**Process hierarchies**

* **Process hierarchy**
  + Parent–Child Relationship
  + The child can itself create new processes
  + Tree of processes
* In **UNIX** OS
  + A process creates another process, the parent process and child process continue to be associated in certain ways
  + A process and all of its children and further descendants together form a process group
* In **Windows** OS
  + All processes are equal (has no concept of process hierarchy)
  + The parent is given a special token (called a handle) that it can use to control the child

**Process Termination**

* + Normal exit (task accomplished) (voluntary)
  + Error exit (voluntary). Ex: nonexistent files, insufficient or incorrect input
  + Fatal error (involuntary). Ex: illegal instructions, division by zero etc.
  + Killed by another process (involuntary). kill system call in Unix, or TerminateProcess in Win32.(in some systems, if the parent terminates)
  + **Voluntary** – using a special system call
  + **Involuntary** – receiving an interruption

**Process States**

* + **Running**: Using the CPU at that instant (executed by the CPU) (or Instructions are being executed)
  + **Ready**: Runnable; temporarily stopped to let another process run; but the CPU available (or The process is waiting to be assigned to a process)
  + **Blocked**: Unable to run until some external event happens (or The process is waiting for some event to occur)
  + **New** (optional): Waiting for same resources to be allocated (or The process is being created)
  + **Terminated** (optional): Keeping same information about the exit state (or The process has finished execution)

**Thread benefits**

* + Responsiveness and better resource sharing
  + Economy:
  + Useful on systems with multiple CPUs.
  + Less time to terminate a thread than a process
  + Less time to switch between two threads within the same process (serve many task with the same purpose)
  + Since threads within the same process share memory and files, they can communicate with each other without invoking the kernel

But, they introduce a number of complications:

**Proposals for achieving mutual exclusion:**

* + **Disabling interrupts**
  + **Lock variables**
  + **Strict alternation:** The two processes strictly alternate in entering their critical regions
  + **Peterson's solution**: Combines the idea of taking turns with the idea of lock variables and strict alternation
  + The TSL instruction
  + Sleep and Wakeup
  + Producer & Consumer
  + Semaphore: (Full, Empty, Mutex)
  + Mutex,
  + **Monitor**: Is a collection of procedures, variables, and data structures that are all grouped together in a special kind of module or package
  + Barrier: When a process **reaches the barrier**, it is **blocked** **until all processes have reached the barrier.**

**Solution: Mutual exclusion with Busy waiting**

**Software proposal**

Lock Variables

Strict Alternation

Peterson's Solution

**Hardware proposal**

Disabling Interrupts

The TSL Instruction

Testing a variable until some value appears is called **busy waiting** (wastes CPU time)

Both Peterson and TSL have defect of requiring “busy waiting“:

+ Waste CPU time

+ Priority inversion

->> Sleep & wakeup

**Message Passing:** Using system calls (like semaphore) with 2 primitives

* + **send** (destination, &message): sends a message to given destination
  + **receive** (source, &message): receives a message from a given resource. If no message is available, the receiver can block until one arrives
  + used for the communicating processes are on different machines connected by a network

**Scheduling**When to schedule

**non-preemptive** scheduling algorithms

* + Picks a process to run, lets it **run until it is blocked or until it voluntarily releases the CPU** (will not be forcibly suspended, no scheduling decisions)
  + Applying to the batch system

**preemptive** scheduling algorithms

* + The process can run (continuously) for **a maximum of some fixed time**. If it is still running at the end of this time, it is **suspended and the scheduler will pick another process to run** (needs timer)
  + Applying to the time-sharing or real time

**Throughput**: The number of processes that complete their execution per time unit.

**Turnaround time:** Amount of time to execute a particular process

= time of completion – arrival time

When there is an excessive amount of page swapping between main memory and secondary storage, the operation becomes inefficient, which is called **thrashing.**

**Starvation**: The low priority processes can be waited indefinitely (cannot be executed if the system occurred errors in runtime) for CPU by higher priority processes

