1 Memory Management

1.1 Requirements

- **Relocation**: Programmer/compiler does not know where program will be in memory. Program may also be moved around during execution. Memory references in the code must be translated.
- **Protection**: Processes shouldn't be allowed to access other's memory without permission. Impossible to know values since program can be relocated, which requires runtime checks. OS cannot anticipate all accesses of a program.
- **Sharing**: Allow several processes to share the same portion of memory. Better to allow each process access to the same copy of a program than have separate copies.

1.2 Methods - Fixed Partitioning

- Memory is divided into partitions which are assigned on demand. These can be equal or variable-sized.
- Inefficient, any program, no matter how small, occupies and entire partition.
- **Strategies**: Equal-sized requires no strategy. Unequal typically assigns each process to the smallest partition which it can fit, wasting the least amount of memory.
- **Partition Assignment**: Can give each partition a queue or have a main queue for the entire memory for processes to wait.

1.3 Methods - Dynamic Partitioning

- Processes are allocated exactly as much memory as required.
- Eventually holes start appearing in memory (unallocated bits when processes finish).
- **External Fragmentation**: Chunks of unallocated memory left from processes finishing. When all over, processes can't find space.
- **Internal Fragmentation**: With fixed page sizes, the last page of a program that can be partially filled.
- Must use **compaction** to shift processes so they are contiguous and free memory is in one block.

1.4 Strategies for Dynamic Partitioning

- **First-fit**: fastest, however can have many processes allocated at front end that must be searched over while searching for a new block. Looks for the first block of memory that fit.
- **Best-fit**: Worst performer, chooses block that is closest in size to request. Causes more computation to be required as external fragments are the smallest too.
- **Next-fit**: Like first fit, but searches for free blocks beyond the last (time) allocated block. Results in event distribution of free block in memory.

1.5 Methods - Buddy System

- Entire space is treated as a single block of 2^u. If a request of size s is such that $2^{u-1} < s \leq 2^u$ then the entire block of 2^u is allocated.
- Otherwise, split into two equal buddies.
- Splitting continues until smallest block $\geq$ s is generated.
- Data structure resembles a binary tree.

2 Virtual Memory

- **Page Fault**: Page table indicates that virtual address isn't in memory, os exception handler is invoked to move data from disk to memory.
- **Translation Lookaside Buffer**: High speed cache to look up page table entries. Stores most recently used page table entries. Uses associative mapping (many page numbers can map to the same TLB index).
  - Given a virtual address, the processor examines the TLB.
  - If a page table entry is present, it's a "hit" and the frame number is returned which leads to the read address.
  - If it isn't ("miss"), the page number is used to look it up, and the TLB is updated with this data.

2.1 Page Sizes

- Multi-level page tables allow for a new page table above another, which then maps to other entries.
- Smaller page side leads to less internal fragmentation, larger page tables, and page tables ending up in virtual memory (as they are so large). This can, in turn, lead to double page faults.
- Small page sizes means more pages fit in memory, leading to fewer page faults. This is due in part to smaller pages that are used frequently staying in memory, instead of large chunks with a small bit used taking up space.
- Secondary memory is designed to transfer large chunks of data efficiently, which favors large page sizes.
- Larger page sizes leads to some useless references being in memory taking up space.

3 Uniprocessor Scheduling

3.1 Aims

- **Response Time**: time it takes a system to reactive to a given input (reduce).
- **Turnaround time**: (TAT) total time spent in the system, waiting time + service time.
- **Throughput**: jobs per minute (inverse of TAT), maximize.

3.2 Types

- **Long-Term Scheduling**
  - Determines which programs are admitted to system for processing
  - Controls degree of multiprogramming
  - Which job to admit? FCFS, Priority, expected exec time, I/O reqs
- **Medium-Term Scheduling**
  - Part of the swapping function
  - Swapping-in decision is based on the need to manage the degree of multiprogramming
- **Short-term scheduling** is the **dispatcher**. It executes most frequently and is invoked when events occur. Including clock interrupts, I/O, OS calls, signals, etc.

3.3 Factors

- Priorities: scheduler should choose higher priority processes over lower priority ones. Uses many ready queues to represent each level. Lower priority processes can suffer starvation.
- Decision Modes:
  - **Nonpreemptive**: Once a process is in the running state, it will continue until termination or blocks self.
  - **Preemptive**: Currently running process can be interrupted and moved to the ready state due to external event. No single process can monopolize processor for long.

