Linux Device Model (LDM)  
  
Explain about the Linux Device Model (LDM)?  
Explain about about ksets, kobjects and ktypes. How are they related?  
Questions about sysfs.  
  
  
                                       Linux Boot Sequence  
Explain about the Linux boot sequence in case of ARM architecture?  
How are the command line arguments passed to Linux kernel by the u-boot (bootloader)?  
Explain about ATAGS?  
Explain about command line arguments that are passed to linux kernel and how/where they are parsed in kernel code?  
Explain about device tree.  
  
                                      Interrupts in Linux  
Explain about the interrupt mechanims in linux?  
What are the APIs that are used to register an interrupt handler?  
How do you register an interrupt handler on a shared IRQ line?  
Explain about the flags that are passed to request\_irq().  
Explain about the internals of Interrupt handling in case of Linux running on ARM.  
What are the precautions to be taken while writing an interrupt handler?  
Explain interrupt sequence in detail starting from ARM to registered interrupt handler.  
What is bottom half and top half.  
What is request\_threaded\_irq()  
If same interrupts occurs in two cpu how are they handled?  
How to synchronize data between 'two interrupts' and 'interrupts and process'.  
How are nested interrupts handled?  
How is task context saved during interrupt.  
  
                                    Bottom-half Mechanisms in Linux  
What are the different bottom-half mechanisms in Linux?  
Softirq, Tasklet and Workqueues  
What are the differences between Softirq/Tasklet and Workqueue? Given an example what you prefer to use?  
What are the differences between softirqs and tasklets?  
Softirq is guaranteed to run on the CPU it was scheduled on, where as tasklets don’t have that guarantee.   
The same tasklet can't run on two separate CPUs at the same time, where as a softirq can.   
When are these bottom halfs executed?  
Explain about the internal implementation of softirqs?  
[Bottom-halves in Linux - Part 1: Softirqs](http://linuxblore.blogspot.com/2013/02/bottom-halves-in-linux-part-1-softirqs.html)  
Explain about the internal implementation of tasklets?  
[Bottom-halves in Linux - Part 2: Tasklets](http://linuxblore.blogspot.com/2013/02/bottom-halves-in-linux-part-2-tasklets.html)  
Explain about the internal implementation of workqueues?  
[Bottom-halves in Linux - Part 3: Workqueues](http://linuxblore.blogspot.in/2013/01/workqueues-in-linux.html)  
Explain about the concurrent work queues.  
  
                                       Linux Memory Management   
What are the differences between vmalloc and kmalloc? Which is preferred to use in device drivers?  
What are the differences between slab allocator and slub allocator?  
What is boot memory allocator?  
How do you reserve block of memory?  
What is virtual memory and what are the advanatages of using virtual memory?  
What's paging and swapping?  
Is it better to enable swapping in embedded systems? and why?  
What is the page size in Linux kernel in case of 32-bit ARM architecture?  
What is page frame?  
What are the different memory zones and why does different zones exist?  
What is high memory and when is it needed?  
Why is high memory zone not needed in case of 64-bit machine?  
How to allocate a page frame from high memory?  
In ARM, an abort exception if generated, if the page table doesn't contain a virtual to physical map for a particular page. How exactly does the MMU know that a virtual to physical map is present in the pagetable or not?  
A Level-1 page table entry can be one of four possible types. The 1st type is given below:   
A fault entry that generates an abort exception. This can be either a prefetch or data abort, depending on the type of access. This effectively indicates virtual addresses that are unmapped.  
In this case the bit [0] and [1] are set to 0. This is how the MMU identifies that it's a fault entry.  
Same is the case with Level-2 page table entry.  
Does the Translation Table Base Address (TTBR) register, Level 1 page table and Level 2 page table contain Physical addresses or Virtual addresses?  
TTBR: Contain physical address of the pgd base  
Level 1 page table (pgd): Physical address pointing to the pte base  
Level 2 page table (pte): Physical address pointing to the physical page frame  
Since page tables are in kernel space and kernel virtual memory is mapped directly to RAM. Using just an easy macro like \_\_virt\_to\_phys(), we can get the physical address for the pgd base or pte base or pte entry.  
  
  
 Kernel Synchronization  
Why do we need synchronization mechanisms in Linux kernel?  
What are the different synchonization mechanisms present in Linux kernel?  
What are the differences between spinlock and mutex?  
What is lockdep?  
Which synchronization mechanism is safe to use in interrupt context and why?  
Explain about the implementation of spinlock in case of ARM architecture.  
Explain about the implementation of mutex in case of ARM architecture.  
Explain about the notifier chains.  
Explain about RCU locks and when are they used?  
Explain about RW spinlocks locks and when are they used?  
Which are the synchronization technoques you use 'between processes', 'between processe and interrupt' and 'between interrupts'; why and how ?  
What are the differences between semaphores and spinlocks?  
  
                            Process Management and Process Scheduling  
What are the different schedulers class present in the linux kernel?  
How to create a new process?  
What is the difference between fork( ) and vfork( )?  
Which is the first task what is spawned in linux kernel?  
What are the processes with PID 0 and PID 1?  
PID 0 - idle task  
PID 1 - init   
How to extract task\_struct of a particular process if the stack pointer is given?  
How does scheduler picks particular task?  
When does scheduler picks a task?  
How is timeout managed?  
How does load balancing happens?  
Explain about any scheduler class?  
Explain about wait queues and how they implemented? Where and how are they used?  
What is process kernel stack and process user stack? What is the size of each and how are they allocated?  
Why do we need seperate kernel stack for each process?  
What all happens during context switch?  
What is thread\_info? Why is it stored at the end of kernel stack?  
What is the use of preempt\_count variable?  
What is the difference between interruptible and uninterruptible task states?  
How processes and threads are created? (from user level till kernel level)  
How is virtual run time (vruntime) calculated?  
  
                                      Timers and Time Management  
What are jiffies and HZ?  
What is the initial value of jiffies when the system has started?  
Explain about HR timers and normal timers?  
On what hardware timers, does the HR timers are based on?  
How to declare that a specific hardware timers is used for kernel periodic timer interrupt used by the scheduler?  
How software timers are implemented?  
  
                                       Power Management in Linux  
Explain about cpuidle framework.  
Explain about cpufreq framework.  
Explain about clock framework.  
Explain about regulator framework.  
Explain about suspened and resume framwork.  
Explain about early suspend and late resume.  
Explain about wakelocks.  
  
                                          Linux Kernel Modules  
How to make a module as loadable module?  
How to make a module as in-built module?  
Explain about Kconfig build system?  
Explain about the init call mechanism.  
What is the difference between early init and late init?  
Early init:  
Early init functions are called when only the boot processor is online.  
Run before initializing SMP.  
Only for built-in code, not modules.  
Late init:  
Late init functions are called \_after\_ all the CPUs are online.  
  
                                         Linux Kernel Debugging  
  
What is Oops and kernel panic?  
Does all Oops result in kernel panic?  
What are the tools that you have used for debugging the Linux kernel?  
What are the log levels in printk?  
Can printk's be used in interrupt context?  
How to print a stack trace from a particular function?  
What's the use of early\_printk( )?  
Explan about the various gdb commands.  
  
                                                  Miscellaneous  
  
How are the atomic functions implemented in case of ARM architecture?  
How is container\_of( ) macro implemented?   
Explain about system call flow in case of ARM Linux.  
What 's the use of \_\_init and \_\_exit macros?  
How to ensure that init function of a partiuclar driver was called before our driver's init function is called (assume that both these drivers are built into the kenrel image)?  
What's a segementation fault and what are the scenarios in which segmentation fault is triggered?  
If the scenarios which triggers the segmentation fault has occurred, how the kernel identifies it and what are the actions that the kernel takes?   
  
Courtesy:  
[Linux kernel and Linux Device Driver Questions](http://www.dummymind.com/linux-device-driver-interview-questions.html)  
<http://www.dummymind.com/linux-device-driver-interview-questions.html>

would expect some of the below topics to appear in a Linux kernel interview:

* user/kernel interface (system calls, procfs/sysfs, ioctl)
* process vs interrupt context
* deferred actions: softirq, tasklets, workqueues, timers
* synchronization between contexts (how would you synchronize access to a shared memory area used from an interrupt handler and a workqueue on a SMP preemptive kernel).
* memory allocation (kmalloc vs vmalloc)
* debugging - tools and strategies for finding bugs.
* are you familiar with the Linux kernel development process? Do you have any patch accepted in the mainline?
* 1. What is the difference in features between kernel 2.2, 2.4 and 2.6 ?
* 2. What are Static and Shared libraries ?
* 3. What is dynamic linking ? What is static linking ?
* 4. What are the advantages of Dynamic linking or Shared libraries ?
* 5. Does gcc search for both static and shared libraries ? Which is searched initially by gcc compiler ?
* 6. What should be done for Shared library based linking in gcc ?
* 7. What should be done for static library based linking in gcc ?
* 8. What is object file and what are symbols ?
* 9. Can you tell the memory layout based on Data,BSS,HEAP and STACK ?
* 10. What are the ways in which linux kernel can be compiled ?
* 11. How will get the driver added into the kernel ? What are Kconfig files ?
* 12. What is a kernel module ?
* 13. What is the difference between insmod and modprobe ?
* 14. How will you list the modules ?
* 15. How do you get the list of currently available drivers ?
* 16. How will you Access userspace memory from kernel ? What are the various methods ?
* 17. What is the use of ioctl(inode,file,cmd,arg) ApI ?
* 18. What is the use of the poll(file, polltable) API ?
* 19. What is the use of file->private\_data in a device driver structure ?
* 20. What is a device number ?
* 21. What are the two types of devices drivers from VFS point of view ?
* 22. What are character devices ?
* 23. How does the character device driver adds and remove itself from the kernel ?
* 24. What is the role of interrupts in a device driver ? How are interrupts handled in device driver ?
* 25. How will you make interrupt handlers as fast as possible ?
* 26. What are the types of softirqs ?
* 27. Difference between Timer Softirq and Tasklet Softirq ?
* 28. What are tasklets ? How are they activated ? when and How are they initialized ?
* 29. What is task\_struct and how are task states maintained ?
* 30. What is rwlock and spinlock ? Briefly explain about both of them ?
* 31. When will you use rwlock instead of spinlock ?
* 32. Can spinlock/rwlock be used in Interrupt handler ?
* 33. Tell about the Memory Layout of a Process in Linux .
* 34. How will you trace the system calls made into the kernel of lInux ?
* 35. What is mmap ? MMAP & malloc ? MMAP & brk ? MMAP adv & dis-adv.
* 36. Tell the relation between Malloc and MMAP
* 37. Advantages of MMAP over Read ?
* 38. Tell the role of brk() in malloc / Tell the relation between heap and brk?
* 39. Example of using MMAP and MUNMAP in C ?
* 40. Tell about the method/steps in Linux Kernel Compilation.
* 41. What is Kmalloc and how does it differ from normal malloc ? or Why can’t we use malloc in kernel code ?
* 42. What happens as soon as a packet arrives from the network in Linux ?
* 43. What is a stack frame, stack pointer & frame pointer ?
* 44. What is a profiler ? Which one have you used ?
* 45. How do you determine the direction of stack growth ?

Below are some of the interview questions related to V4l2.

