Cheatography

Operating System Basics Cheat Sheet

by rpolyano via cheatography.com/24960/cs/6373/

Address Translation

VA -> Page Table (entry) -> Physical memory or swap

Temporal Locality - recently referenced pages that might be again

Spatial Locality - referenced addresses will probably be near other recently referenced

Mappings cached in TLB (Translation Lookaside Buffer)

Page Tables

Entry looks like | M^1 | R^1 | V^1 | $Prot^3$ | PFN^{26} |

Modify Bit - whether or not a page has been written (set on write)

Reference Bit - page has been accessed (set on read/write)

Valid bit - PTE can be used

Protection bits - R/W/X permissions

PFN - Page Frame Number - physical page

Other Page Tables

Hashed - Hash function maps virtual page to bucket of LList of (VPN, PTE)

Inverted - Table of PFNs - > PID, PTE

Fetch Policies

Demand Paging - Swap in the pages you need when you need them

Prepaging - Predict next page that will get loaded and load it now

Replacement Policies

OPT	Swap out page that will not be used for the longest time
FIFO	Swap out oldest page
	Suffers Belady's Anomaly - Fault rate **may** increase with memory size
LRU	Swap out page that has not been accessed for longest time

Replacement Policies (cont)

CLOCK	Approximation of LRU - Go through candidates in a circle. If Ref bit set, clear it, move on. If not set, then swap out.

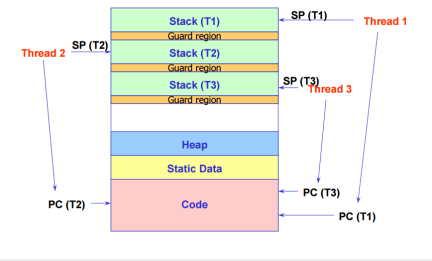
Page Buffering - Maintain a pool of free (uncleared/rescuable) frames. Run replacement when pools gets too empty, run it until pull is full enough. On fault grab frame from pool.

Deadlock

Definition	Set of processes that compete for resources or communicate with each other permanently blocked
Conditions	Mutual Exclusion* - only one process uses resource at once
	Hold and wait* - A process may hold resource while waiting for other resource
	No preemption* - No resource can be forcefully removed from a process
	Circular wait** - A closed chain of processes, where one needs 1+ resources held by the next.

*Necessary for deadlock.
** Necessary and sufficient.

Address Space



Memory partitioning (Fixed vs Dynamic)

- ***Fixed partitioning:*** Every process gets a fixed amount of memory in one of the following ways:

Memory partitioning (Fixed vs Dynamic) (cont)

- **Equal size.** Every process gets the same amount of memory. Internal fragmentation when process too small. Overlay when process too big. No external fragmentation.

- **Unequal size.** Every process gets a chunk that has minimum memory required available. Internal fragmentation. Holes

- ***Dynamic partitioning:*** Size and number of partitions varies

- **Every process** gets as much as needed on start. Need to know size beforehand. Holes. Need relocatable processes.

- ***PAGING***: Split both virtual and physical memory in same chunks called **Pages**

- Every process gets its own **Page Table** which maps its virtual addresses into physical pages (offset is the same, only need to translate frames)

File Buffer Cache

Locality when reading/writing files

Cache blocks/inodes/dir entries that are "hot"

Can use static or dynamic partitioning.
Static - allocate n% at boot. Wasteful.
Dynamic - VM and FS pages into unified cache. Pages of memory can be dynamically allocated between RAM and FS cache

FS - Caching Writes

Writes can be buffered. This allows

Multiple writes in one (e.x. flip many bits of a bitmap)

Scheduling to allow for better disk access

Avoid writes entirely (e.x. set bit to 1, set bit to 0)

Disk Performance

Seek	Moving disk arm to cylinder (1 - 15ms, ~5ms avg, improving slowly)
Rotation	Wait for sector to rotate to head (4ms avg, not improving)

By **rpolyano**
cheatography.com/rpolyano/

Not published yet.
Last updated 11th December, 2015.

