# Section 7: Memory Management

## Question 1

Describe both the logical and physical views of a process’s memory.

### Answer

Logical View: The logical view of the process image is the illusion that the process image is installed in a (more or less) contiguous range of memory addresses that starts at address zero. A process’s logical address space is owned by the process and is separate from all other process images. That is, that each process has its own 0-N logical address space.

Physical View: The physical view of the process image is the reality that all process images share physical memory. Each process is installed into a memory partition at unique addresses in physical memory. Partitions can be relocated to new addresses, and with paged memory management can be placed in non-contiguous (non-adjacent) memory regions.

The translation from logical to physical addresses is primarily accomplished by the system’s Memory Management Unit hardware.

## Question 2

Describe the three types of dynamic partition placement algorithms described in class.

### Answer

First Fit: The operating system scans main memory, starting from the top, to find the first unallocated region of memory into which the process will fit regardless of amount of external fragmentation it creates.   
  
Next Fit: The operating system scans main memory, starting from the partition last allocated, to find the next unallocated region of memory into which the process will fit.  
  
Best Fit: The operating system scans main memory to find the unallocated region of memory that is the closest / best fit i.e. is the smallest unallocated region into which the process will fit.

## Question 3

1. Describe the meaning of process image partition relocation?
2. How is relocation of a process image implemented in the Primitive Memory Management Unit (MMU) described in the slides?
3. What is the unit of relocation in paged memory management?
4. Does Paged Memory Management use absolute or relative logical addresses?

### Answer

1. A process image is placed into and executed from a memory partition. A processes’ image can be relocated (moved) from one partition (region of physical memory) to a different partition throughout its lifetime. For example, slide 4 illustrates P1 relocation from one partition to a second.
2. The Primitive MMU implements relocation with two registers: Base and Bounds. The base register maintains the start of the process’s partition and the bounds maintains the partition’s size. A logical relative address is translated into a physical address by adding the bounds register to the relative logical address.
3. In Paged Memory Management, the unit of relocation is a Page of the process image.
4. The Paged Memory Management uses absolute logical addressing. Paged Memory Management uses the high order N bits of a logical address as an offset into the process’s Page Table to lookup the base address of the physical memory frame assigned to the page. The lower M bits of the logial address provide an offset into the frame to generate the physical address.

## Question 4

What is the purpose of *memory* *overlaying* as described in the text?

### Answer

Overlaying is an outdated technique that allows the execution of programs that are too large to fit into the available physical memory. It allows the programmer to explicitly divide their program into multiple executable modules and then specify which of the modules are moved into memory for execution. Newly loaded modules will replace (overlay) program modules currently in memory.

For example, a video game has many levels and players leave one level and moves into another. The code, maps, texture images, etc. for the current level can be overlaid (replaced) with the next level’s content allowing for many levels to be loaded and played in the same physical memory.

## Question 5

1. Describe the purpose and contents of the page table as described in this section.
2. How many page tables are needed to maintain N processes?

### Answer

1. The page table maintains a number of page table entries each of which maintains process’s mapping from logical page numbers to physical-memory frame numbers.
2. Each process maintains its own page table.

## Question 6

NOTE: When creating the elearning assessment, be sure that superscripts are used.

Consider a simple paging system with the following parameters:

* 232 bytes of physical memory
* A page size of 210 bytes
* 216 pages of logical address space

Note: In order to reduce the max size of the page tables, this system’s memory management hardware constrains each process’s page table to 216 entries. This has the effect of reducing the size of the logical address range to less than the physical address range.

1. How many bits are needed to specify a logical address?
2. How many bytes (words) in a frame?
3. How many bits of a physical address are used to specify the frame address?
4. How many entries in a process’s page table?
5. How many bits in each page table entry?

### Answer

1. The number of bytes in the logical address space is (216 pages) x (210 bytes/page) = 226 bytes. Therefore, 26 bits are required for the logical address.
2. A frame is the same size as a page, 210 bytes.
3. The number of frames in main memory is (232 bytes of main memory) / (210 bytes/frame) = 222 frames. So 22 bits are needed to specify the frame address.
4. There is one entry for each page in the logical address space. Therefore there are 216 entries.
5. In addition to the present bit, 22 (32-10) bits are needed to specify the frame location in main memory, for a total of 22 bits.

## Question 7

NOTE: When creating the elearning assessment, be sure that superscripts are used.

Note: Intel hardware’s page size is 212 or 4096.

1. What is the largest internal fragment that can occur with paging on Intel hardware?
2. What is the largest external fragment that can occur with paging on Intel hardware?

