Process Synchronization 1

- A *race condition* exists in a program when the result of program execution depends on the precise order of execution of the threads of the process or other processes.
- A *critical section* is any piece of code that accesses shared data.
- Mutual exclusion conditions:
    - Ensures that only one thread/process accesses the shared data.
    - No two threads simultaneously in critical section.
    - *Progress*: No thread running outside its critical section may block another thread.
    - *Bounded Waiting*: No thread must wait forever to enter its critical section.
- Peterson's Solution:
    - works for 2 processes only.
    - Uses a flag array of two and an int, loops while the shared memory is free.
- Hardware solutions for locks:
    - disabling interrupts.
        - Disadvantages: gives the user too much power(turning off interrupts), does not work for multi-CPUs.
    - Test and Lock Instruction(TSL)
        - Disadvantages: requires busy waiting.
- TSL and Peterson's Solution violate condition 4 for mutual exclusion, runs for ever,caused by priority inversion.

Process Synchronization 2

- A *semaphore* is an integer variable with the following three operations.
    - Initialize: any non-negative number.
    - Decrement: decrements until a zero is reached, blocks instead of negative.
    - Increment: if zero, then unblock next process, else, increment.
- Binary semaphore is called a *mutex*. *Counting semaphores* are 0 to N.
- Deadlock: Two or more processes are waiting, but can only be unblocked by one of the blocked processes.
- Waking up a process is done signalling.
- Longer critical sections lead to lower throughput.

Classic Synchronization Problems

- List of Classic Synchronization Problems
    - Producer-Consumer:
        - Buffer(N)
        - Producer writes to buffer, writes while the buffer is not full.
        - Consumer reads from buffer, empties the buffer up as it reads.
    - Readers-Writers
        - Only one writer active at a time. Readers can't be active at that time.
        - Multiple readers can be active simultaneously.

    - Dining Philosopher

        - Deadlock: occurs when everyone is trying to use their fork at once.
        - Can only pick up one fork at a time.
        - Needs two forks to eat.
    - Sleeping Barber
        - Barbers sleep if there are no waiting customers.
        - No chairs(buffer) available, customer leaves.
        - Customer wakes up a barber if they are asleep.

Basic Memory Management

- Requirements of memory management:
    - Track inuse parts of memory.
    - Allocate memory to processes.
    - Deallocate memory.
- Instruction-Execution cycle
    - Fetch instruction from memory.
    - Decode instruction.
    - Fetch operands from memory, if necessary.
    - Execute instruction.
    - Store result in memory, if needed.
- Memory Hierarchy
    - Registers
    - On-chip Cache
    - Main Memory
    - Disk
- Fetching data from higher levels or the archy is not desirable, too slow.
- Memory is shared by processes, process can't access each others memory without permission.
- Addresses in the source of the program are usually symbolic.
- Compiler binds addresses into relocatable addresses, loader binds them into absolute address, binding to actual physical memory address is done in execution time.
- Swapping: reallocation of program binary between different main memory and disk locations.
- Logical/virtual memory addresses must be translated to physical addresses.
- Memory Management Unit (MMU): does run-time mapping for addresses, hardware device.

Allocating Memory

- Contiguous memory allocation: each process is contained in a single section of memory.
- Fixed partitioning: every program will occupy an entire portion.
  - Equal vs unequal partition sizes.
  - Unequal partitions placement algorithm will try to minimize wasted memory.
  - One queue per all partitions vs. multiple queues, one per partition.
- *Base*: smallest physical memory location.
- *Limit*: highest physical location.
- Simple MMU: relocation register value is added to the virtual addresses.
- Dynamic partitioning
  - External fragmentation: has holes in memory.
  - Compaction: shift processes to close holes, defragmentation.
  - Dynamic partitioning placement: must decide what block is going to be used.
    - Best-fit: chooses closest size, worst performance, less fragmentation needed, more compaction must be done.
    - First-fit: looks for the first large enough memory block, starts at the beginning of the physical memory.
    - Next-fit: same as above, starts from the last used location, best performance.
- *Paging*: avoids fragmentation and compaction.
- Divides physical memory and process into small equal chunks.
- *Pages*: process chunks.
- *Frames*: physical chunks.
- OS maintains a page table for every process, page to frame table.
- Addresses generated by CPU are divided into
  - Page number: the index in the page table.
  - page offset: physical memory offset.
- Logical and physical space addresses don't have to be the same size.
- Might have internal fragmentation: last frame doesn't have to be full, on average half a page per process.
- Larger page sizes means more space wasted.
- Smaller page sizes means larger tables.

