1.

(a) What are kernel modules?

(b) What are their main advantages?

a. It is an object file which contains compiled binary code that can be loaded and/or unloaded into the kernel as per demand.
b. Advantages
   - Specific module loading/unloading allows to free up memory and other resources.
   - No Reboot and recompile of kernel.

2.

One of the disadvantages of Linux kernel modules is that they are not safe. Since kernel modules run with full kernel privileges, such as Ring 0 in x86 processors, any bug in a kernel module can compromise or crash the entire system.

Design an approach to make kernel modules safe, meaning that an untrusted or buggy kernel module should not be able to crash or compromise the core kernel. Also, discuss the functionality versus safety tradeoffs in your design, i.e. what is the cost of additional safety.

We can develop a system which are how user-space threads operate. User-space threads, if one has a bug the system handles the exception/event and it continues with the other thread working with the consecutive threads have no knowledge about the failure in another thread.

We can adopt the same method in implementing the kernel threads which will in-turn help in a more efficient system.

3.

What are the following? What do they do? Where are they located?

1. Memory Management Unit (MMU) -
2. Translation Lookaside Buffer (TLB)
3. Page tables
4. Swap device

1. is a computer hardware unit through which all the memory references are passed through. It's main job is computation of the virtual memory address to physical address, handles cache operations. Located within the computer's central processing unit.

2. It is a memory cache which is used to reduce the time taken to access a user memory location. It stores the most recent translations of virtual memory to physical memory. It's part of the MMU.
3. A page table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses. It is stored in the kernel address space.
4. It is a space on the hard disk used as a virtual extension of the RAM. It allows the system to assume that it has more storage space than it actually does.

4.

What are TLB misses and page faults? How are they handled (if at all) in systems having (i) architected page tables and (ii) architected TLBs?

TLB miss occurs when the virtual memory address -> physical memory address mapping requested by the system is not present in the TLB.
Page fault occurs when the page which is being accessed the program is not present in the physical memory. It means the page is present in the secondary memory but not yet loaded into a frame of physical memory.
Architected Page Tables will have a TLB too, but it's transparent to software which sees only the page table. This is hardware managed TLB.
Architected TLB, the capacity of the TLB is limited, the operating system will likely have an internal structure resembling a page table for each process. These are software managed TLB's. The operating system has to manipulate the TLB directly.

5.

Consider a machine with B-bit architecture (i.e. virtual address and physical address are B bits long). Size of a page is P bytes.

1. What is the size (in bytes) of the virtual address space of a process?

   2^B

2. How many bits in an address represent the byte offset into a page?

   log P

3. How many bits in an address are needed to determine the page number ?

   B – log P

4. How many page-table entries does a process' page-table contain?

   2 ^ (B- log P)

Explain your answers.

6.

Considering the typical fields of a page table entry, explain the operating system can track

1. Dirty (or updated) memory pages for the purpose of eviction?
2. Every memory write performed by a process to specific memory pages?

1. The contents of the page is saved onto a swap file. Then the kernel overwrites the frame with new page content. OS tracks dirty page table to know whether updating on the page is necessary.

2. Modified Bit – It is used to indicate whether the page has been modified or not. If the page has been modified, then the modified bit is updated and a copy of the page is stored on the hard disk for future references.

7.

In the superpage paper, explain how pre-emptible reservations, incremental promotions and speculative demotions work.

Preempting reservations – When the available physical memory is less, the system preemot frames that are reserved but not yet used. When the system requires physical space, and the desired size is not available, the system has the option of either refusing the allocation and reserving a smaller extent than desired or preempting an existing reservation that has enough unallocated frames.

There is an observation that useful reservations are populated quickly, and that reservations that have not experienced any recent allocations are less likey to be fully allocated in the future.

Incremental promotions – Whenever any superpage-sized within a reservation gets full there is an incremental promotion. Promotion takes place to the smallest superpage size as soon as the population count corresponds to that size. When the population count reaches the next larger supersize page, consecutive promotion takes place.

Speculative Demotion – The system periodically demotes active superpage speculatively to determine if the superpage is being actively used to it's fullest. The OS has no way to know to know that which pages are being accessed in a superpage. The OS demotes the superpage when there a lot of demand of memory so that unused base pages are discovered and evicted.

8.

How is a virtual address converted to a physical address, considering both segmentation and paging in (a) Multics and (b) Pentium architectures?