1. What is V4L2?
2. Why V4L2 Was introduced?
3. how to create a video node using v4l2?
4. What are the core functionalities supported by video nodes?
5. How you can negotiate the data format using v4l2?
6. How you can negotiate Input/Output Method ?
7. What are the sequence of IOCTLs you need to initiate before starting camera sensor capturing?
8. How memory is allocated and mapped using V4L2?
9. How v4l2 Sub devices are created ?
10. Which IOCTL is used to query the capabilities of devices?
11. **Process management:-**
12. 1) [how to manipulate the current process.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
13. 2) [what are kernel thread.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
14. 3) [how threads are implemented in linux kernel.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
15. 4) [What are different state of a process in lunix.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
16. 5) [what is difference between process and thread.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
17. [6) generally what resources are shared between threads.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
18. [7) what is process descriptor](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
19. [8) what is task\_struct.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
20. [9) what is therad\_info structure for.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
21. [10) what was the need of thread\_info structure.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
22. [11) difference betwen fork() and vfork()](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
23. [12) what is process context.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
24. [13) what is zombie process.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
25. [14) how parent less process is handles in linux.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)

28. **Process Scheduling:-**
29. [1) what is process scheduling](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
30. [2) what is cooperative multitasking and pre-emptive multitasking.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
31. [3) what is yielding.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
32. [4) what is limitation of cooperative multitasking.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
33. [5) I/O bound versus Processor bound process.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
34. [6) what is process priority.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
35. [7) What kind of priority is maintained in linux.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
36. [8) what is nice value.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
37. [9) what is virtual run time.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
38. [10) what are the available scheduling classes in linux.](http://www.tutorialsdaddy.com/linux-kernel-driver-interview-preperation/)
39. 11) [which type os scheduling used in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
40. [12)  how next task is picked for scheduling.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
41. [13) what is scheduler entry point in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
42. [14) what is waitqueus.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
43. [15) How context switching is handled in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
44. [16) what is user preemption and kernel preemption](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
46. **Syscalls:-**
47. [1) what is syscalls.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
48. [2) how system calls are implemented in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
49. [3) what happens when process in userspace calls a syscall.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
50. [4) what is the need of verifying parameter in definition of syscall.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
51. [5) what is system calls context.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
52. [6) why it is not recommended to writing new syscall.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)

55. **Interrupts and interrupt handlers:-**
56. [1) what is interrupt](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
57. [2) what is interrupt handler or ISR.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
58. [3) what is top halves and bottom halves.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
59. [4) How interrupt is registered.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
60. [5) what are different interrupt handler flags.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
61. [6) How interrupt are freed.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
62. [7) what are the considerations needs to taken care while writing interrupt handler.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
63. [8) what are shared handlers.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
64. [9) what is interrupt context.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
65. [10) how to disable and enable interrupts.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
66. [11) what are different bottom halves techniques in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
67. [12) what is tasklets , softirq and workqueus and difference among them.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
68. [13) when to choose which bottom halves.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
69. [14) how to implements softirq , tasklets and workwues.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
70. [15) how to schedule tasklet.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
71. [16) what is ksoftirq.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
72. [17) How to disable bottom halves.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
73. [18) How locking between bottomhalves handled.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
74. [19) Why we need preemption.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
76. **Kernel Synchronization:-**
77. [1) what is synchronization](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
78. [2) what is critical section](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
79. [3) what is race condition.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
80. [4) why we need to take care of synchronization](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
81. [5) what is various synchronization techniques in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
82. [6) what is deadlocks.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
83. [7) what is atomic operations.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
84. [8) what is spin locks.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
85. [9) what is reader-writer spin lock.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
86. [10) what is semaphore.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
87. [11) what is binary semaphore.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
88. [12) what is difference between semaphore and spin lock.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
89. [13) when to choose what among spin lock and semaphore.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
90. [14) what is difference between semaphore and mutex.](http://www.tutorialsdaddy.com/courses/intro-to-c/)
91. [15) what is preemption disabling and what is the use of this.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
93. **Kernel timers**
94. [1) what is tick rate and jiffies.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
95. [2) what are the various way of applying delay in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
96. [3) what is blocking and non blocking call](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
97. [4) what is Real time clock (RTC).](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
98. [5) how busy looping is implemented in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
100. **Memory management**
101. [1) how memory is managed in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
102. [2) what are pages.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
103. [3) what are different memory zones in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
104. [4) how to allocated pages.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
105. [5) how to freeing page.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
106. [6) what us kmalloc and what are action modifier we can pass while using kmalloc.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
107. [7) what is zone modifier in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
108. [8) what is vmalloc](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)

111. **Virtual file system:-**
112. [1) what is virtual file system and what is the need of it in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
113. [2) are are different object types in VFS.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
114. [3) what are the operations possible on inode and superblock objects.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
116. **Process Address Space:-**
117. [1) what is process address space.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
118. [2) what is memory descriptor in linux and which structure represents it.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
119. [3) how to allocate and destroy a memory descriptor](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
120. [4) does kernel thread  has any association with mm\_struct.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
121. [5) what is VMA and what are various VMA operations possible in linux kernel.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
122. [6) how to manipulate memory area in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
123. [7) what mmap and do\_mmap().](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
124. [8) what is page tables.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
126. **Device driver  questions:-**
128. [1) What is device driver and what is the need of it.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
129. [2) what are different kind of devices.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
130. [3) what is module in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
131. [4) how mudules are loaded in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
132. [5) difference between insmod and modprobe.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
133. [6) how parameters are shared between driver modules.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
134. [7) what are IOCTLS.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
135. [8) what is syscalls.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
136. [9) what are the benefits of syscalls.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
137. [10) how character driver is registered in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
138. [11) what is init and exit function of a driver.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
139. [12) how and when init and exit function of driver get called.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
140. [13) what is probe function.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
141. [14) when probe is get called.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
142. [15) what is platform devices.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
143. [16) what is device tree.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
144. [17) what are the benefits of device tree over board files.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
145. [18) what is sysfs and procfs.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
146. [19) how logs are printed in linux kernel and what are the logs level available in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
147. [20) what is copy\_to\_user and copy\_from\_user.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
148. [21) what do you mean by kernel configuration and what are the various way of configuring kernel.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
149. [22) what is menuconfig.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
150. [23) what is ioremap](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
151. [24) what is seagmentation fault.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
152. [25) what are the various ways od debugging linux kernel.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
153. [26) how linux kernel boots.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
154. [27) what is zimage and bzimage.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
155. [28)  what are different booting arguments in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
156. [29) how parameter are passed from boot loader to kernel.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
157. [30) what is ATAGS.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
158. [31) from which file kernel execution starts.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
159. [32) what is bootloader.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
160. [33) what is primary and secondary bootloader.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
161. [34) why we need two bootloader.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
162. [35) difference between poll and select.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
163. [36) what is priority inheritance and priority inversion.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
164. [37) what are different type of kernel.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
165. [38) what is DMA.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
166. [39) what is cache coherency.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
167. [40) what is copy on write.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
168. [41) what is highmem and lowmem.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
169. [42) what happens if we pass invalid address from userspace by using ioctls.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
170. [43) what are different ipc mechanism in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
171. [44) what are sockets.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
172. [45) how page fault is handled in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
173. [46) difference between memory based io and port based io.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
174. [47) what is I2c and SPI.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
175. [48) how physical to virtual translations works in linux.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
176. [49) what is thrashing, segmentation and fragmentation.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)
177. [50) what is preempt\_count and what is the need of that.](http://www.tutorialsdaddy.com/courses/linux-kernel-drivers-interview-questions-and-answers/)

**Linux Device Model (LDM)**

Explain about the Linux Device Model (LDM)?

Explain about about ksets, kobjects and ktypes. How are they related?  
  
Questions about sysfs.

**Linux Boot Sequence**

Explain about the Linux boot sequence in case of ARM architecture?

How are the command line arguments passed to Linux kernel by the u-boot (bootloader)?

Explain about ATAGS?

Explain about command line arguments that are passed to linux kernel and how/where they are   
parsed in kernel code?

Explain about device tree.

**Interrupts in Linux**

Explain about the interrupt mechanims in linux?

What are the APIs that are used to register an interrupt handler?

How do you register an interrupt handler on a shared IRQ line?

Explain about the flags that are passed to request\_irq().

Explain about the internals of Interrupt handling in case of Linux running on ARM.

[Solution](http://venkateshabbarapu.blogspot.in/2012/09/interrupt-handling-in-arm.html)

What are the precautions to be taken while writing an interrupt handler?  
  
Explain interrupt sequence in detail starting from ARM to registered interrupt handler.  
  
What is bottom half and top half.  
  
What is request\_threaded\_irq()  
  
If same interrupts occurs in two cpu how are they handled?  
  
How to synchronize data between 'two interrupts' and 'interrupts and process'.  
  
How are nested interrupts handled?  
  
How is task context saved during interrupt.

**Bottom-half Mechanisms in Linux**

What are the different bottom-half mechanisms in Linux?  
Softirq, Tasklet and Workqueues  
  
What are the differences between Softirq/Tasklet and Workqueue? Given an example what you   
prefer to use?  
  
What are the differences between softirqs and tasklets?

* Softirq is guaranteed to run on the CPU it was scheduled on, where as tasklets don’t have that guarantee.
* The same tasklet can't run on two separate CPUs at the same time, where as a softirq can.

When are these bottom halfs executed?

Explain about the internal implementation of softirqs?  
<http://linuxblore.blogspot.com/2013/02/bottom-halves-in-linux-part-1-softirqs.html>  
  
Explain about the internal implementation of tasklets?  
<http://linuxblore.blogspot.com/2013/02/bottom-halves-in-linux-part-2-tasklets.html>  
  
Explain about the internal implementation of workqueues?  
<http://linuxblore.blogspot.in/2013/01/workqueues-in-linux.html>  
  
Explain about the concurrent work queues.

**Linux Memory Management**

What are the differences between vmalloc and kmalloc? Which is preferred to use in device drivers?  
  
What are the differences between slab allocator and slub allocator?  
  
What is boot memory allocator?  
  
How do you reserve block of memory?

What is virtual memory and what are the advanatages of using virtual memory?

What's paging and swapping?

Is it better to enable swapping in embedded systems? and why?

What is the page size in Linux kernel in case of 32-bit ARM architecture?  
  
What is page frame?  
  
What are the different memory zones and why does different zones exist?  
  
What is high memory and when is it needed?  
  
Why is high memory zone not needed in case of 64-bit machine?  
  
How to allocate a page frame from high memory?  
  
In ARM, an abort exception if generated, if the page table doesn't contain a virtual to physical map for a particular page. How exactly does the MMU know that a virtual to physical map is present in the pagetable or not?

A Level-1 page table entry can be one of four possible types. The 1st type is given below:

* A fault entry that generates an abort exception. This can be either a prefetch or data abort, depending on the type of access. This effectively indicates virtual addresses that are unmapped.

In this case the bit [0] and [1] are set to 0. This is how the MMU identifies that it's a fault entry.   
  
Same is the case with Level-2 page table entry.

Does the Translation Table Base Address (TTBR) register, Level 1 page table and Level 2 page table contain Physical addresses or Virtual addresses?

TTBR: Contain physical address of the pgd base  
Level 1 page table (pgd): Physical address pointing to the pte base  
Level 2 page table (pte): Physical address pointing to the physical page frame  
  
Since page tables are in kernel space and kernel virtual memory is mapped directly to RAM. Using just an easy macro like \_\_virt\_to\_phys(), we can get the physical address for the pgd base or pte base or pte entry.

**Kernel Synchronization**

Why do we need synchronization mechanisms in Linux kernel?

What are the different synchonization mechanisms present in Linux kernel?

What are the differences between spinlock and mutex?

What is lockdep?

Which synchronization mechanism is safe to use in interrupt context and why?

Explain about the implementation of spinlock in case of ARM architecture.

[Solution](http://linuxkernelarticles.blogspot.in/2013/02/spinlock-implementation-in-arm.html)

Explain about the implementation of mutex in case of ARM architecture.

[Solution](http://linuxkernelarticles.blogspot.in/2013/02/mutex-implementation-in-arm-architecture.html)  
  
Explain about the notifier chains.  
[Solution](http://www.linuxforu.com/2009/01/the-crux-of-linux-notifier-chains/)

Explain about RCU locks and when are they used?

Explain about RW spinlocks locks and when are they used?  
  
Which are the synchronization technoques you use 'between processes', 'between processe and interrupt' and 'between interrupts'; why and how ?  
  
What are the differences between semaphores and spinlocks?

**Process Management and Process Scheduling**

What are the different schedulers class present in the linux kernel?

How to create a new process?  
  
What is the difference between fork( ) and vfork( )?  
  
Which is the first task what is spawned in linux kernel?  
  
What are the processes with PID 0 and PID 1?  
PID 0 - idle task  
PID 1 - init

How to extract task\_struct of a particular process if the stack pointer is given?  
  
How does scheduler picks particular task?  
  
When does scheduler picks a task?  
  
How is timeout managed?  
  
How does load balancing happens?  
  
Explain about any scheduler class?  
  
Explain about wait queues and how they implemented? Where and how are they used?  
  
What is process kernel stack and process user stack? What is the size of each and how are they allocated?  
  
Why do we need seperate kernel stack for each process?  
  
What all happens during context switch?  
  
What is thread\_info? Why is it stored at the end of kernel stack?  
  
What is the use of preempt\_count variable?  
  
What is the difference between interruptible and uninterruptible task states?  
  
How processes and threads are created? (from user level till kernel level)  
  
How is virtual run time (vruntime) calculated?  
[Solution](http://oakbytes.wordpress.com/2012/07/03/linux-scheduler-cfs-and-virtual-run-time/)

**Timers and Time Management**

What are jiffies and HZ?

What is the initial value of jiffies when the system has started?  
  
Explain about HR timers and normal timers?  
  
On what hardware timers, does the HR timers are based on?  
  
How to declare that a specific hardware timers is used for kernel periodic timer interrupt used by the scheduler?  
  
How software timers are implemented?

**Power Management in Linux**

Explain about cpuidle framework.  
  
Explain about cpufreq framework.  
  
Explain about clock framework.  
  
Explain about regulator framework.  
  
Explain about suspened and resume framwork.  
  
Explain about early suspend and late resume.

Explain about wakelocks.

**Linux Kernel Modules**

How to make a module as loadable module?