### Answer

1. The worst case is a page with a single byte resulting in an internal fragment of size (PageSize -1) or 4095 bytes.
2. A trick question. There is no external fragmentation with paging. (Grader: Do not score this second question).

## Question 8

What is the maximum number of pages that can be addressed in Figure 7.12a?

### Answer

The page number is six bits long or 26 for 64 page entries.

## Question 9

This question refers to the buddy system described in Figure 7.7.

Initially the allocation map is:

128K (A) | 64K (C) | 64K free | 256K free | 256K (D) | 256K free

1. What will the allocation map look like when only Process A exits?
2. What will the allocation map look like when only Process C exits?
3. What will the allocation map look like when only Process D exits?

Please provide three allocation maps i.e. one each for 1, 2, & 3.

### Answer

1. 128K free | 64K (C) | 64K free | 256K free | 256K (D) | 256K free
2. 128K (A) | 128K free | 256K free | 256K (D) | 256K free
3. 128K (A) | 64K (C) | 64K free | 256K free | 512K free

The point is that only sibling nodes are consolidated when both are freed.

## Question 10

What is buffer overflow and how is the following code segment susceptible to it?

int main(void) {  
 char buff[50];  
 int age = 2;  
 …  
 gets(&buff);  
 printf(“You Entered %s\n”, buff);  
 …  
}

### Answer

Buffer overflow occurs when a flaw in the code causes an unchecked input operation to overwrite its assigned input buffer to corrupt adjacent data on the stack or in memory.

In the example, the gets() function will copy input from stdin into a buffer passed to the function until a newline is found. Notice that gets is passed the location of a character buffer ‘buff’ with a fixed length of 50. If the data read from stdin exceeds 50 bytes, gets will begin to overwrite the adjacent integer ‘age’ with the data it is copying.

NOTE: The Unix / Linux library function gets(3) was replaced with the function fgets(3) which requires an additional argument: the buffer size to check for and prevent overflow.

# Section 8: Virtual Memory

NOTE: Please Review Answers Before Posting

## Question 1

What are the two characteristics of simple paging that lead to, and the fundamental idea defines virtual memory? Hint: Three parts in slides.

### Answer

The two characteristics of simple paging are:

1. Because of the indirection between logical and physical addresses, a process’s pages can be located anywhere in memory and can be moved (relocated).
2. The frames occupied by a process’s pages do not need to be contiguous i.e. they can be spread throughout physical memory.

And the fundamental idea in virtual memory is:

1. Not all of a process’s pages need to be maintained in physical memory for the process to execute. Only those pages in the process’s working set need to be resident.

## Question 2

1. Briefly describe Page Fault.
2. Briefly describe Page Fault Rate (PFR).
3. Briefly describe Thrashing.

### Answer

1. A page fault is the interrupt generated by the memory management unit when the process references a page that is not resident in memory i.e. not in the resident set.
2. The page fault rate is a metric that describes the number of page faults that occur over some time period e.g. PFs per minute.
3. Thrashing occurs when a process’s page fault rate (PFR) rises to a point that the process cannot make progress i.e. each time the process is scheduled for execution (is made runnable) a missing page that the process requires and generates a page fault that blocks its execution.

## Question 3

Note: This question assumes a static resident set size.

1. Briefly describe a process’s Resident Set.
2. What is the result of setting a process’s resident set size too small?
3. What is the result of setting a process’s resident set size too large?

### Answer

The resident set is the set of process pages that are resident in (paged into) memory.

A resident set that is too small will cause an increase in the number of page faults and will cause the process to begin thrashing.

A resident set that is too large wastes memory (frames) by maintaining process pages in memory frames that are no longer being referenced by the process. This has the effect of reducing the overall number of processes that can be maintained in memory.

## Question 4

1. Briefly describe a process’s Working Set.
2. Briefly describe the relationship between a process’s working set and its resident set.
3. Generally, how is a process’s page fault frequency used to adjust a process’s resident set size?

### Answer

1. The Working Set is the set of pages that the process currently needs to execute. Any page in the working set that is not resident must be swapped in.
2. The resident set must include at least the working set. If the resident set is smaller than the process’s working set, the process will begin to thrash.
3. If the resident set is smaller than the working set, the PF Frequency will rise providing an indication that the process’s resident set’s size must be increased. When the resident set contains the working set, the PF Frequency will drop. The OS can monitor the PF Frequency using it as an indicator of how to adjust the process’s resident set size up or down.

## Question 5

1. What three pieces of information is maintained in a page table entry (as described in the slides)?
2. How many page table entries are needed with a 32 bit address and a page size of 8192?
3. What is the technique described in the book, and in class, that reduces the number of page table entries needed to be maintained in memory?

