UNIX Memory Management Interview Questions and Answers

1. What is the difference between Swapping and Paging?

**Swapping:** Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems.

**Paging:** Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory. It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

2. What is major difference between the Historic Unix and the new BSD release of Unix System V in terms of Memory Management?

Historic Unix uses Swapping - entire process is transferred to the main memory from the swap device, whereas the Unix System V uses Demand Paging - only the part of the process is moved to the main memory. Historic Unix uses one Swap Device and Unix System V allow multiple Swap Devices.

3. What is the main goal of the Memory Management?

1. It decides which process should reside in the main memory,
2. Manages the parts of the virtual address space of a process which is non-core resident,
3. Monitors the available main memory and periodically write the processes into the swap device to provide more processes fit in the main memory simultaneously.

4. What is a Map?

A Map is an Array, which contains the addresses of the free space in the swap device that are allocatable resources, and the number of the resource units available there.

| Address Units | 1 | 10,000 |

This allows First-Fit allocation of contiguous blocks of a resource. Initially the Map contains one entry - address (block offset from the starting of the swap area) and the total number of resources.

Kernel treats each unit of Map as a group of disk blocks. On the allocation and freeing of the resources Kernel updates the Map for accurate information.

5. What scheme does the Kernel in Unix System V follow while choosing a swap device among the multiple swap devices?

Kernel follows Round Robin scheme choosing a swap device among the multiple swap devices in Unix System V.

6. What is a Region?

A Region is a continuous area of a process's address space (such as text, data and stack). The kernel in a "Region Table" that is local to the process maintains region. Regions are sharable among the process.
7. What are the events done by the Kernel after a process is being swapped out from the main memory?

When Kernel swaps the process out of the primary memory, it performs the following:

1. Kernel decrements the Reference Count of each region of the process. If the reference count becomes zero, swaps the region out of the main memory,
2. Kernel allocates the space for the swapping process in the swap device,
3. Kernel locks the other swapping process while the current swapping operation is going on,
4. The Kernel saves the swap address of the region in the region table.

8. Is the Process before and after the swap are the same? Give reason.

Process before swapping is residing in the primary memory in its original form. The regions (text, data and stack) may not be occupied fully by the process, there may be few empty slots in any of the regions and while swapping Kernel do not bother about the empty slots while swapping the process out.

After swapping the process resides in the swap (secondary memory) device. The regions swapped out will be present but only the occupied region slots but not the empty slots that were present before assigning.

While swapping the process once again into the main memory, the Kernel referring to the Process Memory Map, it assigns the main memory accordingly taking care of the empty slots in the regions.

9. What do you mean by u-area (user area) or u-block?

This contains the private data that is manipulated only by the Kernel. This is local to the Process, i.e. each process is allocated a u-area.

10. What are the entities that are swapped out of the main memory while swapping the process out of the main memory?

All memory space occupied by the process, process's u-area, and Kernel stack are swapped out, theoretically.

Practically, if the process's u-area contains the Address Translation Tables for the process then Kernel implementations do not swap the u-area.

11. What is Fork swap?

"fork()" is a system call to create a child process. When the parent process calls "fork()" system call, the child process is created and if there is short of memory then the child process is sent to the read-to-run state in the swap device, and return to the user state without swapping the parent process. When the memory will be available the child process will be swapped into the main memory.

12. What is Expansion swap?

At the time when any process requires more memory than it is currently allocated, the Kernel performs Expansion swap. To do this Kernel reserves enough space in the swap device. Then the address translation mapping is adjusted for the new virtual address space but the physical memory is not allocated. At last Kernel swaps the process into the assigned space in the swap device. Later when the Kernel swaps the process into the main memory this assigns memory according to the new address translation mapping.

13. How the Swapper works?
The swapper is the only process that swaps the processes. The Swapper operates only in the Kernel mode and it does not uses System calls instead it uses internal Kernel functions for swapping. It is the archetype of all kernel processes.