3.4 Schedulers

- First-Come-First-Serve (FCFS): nonpreemptive scheduler where oldest process is scheduled to run next.
  - Advantage: favors CPU-bound processes. I/O processes wait for CPU bound ones to finish.
  - Disadvantage: Short processes can wait for a long time before running.
- Round Robin (RR): preemptive based on clock (time sliced) interrupt at regular intervals. When an interrupt occurs, process is placed into ready and next job runs.
- Shortest Process Next (SPN): Nonpreemptive, process with shortest **expected** processing time is next. For batch jobs, user is required to estimate the running time . For interactive jobs, OS predicts it. Short processes get priority here and long ones can be starved.
- Shortest Remaining Time (SRT): Preemptive version of $\wedge$. Achieves better turnaround time to $\wedge$ as a short job is given immediate preference to a long one. Hard to estimate remaining time.
- Highest Response Ratio Next (HRRN): Nonpreemptive, uses R= (time spent waiting + service time) / service time decide next process. No starvation possible, shorter jobs preferred.

4 Multiprocessing

4.1 Issues

- Multiprogramming usage— should we allow one application to lock up several cores (maximum speedup)?
- Unless a single queue is used for scheduling, it becomes more difficult to maintain specific disciplines.
- When a single queue is used, a single process can be schedule to run on any processor (master node needed).
- Preemptive schemes (RR) are costly to implement with a single queue approach.

4.2 Real-time systems

- Tasks or processes attempt to control/react to events which must keep up.
- Correctness depends on both result AND time at delivery.
- Critical that the system is reliable, sometimes with failsafe (traffic lights).
- Sometimes include small size, fast context switches, prioritization of scheduling, and special alarms/timeouts.
- **Hard real-time task**: must meet the deadline, causes catastrophic failure/errors w/o.
- **Soft real-time task**: has a deadline that is desired. Makes sense to complete even if late.
- **Periodic tasks**: Only one unit per period T, exactly T apart.
- **Aperiodic task**: has a deadline by which it must finish/start/both.

4.3 Deadline scheduling

- Real time application, not concerned with fairness/response time but with prioritizing tasks based on deadlines.
- Earliest deadlines typically scheduled first.
- Best to know each first:
  - Ready time (periodic tasks know this)
  - Starting/completion deadline
  - Processing time and resource reqs.
  - Priority

4.4 Rate Monotonic Scheduling

- Assigns priorities to tasks on basis of periods.
- Highest priority tasks have the shortest period.
- Always works if the below is satisfied:

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \leq n(2^{\frac{1}{n}} - 1)$$

- Has industrial applications.
- Higher priority tasks have higher frequencies (hz)
- n is the number of tasks, C is cycle time T is processing time?

5 Networking

5.1 Internet Protocol

- Uniform method for host addresses, each host getting a unique one.
- Provides packet delivery mechanisms (**Packet**: standard transfer unit), packets having a header with destination and size and a payload.
- ISO Open Systems Interconnect Model has 7 layers. From top to bottom: Application, presentation, session, transport, network, data link, physical.
- ISO OSI is a framework for specific protocols (FTP/RPC/TCP).

5.2 Low Level Protocols

- Physical layer is the signaling tech— all hardware.
- Data link layer, frame management. MAC addresses, form of hexadecimal letters X XX:XX:XX:XX:XX:XX, 6 pairs.
- Broadcasts are on 6 pairs of FF.
- Examples are wireless, Ethernet, and X.25.
- These addresses are 48 bit.
- Hosts send bits to other hosts in **frames**.
- Hubs take one frame and send it to every other port.

5.3 OSI Layers (cont)

- Network layer
  - Combines networks using IP.
  - Performs packet routing across **gateways**, intermediate hosts.
- Transport Layer
  - TCP makes conns. on network w/ sockets, using IP/ports.
  - Keeps track of order of packet delivery.
  - Ensures all data arrives at destination in order (ACK/SYN).
  - Two basic protocols: TCP and UDP.
  - TCP is transmission control protocol, which is stream-oriented. Unduplicated and reliable.
  - UDP is user datagram protocol which gives no guarantee of delivery or duplication. (more efficient)
- Presentation Layer: converts local representations of data into its canonical form.
- Application layer: provides network services to end users, FTP clients, telnet, SMTP.

5.4 Domain Name System

- Translates symbolic hostnames into IP addresses
- Hierarchical, distributed naming system for things on the internet
- IP uses 32-bit address (4 sets of #s, 0-255)
- Each (sub)domain has 1+ **authoratative** DNS servers that public info to name servers.
- DNS server maintains list of resolutions.

5.5 Sockets

- BSD sockets enable communication between client/server.
- Semantics resemble pipes (files), bidirectional.
- Once one is created it can be bound to a port.
- A server assigns an address to its socket + tells all potential clients.
- Servers are passive and always waiting for clients to do something.
- A client obtains correct socket address of any server.
- Clients are active and run automatically deciding when to use a server.
- Low level ports reserved for OS.
- Each port can be bound to an address and used by an application.