### Answer

1. Mainly each entry maintains the upper N bits of the physical address of the memory frame containing the page. The PT Entry also maintains a ‘Present bit’ that indicates whether the page is resident (1) or must be retrieved from disk (0). The PT entry also maintains a ‘modified bit’ that indicates whether the page has been written to (modified) since it was made resident.
2. A 32 bit address range with a page size of 213 bits permits (32-19) 219 (524288) pages.
3. Dividing the page table into multiple levels (two in the book) greatly reduces the number of PTE that must be maintained in memory with the tradeoff of additional complexity in the design of the memory management unit hardware (Figure 5.8).

## Question 6

What are the two advantages of a large page size described in the slides?

### Answer

The large page size results in fewer page table entries (smaller page tables), a smaller TLB, and overall reduces the amount of processing overhead of memory management on the system.

The large page size increases the efficient the I/O operations that move pages between the swap drive and memory i.e. more information is moved between the swap drive and memory per I/O operation.

## Question 7

Describe the purpose of the Translation Lookaside Buffer.

### Answer

The TLB is a memory cache for page table entries. When the TLB hits, it avoids the additional memory reference needed to retrieve one or more page table entries from memory depending on whether a 1, 2, 3, etc. level page table is being employed. Without a TLB, every virtual memory reference would require two or more memory accesses (i.e. 1, 2, or more page table entries) plus the reference to the target frame.

## Question 8

1. Describe the Operating System’s Fetch Policy.
2. How does the Prepaging Fetch Policy reduce the number of page faults experienced when a process is initially started?

### Answer

The fetch policy determines when a page is made resident i.e. swapped from disk into memory.

NOTE: The following discussion concerning pre-paging only applies to a hard disk drive. There is no seek penalty in reading non-adjacent blocks with a solid state drive.

The Prepaging Fetch Policy will not only swap-in the page being referenced by the page fault, but will swap-in the adjacent page in a single I/O operation. The assumption is that if page i is needed, then the adjacent page i+1 will likely be needed very soon after. By bringing both pages in with the same I/O operation, the time needed to retrieve page i+1 is reduced to almost nothing relative to the time needed to bring in page i. That is, the expense in the I/O operation is moving the disk’s heads to the needed track and waiting for the correct disk sector to move into position. The time needed to retrieve an adjacent sector is almost nil compared to the setting up and retrieving the first.

## Question 9

1. Describe the goal of a Page Replacement Policy.
2. In theory, how does the Least Recently Used page replacement policy select a page to be replaced when a new page must be made resident?
3. Describe how the ‘use bit’ marks pages in the Clock Algorithm. How is the use bit used to select a page for replacement?

### Answer

To reduce the page fault rate by selecting pages for replacement that will not be referenced for the longest amount of time.

The LRU policy selects the page that has not been referenced in the longest amount of time. According to the principle of locality, the LRU page is least likely to be referenced in the near future.

The frame table maintains a use bit (UB) for every memory frame. The UB is an indicator of whether the frame has been referenced in the recent past. The Clock page replacement algorithm scan the frame table until a UB = 0 is located which is then selected for replacement. During this scan, if the frame’s UB = 1, it is set to 0 and the next frame examined. Every time a process references a frame (i.e. references the page in the frame), the UB is set to 1. Therefore, the only frames (pages) that have not been recently referenced will be selected for replacement.

Note: You may be required to explain the clock algorithm in an exam question.

## Question 10

1. Describe the goal of a Load Control Policy.
2. Describe how a process’s Working Set can be used to determine which processes to maintain in a ready state or which to suspend.
3. Describe the L=S Criteria approach to determining how many processes to maintain in a ready state.

### Answer

The load control policy determines the number of processes to make active (i.e. not suspended) with the goal of maximizing processor utilization. Too few processes reduce the chance that a process will be in a ‘ready’ state (not blocked for I/O). Too many processes reduces the resident set sizes of individual processes and so increases process thrashing.

The Working Set method schedules processes for execution whose current working set is contained in its resident set and suspends processes whose working set is not currently resident.

The L=S Criterion method maintains the number of executable processes such that the average time between page faults is equal to the average time needed to swap a page from disk (length of the IO operation). Too high a page fault rate indicates that too many processes are active and some need to be suspended to free up frames for the remaining processes.

# Section 9: Distributed Processing

## Question 1

Describe the difference between Synchronous and Asynchronous messaging.

### Answer

Synchronous messaging almost always describes a client-server architecture and implies a synchronization between the client and server processes. That is when the client makes its request of the server, the client blocks its execution until a response is received. In practical terms, the client’s request should be time limited and a “timeout” error (or exception) will be generated if the server’s response is not received.