How to make a module as in-built module?  
  
Explain about Kconfig build system?  
  
Explain about the init call mechanism.  
[Solution](http://linuxgazette.net/157/amurray.html)  
  
What is the difference between early init and late init?

Early init:

* Early init functions are called when only the boot processor is online.
* Run before initializing SMP.
* Only for built-in code, not modules.

Late init:

* Late init functions are called \_after\_ all the CPUs are online.

**Linux Kernel Debugging**

What is Oops and kernel panic?

Does all Oops result in kernel panic?

What are the tools that you have used for debugging the Linux kernel?

What are the log levels in printk?

Can printk's be used in interrupt context?

How to print a stack trace from a particular function?

What's the use of early\_printk( )?

Explan about the various gdb commands.

**Miscellaneous**

How are the atomic functions implemented in case of ARM architecture?  
[Solution](http://linuxkernelarticles.blogspot.in/2013/02/atomic-function-implementation-in-arm.html)  
  
How is container\_of( ) macro implemented?   
  
Explain about system call flow in case of ARM Linux.

What 's the use of \_\_init and \_\_exit macros?

How to ensure that init function of a partiuclar driver was called before our driver's init function is called (assume that both these drivers are built into the kenrel image)?

What's a segementation fault and what are the scenarios in which segmentation fault is triggered?

If the scenarios which triggers the segmentation fault has occured, how the kernel identifies it and what are the actions that the kernel takes?

**Questions On Storage Class Specifier**

#### 1. What are storage class specifier?

**Ans**: Please refer this link : <http://cinterviewquestionandanswer.blogspot.in/2014/01/storage-class-specifier.html>

#### 2. What is static variable?

**Ans** :  
  
There are 3 main uses for the static.  
  
1. If you declare within a function: It retains the value between function calls.  
  
2. If it is declared for a function name: By default function is extern..so it will be visible from other files if the function declaration is as static..it is invisible for the outer files.  
  
3. Static for global variables: By default we can use the global variables from outside files If it is static global..that variable is limited to with in the file.

#### 3. What is difference between static and extern?

**Ans:**  
"The **static** storage class is used to declare an identifier that is a local variable either to a function or a file and that exists and retains its value after control passes from where it was declared. This storage class has a duration that is permanent. A variable declared of this class retains its value from one call of the function to the next. The scope is local. A variable is known only by the function it is declared within or if declared globally in a file, it is known or seen only by the functions within that file. This storage class guarantees that declaration of the variable also initializes the variable to zero or all bits off.  
The **extern** storage class is used to declare a global variable that will be known to the functions in a file and capable of being known to all functions in a program. This storage class has a duration that is permanent. Any variable of this class retains its value until changed by another assignment. The scope is global. A variable can be known or seen by all functions within a program. ."

#### 4. What is difference between static local and static global variable?

**Ans:**   
**Static global  :**  
Static variable has scope only in the file in which it is declared. it can't be accessed in any other file but its value remains intact if code is running in some other file means lifetime is in complete program .  
**Static local:**  
static local variable has scope in that function in which it is declared means it can't be used in other functions in the same file also, means scope is limited to the function in which it is declared while its life time is also through out the program.

#### 5. Can we declare static variable in header file?

**Ans:**   
You can’t declare a static variable without defining it as well (this is because the storage class modifiers static and extern are mutuallyexclusive). A static variable can be defined in a header file, but this would cause each source file that included the header file to have its own private copy of the variable, which is probably not what was intended.

#### 6. Can we declare main() function as static?

**Ans:**  
No. The C spec actually says somewhere in it  that the main function cannot be static.  
The reason for this is that static means "don't let anything outside this source file use this object". The benefit is that it protects against name collisions in C when you go to link (it would be bad bad bad if you had two globals both named "is\_initialized" in different files... they'd get silently merged, unless you made them static). It also allows the compiler to perform certain optimizations that it wouldn't be able to otherwise. These two reasons are why static is a nice thing to have.  
Since you can't access static functions from outside the file, how would the OS be able to access the main function to start your program? That's why main can't be static.  
Some compilers treat "main" specially and might silently ignore you when you declare it static.

#### 7. Draw memory layout of C program?

**Ans :**  
**Refer This link:** [**http://cinterviewquestionandanswer.blogspot.in/2014/01/memory-layout-of-c-programs.html**](http://cinterviewquestionandanswer.blogspot.in/2014/01/memory-layout-of-c-programs.html)   
 **8.What is volatile variable means?**  
volatile has nothing to deal with storage class.  
volatile just tells the compiler or force the compiler to "not to do the optimization" for that variable. so compiler would not optimize the code for that variable and reading the value from the specified location, not through interal register which holds the previous value.  
So, by declaring variable as volatile.. it gives garrantee that you will get the latest value, which may be alterred by an external event.  
your code may be work fine if haven't declare that variable as volatile, but there may be chance of not getting correct value sometimes.. so to avoid that we should declare variable as volatile.  
volatile is generally used when dealing with external events, like interrupts of hardware related pins.   
  
**Example**. reading adc values.  
  
**const voltile** means you can not modify or alter the value of that variable in code. only external event can change the value.  
controller pins are generally defines as volatile. may be by declaring variable as volatile controller will do "read by pin" not "read by latch"... this is my assumtion. may be wrong...  
but still there is lots of confusion when to choose variable as volatile..  
A variable should be declared volatile whenever its value could change unexpectedly. In practice, only three types of variables could change:  
Memory-mapped peripheral registers  
Global variables modified by an interrupt service routine  
Global variables within a multi-threaded application

#### 9. What does keyword const means?

**Ans:**  
The const qualifier explicitly declares a data object as something that cannot be changed. Its value is set at initialization. You cannot use const data objects in expressions requiring a modifiable lvalue. For example, a const data object cannot appear on the lefthand side of an assignment statement  
int const volatile var

#### 10. What do the following declaration means?

const int a;  
int const a;  
const int \*a;  
int \* const a;  
int const \* a const;

#### 11. Can we use const keyword with volatile variable?

**Ans:**  
Yes. The const modifier means that this code cannot change the value  
of the variable, but that does not mean that the value cannot be  
changed by means outside this code. For instance, in the example  the timer structure was accessed through a volatile const pointer. The function itself did not change the value of the timer, so it was declared const. However, the value was changed by hardware on the computer, so it was declared volatile. If a variable is both const and volatile, the two modifiers can appear in either order.  
  
 **Pointer String and array question :**

#### 1. What are pointers?

**Ans:**  
A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is:

type \*var-name;

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk \* you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Following are the valid pointer declaration:

int \*ip; /\* pointer to an integer \*/

double \*dp; /\* pointer to a double \*/

float \*fp; /\* pointer to a float \*/

char \*ch /\* pointer to a character \*/

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

#### 2. What is dangling pointer?

**Ans:**  
A [**dangling pointer**](http://en.wikipedia.org/wiki/Dangling_pointer) points to memory that has already been freed. The storage is no longer allocated. Trying to access it might cause a Segmentation fault.  
Common way to end up with a dangling pointer:

char\* func()

{

char str[10];

strcpy(str,"Hello!");

return(str);

}

//returned pointer points to str which has gone out of scope.

You are returning an address which was a local variable, which would have gone out of scope by the time control was returned to the calling function. **(Undefined behaviour)**  
Another common dangling pointer example is an access of a memory location via pointer, after free has been **explicitly** called on that memory.

int \*c = malloc(sizeof(int));

free(c);

\*c = 3; //writing to freed location!

#### 3. What is NULL pointer?

**Ans:**  
Null pointer is special reserved value of a pointer. A pointer of any type has such a reserved value. Formally, each specific pointer type (int\*, char\*) has its own dedicated null-pointer value. Conceptually, when a pointer has that null value it is not pointing anywhere.  
  
**4. What is void Pointer?**  
**Ans:**  
Void pointer or generic pointer is a special type of pointer that can be pointed at objects of any data type. A void pointer is declared like a normal pointer, using the void keyword as the pointer’s type.  
  
Pointers defined using specific data type cannot hold the address of the some other type of variable i.e., it is incorrect in C++ to assign the address of an integer variable to a pointer of type float.  
  
Example:  
  
float \*f; //pointer of type float  
int i;  //integer variable  
f = &i; //compilation error  
  
The above problem can be solved by general purpose pointer called void pointer.  
  
Void pointer can be declared as follows:  
  
void \*v // defines a pointer of type void  
  
The pointer defined in this manner do not have any type associated with them and can hold the address of any type of variable.  
  
Example:  
  
void \*v;   
int \*i;  
int ivar;  
char chvar;  
float fvar;  
v = &ivar; // valid  
v = &chvar; //valid  
v = &fvar; // valid  
i = &ivar; //valid  
i = &chvar; //invalid  
i = &fvar; //invalid  
  
**5. What is memory leakage? How can we avoid it?**  
**Ans :**  
Memory leak occurs when programmers create a memory in heap and forget to delete it.  
Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

/\* Function with memory leak \*/

#include

void f()

{

int \*ptr = (int \*) malloc(sizeof(int));

/\* Do some work \*/

return; /\* Return without freeing ptr\*/

}

To avoid memory leaks, memory allocated on heap should always be freed when no longer needed.

/\* Function without memory leak \*/

#include ;

void f()

{

int \*ptr = (int \*) malloc(sizeof(int));

/\* Do some work \*/

free(ptr);

return;

}

#### 6. What is the size of pointer in 32 bit machine?

**Ans:**  
Sizeof of pointer in 32 bit machine is always 4 bytes.

#### 7. Write a program to find weather machine is 32 bit or 64 bit?

**Ans:**  
int main()  
{  
    int \*p = NULL;  
    if(sizeof(p) == 4)  
        printf("Machine is 32 bit\n");  
    else  
        printf("Machine is 64 bit\n");  
    return 0;  
}

#### 8. What is array?

**Ans:**  
In C programming, one of the frequently arising problem is to handle similar types of data. For example: If the user want to store marks of 100 students. This can be done by creating 100 variable individually but, this process is rather tedious and impracticable. These type of problem can be handled in C programming using arrays.  
An array is a sequence of data item of homogeneous value(same type).

#### 9. What is difference between array and pointer?

**Ans:**

An array is an array and a pointer is a pointer, but in most cases array names are *converted* to pointers.  
Here is an array:  
int a[7]   
**a** contains space for seven integers, and you can put a value in one of them with an assignment, like this:  
a[3] = 9;   
Here is a pointer:  
int \*p;   
**p** doesn't contain any spaces for integers, but it can point to a space for an integer. We can for example set it to point to one of the places in the array **a**, such as the first one:  
p = &a[0];   
What can be confusing is that you can also write this:  
p = a;   
This does *not* copy the contents of the array **a** into the pointer **p** (whatever that would mean). Instead, the array name **a** is converted to a pointer to its first element. So that assignment does the same as the previous one.  
Now you can use p in a similar way to an array:  
p[3] = 17;   
The reason that this works is that the array dereferencing operator in C, "[ ]", is defined in terms of pointers. **x[y]** means: start with the pointer **x**, step **y** elements forward after what the pointer points to, and then take whatever is there. Using pointer arithmetic syntax, **x[y]** can also be written as **\*(x+y)**.  
For this to work with a normal array, such as our **a**, the name **a** in **a[3]** must first be converted to a pointer (to the first element in **a**). Then we step 3 elements forward, and take whatever is there. In other words: take the element at position 3 in the array. (Which is the fourth element in the array, since the first one is numbered 0.)  
So, in summary, array names in a C program are (in most cases) converted to pointers. One exception is when we use the **sizeof** operator on an array. If **a** was converted to a pointer in this contest, **sizeof(a)** would give the size of a pointer and not of the actual array, which would be rather useless, so in that case **a** means the array itself.

#### 10. Can we increment the base address of array? Justify your answer.

**Ans:**  
No**,** Becauseonce we initialize the array variable, the pointer points base address only & it's fixed  and constant pointer.