14. What are the processes that are not bothered by the swapper? Give Reason.

1. Zombie process: They do not take any up physical memory.
2. Processes locked in memories that are updating the region of the process.
3. Kernel swaps only the sleeping processes rather than the 'ready-to-run' processes, as they have the higher probability of being scheduled than the Sleeping processes.

15. What are the requirements for a swapper to work?

The swapper works on the highest scheduling priority. Firstly it will look for any sleeping process, if not found then it will look for the ready-to-run process for swapping. But the major requirement for the swapper to work the ready-to-run process must be core-resident for at least 2 seconds before swapping out. And for swapping in the process must have been resided in the swap device for at least 2 seconds. If the requirement is not satisfied then the swapper will go into the wait state on that event and it is awaken once in a second by the Kernel.

16. What are the criteria for choosing a process for swapping into memory from the swap device?

The resident time of the processes in the swap device, the priority of the processes and the amount of time the processes had been swapped out.

17. What are the criteria for choosing a process for swapping out of the memory to the swap device?

1. The process's memory resident time,
2. Priority of the process and
3. The nice value.

18. What do you mean by nice value?

Nice value is the value that controls {increments or decrements} the priority of the process. This value that is returned by the nice() system call. The equation for using nice value is:

\[ \text{Priority} = \left( \frac{\text{recent CPU usage}}{\text{constant}} \right) + (\text{base- priority}) + (\text{nice value}) \]

Only the administrator can supply the nice value. The nice() system call works for the running process only. Nice value of one process cannot affect the nice value of the other process.

19. What are conditions on which deadlock can occur while swapping the processes?

1. All processes in the main memory are asleep.
2. All "ready-to-run" processes are swapped out.
3. There is no space in the swap device for the new incoming process that are swapped out of the main memory.
4. There is no space in the main memory for the new incoming process.

20. What are conditions for a machine to support Demand Paging?

1. Memory architecture must based on Pages,
2. The machine must support the 'restartable' instructions.

21. What is "the principle of locality"?
It's the nature of the processes that they refer only to the small subset of the total data space of the process. i.e. the process frequently calls the same subroutines or executes the loop instructions.

22. What is the working set of a process?

The set of pages that are referred by the process in the last "n", references, where "n" is called the window of the working set of the process.

23. What is the window of the working set of a process?

The window of the working set of a process is the total number in which the process had referred the set of pages in the working set of the process.

24. What is called a page fault?

Page fault is referred to the situation when the process addresses a page in the working set of the process but the process fails to locate the page in the working set. And on a page fault the kernel updates the working set by reading the page from the secondary device.

25. What are data structures that are used for Demand Paging?

Kernel contains 4 data structures for Demand paging. They are,

1. Page table entries,
2. Disk block descriptors,
3. Page frame data table (pfdata),
4. Swap-use table.

26. What are the bits that support the demand paging?

Valid, Reference, Modify, Copy on write, Age. These bits are the part of the page table entry, which includes physical address of the page and protection bits.

27. How the Kernel handles the fork() system call in traditional Unix and in the System V Unix, while swapping?

Kernel in traditional Unix, makes the duplicate copy of the parent's address space and attaches it to the child's process, while swapping. Kernel in System V Unix, manipulates the region tables, page table, and pfdata table entries, by incrementing the reference count of the region table of shared regions.

28. Difference between the fork() and vfork() system call?

During the fork() system call the Kernel makes a copy of the parent process's address space and attaches it to the child process.

But the vfork() system call do not makes any copy of the parent's address space, so it is faster than the fork() system call. The child process as a result of the vfork() system call executes exec() system call. The child process from vfork() system call executes in the parent's address space (this can overwrite the parent's data and stack ) which suspends the parent process until the child process exits.

29. What is BSS(Block Started by Symbol)?

A data representation at the machine level, that has initial values when a program starts and tells about how much space the kernel allocates for the un-initialized data. Kernel initializes it to zero at run-time.
30. What is Page-Stealer process?