Asynchronous messaging describes a message-driven (event-driven) architecture. Interacting processes have a peer-to-peer relationship, are not synchronized, and each process is capable of sending and receiving messages.

## Question 2

1. Describe how information flows between two endpoints in a TCP Socket.
2. Describe the relationship between InputStream and OutputStream with respect to the flow of information.
3. How is blocking I/O of an InputStream used to regulate this flow.

### Answer

A TCP Socket represents a two-way communication channel between two processes. Each end of this channel has two endpoints that allow the processes to exchange information i.e. streams of bytes. Each endpoint allows a process to both receive bytes from, and to send bytes to the other process.

To accomplish this each endpoint is supported by an InputStream and OutputStream. The InputStream has a read() operation that is used to read the bytes sent by the other process. The OutputStream has a write() operation that is used to send bytes to the other process.

The InputStream of each endpoint is blocking. That is, when a process read()’s from its InputStream, the process blocks if the stream is empty and will continue to block until the other process writes data into it’s OutputStream.

## Question 3

Describe the responsibility of the two layers in the TCP/IP protocol.

### Answer

IP or Internet Protocol is responsible for transporting fixed length packets of data (bytes) from one machine to another machine across the IP network. The IP network supports the ‘routing’ of packets between connected machines across the world-wide Internet.

TCP or Transport Control Protocol is responsible for implementing the socket mechanism by breaking the stream of bytes transmitted by a process on its socket endpoint’s OutputStream into many separate packets that are routed on the IP network to the destination machine. TCP is also responsible for re-assembling the packets in the correct order to reproduce the original stream of bytes on the InputStream side of the receiving process.

## Question 4

1. Describe the two-phase message protocol that exists between client and server processes.
2. How does blocking I/O enter into the implementation of the client-server protocol?

### Answer

The Client-Server protocol defines the exchange of two messages between two (client & server) processes. The protocol starts with the Client Process sending a request message to the Server Process. This is accomplished by using a TCP/IP socket. The Server Process waits for the client’s message and ‘processes’ the message in some application-specific manner. This request message processing produces a result or response to the client’s message. The result is returned to the client by the server using the same TCP/IP Socket.

The server waits for the client’s message to arrive by blocking on a read() of its TCP socket endpoint. The server process’s read() on its InputStream blocks until the client sends a message across the socket. Conversely, after the client sends its request message, the client reads and blocks on its endpoint’s InputStream until the server sends its response message.

## Question 5

What are the names and purposes of each of the tiers in a Three-Tier Architecture?

### Answer

**Presentation Tier**: The presentation tier contains the software and services that presents information to, and gathers information from, the system’s users. This may be an application running on a phone or on a browser.

**Service Tier**: The service tier contains the software and services that implement the application information processing and business rules. The service tier software architecture is typically made up of Controller (see the design pattern) and services that are invoked by client requests from the presentation tier.

**Data Tier**: The data tier contains the software and services responsible for persisting data in databases. The persistence of information is often complex and tightly coupled to the specific database and schema. So it is wise to encapsulate the implementation of persistence from the services in the service tier.

It is common for the components / processes that make up each tier to run on separate machines, sometimes using a cluster to increase capacity. These processes communicate with each other using Remote Procedure Calls or other distributed processing technology.

## Question 6

Fill in the blanks:

1. The business process describes the <A> though the enterprise
2. A business process is made up of <B> that accepts information as input, process the information, and generates results.

### Answer

1. flow of information
2. processing steps

## Question 7

1. Describe the role a Producer plays in implementing a message-based architecture and its relationship to message destinations.
2. Describe the role a Consumer plays in implementing a message-based architecture and its relationship to message destinations.
3. T|F: A component is either a Producer or a Consumer, but never both.

### Answer

Producers produce messages. The producer will process and produce a result that is used by different consuming component. The producer will deliver its messages to a message-type specific destination. The destination will forward the messages to consumers that have registered interested in messages of specific types.

Consumers consume (accept and process) messages. The consumer will accept the message and execute some application-specific code with the information provided by the message.

Flase: Many (most) message processing processes are both message consumers (accepting and processing a message) and produce messages as a result of their message processing behavior.

## Question 8

1. Select the option that makes this statement true: Asynchronous messaging systems are <more> <less> coupled that synchronous messaging systems.
2. Describe why Client-Server is considered strongly coupled.
3. Describe whey Messaging is considered weakly coupled.

### Answer

Asynchronous messaging systems are less coupled that synchronous messaging systems.

A synchronous messaging system is made up of clients that require a reference to a server in order to invoke an action on the server and (usually) receive a result / response. This reference is a pointer variable in a language or the network address of the machine hosting the server process.