#### 11.  What is the output of program.

#### int a[5] = {1,2,3,4,5};

#### int \*ptr = (int\*) (&a +1);

#### int \*ptr = (int\*) (a+1);

**Ans:**

you see, for your program above, a and &a will have the same **numerical value**,and I believe that's where your whole confusion lies.You may wonder that if they are the same,the following should give the **next address** after a in both cases,going by pointer arithmetic:

(&a+1) and (a+1)

But it's not so!!**Base address** of an array (a here) and **Address** of an array are not same! a and &a might be same numerically ,but they are not the same **type**. a is of type int\* while &a is of type int (\*)[5],ie , &a is a pointer to (address of ) and array of size 5.But a as you know is the **address of the first element of the array**.Numerically they are the same as you can see from the illustration using **^** below.  
But when you increment these two pointers/addresses, ie as (a+1) and (&a+1), the arithmetic is totally different.While in the first case it "jumps" to the address of the next element in the array, in the latter case it **jumps by 5 elements as that's what the size of an array of 5 elements.**

#### 12. What is output of the program?

int main()

{  
            int arr[10];  
            int \*ptr = arr;  
            ptr++;  
            arr++;  
            return 0;   
}  
 **Ans:**  
The statement arr++ will give you lvalue error. Because hear you are trying to increment base address of array and by default base address of array is constant pointer(constant value) so,  
**arr  =  arr+1;**  
**i.e according to rule on LHS of assignment operator there always should be lvalue i.e variable not constant.**

#### 13. What is string?

**Ans:**  
The string in C programming language is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

#### 14. What is difference between these two,

**char ptr[] = "string";**  
**char \*ptr = "string";**  
  
**Ans:**

The two declarations are not the same.  
First one is the array of character i.e. string and second one is the string literals.  
char ptr[] = "string"; declares a char array of size 7 and initializes it with the characters s ,t,r,i,n,g and \0. You are **allowed** to modify the contents of this array.   
char \*ptr = "string"; declares ptr as a char pointer and initializes it with address of *string literal* "string" which is *read-only*. Modifying a string literal is an **undefined behavior**. What you saw(seg fault) is one manifestation of the undefined behavior.

#### 15. Write a program to find size of variable without using sizeof operator?

**Ans:**   
  
#define sizeof(var)     ( (char\*)(&var+1) - (char\*) (&var))  
  
int main()  
{  
    int val;  
    printf("size of = %d\n", SIZEOF(val));  
    return 0;  
}

#### 16. Write a program to find sizeof structure without using size of operator?

**Ans:**  
  
#define SIZEOF(var)     ( (char\*)(&var+1) - (char\*) (&var))  
  
int main()  
{  
        struct s {  
                int a;  
                char b;  
                int c;  
        };  
        struct s obj[1];  
  
    printf("size of = %ld\n", SIZEOF(obj));  
    return 0;  
}

#### 17. What is the output of following program?

int main()  
{  
      char str[] = "vishnu";  
      printf("%d %d\n",sizeof(str),strlen(str));  
}  
  
**Ans:**   
**7 6**  
Here char str[] = " 'v'. 'i' ,'s','h','n',u','\0' ";  
so sizeof operator always count null character whereas strlen skip null character.

#### 18. Write a program to implement strlen(), strcpy(),strncpy(), strrev(),strcmp() function"?

**Ans:**

### 1. strlen:

int my\_strlen(const char \* str)  
{  
    int count;  
    while(\* str != '\0') {  
        count++;  
        str++;  
    }  
    return 0;  
}

#### 2. strcpy:

void my\_strcpy(char \* dest, const char\* src)  
{  
    while(\* src != '\0') {  
        \*dest = \*src;  
        dest++;  
        src++;  
    }  
    \*dest = '\0';  
}

#### 3. strrev:

void my\_strrev(char \*str,size)  
{  
    int i;  
    char temp;  
    for(i=0;i<=size/2;i++) {  
        temp = str[i];  
        str[i] = str[size-i];  
        str[size-i] = temp;  
    }  
}  
  
int main()  
{  
    char str[10] = "vishnu";  
    int len;  
    len = strlen(str);  
    my\_strrev(str,len-1);  
    printf("strrev = %s\n",str);  
    return 0;  
}

#### 4. strcmp :

void my\_strcmp(char \* dest, const char\* src)  
{  
     while(\*str != '\0' && \*dest != '\0') {  
        str++;  
        dest++;  
    }  
    return (\*src - \*dest);  
}  
  
int main()  
{  
    char str[10];  
    char dest[10];  
    int i;  
    i = my\_strcmp(dest,src);  
    if(i == 0 )   
        printf("strings are equal\n");  
    if(i < 0)   
        printf(" string1 is less than string2\n");  
    if(i > 0)   
        printf("string2 is greter than string1\n");  
    return 0;  
}

#### 5. strncpy

void my\_strncpy(char \* dest, const char\* src,int n)  
{  
    while(\*src != '\0' && n != 0) {  
        \*dest = \*src;  
        dest++;  
        src++;  
        n--;  
    }  
    while(n) {  
        \*dest = '\0';  
        n--;  
    }  
}

#### 19 . Write a program to implement above function using recursion?

#### 20 . Write a program to check weather  string is palindrome or not?

int main()  
{  
    char string[25], reverse\_string[25] = {'\0'};  
    int  i, length = 0, flag = 0;  
    fflush(stdin);  
    printf("Enter a string \n");  
    gets(string);  
    for (i = 0; string[i] != '\0'; i++) {  
        length++;  
    }  
      
    for (i = length - 1; i >= 0; i--) {  
        reverse\_string[length - i - 1] = string[i];  
    }  
    if(strcmp(string,reverse\_string) == 0 )  
        printf("%s is palindrome\n");  
    else  
        printf("%s is not palindrome\n");  
    return 0;   
}

#### 21 . Write a program to count total number of vowel,consonant present in given string?

**Ans:**  
  
int main()  
{  
  
    char sentence[80];  
    int i, vowels = 0, consonants = 0, special = 0;  
    printf("Enter a sentence \n");  
    gets(sentence);  
         for (i = 0; sentence[i] != '\0'; i++) {  
  
        if ((sentence[i] == 'a' || sentence[i] == 'e' || sentence[i] == 'i' || sentence[i] == 'o' || sentence[i] == 'u') || (sentence[i] == 'A' || sentence[i] == 'E' || sentence[i] == 'I' || sentence[i] == 'O' || sentence[i] == 'U')) {  
        vowels = vowels + 1;  
        }  
        else {  
            consonants = consonants + 1;  
        }  
        if (sentence[i] =='t' ||sentence[i] =='\0' || sentence[i] ==' ') {  
            special = special + 1;  
        }  
  
    }  
    consonants = consonants - special;  
      printf("No. of vowels in %s = %d\n", sentence, vowels);  
    printf("No. of consonants in %s = %d\n", sentence, consonants);  
    return 0;  
  
}

#### 22. Write a function to find whether machine is little endian or big endian.?

**Ans:**  
void is\_little\_or\_big(int n)  
    {  
        int num = 0x01;  
        char \* ptr = (char\*)num;  
        if(\*ptr == 1)  
            printf("little endian\n");  
        else  
            printf("big endian\n");  
    }  
    **or**  
      
    void is\_little\_or\_big()  
    {  
        enum union {  
            int a;  
            char c[4];  
        };  
        enum endian obj;  
        obj.i = 1;  
        if(obj.c[0] == 1)   
            printf("Machine is little endian\n");  
        else  
            printf("machine is big endian\n");  
    }

#### Write a program to find occurrence of particular key in given string?

#### Write a program to move all 0's to one side and 1's on ther side of array?

#### Write a program to find largest element in an array?

#### Write a program to find second largest element from array?

**Bit Manipulation**  **:**

#### 1. Write a program to count total number of setbit in 32 bit number?

int countset(int num)

{

           int count = 0;

           while (num) {

                    if( ((num) & 1) == 1)

                           count++;

                     num = num >> 1;

              }

             return count;

}

or

unsigned int countsetbit(int num)

{

           int count = 0;

           while(num != 0) {

                      count ++;  
                      num = num & (num-1);

              }

             return count;

}

#### 2. Write a program to set n th bit in 32 bit number?

int setbit(int num, int pos)

{

           num = num | 1 << pos;

}

#### 3. Write a program to count total number of reset bit in 32 bit integer?

 int setbit(int num)

{

           int count = 0;

           while (num) {

                    if( ((num) & 1) == 0)

                           count++;

                     num = num >> 1;

              }

             return count;

}

#### 4. Write a program to reset nth bit in 32 bit number?

int resetbit(int num, int pos)

{

           num = num &  ~(1 << pos);

}

#### 5. Write a program to swap nibble of a 1byte data?

#### 6. Write a program to swap two variable using bitwise operator?

void swap(int a, int b)  
{  
         a = a ^ b;  
         b = a ^  b;  
         a = a ^ b;  
}

#### 7. Write a program to find number is even or odd?

void evnodd(int num)  
{  
       if( (num) & (1) )  
           printf("odd");  
      else  
           printf("even");   
}

#### 8. Write a program to find number is power of 2 or not?

void  power(int num)  
{  
       if( !( (num) & (num-1) ) )  
           printf("power of 2");  
      else  
           printf("num is not power of 2");   
}

#### 9 Write a function to swap even bits with consecutive odd bits in a number.

#### e.g. bo swapped with b1,b2 sawpped with b3 and so on.

 Given an unsigned integer, swap all odd bits with even bits. For example, if the given number is 23 (**0**0**0**1**0**1**1**1), it should be converted to 43 (0**0**1**0**1**0**1**1**). Every even position bit is swapped with adjacent bit on right side (even position bits are highlighted in binary representation of 23), and every odd position bit is swapped with adjacent on left side.  
If we take a closer look at the example, we can observe that we basically need to right shift (>>) all even bits (In the above example, even bits of 23 are highlighted) by 1 so that they become odd bits (highlighted in 43), and left shift (<<) all odd bits by 1 so that they become even bits. The following solution is based on this observation. The solution assumes that input number is stored using 32 bits.  
Let the input number be x  
1) Get all even bits of x by doing bitwise and of x with 0xAAAAAAAA. The number 0xAAAAAAAA is a 32 bit number with all even bits set as 1 and all odd bits as 0.  
2) Get all odd bits of x by doing bitwise and of x with 0x55555555. The number 0x55555555 is a 32 bit number with all odd bits set as 1 and all even bits as 0.  
3) Right shift all even bits.  
4) Left shift all odd bits.  
5) Combine new even and odd bits and return.

// C program to swap even and odd bits of a given number

#include

unsigned int swapBits(unsigned int x)

{

    // Get all even bits of x

    unsigned int even\_bits = x & 0xAAAAAAAA;

    // Get all odd bits of x

    unsigned int odd\_bits  = x & 0×55555555;

    even\_bits >>= 1;  // Right shift even bits

    odd\_bits <<= 1;   // Left shift odd bits

    return (even\_bits | odd\_bits); // Combine even and odd bits

}

// Driver program to test above function

int main()

{

    unsigned int x = 23; // 00010111

    // Output is 43 (00101011)

    printf("%u ", swapBits(x));

    return 0;

}

Output:

43

#### 10. Write a function to set a particular bit.

unsigned int setbit(unsigned inr num,int pos)

{

num = num | (1 << pos);

return num;

}

#### 11. Write a function to clear a particular bit.

unsigned int clear(unsigned inr num,int pos)

{

num = num & ~ (1 << pos);

return num;

}

#### 12. Write a function to toggle particular bit.

unsigned int togglebit(unsigned inr num,int pos)

{

num = num ^ (1 << pos);

return num;

}

#### 13. Write a function to swap even bits with consecutive odd bits in a number.

#### e.g. b0 swapped with b1, b2 swapped with b3 and so on.

unsigned int swap\_bits(unsigned int num)

{

return (num >> 1 & 0x55555555) | (num << 1 & 0xAAAAAAAA);

}

#### 14. Write a function to swap odd bits in a number.

#### e.g. b1 swapped with b3, b5 swapped with b7 and so on.

unsigned int swap\_odd\_bits(unsigned int num)

{

return (num >> 2 & 0x22222222) |

  (num << 2 & 0x88888888) |

       ( num   & 0x55555555) ;

}

#### 15. Write a function to swap even bits in a number.

#### e.g. b0 swapped with b2, b4 swapped with b6 and so on.

unsigned int swap\_even\_bits(unsigned int num)  
{  
 return (num >> 2 & 0x11111111) |  
        (num << 2 & 0x44444444) |  
        ( num   & 0xAAAAAAAA);}

#### 16. Write a function to find out the number of 1s in a number.

unsigned int num\_of\_ones(unsigned int num)

{

if( (count\_ones(num) & 1)

return ODD;

else

return EVEN;

}

#### 17. Write a function for finding the first lowest bit set in a number.

unsigned int first\_lowest\_bit(unsigned num)

{

int count =0;

while(num) {

count ++;

if( (num) & 1 == 1)

break;

num = num >> 1;

}

return count;

}

#### 18. Write a function for finding the higest bit set in a number.

unsigned int first\_highest\_bit(unsigned num)

{

int count =0;

while(num) {

count ++;

if( (num & (1 << 31) ) == 1)

break;

num = num << 31;

return count;

}

#### 19 Write a function for reversing the bits in a number.

unsigned int reverse\_bit(unsigned num)

{

unsigned int NO\_OF\_BITS = sizeof(num) \* 8;

unsigned int temp,rev=0;

for(i=0; i <= NO\_OF\_BITS -1 ;i++) {

if(temp) {

rev |= (1 << ((NO\_OF\_BITS-1)-i);

}

return rev;

