Processes & Threads

Processes

- **Process**: Program in execution.
- **Multiprogramming**: Concurrent processes on one core.
- **Multiprocessing**: One process on multiple cores.
- **Address Space**: Code (Text), Static Data, Heap, Stack.
- **PCB (Process Control Block)**: `task_struct`, pointed to by `current`.
- **PCB Fields**: PID, TGID, state, registers, `mm_struct`, family pointers, exit code.
- **Process States**:
  - **Running**: On CPU.
  - **Ready**: Waiting for CPU.
  - **Waiting (Sleeping)**: For event. `TASK_INTERRUPTIBLE`, `TASK_UNINTERRUPTIBLE`.
  - **Stopped**: Suspended.
  - **Zombie**: Terminated, PCB exists for `wait()`.
- **Process Family**: **Idle** (PID 0) → **init** (PID 1). **Real Parent**, **Parent**, **Orphans** (adopted by init).
- **Syscalls**: `fork()`, `exec()`, `wait()`/`waitpid()`, `exit()`, `kill()`, `getpid()`, `getppid()`.
- `fork()`: returns 0 to child, PID to parent.

Threads

- **Threads**: Execution units within a process.
- **Multithreading**: Process with multiple threads.
  - **Shared**: Memory space, FDT, directory, signal handlers.
  - **Unique**: Stack, registers (PC, SP), TID.
- **Thread Group**: Threads sharing a PID (TGID). `getpid()` → TGID, `gettid()` → TID.
- `exit()` terminates the entire thread group.
- **Pthreads**: `pthread_create()`, `pthread_join()`, `pthread_exit()`.
- **User Level Threads (ULTs)**: Managed by user-library, no true parallelism.
- **Kernel Threads**: OS-level tasks, no memory space switch.

Signals

- **Signal**: Asynchronous notification.
- Wakes a waiting process.
- **Handling**: Custom handler (`sigaction()`), ignore (`SIG_IGN`), default (`SIG_DFL`).
- **Masking**: Block signals (`sigprocmask()`), becomes "pending".
- **Unblockable Signals**: `SIGKILL`, `SIGSTOP`.
- **Common Signals**: `SIGSEGV`, `SIGILL`, `SIGCHLD`, `SIGINT`, Signal 0.

IPC & System Calls

- **clone()**: Creates processes/threads. Flags: `CLONE_VM`, `CLONE_FILES`, `CLONE_SIGHAND`, `CLONE_THREAD`.
- **Copy-on-Write (COW)**: `fork()` optimization, duplicates page on write.
- **Pipe**: Unidirectional buffer (`int fd[2]`).
- **FIFO (Named Pipe)**: Filesystem pipe (`mkfifo()`) for unrelated processes.

Boot & Modules

- **Booting Sequence**: BIOS → MBR → Boot Loader (GRUB) → Kernel (`/sbin/init`).
- **Kernel Modules**: Dynamically loadable code (`init_module()`, `cleanup_module()`).
- **Device Files**: In `/dev`. **Major** (driver), **minor** (instance).
  - **Character**: Stream-based.
  - **Block**: Addressable.
  - **Pseudo**: Virtual (`/dev/null`).
- `file_operations` struct: Driver function pointers, registered with `register_chrdev()`.

CPU Scheduling

Concepts

- **Time Sharing**: CPU virtualization.
- **Process Types**: **I/O-bound** (latency-sensitive), **CPU-bound**.
- **Preemption**: Forcible process stop, via **Quantum**.
- **Scheduling Metrics**:
  - $T_{wait} = T_{start} - T_{submit}$.
  - $T_{resp} = T_{end} - T_{submit}$.
  - Slowdown: $1 + \frac{T_{wait}}{T_{run}}$.

Batch (Non-Preemptive) Schedulers

- **FCFS**: First-Come, First-Served.
- **SJF**: Shortest Job First.
- **EASY**: FCFS with **backfilling**.

Preemptive Schedulers

- **Round Robin (RR)**: Quantum-based circular queue.
- **SRTF**: Shortest Remaining Time First.
- **Selfish RR**: Two-level queue with aging.
- **Gang Scheduling**: RR for parallel jobs.

Linux Schedulers

- **Linux $\leq$ 2.4** ($O(n)$):
  - **HZ**: Time interrupts per second.
  - **Tick**: Time between time interrupts.
  - **Policies**: `SCHED_RR`, `SCHED_FIFO`, `SCHED_OTHER`.
  - **Epoch**: Cycle for all runnable tasks.
  - **Priority**: Static (`nice`) + Dynamic (`counter`).
  - New epoch counter: $C_{new} = \lfloor \frac{C_{old}}{2} \rfloor + P_{static}$.
  - `schedule()` finds best `goodness()`.
- **CFS** ($O(\log N)$):
  - **Goal**: Fair CPU time via equal `vruntime`.
  - **vruntime**: Task with lowest runs.
  - $\Delta vruntime = \Delta T_{actual} \times \frac{W_{ideal}}{W_{task}}$.
  - **Data Structure**: Red-black tree ordered by `vruntime`.