6 Distributed Processing

6.1 Applications

- **Databases**:
  - Databases is a very common family of distrib. proc w/ client/servers.
  - The server in a database maintains it
  - Clients use transactions to interact, usually over IP. Many clients can coexist.
- Thinclients (dumb terminals / VM)
- **Three tier models**:
  - Application software distributed on all 3.
  - Users use a thin client, while the backend has "legacy applications"
  - Middle tier has gateway, protocol conversion, mapping, and is both a client and a server

6.2 Issues

- Lack of standards leads to **middleware**.
  - Sets of tools to provide uniform style of access and usage across a platform.
  - Provides standard programming interfaces/protocols to sit in the middle of client/servers.
  - Handles complexities and disparities.
  - **SOA**, service oriented architecture. Services with well defined interfaces are given to other groups to maintain.
  - XML via HTTP is a popular interface for communication between services.
  - RPC is another standard (remote procedure calls).
- Message passing schemes
  - Guarantees delivery if possible
  - Send the message out without reporting failure/success reduces overhead and allows queueing.
  - Blocking: send does not return control to sending process until sent, does not return control until ACK'd, or not returned until buffered for send.
  - Nonblocking: process is not suspended as result of send/received, difficult to debug but efficient.

6.3 Clusters

- Alternative to SMP— group of interconnected comps working as unified source, acts as once machine.
- SMP is easier to manage and configure.
- SMP takes up less space, uses less power.
- Clusters are better for incremental / abs scalability
- $\wedge$ superior in terms of availability
- $\wedge$ Better price/performance.
- Load balancing is a problem for clusters.
- Failure management is another issue— failure tolerant or highly available?

7 Glossary

7.1 Exam 1

- **Address Translator** – A functional unit that transforms virtual addresses to real addresses
- **Busy waiting** – the repeated execution of a loop of code while waiting for an event to occur
- **Context Switch** – an operation that switches the processor from one process to another, by saving all the process control block, registers, and other information for the first and replacing them with the process information for the second
- **Direct Memory Access (DMA)** – a form of I/O in which a special module, called a DMA module, controls the exchange of data between main memory and an I/O device. The processor sends a request for the transfer block of data to the DMA module and is interrupted only after the entire block has been transferred.
- **File Allocation Table (FAT)** – a table that indicates the physical location on secondary storage of the space allocated to a file. There is one file allocation table for each file.
- **Job Control language (JCL)** – a problem-oriented language that is designed to express statements in a job that are used to identify the job or to describe its requirements to an operating system
- **Kernel** – a portion of the operating system that includes the most heavily used portions of software. Generally, the kernel is maintained permanently in main memory. The kernel runs in a privileged mode and responds to calls from processes and interrupts from devices.
- **Race Condition** – an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence in order to be done correctly.
- **Starvation**: process is delayed forever as other processes are always given preference.
- **Time Sharing** – the concurrent use of a device by a number of users
- **Time Slicing** – a mode of operation in which two or more processes are assigned quanta of time on the same processor

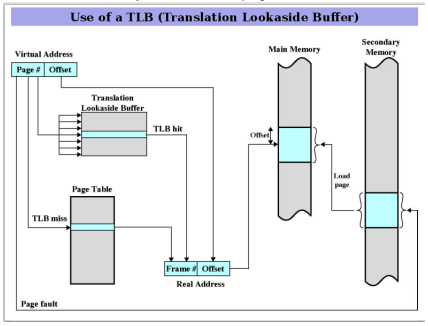
7.2 Exam 2

- **Base Address**: an address that is used as the origin in the calculation of addresses in the execution of a computer program.
- **Dynamic Relocation**: a process that assigns new absolute addresses to a computer program during execution so that the program may be executed from a different area of main storage.
- **Indexed Sequential Access**: pertaining to the organization and accessing of the records of a storage structure through an index of the keys that are stored in arbitrarily partitioned sequential files.
- **Logical Address**: a reference to a memory location independent of the current assignment of data to memory. A translation must be made to a physical address before the memory access can be achieved.
- **Page**: in virtual storage, a fixed length block that has a virtual address and that is transferred as a unit between main memory and secondary memory.
- **Paging**: the transfer of pages between main memory and secondary memory.
- **Physical Address**: the absolute location of a unit of data in memory (e.g., word or byte in main memory, block on secondary memory).
- **Sequential Access**: the capability to enter data into a storage device or a data medium in the same sequence as the data are ordered or to obtain data in the same order as they were entered.
- **Sequential File**: a file in which records are ordered according to the values of one or more key fields and processed in the same sequence from the beginning of the file.
- **Spooling**: the use of secondary memory as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer.
- **Trojan Horse**: secret undocumented routine embedded within a useful program. Execution of the program results in execution of the secret routine.
- **Virtual Address**: the address of a storage location in virtual storage.