}

#### 20. Write a code to extract nth to mth bit, where n

(num >> n) & ~(~ 0 << (m-n+1))  
 **21. write a code for toggling nth to m bits,where n < m**

num ^ ((~ 0 << n) & ( ~0 >> (31-m)))

**22. Write a code for setting nth to mth bit, where n < m**

num | ((~0 << n) & (~0 >>(31-m)))

#### 23. write a code for clearing nth to mth bit, where n  < m

num & ~((~0 << n) & (~0 >> (31-m)))

## Link List Question:

#### 1. How to check whether linked list is circular or not.

**Ans**:

**s**truct node {

    int data;  
    struct node \*next;  
};  
struct node \*head = NULL;  
  
void checkcircular(struct node \*head)  
{  
    struct node \* slow = head;  
    struct node \* fast = head;  
    while( fast && fast->next) {  
        if(slow == fast->next->next) {  
            printf("Circular\n');  
            break;  
        }  
        else {              
            slow = slow->next;  
            fast = fats->next->next;  
        }  
    }  
}

#### 2. How would you find a loop in a singly linked list?

struct node {  
    int data;  
    struct node \*next;  
};  
  
struct node \*head = NULL:  
  
void detectloop(struct node \* head)  
{  
    struct node \* slow = head;  
    struct node \* fast = head;  
    while(slow && fast && fast->next) {  
        slow = slow->next;  
        fast = fast->next->next;  
        if (slow == fast) {  
            printf("Loop detected\n');  
            break;  
        }  
    }  
}

#### 3. Write a C function to print the middle of a given linked list.

struct node {  
    int data;  
    struct node \*next;  
};  
  
void findmiddle(struct node \*head)  
{  
    struct node \* slow = head;  
    struct head \* fast = head;  
    while (fast != NULL && fast->next != NULL) {  
        slow = slow->next;  
        fast = slow->next->next;  
    }  
    printf(" Middle element is %d\n", slow->data);  
}

#### 4. Write a c program to get the intersection point of two singly linked lists.

struct node {  
    int data;  
    struct node \*next;  
};  
  
struct node \* head = NULL:  
  
int count\_node(struct node \* head)   
{  
    int count = 0;  
    struct node \* current = head;  
    while (current != NULL) {  
        count ++;  
        current = current->next;  
    }  
    return count;  
}  
  
struct get\_intersection\_mod(int d,struct node \* head1, struct node \* head2)  
{  
    struct node current1 = head1;  
    struct node current2 = head2;  
      
    for(i=0;i        cureent1 = current->next;  
    }  
    while(cuttent1 != NULL && current2 != NULL) {  
        if(current1 == current2) {  
            printf(" intersection node =%d\n", current1->data);  
            break;  
        }  
        current1 = current1->next;  
        current2 = current2->next;  
    }  
}  
  
void get\_intersecton(struct node \* head1, struct node \* head2)  
{  
    struct node \*current = head1;  
    struct node \*current = head2;  
    int c1, c2;  
    c1 = count\_node(current1);  
    c2 = count\_node(current2);  
    if(c1 > c2) {  
        d= c1 -c2;  
        get\_intersection\_mod(d,head1,head2);  
    }  
    else {  
        d = c2 -c1;  
        get\_intersection\_mod(d, head1,head2);  
    }  
}

<http://cinterviewquestionandanswer.blogspot.in/2014/06/function-pointers-and-callbacks-in-c.html>

### C String interview Question

**1. What will be output when you will execute following c code?**  
#include  
void main(){  
char arr[7]="Network";  
printf("%s",arr);  
}  
Explanation:  
Size of a character array should one greater than total number of characters in any string which it stores. In c every string has one terminating null character. This represents end of the string.So in the string “Network” , there are 8 characters and they are ‘N’,’e’,’t’,’w’,’o’,’r’,’k’ and ‘\0’. Size of array arr is seven. So array arr will store only first sevent characters and it will note store null character.  
As we know %s in prinf statement prints stream of characters until it doesn’t get first null character. Since array arr has not stored any null character so it will print garbage value.  
  
  
 **2.What will be output when you will execute following c code?**  
#include  
void main(){  
    char arr[11]="The African Queen";  
    printf("%s",arr);  
}  
*Explanation:*  
Size of any character array cannot be less than the number of characters in any string which it has assigned. Size of an array can be equal (excluding null character) or greater than but never less than.  
 **3.What will be output when you will execute following c code?**  
#include  
void main(){  
    int const SIZE=5;  
    int expr;  
    double value[SIZE]={2.0,4.0,6.0,8.0,10.0};  
    expr=1|2|3|4;  
    printf("%f",value[expr]);  
}  
*Explanation:*  
Size of any array in c cannot be constantan variable.  
 **4.What will be output when you will execute following c code?**  
#include  
enum power{  
    Dalai,  
    Vladimir=3,  
    Barack,  
    Hillary  
};  
void main(){  
    float leader[Dalai+Hillary]={1.f,2.f,3.f,4.f,5.f};  
    enum power p=Barack;  
    printf("%0.f",leader[p>>1+1]);  
}  
**Explanation:**  
Size of an array can be enum constantan.  
Value of enum constant Barack will equal to Vladimir + 1 = 3 +1 = 4  
So, value of enum variable p  = 4  
leader[p >> 1 +1]  
= leader[4 >> 1+1]  
=leader[4 >> 2]   //+ operator enjoy higher precedence than >> operator.  
=leader[1]  //4>>2 = (4 / (2^2) = 4/4 = 1  
=2  
 **5.What will be output when you will execute following c code?**  
#include  
#define var 3  
void main(){  
    char \*cricket[var+~0]={"clarke","kallis"};  
    char \*ptr=cricket[1+~0];  
    printf("%c",\*++ptr);  
}  
**Explanation:**  
In the expression of size of an array can have micro constant.  
var +~0 = 3 + ~0 = 3 + (-1)  = 2  
Let’s assume string “clarke” and “kallis” has stored at memory address 100 and 500 respectively as shown in the following figure:  
For string “clarke”:  
For string “kallis”:  
In this program cricket is array of character’s pointer of size 2. So array cricket will keep the memory address of first character of both strings i.e. content of array cricket is:  
cricket[2] = {100,500}  
ptr is character pointer which is pointing to the fist element of array cricket. So, ptr = 100  
Now consider on \*++ptr  
Since ptr = 100 so after ++ptr , ptr = 101  
\*(++ptr) = \*(101) = content of memory address 101. From above figure it is clear that character is l.  
 **6.What will be output when you will execute following c code?**  
#include  
void main(){  
    char data[2][3][2]={0,1,2,3,4,5,6,7,8,9,10,11};  
    printf("%o",data[0][2][1]);  
}  
*Explanation:*  
%o in printf statement is used to print number in the octal format.  
 **7.What will be output when you will execute following c code?**  
#include  
void main(){  
    short num[3][2]={3,6,9,12,15,18};  
    printf("%d  %d",\*(num+1)[1],\*\*(num+2));  
}  
*Explanation:*  
\*(num+1)[1]=\*(\*((num+1)+1))=\*(\*(num+2))=\*(num[2])=num[2][0]=15And\*\*(num+2)=\*(num[2]+0)=num[2][0]=15  
 **8.What will be output when you will execute following c code?**  
#include  
void main(){  
    char \*ptr="cquestionbank";  
    printf("%d",-3[ptr]);  
}  
Explanation:  
-3[ptr]=-\*(3+ptr)=-\*(ptr+3)  
=-ptr[3]  
=-103  //ASCII value of character ‘e’ is 103  
  
**9.What will be output when you will execute following c code?**  
#include  
void main(){  
    long  myarr[2][4]={0l,1l,2l,3l,4l,5l,6l,7l};  
    printf("%ld\t",myarr[1][2]);  
    printf("%ld%ld\t",\*(myarr[1]+3),3[myarr[1]]);  
    printf("%ld%ld%ld\t" ,\*(\*(myarr+1)+2),\*(1[myarr]+2),3[1[myarr]]);   
}  
Explanation:  
Think yourself.  
 **10.What will be output when you will execute following c code?**  
#include  
void main(){  
    int array[2][3]={5,10,15,20,25,30};  
    int (\*ptr)[2][3]=&array;  
    printf("%d\t",\*\*\*ptr);  
    printf("%d\t",\*\*\*(ptr+1));  
    printf("%d\t",\*\*(\*ptr+1));  
    printf("%d\t",\*(\*(\*ptr+1)+2));  
}  
Explanation:  
ptr is pointer to two dimension array.  
\*\*\*ptr  
=\*\*\*&array  //ptr = &array  
=\*\*array //\* and & always cancel to each other  
=\*arr[0]  // \*array = \*(array +0) = array[0]  
=array[0][0]  
= 5  
Rests think yourself.  
 **11.What will be output when you will execute following c code?**  
#include  
void main(){  
    static int a=2,b=4,c=8;  
    static int \*arr1[2]={&a,&b};  
    static int \*arr2[2]={&b,&c};  
    int\* (\*arr[2])[2]={&arr1,&arr2};  
    printf("%d %d\t",\*(\*arr[0])[1],  \*(\*(\*\*(arr+1)+1)));  
}  
**Explanation:**  
Consider on the following expression:  
\*(\*arr[0])[1]  
=\*(\*&arr1)[1]  //arr[0] = &arr1  
=\*arr1[1]   //\* and & always cancel to each other  
=\*&b  
=b  
=4  
*Consider on following expression:*  
\*(\*(\*\*(arr+1)+1))  
= \*(\*(\*arr[1]+1))  //\*(arr+1) = arr[1]  
= \*(\*(\*&arr2+1))  //arr[1] = &arr2  
=\*(\*(arr2+1))  //\*&arr2 = arr2  
=\*(arr2[1])  //\*(arr2+1) = arr2[1]  
=  \*&c    //arr2[1] = &c  
=  c  
= 8  
 **12.What will be output when you will execute following c code?**  
#include  
#include  
double myfun(double);  
void main(){  
    double(\*array[3])(double);  
    array[0]=exp;  
    array[1]=sqrt;  
    array[2]=myfun;  
    printf("%.1f\t",(\*array)((\*array[2])((\*\*(array+1))(4))));   
}  
double myfun(double d){  
       d-=1;  
       return d;  
}  
Explanation:  
array is array of pointer to such function which parameter is double type data and return type is double.  
Consider on following expression:  
(\*array)((\*array[2])((\*\*(array+1))(4)))  
= (\*array)((\*array[2])((\*array[1])(4)))  
//\*(array+1) = array[1]  
= (\*array)((\*array[2])(sqrt(4))))  
//array[1] = address of sqrt function  
= (\*array)((\*array[2])(2.000000)))  
= (\*array)(myfun(2.000000)))  
// array[2] = address of myfunc function  
=(\*array)(1.000000)  
=array[0](1.000000)  
=exp(1.000000)  
 **13.What will be output when you will execute following c code?**  
#include  
typedef struct{  
    char \*name;  
    double salary;  
}job;  
void main(){  
    static job a={"TCS",15000.0};  
    static job b={"IBM",25000.0};  
    static job c={"Google",35000.0};  
    int x=5;  
    job \* arr[3]={&a,&b,&c};  
    printf("%s  %f\t",(3,x>>5-4)[\*arr]);  
}  
double myfun(double d){  
       d-=1;  
       return d;  
}  
*Explanation:*  
(3,5>>5-4)[\*arr]  
=(3,5>>5-4)[\*arr] //x=5  
= (3,5>>1)[\*arr] //- operator enjoy higher precedence than >>  
= (3,2)[\*arr]  //5>>1 = 5/(2^1) = 5 /2 = 2  
= 2[\*arr]  //In c comma is also operator.  
= \*(2 + \*arr)  
= \*(\*arr + 2)  
=\*arr[2]  
=\*(&c) //arr[2] = &c  
=c   // \*  and & always cancel to each other.  
So,  
printf("%s  %f\t",c);  
=> printf("%s  %f\t", "Google",35000.0);  
 **14.What will be output when you will execute following c code?**  
#include  
union group{  
    char xarr[2][2];  
    char yarr[4];  
};  
void main(){  
    union group x={'A','B','C','D'};  
    printf("%c",x.xarr[x.yarr[2]-67][x.yarr[3]-67]);  
}  
*Explanation:*  
In union all member variables share common memory space.  
So union member variable, array xarray will look like:  
{  
{‘A’,’B’},  
{‘C’,’D’}  
}  
And union member variable, array yarray will look like:  
{  
{‘A’,’B’,’C’,’D’}  
}  
x.xarr[x.yarr[2]-67][x.yarr[3]-67]  
= x.xarr[‘C’-67][‘D’-67]  
= x.xarr[67-67][68-67]  
//ASCII value of ‘C’ is 67 and ‘D’ is 68.  
x.xarr[0][1]  
=’B’  
 **15.What will be output when you will execute following c code?**  
#include  
void main(){  
    int a=5,b=10,c=15;  
    int \*arr[3]={&a,&b,&c};  
    printf("%d",\*arr[\*arr[1]-8]);  
}  
*Explanation:*  
Member of an array cannot be address of auto variable because array gets memory at load time while auto variable gets memory at run time.  
 **16.What will be output when you will execute following c code?**  
#include  
void main(){  
    int arr[][3]={{1,2},{3,4,5},{5}};  
    printf("%d %d %d",sizeof(arr),arr[0][2],arr[1][2]);  
}  
*Explanation:*  
If we will not write size of first member of any array at the time of declaration then size of the first dimension is max elements in the initialization of array of that dimension.  
So, size of first dimension in above question is 3.  
So size of array = (size of int) \* (total number of elements) = 2 \*(3\*3) = 18  
Default initial value of rest elements are zero.  So above array will look like:  
{  
{1,2,0}  
{3,4,5},  
{5,0,0}  
}           
 **17.What will be output when you will execute following c code?**  
#include  
void main(){  
    int xxx[10]={5};  
    printf("%d %d",xxx[1],xxx[9]);  
}  
*Explanation:*  
If we initialize any array at the time of declaration the compiler will treat such array as static variable and its default value of uninitialized member is zero.  
 **18.What will be output when you will execute following c code?**  
#include  
#define WWW -1  
enum {cat,rat};  
void main(){  
    int Dhoni[]={2,'b',0x3,01001,'\x1d','\111',rat,WWW};  
    int i;  
    for(i=0;i<8 i="" span="">  
         printf(" %d",Dhoni[i]);  
}  
Explanation:  
Dhoni[0]=2  
Dhoni[1]=’b’ =98  //ASCII value of character ‘b’ is 98.  
Dhoni[2]=  0x3  =  3  //0x represents hexadecimal number. Decimal value of hexadecimal 3 is also 3.  
Dhoni[3]=01001 = 513 //Number begins with 0 represents octal number.  
Dhoni[4]  = ‘\x1d’ = 29 //’\x1d’ is hexadecimal character constant.  
Dhoni[5] = ‘\111’ = 73 //’\111’ is octal character constant.  
Dhoni[6] =rat = 1  //rat is enum constant  
Dhoni[7] = WWW = -1  //WWW is macro constant.  
  