Virtual Memory

- Only part of the virtual memory space is mapped to the physical memory.
- Demand paging: only bring a page into memory when its needed.
- *Page fault*: when trying to access an invalid page.
- Handling a page fault:
  - Trap OS.
  - Read page from memory.
  - Allocate CPU to another process until the read operation is done.
  - System will interrupt when the process is done.
  - Add page to the page table.
  - Wait for the process turn again.

    - Perform interrupted process.
- Avg page fault time = (1 - p) * memory time(cache) + p * page fault time(main memory)
- Page replacement: when a process needs a page and there is not space in the frame.
- Victim: a page that will sacrificed for another.
- FIFO: First In First Out
- LRU: Least Recently Used
    - Lots of overhead
    - LRU Approximation
        - Exhibits temporal and spatial localities.
    - Other: LFU, MFU, Least/Most Frequently Used
- Processes exhibit locality of reference, only access a small section of pages call working set.

Page Table Implementation
- Translation Lookaside Buffer (TLB): a small cache for page tables.
- Effective Access Time (EAT): hit time * hit ratio + miss time * miss ratio
- Protection bits determine if the page is read-write or read-only
- Page tables structure:
    - MultiLevel
    - Hashed
    - Inverted

Program Structure
- Choosing different data/programming structures can increase locality.
- Array vs. Linked list

	Array	Linked List
size	fixed: resizing is expensive	dynamic
insertion and deletion	inefficient, usually shifs.	efficient
Access	efficient indexing	no indexing
memory waste	yes	no
locality	yes	no

- Stack has good locality.
- Hash table has poor locality.

Paging: Design Issues
- *Thrashing*: a process that causes page faults every few instructions.
- High page fault will cause

- low CPU utilization
  - OS will try increase the degree of multiprogramming
- Each process will require a minimum number of pages, determined by hardware.
- Global replacement
  - Selects a frame for the set of all frames, chooses what page to replace based on the algorithm.
  - Process can't control its own fault rate.
- Local replacement
  - Assumes fixed frame allocation.
  - A thrashing process can't cause another to thrash when stealing from its resources.
- Working set varies over time, dynamic frames/frame sizes needed.
- Global is better as it has better utilization.
- Fixed/Proportional Allocation: All processes start with a fixed/proportional number of pages.
- Prepaging: when fetching a new page, bring in adjacent pages.
- Copy-on-Write: allow parent and child processes to use the same pages
- I/O interlock: pages that are in use cannot be replaced.
- Windows uses prepaging, clustering.
- Solaris: global replacement
  - If the number of free frames falls under a threshold, pageout is called.
  - Pageout scans all pages and sets reference bit to 0, will later check if that page has been written to, if not, then its that page is freed.
- Linux: global replacement
  - LRU and others
- Android
  - No swapping.
  - Implements paging.
  - Apps are forked from Zygote.
  - Static data is mapped to specific pages.
  - Some dynamic memory is shared by android and applications.
  - Non-foreground applications are in the LRU cache, allows faster app switching.

Mass Storage Systems

- Mass Storage requirements
  - Large space.