Disk Performance (cont)

Transfer	Moving data from platter to controller (~100MB/s, sector ~5microseconds, Improving quickly)

Log FS Mapping and GC

Mappings to inodes are stored in imaps, placed in chunks after new information. Fixed location on disk called Checkpoint Region contains pointers to latest pieces of imap. Keep 2 CRs at opposite ends of the disk for redundancy.

Garbage Collection periodically goes through segment by segment and frees up files, since new versions of files are written to new places on disk. (LOG FS is circular)

CPU Scheduling

FCFS	NP, First Come First Serve
SJF	NP or P, Choose thread with shortest (remaining) processing time. Optimal avg wait time
RR	P, Circular Q, good for time sharing systems
Priority Scheduling	NP or P, Highest priority job selected from ready Q

NP - Nor Preemptive Preemptive
Priority Inversion - Low priority blocks resource, high priority doesn't run.

Filesystem consistency

Superblock	Sanity checks. Use another copy if corrupted
Free blocks	Ensure inodes in use are marked in bitmap. Trust inodes.
Inode state	Must have valid mode. Remove if can't fix
Inode links	Verify links. Traverse tree and count links. Move unlinked inodes to lost+found
Duplicates	Two inodes refer to same block

Filesystem consistency (cont)

Bad blocks	bad pointers. Remove from inode
Directory checks	Make sure . and .. are first. Each inode in dir is allocated. No dir linked more than once.

Fast File System

Used cylinder (block) groups.

Data blocks in same file allocated in same group

Entries in same directory allocated in same group

Inodes for files are allocated in same group as file data

Large files get broken up into chunks and allocated in different cylinder groups

Groups reduce number of long seeks.
Must have free space in cylinder groups (10% of disk reserved free)

Disk Scheduling Policies

FCFS - First Come First Serve (i.e. no scheduling) - Good under small load

SSTF (Shortest Seek Time First) - Minimize arm movement, maximize request rate, favor middle blocks

SCAN (elevator) - Order requests by direction (do all in direction one, then all in reverse)

C-SCAN (Typewriter) - Like SCAN, but only in one direction.

LOOK/C-LOOK - Same as SCAN/C-SCAN but only move as far as last request in each direction.

Journaling

Log TxBegin , log inode bitmap, log block bitmap, log block data. Once done, log txend (transaction committed)

Write data (checkpoint step). Make sure TxEnd scheduled last.

Avoid writing data to journal by writing data first, then journal, then actual metadata

Log FS

Like FFS but everything gets logged in order.

Superblock | summary | inode | data | data | inode | data | superblock | summary etc

Summary has pointer to next summary or next super block.

RAID (Redundant Array of Independent Disks

RAID1 (Data duplication)	mirror images, redundant full copy. Wastes space
Parity Information	XOR each bit from2 drives, store checksum on 3rd
RAID 0	Write half of data on one drive and one on the other

CPU Scheduling Goals

All Systems	Fairness, Avoid starvation, Policy enforcement, Best use of resources
Batch System	Throughput (jobs/hour), Turnaround (min time between submit and complete), CPU Util (keep CPU busy)
Interactive System	Response time (min time from submit to start), Proportionality (simple tasks finish faster)
Real Time System	Meet deadlines, predictability

Threads

Kernel - Less state to alloc and init than process => cheaper concurrency

User - Managed by runtime systems. Small, fast, has PC, registers, stack, small TCB. Creating, switching, sync done by procedures (no kernel). (Disadv.) - Not well integrated with OS, Scheduling.

Hybrid - Can have many kernel threads associated with many user threads, or many user threads associated with a few kernel threads.

By **rpolyano**
cheatography.com/rpolyano/

Not published yet.
Last updated 11th December, 2015.