Multics –
Pentium architectures - Virtual address is composed of two parts. Segment selector and offset.

9.

In a file-system,

(a) What is meta-data?

(b) Where is meta-data stored?

(c) Why is it important for a file system to maintain the meta-data information?

(d) List some of the typical information that is part of the meta-data.

a) Metadata is data about the document and includes information such as the file creation date, author, last accessed date etc.
b) on-disk structure called an "inode".
c) It classifies and organizes the data. To navigate faster through data repository to find accurate data.
d) size of the file, author, data of creation of the file, description of the file.

10.

Assume that the Size of each disk block is B bytes.

Address of each disk block is A bytes long.

The top level of a UNIX i-node contains D direct block addresses, one single-indirect block address, one double-indirect block address, and one triple-indirect block address.

(a) What is the size of the largest “small” file that can be addressed through direct block addresses?
(b) What is the size of the largest file that can be supported by the UNIX inode?

Explain your answers.

a. BD bytes – D direct blocks, each disks is B bytes
b. Size of the largest file will be (BD + (B/A) * B + (B/A) * (B/A) * B + (B/A) * (B/A) * (B/A)*B) bytes

11.

For a RAID system with N disks, including data and parity, compare the level of parallelism provided by RAID 1, RAID 3, RAID 4, and RAID 5 for multiple simultaneous

(i) read I/O operations,

(ii) write I/O operations, and

(iii) combination of read and write I/O operations?

Explain your answers.

1. Read I/O operations
   - RAID 1 – Both the disk can be read in parallel hence N
   - RAID 3 – 1, Sequential read has better performance.
   - RAID 4 – N+1 disks, Good random reads.
   - RAID 5 – N+1 disks, Read operations very fast.
2. Write I/O operations
   - RAID 1 – Although there are two disks writing in parallel, they're writing the same data twice, hence N/2
   - RAID 3 – 1, Sequential write has great performance.
   - RAID 4 – 1, Slow write operations
   - RAID 5 – (N+1)/2, Write operations are slow.
3. Read and Write operations
   - RAID 1 – N disks, write performance slower compared to read.
   - RAID 3 – N+1 disks, Sequential read and write
   - RAID 4 – N+1 disks, Good random reads but slow writes
   - RAID 5 – N+1 disks, write performance is slower compared to read.

12.

What are the five I/O models? How do they handle the two stages of data reception?

Blocking I/O

User space application you may do a read/recv() system call. Control transferred to OS.

OS would check if the data is available or not.

If the data is not available, your process would be moved to blocked state.

OS internally will move it to blocked state on the event of Data Ready.

When data is ready, process has woken up, from blocked -> ready state -> running state.

After that the kernel will copy the data from the kernel to user space.

Once both the process is complete then it returns back and makes progress.

Non-blocking I/O (polling)

Read() -> OS , if data not available then OS returns backs with ‘Data not ready’ with “EWOULDBLOCK’ indicating that the user program can continue it’s execution and return after a while for the data.

This indicates that the process will remain in the running state.

When read() -> OS, data is available, OS invokes COPY_TO_USER, process is blocked (process is not making progress).

When copy to user is complete, then OS returns ok to process and it continue.

I/O Multiplexing – select()

Combination of various IP models. Single thread to be handling IO operations on multiple file descriptors.

Use a select() system call to tell the OS that process will wait for IO operations on multiple file descriptors.

When the process is blocked, it’s blocked on multiple events, blocks all multiple file descriptors.

Once the data is available on any of the descriptors, read() system call will block the process while copy_to_user is complete.

In the second phase it will block one by one on the multiple file descriptors rather than block all at once.

Signal driven I/O

No periodic checking as Non-blocking I/O.

Process will establish a SIGIO with OS, if data not ready,OS would make note of the it, process can continue with it’s execution.

When data is ready, OS signals the process about the data is ready.

When process calls the read() system call, copy_to_user function copies the data, during which the process is blocked and then returns back to the process.

Improvement on Non-blocking I/O

Asynchronous I/O

In this both the phases, OS will not block the process.

Process calls the aio_read() system call, OS keeps note of what and where the data is needed and returns the control back to the process.

Process can continue and the OS would internally issue the IO operation, issue the data, copy_to_user into the user space while all along the process is not blocked.

When the copy is complete, OS will inform the process via a signal that the data has been copied.