**19.What will be output when you will execute following c code?**  
#include  
void main(){  
    long double a;  
    signed char b;  
    int arr[sizeof(!a+b)];  
    printf("%d",sizeof(arr));  
}  
Explanation:  
Size of data type in TURBO C 3.0 compiler is:  
S.N.  
Data type  
Size(In byte)  
1  
char  
1  
2  
int  
2  
3  
double  
8  
Consider on the expression: !a + b  
! Operator always return zero if a is non-zero number other wisie 1. In general we can say ! operator always returns an int type number. So  
!a +b  
=! (Any double type number) + Any character type number  
= Any integer type number + any character type number  
= Any integer type number  
Note: In any expression lower type data is always automatically type casted into the higher data type. In this case char data type is automatically type casted into the int type data.  
So sizeof (!a +b) = sizeof(Any int type number)  = 2  
So size of array arr is 2 and its data type is int. So  
sizeof(arr) = size of array \* sizeof its data type = 2\* 2= 4  
  
**20.What will be output when you will execute following c code?**  
#include  
void main(){  
    char array[]="Ashfaq \0 Kayani";  
    char \*str="Ashfaq \0 Kayani";  
    printf("%s %c\n",array,array[2]);  
    printf("%s %c\n",str,str[2]);  
    printf("%d %d\n",sizeof(array),sizeof(str));  
}  
*Explanation:*  
A character array keeps the each element of an assigned array but a character pointer always keeps the memory address of first element.   
As we know %s in prints the characters of stream until it doesn’t any null character (‘\0’).  So first and second printf function will print same thing in the above program.  But size of array is total numbers of its elements i.e. 16 byte (including ending null character). While size of any type of pointer is 2 byte (near pointer).

# [Qualcomm Interview Questions](http://embedded-thoughts.blogspot.com/2013/07/qualcomm-interview-questions.html)

How many interrupt lines you have been used in your driver development.  
UART : 1 (RX)  
Ethernet: 2 (RX & TX)  
I2C : what is I2C Clock Stretch, Bus line clear.  
  
How do u debug when an process aborted.  
  
How many Exceptions are there in ARM.  
  
what are the methods or functions to raise execute the interrupt process when interrupt occurs.  
  
  
C-Questions  
1) Find the size of a structure with out size of operator.

struct XYZ{

int x;

float y;

char z;

};

int main(){

struct XYZ arr[2];

int sz = (char\*)&arr[1] - (char\*)&arr[0];

printf("%d",sz);

return 0;

}

OS  
2) what is the output of the below program  
    Function(void)  
   {  
     fork(); fork(); fork();  
    printf("Hello world");  
   }  
ans: Hello world will be printed for 8 times.  
  
L&T Interview Questions:  
  
1) Timer Interrupts and related drivers.  
2) How to delete similar nodes from a linked list, not by traversing every node.  
3) ARM Internal Architecture.  
4) Power Management Techniques - Experience.  
5) Explain me any driver you worked on.  
6) Interrupt based driver.  
7) Advantages and disadvantages of Polling & Interrupt drivers.  
8) Android Frame work.  
9) What are the different modes in ARM.  
10) How a interrupt will be handled  in Linux.  
11) What is the difference between IRQ & FIQ.  
  
  
Work:  
1) Touch screen Device drivers.  
2) USB Device driver Experience.  
3) Power Management Techniques.  
  
  
  
Some more questions:  
1.how the synchronisation works in single processor system(spinlock\_irq\_save).

2.how external devices access the physical address.

3.can we access the variable in all the functions below

func1()

func2()

static int a;

func3()

func4()

func1()

func2()

int a;

func3()

func4()

4.why stack is always in higher location.

5. what is the difference between global and static

6. have u debug any issues using jtag/ any other debugger.

if yes, how can we get the call stack in the debugger.

<https://www.careercup.com/page?pid=linux-kernel-interview-questions>

what is basic difference in insmod & modprobe ?

modprobe reads the modules from /lib/modules/$(uname -r)/modules.dep.bin (or without the .bin suffix if the other file is not available). From the same file, dependencies are loaded.  
  
modprobe accepts the name of a .ko file in /lib/modules/$(uname -r) (e.g. nvidia-current for the file dkms/nvidia-current.ko) and aliases (modules.alias.bin). Builtins (modules.alias.bin) are recognized as well, but since these modules are loaded by default, there is not point in modprobing this kind of modules.  
  
insmod on the other hand accepts paths to files. The module does not have to reside in /lib/modules/$(uname -r), but dependencies are not automatically loaded. This is the lower program used by modprobe to load modules.

everyone will prefer modprobe because insmod have no capability to resolve dependency issue .

# [How to know when a schedule() call is returning because of a signal?](http://stackoverflow.com/questions/1832033/how-to-know-when-a-schedule-call-is-returning-because-of-a-signal)

# [when schedule() returns?](http://stackoverflow.com/questions/15083898/when-schedule-returns)

in case of blocking IO, say, driver read, we call wait\_event\_interruptible() with some condition. When the condition is met, read will be done. I looked into wait\_event\_interruptible() function, it checks for condition and calls schedule(). schedule() will look for the next runnable process and does context switch and other process will run. Does it mean that, the next instruction to be executed for the current process will be inside schedule() function when this process is woken up again?

1. If yes, if multiple process voluntarily calls schedule, then all processes will have next instruction to be executed once after it gets woken up will be well inside schedule()?
2. In case of ret\_from\_interrupt, schedule() is called. When it will return? as iret is executed after that.

http://developerweb.net/viewforum.php?id=70

Questions regarding both Clients and Servers (TCP/SOCK\_STREAM) [/size]

[\*]How can I tell when a socket is closed on the other end?

[\*]What's with the second parameter in bind()?

[\*]How do I get the port number for a given service?

[\*]If bind() fails, what should I do with the socket descriptor?

[\*]How do I properly close a socket?

[\*]When should I use shutdown()?

[\*]Please explain the TIME\_WAIT state.

[\*]Why does it take so long to detect that the peer died?

[\*]What are the pros/cons of select(), non-blocking I/O and SIGIO?

[\*]Why do I get EPROTO from read()?

[\*]How can I force a socket to send the data in its buffer?

[\*]Where can I get a library for programming sockets?

[\*]How come select says there is data, but read returns zero?

[\*]Whats the difference between select() and poll()?

[\*]How do I send [this] over a socket?

[\*]How do I use TCP\_NODELAY?

[\*]What exactly does the Nagle algorithm do?

[\*]What is the difference between read() and recv()?

[\*]I see that send()/write() can generate SIGPIPE. Is there any advantage to handling the signal, rather than just ignoring it and checking for the EPIPE error?

[\*]After the chroot(), calls to socket() are failing. Why?

[\*]Why do I keep getting EINTR from the socket calls?

[\*]When will my application receive SIGPIPE?

[\*]What are socket exceptions? What is out-of-band data?

[\*]How can I find the full hostname (FQDN) of the system I'm running on?

[\*]How do I monitor the activity of sockets?

Writing Client Applications (TCP/SOCK\_STREAM)

\* How do I convert a string into an internet address?

\* How can my client work through a firewall/proxy server?

\* Why does connect() succeed even before my server did an accept()?

\* Why do I sometimes lose a server's address when using more than one server?

\* How can I set the timeout for the connect() system call?

\* Should I bind() a port number in my client program, or let the system choose one for me on the connect() call?

\* Why do I get "connection refused" when the server isn't running?

\* What does one do when one does not know how much information is comming over the socket? Is there a way to have a dynamic buffer?

\* How can I determine the local port number?

Writing Server Applications (TCP/SOCK\_STREAM)

\* How come I get "address already in use" from bind()?

\* Why don't my sockets close?

\* How can I make my server a daemon?

\* How can I listen on more than one port at a time?

\* What exactly does SO\_REUSEADDR do?

\* What exactly does SO\_LINGER do?

\* What exactly does SO\_KEEPALIVE do?

\* How can I bind() to a port number < 1024?

\* How do I get my server to find out the client's address / hostname?

\* How should I choose a port number for my server?

\* What is the difference between SO\_REUSEADDR and SO\_REUSEPORT?

\* How can I write a multi-homed server?

\* How can I read only one character at a time?

\* I'm trying to exec() a program from my server, and attach my socket's IO to it, but I'm not getting all the data across. Why?

Writing UDP/SOCK\_DGRAM applications

\* When should I use UDP instead of TCP?

\* What is the difference between "connected" and "unconnected" sockets?

\* Does doing a connect() call affect the receive behaviour of the socket?

\* How can I read ICMP errors from "connected" UDP sockets?

\* How can I be sure that a UDP message is received?

\* How can I be sure that UDP messages are received in order?

\* How often should I re-transmit un-acknowleged messages?

\* How come only the first part of my datagram is getting through?

\* Why does the socket's buffer fill up sooner than expected?

Advanced Socket Programming

\* How would I put my socket in non-blocking mode?

\* How can I put a timeout on connect()?

\* How do I complete a read if I've only read the first part of something, without again calling select()?

\* How to use select routine

\* RAW sockets

\* Restricting a socket to a given interface

\* Receiving all incoming traffic through a RAW-socket?

\* Multicasting

\* Getting IP header of a UDP message

\* To fork or not to fork?

\* Connect with timeout (or another use of select() )

Bugs and Strange Behaviour

\* send() hangs up when sending to a switched off computer

\* Error when using inetd

Connect with timeout (or another use for select() )

To fork or not to fork?

Is it possible to receive all the incoming packets ignoring target port, source address/port and seq/ack-numbers, through a raw-socket?

How do I restrict a socket to a specific inteace i.e. that it only listens and accepts from the given interface ?

How would a socket using the SOCK\_RAW protocol be used, and what, if any, advantages would it have over SOCK\_DGRAM or SOCK\_STREAM protocols ?

How to use select call for multiplexing ?

How can I put a timeout on connect()?

How would I put my socket in non-blocking mode?

How to be sure that UDP messages are received in order ?

How do I monitor the activity of sockets?

How can I find the full hostname (FQDN)?

What are socket exceptions? What is out-of-band data?

how to do handling SIGPIPE signal or EPIPE errno ?

Whats the difference between select() and poll()?

Where can I get a library for programming sockets?

Pros/cons of select(), non-blocking I/O and SIGIO ?

How do I get the port number for a given service?

How to tell when a socket is closed on the other end?

How do I convert a string into an internet address?

How can my client work through a firewall/proxy server ?

How can I determine the local port number?

How to find out the client's address/hostname ?

How should I choose a port number for my server?

Difference between SO\_REUSEADDR and SO\_REUSEPORT ?

How can I write a multi-homed server?

When should I use UDP instead of TCP?