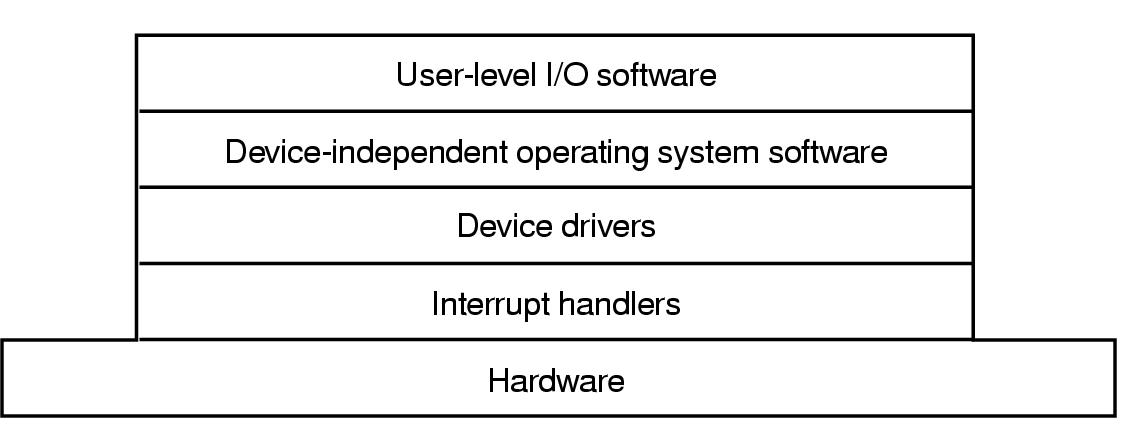
**Solution**

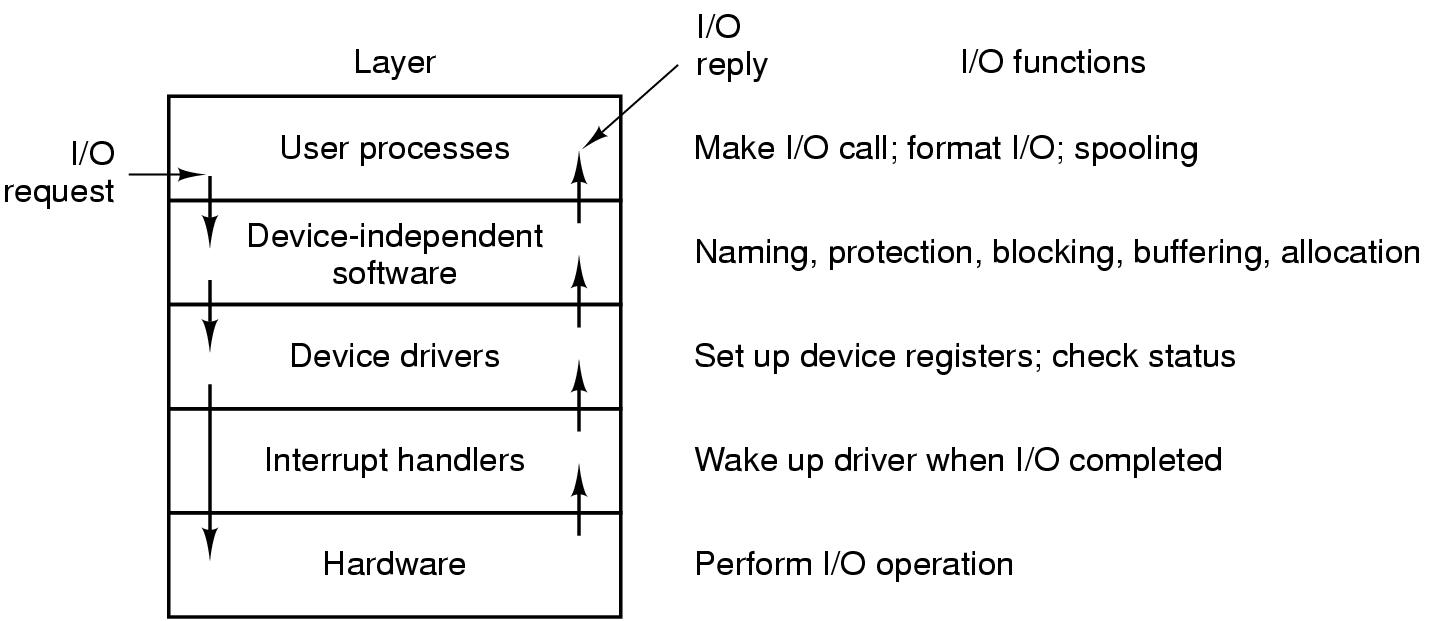
**Aging:** Is a technique of **gradually increasing the priority of processes** that wait in the system for a long time (using the clock interrupt)

**Ex**: every 15 minutes, decreasing the priority of a waiting process from (1 → 127), means that the priority with 127 → 1 at least 32 hours

**Memory Management Unit** (MMU): A **mapping unit** from virtual addresses into physical addresses

**Layers of the I/O Software System**





**Page Fault** – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

<https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>

<https://getphilnitscertified.wordpress.com/2011/10/09/2007-april-fe-am-q26/>

Resource Deadlocks

* + A resource that is owned by a deadlocked process

**Four conditions** must hold for there to be a deadlock

* + **Mutual exclusion condition**
    - Each resource is either currently assigned to exactly one process or is available
  + **Hold and wait condition.**
    - A process currently holding at least one resource is waiting to acquire additional resources held by other processes
  + **No preemption condition**.
    - A resource can be released only voluntarily by the process holding it, after that process has completed its task
  + **Circular wait condition**
    - There must be a circular chain of two or more process, each of which is waiting for a resource held by the next member of chain

Preemptable: Is resources that can be **taken away from** the **process** **owning** it **with no ill effects** (either on system or others)

**Page table structure:**  
1. Page frame number (goal)

2. Present/absent bit

3. Protection bit

4. Modified and Referenced Bit

5. Bit that allows caching to be disabled

**Page table purpose:** Designed to map virtual pages onto page frames (page table being the function, virtual page number the argument, and physical frame number as a result), producing physical memory address

 For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4-KB page and for an 8 KB page: 20000, 32768, 60000.

For case (a) for 4KB:4KB = 4096 B / page  
             Page number = INT(20000/4096) = 4.  
             Offset = 20000 - 4\*4096 =36166  
             Analogously(page , offset)