    - Persistence data.
    - Allows simultaneous access to process.
- Magnetic disks and Magnetic tape.
- Tranks: rings.
- Sector: a piece of the ring.
- Reading/Writing operation
    - Position the arm head to the correct cylinder.
    - Rotate the platter until the desired sector is reached.
- Access time
    - Seek time: arm head positioning, most expensive.
    - Rotational time.
    - Transfer time.
- Bus types:
    - Enhanced integrated drive electrics (EIDE)
    - Advanced Technology Attachment (ATA)
    - Serial ATA (SATA)
    - Universal serial bus (USB)
    - Small computer-systems interface (SCI)
- Controllers: carries data transfers operations.
    - Host: motherboard end of the bus.
    - Disk: built into disk drives.
- Host controller contacts the disk controller to process commands.
- Disk controllers have caches.
- Network attached storage (NAS): storage accessed over the network.
    - Common protocols: Network File System (NFS), Common Internet File System (CIFS). Implemented use Remote Procedure Calls (RPCs).
- Storage Area Network (SAN).
- Disk Scheduling
    - Pending request table, is implemented in the disk software, indexed by cylinder number.
    - First-Come, First-Serve (FCFS)
        - Fair, predictable order.
        - May swing around as some locations might have large seeks between them.
    - Shortest Seek Time First (SSTF)
        - Minimizes seek time.
        - May cause starvation.
    - Elevator (SCAN)
        - Requests at each ends may take a while.

    - C-SCAN: elevator with not return trip.
- Optimization: lookahead, needs cache.
- Redundant Array of Inexpensive Disks (RAID)

- Parallelism, improves performance.
  - Striping: divides data to differnet disks to allow parallelism, called interleaving. Can be done to bits, bytes, or blocks.
  - Mirroring: creates two copies of the data and stores them on different disks. Expensive, slower write, better read.
- Information redundancy, improves readability.

- RAID levels
  - level 0: non-redundant striping.
  - level 1: mirrored disks.
  - level 2: memory-style error-correcting codes.
  - level 3: bit-interleaved parity. Odd/Even parity check.
  - level 4: block-interleaved parity.
  - level 5: block-interleaved distributed parity.
  - level 6: mirror, strip, and error check.

File Systems 1

- Files: collections of related information recorded on secondary storage.
- File attributes: Name, size, location, type, protection, creation time.
- Files systems
  - Unix: Unix File System (UFS)
  - Windows: File Allocation Table (FAT), New Technology File System (NTFS).
  - Linux: extended file system (1,2, 3) (ext#)
  - Google File System (GFS)
- File control block (FCB): holds the file information.
- Directory Implementation
  - Linear list
  - Hash Table
- Memory Structures
  - Directory-Structure Cache: holds information about recently accessed directories
  - System-wide open file table: contains copies of the FCB of every open file.
  - Per-process open-file table: contains pointers to the open files from the System-wide open file table.
  - Buffers: buffers file data.
- File creation: create a FCB, updates current entries.
- File Opening: OS checks if the file is open by another process.
  - If it is, add a new entry to the per-process open-file table.
  - Else, search for the file. Copy the FCB to the System-wide open file table. Finally, add a new entry to the per-process open-file table.
  - Return the file descriptor.
- Master Boot Record (MBR): sector 0. The end of the sector contains the partition table.
- First block in the active partition is the boot block.
- Allocation methods
  - Contiguous Allocation.

        - Easy to implement, read performance is excellent.
        - Fragmentation will need periodic compaction.
        - Good for burnables: CDs, DVDs
    - Linked List Allocation.
        - No fragmentation.
        - Random access is slow.
    - Linked List Allocation using Index.
        - Needs a File Allocation Table (FAT).
        - No disks references needed.
        - Entire table is in the memory.
    - Index Allocation.
        - A block contains the addresses of all blocks.
        - Index block should be small to reserve resources. Too small may not enough, too large maybe a waste or resources.
- Index Allocation Mechanisms
    - Linked List
    - Multilevel index
    - Combined: direct and indirect blocks.
- Free Space Management
    - Bitmap: map for the memory blocks. 0 for full and 1 if empty.
    - Linked List: links together all free disk blocks.

File Systems 2

- NFS client connects to the NFS machine.
- Directories can be mounted. They could be remote machines, and not necessarily the same system.
- Export List: where to export, serve side.
- File handle: set of information.
-