Modify the source code of the top-level Makefile CC = $ (CROSSCOMPILE) gcc -> CC = $ (CROSSCOMPILE) gcc -g to make students vmlinux contains debug information

All generated .o's rule in addition to other parameters -c together with a copy outside the parameters used to

generate the .o file CC -E -dD -C $ <> / preprocessing / $ (shell pwd) / $ <preprocessed file generated from this file which can easily find c source file macro definitions

objdump -h vmlinux> vmlinux.txt display linux kernel pieces of information, such as the start of the virtual address of the segment, the length of the segment

objdump -S -l -z vmlinux> vmlinux.txt disassembly vmlinux to mix the code vmlinux.txt, vmlinux.txt containing a compilation and c source file,

objdump -S -l -z -j xxxx (section name) vmlinux> vmlinux.txt disassemble the linux kernel to the file vmlinux.txt in paragraph xxxx.

objdump -x vmlinux> header information for all segments x.txt vmliux, where the mouth of the bag vmlinux entry address

objdump -debugging vmlinux> debugging.txt debug a lot of useful information, such as function names, structures, definitions, etc.

objdump params -S disassemble the source code as possible, especially when compile time when -g specifies that debugging parameters,

the effect is obvious. It implies the -d parameter. -l with the file name and line number marked corresponding object code only and -d, -D or -r together and

use that use -ld -d difference is not great, useful for source-level debugger, and compiler use requirements like the -g debug compiler option.

[-l | -line-Numbers]

[-S | -source] Mixed assembly

[-z | -disassemble-Zeroes]

[-j Section | -section = section]

[-prefix-Addresses]

Interrupts:

The following are two ways of requesting CPU attention:

Interrupt

Polling

What are interrupts ?

What are interrupt handlers?

An interrupt is an event which is raised by software or hardware when its needs the CPU's attention

types of interrupts. Broadly speaking, we can split interrupts into 2 major classes:

External or hardware generated interrupts;

kernel/irq/irqdesc.c.

include/linux/irqdesc.h

early\_irq\_init()

I/O interrupts;

Timer interrupts;

Interprocessor interrupts.

Software-generated interrupts: software exception : Faults, Traps, Aborts.

A fault is an exception reported before the execution of a "faulty" instruction (which can then be corrected). If corrected, it allows the interrupted program to be resume.

A trap is an exception which is reported immediately following the execution of the trap instruction. Traps also allow the interrupted program to be continued just as a fault does.

An abort is an exception that does not always report the exact instruction which caused the exception and does not allow the interrupted program to be resumed.

We can determine the priorities from the highest to the lowest in the following table:

Hardware Reset and Machine Checks

| 1 | RESET

| | - Machine Check

+--------------+-------------------------------------------------+

| | Trap on Task Switch |

| 2 | - T flag in TSS is set |

+--------------+-------------------------------------------------+

| | External Hardware Interventions |

| | - FLUSH |

| 3 | - STOPCLK |

| | - SMI |

| | - INIT |

+--------------+-------------------------------------------------+

| | Traps on the Previous Instruction |

| 4 | - Breakpoints |

| | - Debug Trap Exceptions |

+--------------+-------------------------------------------------+

| 5 | Nonmaskable Interrupts |

+--------------+-------------------------------------------------+

| 6 | Maskable Hardware Interrupts |

+--------------+-------------------------------------------------+

| 7 | Code Breakpoint Fault |

+--------------+-------------------------------------------------+

| 8 | Faults from Fetching Next Instruction |

| | Code-Segment Limit Violation |

| | Code Page Fault |

+--------------+-------------------------------------------------+

| | Faults from Decoding the Next Instruction |

| | Instruction length > 15 bytes |

| 9 | Invalid Opcode |

| | Coprocessor Not Available |

| | |

+--------------+-------------------------------------------------+

| 10 | Faults on Executing an Instruction

| | Overflow |

| | Bound error |

| | Invalid TSS |

| | Segment Not Present |

| | Stack fault |

| | General Protection |

| | Data Page Fault |

| | Alignment Check |

| | x87 FPU Floating-point exception |

| | SIMD floating-point exception |

| | Virtualization exception |

Each processor has its own thread that is called ksoftirqd/n where the n is the number of the processor. We can see it in the output of the systemd-cgls util:

processor specific kernel thread that called ksoftirqd.

$ systemd-cgls -k | grep ksoft

The spawn\_ksoftirqd function starts this these threads.

why same TASKLET can't execute on two core simultaneously?

is nested interrupt supported ? if yes than how it works ?

what is interrupt latency and how it can be reduced ?

IRQ VS ISR vs interrupt handler ?

TASKLET vs SOFTIRQ vs WORKWUEUE vs TOP HALF vs BOTTOM HALF ?

Given a pid, how will you distinguish if it is a process or a thread ?

Do ps -AL | grep pid

1st column is parent id and the second column is thread (LWP) id. if both are same then its a process id otherwise thread.

Where are macros stored in the memory?

What is the difference between kill-6 and kill -9?

IGKILL and SIGABRT are two type of signals that are sent to process to terminate it.

SIGKILL is equivalent of "kill -9" and is used to kill zombie processes, processes that are already dead and waiting for their parent processes to reap them.

SIGABRT is equivalent of "kill -6" and is used to terminate/abort running processes.

SIGKILL signal cannot be caught or ignored and the receiving process cannot perform any clean-up upon receiving this signal.

SIGABRT signal can be caught, but it cannot be blocked.

Linux Kernel In short

What is a Linux Device Driver Model ?

How to Pass Command Line Arguments to a Kernel Module?

What are the Possible Task States ?

What is a Loadable Kernel Module?

Monolithic and Micro Kernel

Why do we need two bootloaders viz. primary and secondary?

Given a pid, how will you distinguish if it is a process or a thread ?

What is memory leak?

What is a linux kernel ? isit a process or thread?

How function pointers are shared across different processes? using which IPCs?

What are the Synchronization techniques used in Linux Kernel?

On x86-32 Linux, at which address the code segment of the program and stack starts?

Why Kernel Code running in interrupt context cannot sleep?

How does User Space memory Layout look like?

How will the User Space mapping look like when our RAM is less than 896 MB?

How will the Kernel Space mapping look like when our RAM is less than 896 MB?

Explain the module loading in Linux.

How does Linux Manage memory?

When should one use Polling and when should one use Interrupts?

What is the difference between IRQ and FIQ in case of ARM?

Where are macros stored in the memory?

What is a kernel Panic?

What is OOPS in Linux Kernel?

How to avoid memory leak in Linux?

What is a buss error? What are the common causes of bus error?

What is a device tree in Linux?

How function pointers are shared across different processes? using which IPCs?

What is the difference between a Mutex and a Semaphore?

What are the important applications of touch command in linux?

How a system call is executed in X86 architecture?

How is a system call executed in ARM architecture?

Compare I2C and SPI protocols

What is a callback function?

Supported tests by the watchdog daemon to check the system status:

Is the process table full?

Is there enough free memory?

Are some files accessible?

Have some files changed within a given interval?

Is the average work load too high?

Has a file table overflow occurred?

Is a process still running? The process is specified by a pid file.

Do some IP addresses answer to ping?

Do network interfaces receive traffic?

Is the temperature too high? (Temperature data not always available.)

Execute a user defined command to do arbitrary tests.

Execute one or more test/repair commands found in /etc/watchdog.d. These commands are called with the argument test or repair

Counter: Counts and indicates the number of signals (events) input at any interval.

Timer: Counts the number of signals input at a constant interval to indicate the elapsed time.

A watchdog timer is a piece of hardware, often built into a microcontroller that can cause a processor reset when it

judges that the system has hung, or is no longer executing the correct sequence of code.

A watchdog timer is a piece of hardware that can be used to automatically detect the system problem like fault condition, hang and send a trigger pulse to reset the system.

You also can say it is a failsafe mechanism that can sense if a system stop functioning.

what is the difference between hardware watchdog and software watchdog ?

How can a CPU dynamically change its clock frequency?

My Intel CPU changes clock speed depending on the usage, but how does it decide what clock speed to run at?

Is the clock speed determined by the OS software using an algorithm, or is it hardware based?

Is it dependent on the # of interrupts? The cache turnover? Does the CPU itself set its own clock?

Or does a separate controller set it? Or software?

How fast cpu clock do you need to control camera?

What exactly will the CPU be doing? Image capture? Image processing? Communications?

Will image processing be performed on an entire captured image or on streamed video on a pixel-by-pixel basis? How complex would the image processing be?

If you haven't figured it out, your question is impossible to answer in it's current form.

What is clock cycle, machine cycle, instruction cycle in a microprocessor?

How does a CPU change frequency?

The clock that the CPU uses does not come directly from the crystal oscillator in the motherboard !

The (fixed frequency) clock from the crystal oscillator is first fed to a system called a Phase Locked Loop or PLL.

The simplest PLLs can only multiply the fixed frequency clock from the crystal oscillator so if the crystal oscillator runs at

40 MHz then such a PLL can generate 80 MHz, 120 MHz, 160 Mhz, etc. Basically n times the crystal clock frequency.

Is there a physical limit to CPU speed? How high is it?

What is the highest CPU speed reached so far?

**Important question:**

Operating System Interview Questions:

Kernel Synchronization what are the differences between mutex and semaphore?

What is a race condition and how to avoid a race condition?

What is a critical region?

What are atomic operations?

RTOS:

What are the differences between general purpose OS and and RTOS?

What are the characteristics of an RTOS?

What is the difference between a hard real time system and a soft real time system?

What is priority inversion and how to solve that problem?

What is priority inheritance?

Process management:

1) how to manipulate the current process

2) what are kernel thread.

3) how threads are implemented in linux kernel.

4) What are different state of a process in lunix.

5) what is difference between process and thread.

6) generally what resources are shared between threads.

7) what is process descriptor

8) what is task\_struct.

9) what is therad\_info structure for.

10) what was the need of thread\_info structure.

11) difference betwen fork() and vfork()

12) what is process context.

13) what is zombie process.

14) how parent less process is handles in linux.

Process Scheduling

1) what is process scheduling

2) what is cooperative multitasking and pre-emptive multitasking.

3) what is yielding.

4) what is limitation of cooperative multitasking.

5) I/O bound versus Processor bound process.

6) what is process priority.

7) What kind of priority is maintained in linux.

8) what is nice value.

9) what is virtual run time.

10) what are the available scheduling classes in linux.

11) which type os scheduling used in linux.

12) how next task is picked for scheduling.

13) what is scheduler entry point in linux.

14) what is waitqueus.

15) How context switching is handled in linux.

16) what is user preemption and kernel preemption

Syscalls:

1) what is syscalls.

2) how system calls are implemented in linux.

3) what happens when process in userspace calls a syscall.

4) what is the need of verifying parameter in definition of syscall.

5) what is system calls context.

6) why it is not recommended to writing new syscall.

Interrupts and interrupt handlers

1) what is interrupt

2) what is interrupt handler or ISR.

3) what is top halves and bottom halves.

4) How interrupt is registered.

5) what are different interrupt handler flags.

6) How interrupt are freed.

7) what are the considerations needs to taken care while writing interrupt handler.

8) what are shared handlers.

9) what is interrupt context.

10) how to disable and enable interrupts.

11) what are different bottom halves techniques in linux.

12) what is tasklets , softirq and workqueus and difference among them.

13) when to choose which bottom halves.

14) how to implements softirq , tasklets and workqueus.

15) how to schedule tasklet.

16) what is ksoftirqd.

17) How to disable bottom halves.

18) How locking between bottom halves handled.

19) Why we need preemption.

Explain about the interrupt mechanims in linux?

What are the APIs that are used to register an interrupt handler?

How do you register an interrupt handler on a shared IRQ line?

Explain about the flags that are passed to request\_irq().

Explain about the internals of Interrupt handling in case of Linux running on ARM.

What are the precautions to be taken while writing an interrupt handler?

Explain interrupt sequence in detail starting from ARM to registered interrupt handler.

What is bottom half and top half.

What is request\_threaded\_irq()

If same interrupts occurs in two cpu how are they handled?

How to synchronize data between 'two interrupts' and 'interrupts and process'.

How are nested interrupts handled?

How is task context saved during interrupt.

What are the considerations needs to taken care while writing interrupt handler ?

How to disable and enable interrupts ?

Bottom-half Mechanisms in Linux

What are the different bottom-half mechanisms in Linux?

Softirq, Tasklet and Workqueues

What are the differences between Softirq/Tasklet and Workqueue? Given an example what you prefer to use?

When to choose which bottom halves ?

Softirq is guaranteed to run on the CPU it was scheduled on, where as tasklets don’t have that guarantee.

The same tasklet can't run on two separate CPUs at the same time, where as a softirq can.

When are these bottom halfs executed?

Explain about the internal implementation of softirqs?

Explain about the concurrent work queues.