An asynchronous messaging system is made up of message producers and consumers. Neither of these components is directly coupled to the other. Producers produce messages. Consumers consume messages. They share a common medium (a message server) which routes messages from producers to consumers. However consumers have no knowledge of the entity creating their inputs and producers have no knowledge of the entity consuming the information they produce. In most cases, the producer will continue to produce messages even when there are no consumers present in the system.

## Question 9

1. What are the two types of message destinations supported by most message server?
2. How does one destination differ from the other in terms of how a message is delivered to a set of registered consumers?

### Answer

**Message Queue**: A queue is a named destination that when shared by two or more consumers will route a message to exactly one of the consumers. Queues are point-to-point destinations.

**Message Topics**: A topic is a named destination that when shared by two or more consumers will route a message to every one of the consumers. Topics are broadcast destinations.

## Question 10

1. Describe the role the Load Balancer plays in implementing a cluster of web servers.
2. How does the Load Balancer promote performance scaling?
3. How does the Load Balancer support high-availability?

### Answer

The Load Balancer is a device (hardware or software) that is attached to each server in the cluster. The LB receives all client requests and forwards each (request) to one of the clustered servers. It is important to note that all of the cluster’s servers is running the same services (software), so any server is capable of processing the client’s requests.

The client’s requests are distributed evenly among M servers allowing M requests to be processed concurrently. This increases the performance of the system M-fold. This is the theory anyway, because the actual scaling in performance depends on how much of the request processing can be executed concurrently i.e. see Amdahl’s rule.

If any of the cluster’s servers crashes, the load balancer will detect the failure and remove the failed server from the cluster i.e. stop forwarding client requests. High availability means that the remaining M-1 servers remain operational the service itself remains available to the clients.

# Section 10: Multiprocessor Scheduling

## Question 1

1. Describe the difference between Static Assignment vs. Dynamic Assignment of threads to Processors.
2. As described in the slides, how are both strategies implemented?

### Answer

1. With static assignment, newly created threads are attached to, and execute on, the same processor throughout is lifetime (unless moved by a load balancer).   
   With dynamic assignment, ready threads are assigned by the dispatcher to any available processor i.e. threads migrate between processors.
2. Static assignment is implemented with per-processor ready queues.   
   Dynamic assignment is implemented with a single, global ready queue.

## Question 2

What are the three disadvantages of dispatching processes to multiple processors from a single, shared ready queue (as opposed to having a ready queue per processor)?

### Answer

1. The shared global queue must be thread safe so mutually exclusive access to the shared queue (i.e. locking, etc.) may become a bottleneck.
2. Threads are unlikely to be re-scheduled on the same processor which makes each processor’s local memory cache less effective when threads move from processor to processor. (important)
3. If threads from all processes are dispatched for a single queue, it is unlikely all of the threads belonging to a single process will scheduled at the same time with the other threads belonging to the same process. (important)

## Question 3

In terms of processor cache utilization, what is the advantage of repeatedly dispatching threads to the same processor?

### Answer

As a thread executes, its current locality causes certain text and data pages to be loaded into its processor’s cache (L1 & L2). If the thread is re-scheduled to the same processor, the processor’s cache remains ‘hot’ i.e. all of the thread’s pages will remain in cache. Contrast with the situation where a thread executes on a different processor; each thread starts its execution with a cold cache that must be reloaded with the thread’s text and data pages.

## Question 4

In terms of throughput, what is the advantage of Gang Scheduling?

### Answer

Gang scheduling attempts to keep all of a process’s threads executing simultaneously. This is important when a process’s threads are tightly coupled, as they often are. Tightly coupled threads will block when synchronizing with a non-running peer thread. The blocking results in a type of ‘processor thrashing’ where a thread executes for a short period and blocks waiting to synchronize with a peer. When all of the threads are running simultaneously, there is no synchronization blocking, the process completes sooner.

## Question 5

1. Describe the meaning of Load Balancing in the context of Processor Assignment.
2. Is a Load Balancer needed with dynamic or static processor assignment?

### Answer

1. Load balancing is the reassignment of threads from assigned processors to other less heavily loaded processors. The ‘Load Balancer’ is a task that periodically runs and examines the load each processor is under. A processor’s load may be measured in terms of the number of threads assigned to the processor’s queue (static assignment). The goal is to spread the system’s processing load evenly across all of the system’s processors.
2. Load balancing is only needed with static assignment. Dynamic assignment naturally spreads the load as threads are assigned from a single global queue to the next available (idle) processor.