This is the Kernel process that makes rooms for the incoming pages, by swapping the memory pages that are not the part of the working set of a process. Page-Stealer is created by the Kernel at the system initialization and invokes it throughout the lifetime of the system. Kernel locks a region when a process faults on a page in the region, so that page stealer cannot steal the page, which is being faulted in.

31. Name two paging states for a page in memory?

The two paging states are:

1. The page is aging and is not yet eligible for swapping,
2. The page is eligible for swapping but not yet eligible for reassignment to other virtual address space.

32. What are the phases of swapping a page from the memory?

1. Page stealer finds the page eligible for swapping and places the page number in the list of pages to be swapped.
2. Kernel copies the page to a swap device when necessary and clears the valid bit in the page table entry, decrements the pfdata reference count, and places the pfdata table entry at the end of the free list if its reference count is 0.

33. What is page fault? Its types?

Page fault refers to the situation of not having a page in the main memory when any process references it. There are two types of page fault:

1. Validity fault,
2. Protection fault.

34. In what way the Fault Handlers and the Interrupt handlers are different?

Fault handlers are also an interrupt handler with an exception that the interrupt handlers cannot sleep. Fault handlers sleep in the context of the process that caused the memory fault. The fault refers to the running process and no arbitrary processes are put to sleep.

35. What is validity fault?

If a process referring a page in the main memory whose valid bit is not set, it results in validity fault. The valid bit is not set for those pages:

1. that are outside the virtual address space of a process,
2. that are the part of the virtual address space of the process but no physical address is assigned to it.

36. What does the swapping system do if it identifies the illegal page for swapping?

If the disk block descriptor does not contain any record of the faulted page, then this causes the attempted memory reference is invalid and the kernel sends a "Segmentation violation" signal to the offending process. This happens when the swapping system identifies any invalid memory reference.

37. What are states that the page can be in, after causing a page fault?

1. On a swap device and not in memory,
2. On the free page list in the main memory,
3. In an executable file,
4. Marked "demand zero",
5. Marked "demand fill"

38. In what way the validity fault handler concludes?

1. It sets the valid bit of the page by clearing the modify bit.
2. It recalculates the process priority.

39. At what mode the fault handler executes?

At the Kernel Mode.

40. What do you mean by the protection fault?

Protection fault refers to the process accessing the pages, which do not have the access permission. A process also incur the protection fault when it attempts to write a page whose copy on write bit was set during the fork() system call.

41. How the Kernel handles the copy on write bit of a page, when the bit is set?

In situations like, where the copy on write bit of a page is set and that page is shared by more than one process, the Kernel allocates new page and copies the content to the new page and the other processes retain their references to the old page. After copying the Kernel updates the page table entry with the new page number. Then Kernel decrements the reference count of the old pfdata table entry.

In cases like, where the copy on write bit is set and no processes are sharing the page, the Kernel allows the physical page to be reused by the processes. By doing so, it clears the copy on write bit and disassociates the page from its disk copy (if one exists), because other process may share the disk copy. Then it removes the pfdata table entry from the page-queue as the new copy of the virtual page is not on the swap device. It decrements the swap-use count for the page and if count drops to 0, frees the swap space.

42. For which kind of fault the page is checked first?

The page is first checked for the validity fault, as soon as it is found that the page is invalid (valid bit is clear), the validity fault handler returns immediately, and the process incur the validity page fault. Kernel handles the validity fault and the process will incur the protection fault if any one is present.

43. In what way the protection fault handler concludes?

After finishing the execution of the fault handler, it sets the modify and protection bits and clears the copy on write bit. It recalculates the process-priority and checks for signals.

44. How the Kernel handles both the page stealer and the fault handler?

The page stealer and the fault handler thrash because of the shortage of the memory. If the sum of the working sets of all processes is greater that the physical memory then the fault handler will usually sleep because it cannot allocate pages for a process. This results in the reduction of the system throughput because Kernel spends too much time in overhead, rearranging the memory in the frantic pace.