Context Switch

- **Context**: Process state.
- **Overhead**: **Direct** (save/load state), **Indirect** (cache pollution).
- **Types**: **Forced** (interrupt), **Initiated** (syscall).
- **Mechanism**: State saved to kernel stack. **TSS** updated.
- **Flow**: `schedule()` → `context_switch()` → `_switch_to_asm()` → `_switch_to()`.

Synchronization

Concepts

- **Race Condition**: Timing-dependent outcome.
- **Critical Section**: Requires mutual exclusion.
- **Atomicity**: All-or-nothing operation.
- **Memory Consistency**: Order of memory ops, enforced by **memory fence**.
- **Amdahl's Law**:

$$Speedup = \frac{1}{s + \frac{1-s}{n}} \leq \frac{1}{s}$$

Mechanisms

- **Lock**: `acquire()` and `release()`.
- **Spinlock**: Busy-waits for lock.
- **Mutex**: May block (sleep) if lock is unavailable.
- **Semaphore**: Counter with wait queue. `wait()` (P), `signal()` (V).
- **Condition Variable**: Wait for condition inside critical section. `cond_wait`, `cond_signal`.

Deadlocks

- **Deadlock**: Processes blocked waiting for each other's resources.
- **Livelock**: State changes, no progress.
- **Starvation**: Process perpetually denied resources.
- **Four Necessary Conditions**: Mutual Exclusion, Hold & Wait, No Preemption, Circular Wait.
- **Handling**:
  - **Prevention**: Violate one condition (e.g., lock ordering).
  - **Avoidance**: **Banker's Algorithm**.
  - **Detection & Recovery**: Find cycle and kill process.

Networking

- **Protocol Stack**: L5-App, L4-Transport, L3-Network, L2-Link.
- **TCP**: Reliable, connection-oriented, stream-based.
- **UDP**: Unreliable, connectionless, datagram-based.
- **IP Address**: Logical host address (network part + host part).
- **Socket**: Communication endpoint (FD). 5-tuple identifies connection.
- **TCP Flow**: Server: `socket` → `bind` → `listen` → `accept`. Client: `socket` → `connect`.
- **Ethernet (L2)**: LAN technology, uses **MAC addresses**.
- **Key Protocols**: **ARP** (IP → MAC), **DHCP** (dynamic IP), **NAT** (private ↔ public IP).

Virtual Memory

Concepts

- **Virtual Memory (VM)**: Private, contiguous address space abstraction.
- **Page** (virtual, 4KB) & **Frame** (physical).
- **Address Translation**: **MMU** translates VA → PA via page tables.
- **Page Table**: Maps virtual pages to physical frames.
- **PTE (Page Table Entry)**: Contains frame number and bits: Present, Dirty, Accessed, R/W, U/S.
- **Page Fault**: Trap on access to unmapped page.
  - **Major Fault**: Fetch from disk.
  - **Minor Fault**: Page in memory, mapping missing.
- **TLB**: Hardware cache for VA → PA translations.

Paging & Swapping

- **On-Demand Paging**: Load pages on first access.
- **Swap Area**: Disk space for evicted pages.
- **Paging out**: Copying page from DRAM to disk.
- **kswapd**: Kernel thread to reclaim page frames.
- **Thrashing**: Excessive paging, little progress.

Page Replacement Algorithms

- **Goal**: Choose victim page to evict.
- **Belady's Optimal**: Replace page used furthest in future.
- **LRU**: Replace least recently used page.
- **Clock Algorithm**: LRU approximation with reference bit.
- **Linux PFRA**: Two lists (**active**, **inactive**) to approximate LRU.

Linux Implementation

- **Multi-Level Page Tables**: 4-level on x86-64. **CR3** points to top level.
- **mm_struct**: Process memory descriptor.
- **vm_area_struct (VMA)**: Contiguous memory region.
- **mmap()**: Creates a new VMA.
- **Copy-on-Write (COW)**: On `fork()`, share pages, copy on write.

Storage & Filesystems

Storage Devices

- **HDD**: Access time = seek time + rotational latency.
- **SSD**: Flash memory, low random access latency.
- **DMA**: Device-to-memory transfer without CPU.

RAID

- **RAID 0 (Striping)**: Best performance, no redundancy.
- **RAID 1 (Mirroring)**: Full redundancy.
- **RAID 4 (Parity Disk)**: Write bottleneck on parity disk.

- **RAID 5 (Distributed Parity)**: No parity disk bottleneck.
- **RAID 6 (Dual Parity)**: Protects against two disk failures.

Filesystem Concepts

- **File**: Logical unit of information.
- **inode**: File metadata structure.
- **dirent**: Maps filename to inode number.
- **Links**: **Hard Link** (same inode), **Symbolic Link** (path string).
- **File Descriptor (FD)**: Integer handle for an open file, index into **FDT**.
- FDT entry → system-wide **File Object** → **inode**.

Filesystem Implementation

- **Layout**: Superblock, bitmaps, inode table, data blocks.
- **Allocation**: **Multi-Level Index**, **Extents**, **FAT**.
- **Journaling**: Transaction log for consistency.
- **Path Resolution**: Traverse directories to find inode.