## Question 6

1. Describe the meaning of ‘Resource Aware Thread Placement’.
2. Describe the example discussed in class.

### Answer

1. Resource Aware Thread Placement has the scheduler assigning threads to processes which optimize system performance because of resources available to specific processors.
2. The example given in class was the placement of threads from the same process on processors that share L1 or L2 cache. The reason is that because threads from the same process share the same text and data memory blocks, their placement on adjacent processors will make the best use of the cache memory shared between those processors.

## Question 7

Describe the meaning of, and relationships between, events, tasks, and deadlines in the context of real-time operating systems.

### Answer

An Event is a signal (usually an interrupt) that triggers the execution of a task.

A Task is code that is executed (triggered) in response to the event. A task can also be regularly scheduled for execution by the RT OS.

A Deadline is a measurement of time between when an event occurs and the completion of the execution of a task that responds to the event and generates an output. The system must respond to, and produce the output within a specific amount of time (its deadline) or suffer some type of failure.

From the example of automotive airbag controllers given in class, a collision (**event**) is detected by an accelerometer and signals the automobile’s on-board computer. The system executes a specific **task** that examines the input and decides whether the air bags should be deployed. The system must respond to the ‘crash’ event and cause the airbags to deploy within a certain amount of time (**deadline**) otherwise the airbag effectiveness will be reduced or even become a hazard to the passengers.

## Question 8

What are the special characteristics of real-time tasks executed in response to incoming events?

### Answer

1. Execution of the task is triggered by an interrupt, not the OS scheduler.
2. Memory hosting the task’s code and data segments cannot be swapped out to disk i.e. no virtual memory.
3. Tasks cannot make any blocking system calls i.e. calls to IO or network communication.
4. The task must be designed and verified to execute well within the deadline imposed by the application and the processing capacity of the processor. Worse-case path analysis must be employed to ensure that no possible combination of state and inputs can cause the task complete after its deadline.

## Question 9

What are two types of tasks dispatched by real-time operating systems?

### Answer

1. Tasks that are executed in response to events from external sources.
2. Tasks that are periodically executed (scheduled) by the operating system.

# Section 11: I/O Management

For Exam: Your answers will be penalized if you describe the disk blocks used to maintain file contents as “Memory”.

For Exam: Know the seven components of the Layered I/O Architecture.

## Question 1

Name and briefly describe the two operating systems design objectives for I/O Management?

### Answer

Efficiency: To obtain the best overall performance from processing (executing) the requests submitted to an I/O device e.g. data transfer rate or disk utilization.

Generality: To provide a standard interfaces between the layers in the operating system’s design. For example, the upper-layer of the OS stack provides a common API that can be used by a program to access most type of I/O devices. The lower-layers of the stack provide a driver interface that is standard across all vendors or drive hardware interfaces.

## Question 2

1. What is a Device Driver and what role does it play in OS design?
2. How does the use of device drivers support the goal of *Generality in OS Design*?

### Answer

1. The device driver is a software module (e.g. a library) that is loaded and installed into the OS between the I/O subsystems and the hardware interfaces to the physical device. For example, a driver is written to a specific operating system (Linux, Windows, etc.) and a specific device controller (a specific type SATA Disk Controller, USB Controller, Graphics Processor, etc.).
2. The driver’s main purpose is to translate the generic I/O commands issued by the OS subsystem (read / write block) into commands executed by the system’s hardware. In this way the generic operations issued by the OS can be translated into vendor-specific hardware commands needed to execute / fulfill those commands. New devices can be installed on the system without modification to the operating system.

## Question 3

1. Describe why we describe drives as ‘block devices’.
2. Explain how data addressed by block devices.
3. Explain how data (e.g. a record) stored on a block device is updated i.e. read-modify-write.

### Answer

1. Block devices manages its data in fixed sized blocks. Data is transferred to / from these devices in blocks.
2. The OS commands the device to retrieve data, or save data as specific block addresses starting at one (first block) though block N where N is the capacity of the drive (in bytes) divided by the block size.
3. Data Update:
   1. The block containing the record must be read from the device into memory.
   2. The record within the block is updated in memory.
   3. The entire block is written back to the drive. This includes the data surrounding the record which was not updated.

The key point is that data is not updated directly on the drive. The block containing the data to be updated must first be copied into memory where it can be accessed by the processor.

## Question 4

1. What is I/O buffering?
2. In terms of process and virtual memory management, what problem does I/O buffering solve?

### Answer

I/O buffering is staging data read-from or written-to an I/O device in memory buffers owned by the OS kernel. For example, when a process wishes to write a block of memory out to disk, the data is first copied from the process’s own memory space into a kernel I/O buffer. The opposite is true for read operations.