8 QA Pool

8.1 Memory Management

- **Page fault trap** interrupt is created when a desired page frame isn't in RAM.
- **Valid bit** is set in the page table when it is not in RAM. This is how it knows it isn't.
- **Dirty bit**s are set when a page frame has been modified.
- **Locality**: process executes in clustered pages (low fragmentation). Bad is opposite.
- **Global allocation**: allows a proc. to select a replacement frame from a set of all frames, even if it's given to some other process; processes can steal frames.
- **Working set allocation**: assumes processes execute in localities. Each process should be alloc'd enough frames for current set.
- Which of the two above are more affected by bad locality? Working set as those with bad locality have poorly defined working sets, so more page faults.
- Page fault timeline:
  - OS blocks proc, puts in waiting queue
  - Ready queue proc. is selected to run
  - DMA is initiated to load faulted page
  - Page replacement strat. ran
  - Page table updated to reflect change
- TRUE: When a DMA takes place, processor does other things.
- FALSE: DMA interrupts CPU by stealing cycles.
- Largest program that can execute on a comp. with 24-bit virtual addresses is a 2^{24} byte program.
- Same question as $\wedge$ but with hardware: can't tell, need to know virt. size.
- Address in a **TLB** entry **PTE** is physical.
- Flags in a PTE: valid bit, reference bit, dirty bit.
- **Hit-ratio**: in 2 level memory system (RAM-HD/cache-RAM), it is the frac of mem accesses found in master (first).
- FALSE: Misses in the **TLB** guarantee entries in the page table entry— *X may not be resident in RAM.*
- TRUE: It is possible that page tables are stored in virtual memory.
- TRUE: page sizes must be large enough to offset high cost of page faults.
- TRUE: in virt. mem. system, cannot run program whose size $>$ main mem.



- Valid bit states whether the page table entry has the disk address.
- **Demand Paging**: only load virtual pages as accessed.
- **Prepaging**: bring more pages in than needed, ones that follow.

8.2 Sockets

- The second argument to listen(sock,#) determines wait queue size, not max.
- Fat client models DO take advantage of desktop power (they ARE desktops).
- All of the following socket commands descriptors return -1 on failure.
- socket(..., SOCKET_STREAM, 0). Creates TCP socket and rets. descriptor.
- bind(sd, -struct cast madness-, len). Binds the definition of a socket to a port #.
- socket(..., SOCKET_SOCK_DGRAM, 0). Creates a UDP socket and rets. descriptor.
- accept(sd, -cast madness-, len). Blocks until client connection received, returns descriptor when it happens.

8.3 Scheduling and Processing

- Direct goals of proc. schedulers: improve response time, throughput, TAT, efficiency

8.4 Security and Triumph of the Nerds

- Most antivirus software uses emulation / heuristics.
- Logic bombs, trojan horses, and viruses require a host program to operate.
- "Bots" attack require: attack software, many vulnerable machines, and locating those machines.
- Dennis Richie and Ken Thompson basically made C/Unix. **Worship them**.
- Bill Gates and Paul Allen started MS in 1975.
- Xerox/PARC made drop down menus, the mouse, windows, etc.
- Steve Jobs and Steve Wozniac co-founded Apple. The former then started NeXT and was the CEO of pixar.
- MSDOS was mostly QDOS, which Tim Patterson wrote, owned by CL Computer Productions, cloned by CPM, which was written by Gary Kildall.
- Jobs saw a GUI at PARC that inspired him to computer real good.
- $\wedge$ also saw OOP and e-mail which he ignored
- $\wedge$ created Lisa after $\wedge$ which flopped. Then created Macintosh (2nd).
- BASIC language interpreter kickstarted MS into the microcomp. bidness.
- MS got into the OS market when Kildall didn't persue IBM when they wanted a new OS. His wife/attourny didn't want to sign an NDA. Gates saw this and jumped in.
- Apple purchased NeXT and its OS NExTStep in 1996.
- **Killer App**: something so useful people buy comps to run it.
- Apple II's $\wedge$ was Visicalc.
- IBM's PC's $\wedge$ was Lotus 1-2-3.
- Macintosh's $\wedge$ was a WYSIWYG desktop publisher.
- IBM didn't create their own OS on their first PC as they wanted to make it super fast, under a year. They couldn't design much, just had to slap it together.
- Compaq had to reverse engineer ROM-BIOS from IBM's first PC as it was proprietary.
- IBM decided on an open architecture to save time. They bought computer components off the shelf and assembled them, hence "open architecture". As a result IBM also had to buy the OS and other software from other vendors.
- Ed Roberts at MITS built the first commercially available PC in 1975.
- Gordon Moore is an Intel founder.
- Altair 8800 was the world's first personal computer and was designed by Ed Roberts in 1975.
- First mass market PC company is Apple.

8.5 Weird Questions

- RAID #'s: 1, mirror; 0, merge drives (splits data even); 10—2, striping, error detection, and fault tolerance (common).