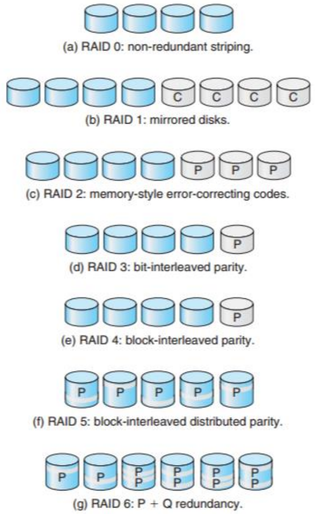
Seek time >> Rotational time ->Minimize seek time. Seek time≈seek distance

first-come, first-served (**FCFS**)

shortest-seek-time-first (**SSTF**): Selects the request with the minimum seek time from the current head position, may cause starvation of some requests. **SCAN(elevator)** algorithm: the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. **C-SCAN**: restart from the beginning. **LOOK**: goes only as far as the final request in each direction, then reverses direction immediately.

RAID Structure

RAID – Redundant Arrays of Independent Disks, multiple disk drives provides reliability via redundancy. Frequently combined with NVRAM(Non-volatile RAM) to improve write performance.



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

IO Concept

Polling: Determines state of device. Busy-wait cycle to wait for I/O from device.

Buffering: Store data in memory while transferring between devices. To cope with device speed mismatch. To cope with device transfer size mismatch. To maintain "copy semantics".

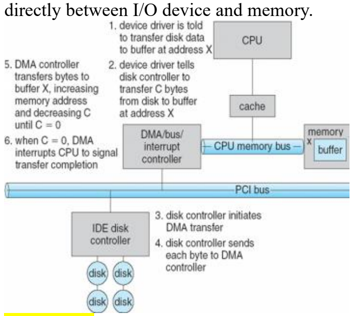
Caching: fast memory holding copy of data. Always just a copy, key to performance.

Spooling (Simultaneous Peripheral Operation On-Line): hold output for a device. If device can serve only one request at a time. i.e., Print

Blocking: the process suspended until I/O completed. **Nonblocking** - I/O call returns as much as available. e.g. video. **Asynchronous**: process runs while I/O executes.

Direct Memory Access(DMA)

Used to avoid programmed I/O for large data movement. Bypasses CPU to transfer data directly between I/O device and memory.


1. device driver is told to transfer disk data to buffer at address X
CPU
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
cache
6. when C = 0, DMA interrupts CPU to signal transfer completion
DMA/bus/interrupt controller
CPU memory bus
memory
X
buffer
PCI bus
IDE disk controller
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
disk
disk
disk
disk

I/O Devices

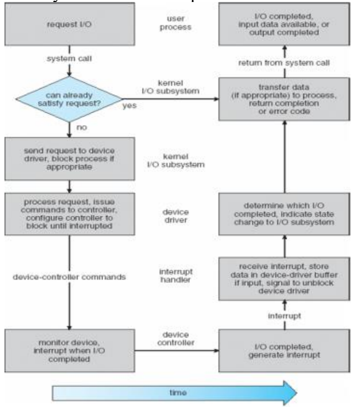
I/O system calls encapsulate device behaviors in generic classes. Device-driver layer hides differences among I/O controllers from kernel.

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Block devices include disk drives. Commands include read, write, seek. Raw I/O or file-system access. Memory-mapped file access.

Character devices include keyboards, mice, serial ports.

Life Cycle of An I/O Request


request I/O
user process
I/O completed, input data available, or output completed
system call
return from system call
can already satisfy request?
kernel I/O subsystem
yes
transfer data (if appropriate) to process, return completion or error code
no
send request to device driver, block process if appropriate
kernel I/O subsystem
process request, issue commands to controller, configure controller to block until interrupted
device driver
determine which I/O completed, indicate state change to I/O subsystem
device-controller commands
interrupt handler
receive interrupt, store data in device-driver buffer if input, signal to unblock device driver
interrupt
monitor device, interrupt when I/O completed
device controller
I/O completed, generate interrupt
time