Buffering allows the pages containing the process’s source or destination buffers to be removed from memory i.e. to be swapped out or suspended. Without buffering, the process’s memory involved with the transfer would need to remain resident and this would interfere with the OS’s ability to manage virtual memory.

## Question 5

1. From the slides, what is the danger of caching disk blocks in kernel-space memory buffers?
2. What system call does the Linux OS provide to address this problem?

### Answer

1. There is the danger of losing data that was written by the process but has not been physically written to disk because the write operation is waiting to be scheduled or is being cached.
2. The Linux/UNIX OS provide the system call is fsync(). The call fsync() forces the OS to write the data block cached in memory out to the disk. This includes not only cached data blocks, but the blocks maintaining file system data structures such as the FAT and Unallocated Block List.

## Question 6

1. What purpose do drive partitions serve?
2. What is the relationship between a drive’s block capacity and the partitions maintained on the drive?
3. How are the blocks in a partition addressed?

### Answer

1. Partitions allow the system administrators to allocate drive storage blocks to hosting file systems in units smaller that the entire drive’s capacity. For example, instead of allocating the entire drive’s capacity to a single file system, 2 or more partitions can be created and used to host 2 separate file systems on the same physical drive.
2. The drive’s capacity of N blocks is divided among the 2 or more partitions on the drive e.g. A 1000 block drive with blocks 0-999; P1: 0-249; P2: 250-999.
3. The drive’s set of N blocks is divided into consecutive regions (partitions). Each partition is addressed in its own 0-M block address range.

## Question 7

How is the Least Frequently Used replacement policy implemented for disk block cache replacement? (See text)

### Answer

Each disk block cache entry maintains a counter that is incremented each time the cache block is referenced. Blocks with the smallest reference count are selected for replacement.

## Question 8

1. In RAID terms, what is Data Striping?
2. In terms of RAID0, describe the difference between the logical view of the drive seen by the operating system and the RAID’s physical implementation across N disks.

### Answer

1. Data Striping is storing the blocks that make up a file system across several drives (partitions) in the RAID. Pg 498.
2. Logically the OS see a RAID0 drive as N\*M blocks where N is the number of disks in the RAID and M is the number of blocks per disk (assuming that all disks are of the same size). Logical block 0 is mapped to disk 0 block 0. Logical block 1 is mapped to disk 1 block 0. Logical block N is mapped to disk 0 block 1. Etc.

## Question 9

1. What are three advantages of RAID1 (mirroring) given in the book? (Assume a two disk RAID)
2. What is the primary disadvantage of RAID1?

### Answer

Raid 1 is also known as “disk mirroring” where data is duplicated across (written to and read from) two disks.

1. Data on the logical RAID drive is protected from a single-disk failure.
2. A read request can be serviced by either of two disks and two read requests can be executed concurrently.
3. Even though both disks need to be updated for a write operation, the write can be executed concurrently on both disks.

Disadvantage: Because a block is mirrored on two disks, RAID1 uses only half the physical capacity of the RAID.

## Question 10

1. What advantage does RAID2-6 have over RAID1?
2. What disadvantage does RAID2-6 have when compared to RAID1? Hint: How is data retrieved when a disk is lost?

### Answer

The advantage of RAID2-6 is that redundancy is obtained with only a single additional disk that maintains the stripe’s parity information. (unlike RAID1 which requires 2x the disks).

When RAID2-6 loses a data disk, the data must be reconstructed dynamically by reading from each of the remaining blocks in the missing block’s strip plus the parity block to reconstruct data from the missing disk. That is, N-1 block reads to reconstruct blocks from the missing disk.

# Section12: File Management

For Exam: Your answers will be penalized if you describe the disk blocks used to maintain file contents as “Memory”.

## Question 1

Describe the purpose of the File Descriptor.

### Answer

File descriptors are a resource created and managed by the operating system. File descriptors are a data structure created by system calls that create new files or open existing files. The file descriptor is used by system calls that manipulate the contents of a file i.e. read, write, seek, etc. The file descriptor maintains state information about the open file e.g. the stream index (position) of the next read or write operation on the open file.

## Question 2

1. Describe the purpose of the File Allocation Table.
2. Describe the purpose of the Unallocated Block List.

### Answer

1. The FAT is the data structure used to maintain the association between a file and the blocks allocated to the file including the ordering of the blocks.
2. The UBL is the list (set) of blocks that are currently not allocated to any file. It is the file system’s free space.

## Question 3

1. How are file systems allocated to a drive?
2. What determines the size of a file system?
3. What is the root file system?
4. What is a child file system?
5. What does it mean to *mount* a child file system?

### Answer

1. A file system is allocated to a partition. That is, the blocks contained in a partition are allocated to the file system installed in the partition.
2. A file system’s size is determined by the number of blocks allocated to the partition.
3. The root file system is the single file system that is activated when the system boots. The root FS maintains the executables, libraries, and data files needed to boot the Operating System.
4. Child file systems are separate file systems which have been created by the administrators to organize data other than the operating system. For example, it is common to place the user’s home directories in a separate file system. Or to put database or other application data files in a separate file system.
5. Multiple file systems can be arranged in a tree with the root FS as the tree’s root and zero or more FS mounted at mount points in a mounted FS. Normally, all child FS are mounted in the root FS.

## Question 4

1. What purpose does the ‘free block bitmap’ serve?
2. What is the formula for calculating the size of the free block bitmap?
3. How large of an array (in bytes) will be needed to maintain a 1 Gig disk (1073741824) with 512-byte blocks?

### Answer

1. The free block bitmap is a technique for maintaining the unallocated blocks in a file system i.e. identifying the blocks not yet allocated to a file. The bitmap is a logical array of N bits (0|1) that represent whether block i is allocated or free (where N is the number of FS blocks).
2. 262144

## Question 5

1. What is a File System Extent?
2. What is the relationship between an extent and a file?
3. Why are large extents desired from the file allocation method?

### Answer

1. An Extent is a range of contiguous (adjacent) disk blocks that has been allocated to a file or found in the file system’s free list.
2. A file can be allocated as one or extents where extent sizes range from a single block to an extent that encompass the entire file.
3. Large extents are encouraged because as sets of contiguous blocks, an extent can be more efficiently retrieved from the drive.

## Question 6

1. Describe Contiguous File Allocation.
2. What are the two disadvantages of contiguous allocation?
3. What information is maintained in the FAT for contiguous allocation?

### Answer

1. Each file is allocated (laid out in) a contiguous range of drive blocks i.e. in a single extent.
2. 1) External fragmentation becomes a problem when the disk fills with many small regions of unallocated extents. 2) Extending a file requires moving the file from its current extent to another larger extent.
3. The FAT maintains the index of the first block and the number of blocks allocated to the file / extent.

## Question 7

1. Describe Chained File Allocation.
2. What two disadvantage does chained allocation offer?
3. What information is maintained in the FAT for chained allocation?

### Answer

1. The blocks allocated to a file maintain as a ‘linked list’ i.e. each block maintains the index of the next block in the file. Each block maintains the index of the next block.
2. The use of a pointer in each block 1) reduces the space available in the block for file data and 2) causes the amount of data stored in the block to be something other than a power of two.
3. The FAT maintains the index of the first block and the number of blocks in the chain. Optionally, an end of chain / file marker can mark the last block in the chain.

## Question 8

1. Describe the purpose of the ‘Inode’.
2. What information is maintained in the FAT for Indexed Allocation with Inodes?
3. How is inode overflow handled in the Linux filesystem?

### Answer

1. The Inode (Index Node or Index Block) is a disk block that maintains an ordered list of block numbers (addresses). The blocks in the list contain the file’s contents.
2. The FAT contains the file name and block address of the file’s inode.
3. When the size of the file exceeds the number of direct block addresses in the root inode, the root node maintains pointers to indirect address blocks which is a block containing file block addresses. Indirect address blocks can be nested like trees allowing for very large file sizes. See Slide 44 “Linux Nested Block Structure”.

## Question 9

What is the reason for file system consistency checks?

### Answer

For reasons of efficiency, the file system’s metadata structures (FAT and Unallocated Block List) are cached in memory. This information needs to written to disk before the system is shut down and the contents of the in-memory cache are lost.

If the system loses power, or otherwise crashes, the cached state of the file system will be lost. For this reason, the operating system frequently flushes (writes) these caches to disk. However, if a change to the file system is made and the system crashes before the flush, the on-disk image of the files system may be corrupted.

The file system consistency check is a system utility the scans the entire file system and locates any unattached disk blocks i.e. blocks not allocated to a file and not maintained in the free block list. The utility usually reattaches these lost blocks to the free block list.

## Question 10

What is the purpose of a Journaled File System?

What problem does a JFS solve?

### Answer

A Journaled file system maintains a journal file that the OS uses to record changes to the file system’s metadata and optionally changes to data blocks. The journal file is persistently maintained on disk as another type of file system metadata. Its purpose is to record changes to the file system more efficiently (with fewer I/O operations) than updating the metadata data structures (FAT and Unallocated Blocks) each time a change is made to the file system. In the event of a system crash, the changes recorded in the journal file can be played back to recover the state of the file system just prior to the crash.

A JFS is used to reduce amount of time needed to perform a file system consistency check needed to recover the file system after a system crash.