How to schedule tasklet or bottom half ?

How to disable bottom halves ?

What is ksoftirqd ?

How locking between bottom halves handled ?

Kernel Synchronization:

1) what is synchronization

2) what is critical section

3) what is race condition.

4) why we need to take care of synchronization

5) what is various synchronization techniques in linux.

6) what is deadlocks.

7) what is atomic operations.

8) what is spin locks.

9) what is reader-writer spin lock.

10) what is semaphore.

11) what is binary semaphore.

12) what is difference between semaphore and spin lock.

13) when to choose what among spin lock and semaphore.

14) what is difference between semaphore and mutex.

15) what is preemption disabling and what is the use of this.

Why do we need synchronization mechanisms in Linux kernel?

What are the different synchonization mechanisms present in Linux kernel?

What are the differences between spinlock and mutex?

What is lockdep?

Which synchronization mechanism is safe to use in interrupt context and why?

Explain about the implementation of spinlock in case of ARM architecture.

Explain about the implementation of mutex in case of ARM architecture.

Explain about the notifier chains.

Explain about RCU locks and when are they used?

Explain about RW spinlocks locks and when are they used?

Which are the synchronization technoques you use 'between processes', 'between processe and interrupt' and 'between interrupts'; why and how ?

What are the differences between semaphores and spinlocks?

Process Management and Process Scheduling

What are the different schedulers class present in the linux kernel?

How to create a new process?

What is the difference between fork( ) and vfork( )?

Which is the first task what is spawned in linux kernel?

What are the processes with PID 0 and PID 1?

PID 0 - idle task and PID 1 - init .

How to extract task\_struct of a particular process if the stack pointer is given?

Timers and Time Management:

1) what is tick rate and jiffies.

2) what are the various way of applying delay in linux.

3) what is blocking and non blocking call

4) what is Real time clock (RTC).

5) how busy looping is implemented in linux.

What are jiffies and HZ?

What is the initial value of jiffies when the system has started?

Explain about HR timers and normal timers?

On what hardware timers, does the HR timers are based on?

How to declare that a specific hardware timers is used for kernel periodic timer interrupt used by the scheduler?

How software timers are implemented?

Memory management:

1) how memory is managed in linux.

2) what are pages.

3) what are different memory zones in linux.

4) how to allocated pages.

5) how to freeing page.

6) what us kmalloc and what are action modifier we can pass while using kmalloc.

7) what is zone modifier in linux.

8) what is vmalloc

Virtual file system

1) what is virtual file system and what is the need of it in linux.

2) are are different object types in VFS.

3) what are the operations possible on inode and superblock objects.

Linux Memory Management

What are the differences between vmalloc and kmalloc? Which is preferred to use in device drivers?

What are the differences between slab allocator and slub allocator?

What is boot memory allocator?

How do you reserve block of memory?

What is virtual memory and what are the advanatages of using virtual memory?

What's paging and swapping?

Is it better to enable swapping in embedded systems? and why?

What is the page size in Linux kernel in case of 32-bit ARM architecture?

What is page frame?

What are the different memory zones and why does different zones exist?

What is high memory and when is it needed?

Why is high memory zone not needed in case of 64-bit machine?

How to allocate a page frame from high memory?

In ARM, an abort exception if generated, if the page table doesn't contain a virtual to physical map for a particular page. How exactly does the MMU know that a virtual to physical map is present in the pagetable or not?

Process Address Space:

1) what is process address space.

2) what is memory descriptor in linux and which structure represents it.

3) how to allocate and destroy a memory descriptor

4) does kernel thread has any association with mm\_struct.

5) what is VMA and what are various VMA operations possible in linux kernel.

6) how to manipulate memory area in linux.

7) what mmap and do\_mmap().

8) what is page tables.

Linux Device Driver :

Explain about the Linux Device Model (LDM)?

Explain about about ksets, kobjects and ktypes. How are they related?

1) What is device driver and what is the need of it.

2) what are different kind of devices.

3) what is module in linux.

4) how mudules are loaded in linux.

5) difference between insmod and modprobe.

6) how parameters are shared between driver modules.

7) what are IOCTLS.

8) what is syscalls.

9) what are the benefits of syscalls.

10) how character driver is registered in linux.

11) what is init and exit function of a driver.

12) how and when init and exit function of driver get called.

13) what is probe function.

14) when probe is get called.

15) what is platform devices.

16) what is device tree.

17) what are the benefits of device tree over board files.

18) what is sysfs and procfs.

19) how logs are printed in linux kernel and what are the logs level available in linux.

20) what is copy\_to\_user and copy\_from\_user.

21) what do you mean by kernel configuration and what are the various way of configuring kernel.

22) what is menuconfig.

23) what is ioremap

24) what is seagmentation fault.

25) what are the various ways od debugging linux kernel.

26) how linux kernel boots.

27) what is zimage and bzimage.

28) what are different booting arguments in linux.

29) how parameter are passed from boot loader to kernel.

30) what is ATAGS.

31) from which file kernel execution starts.

32) what is bootloader.

33) what is primary and secondary bootloader.

34) why we need two bootloader.

35) difference between poll and select.

36) what is priority inheritance and priority inversion.

37) what are different type of kernel.

38) what is DMA.

39) what is cache coherency.

40) what is copy on write.

41) what is highmem and lowmem.

42) what happens if we pass invalid address from userspace by using ioctls.

43) what are different ipc mechanism in linux.

44) what are sockets.

45) how page fault is handled in linux.

46) difference between memory based io and port based io.

47) what is I2c and SPI.

48) how physical to virtual translations works in linux.

49) what is thrashing, segmentation and fragmentation.

50) what is preempt\_count and what is the need of that.

Linux Kernel Modules:

How to make a module as loadable module?

How to make a module as in-built module?

Explain about Kconfig build system?

Explain about the init call mechanism.

What is the difference between early init and late init?

Linux Kernel Debugging

What is Oops and kernel panic?

Does all Oops result in kernel panic?

What are the tools that you have used for debugging the Linux kernel?

What are the log levels in printk?

Can printk's be used in interrupt context?

How to print a stack trace from a particular function?

What's the use of early\_printk( )?

Explan about the various gdb commands.

Miscellaneous:

How are the atomic functions implemented in case of ARM architecture?

How is container\_of( ) macro implemented?

Explain about system call flow in case of ARM Linux.

What 's the use of \_\_init and \_\_exit macros?

How to ensure that init function of a partiuclar driver was called before our driver's init function is called (assume that both these drivers are built into the kenrel image)?

What's a segementation fault and what are the scenarios in which segmentation fault is triggered?

If the scenarios which triggers the segmentation fault has occurred, how the kernel identifies it and what are the actions that the kernel takes?

Power Management in Linux:

Explain about cpuidle framework.

Explain about cpufreq framework.

Explain about clock framework.

Explain about regulator framework.

Explain about suspened and resume framwork.

Explain about early suspend and late resume.

Explain about wakelocks.

Linux Boot Sequence:

Explain about the Linux boot sequence in case of ARM architecture?

How are the command line arguments passed to Linux kernel by the u-boot (bootloader)?

Explain about ATAGS?

Explain about command line arguments that are passed to linux kernel and how/where they are parsed in kernel code?

Explain about device tree.

How does scheduler picks particular task?

When does scheduler picks a task?

How is timeout managed?

How does load balancing happens?

Explain about any scheduler class?

Explain about wait queues and how they implemented? Where and how are they used?

What is process kernel stack and process user stack? What is the size of each and how are they allocated?

Why do we need seperate kernel stack for each process?

What all happens during context switch?

What is thread\_info? Why is it stored at the end of kernel stack?

What is the use of preempt\_count variable?

What is the difference between interruptible and uninterruptible task states?

How processes and threads are created? (from user level till kernel level)

How is virtual run time (vruntime) calculated?

Explaining Workqueues, flags and how they fit into the kernel

Difference between uart\_register\_driver and platform\_driver\_register?

How to do a single dma transaction in kernel?

Translating virtual address to physical address in kernel space

http://stackoverflow.com/questions/35100766/how-to-make-usb-ethernet-driver-for-android

What is mknod and it's usage ?

In how many ways we can allocate device number ?

How can we allocate device number statically ?

How can we allocate device number dynamically ?

How can we Free Device Numbers ?

Can we have same major number for more than one device file ?

What is minor number and it's usage ?

What is range of major and minor numbers?

How to retrieve major and minor number from dev\_t type ?

How can i use my own major and minor number for a device file ?

How to see statically assigned major numbers ?

how interrupt id and ISR is mapped?

How to pin a interrupt to a CPU in driver

How many maximum different CPU-Cores can be used to processing of one IP-packet?

What are the different modes in ARM.

Timer Interrupts and related drivers.

What is the difference between IRQ & FIQ.

how the synchronisation works in single processor system(spinlock\_irq\_save).

how external devices access the physical address.

what is basic difference in insmod & modprobe.

What is the difference between Platform driver and normal device driver..?

What happens to the idle thread when a core is taken offline logically?

What is meant by system call?

What is the difference between programs and Kernel Modules?

What is the difference between user space and kernel space?

What is the importance of /proc and /sys File system ?

What is the difference between printf() and printk() ?

How a user mode is transferred to kernel mode?

How system call causes change from user to kernel space

Which RTOS worked on. Difference between various OS/RTOS

Measure of performance of OS. Define performance

What are the various code optimization techniques used

memory leak deduction and various ways of handling

How to proceed if system is sluggish

How to determine if some high prio task is hogging CPU

How debug prints/system trace could help solve above issues

Important things to look for in code reviews

Understanding of schematics

Why driver code in not written in C++/Java

How to debug while system is running

Run Time optimization

what is repeate sequence in I2C?

How this signal will look on CRO?

How many lines required for SPI communication?

Do you need to change Clock polarity and phase for SPI?

Who has control of SPI clock?

What is deadlock? How to come out of deadlock?

what is difference between mutex and semaphore?

Explain board bring up.

how will you verify I2C communication?How will you know if there was a software or hw problem?

How will you know if the I2C lines are noisy?

Have you worked in crash dump?

How will you solve memory crash, what steps would you take if a crash occurs?

Data sharing between ISR and threads?

what is NAND and NOR flash, diff between them?

What monitoring tools have you used like I2C sniffer

what is scheduler?

who schedules the scheduler

what is cache coherence?

Can a scheduler can be locked?

At what frequency scheduler looks for threads/processes ready for schedule?

what actually does a system does in a tick?

When cahche is enabled in a operating system ,DMA is enabled,how does DMA access the cache?

what is context switch ?

Diff btwn SPI & I2C

When a signal is raised on a interrupt line what will be passed to processor either interrupt number or anything else?

How kernel comes to know which device raised an interrupt when interrupt is shared.

How can an application can talk to specific device in this below case ?

having Hard Disk devices

SATA,SATA1,SATA2

if their a devce driver registered for these devices and a single drivere is used to manage thesr devies then how can an application talk to a sepcific device

Ex: if i want to read from SATA1 hard disk and Write to SATA0 and SATA1 hard disk.

How do you establish a sync mechanisam in above hard disk scenarios.

When a Mutex lock is aquired by a low priority task and High priority trys to acquires the lock

will the low priority task will be pre emitted

The same above scenario is with sema and spinlock

How can two slaves can communicate with master in i2c protocol at a time?

How can we address different devices from same vendor i2c?

What is I2C protocol,I2C- dummy write,repeated start,arbitration,synchronization between masters

How to know whether a interrupt is an edge triggered or level triggered from registering an interrupt handler?

what is i2c bus error

What are the different data transfer modes?

What are the different types of device descriptors?

How many END point descriptors can be present per device?

DO you know container macros? In all most all kernel drivers it is used?

What is the clock frequency used in your I2C driver designed?

What is the start bit condition in I2C?

Can interrupts sleep?why?

How the master knows what is the start condition ?

while in between I2C communication what happens if clocks happends to be dragged low which is not as per i2c standard?

What if the slave device is not responding or no acknowledge bit is sent by slave device?

What is the difference between kmalloc and vmalloc?

how to get physically contiguousness memory allocation if kmalloc is giving logical contiguousness allocations?

what is the entry points in kernel?

Static Drivers vs Dynamic Driver?

How to write a makefile to build a customized kernel

How to debug crash, what is the first line you see when you see a crash/oops message, explain crash console

How do you build only a static (.a) library for kernel modules.

Kernel thread vs user space thread and kernel process vs user space process.

What was the need of thread\_info structure?

Thread switching and process switching in linux kernel ?

How context switching is handled in linux?

How parent less process is handles in linux ?

What is yielding?

1. Tell me about MIPI interface?

2. What is i2c bus and how it works?

3. How you will analyse crash dumps?

4. knowledge about GIT?

5. What is YOKTO?

6. If i2c client is not giving ack how will you identify the issues?

7. What will be your approach for any issue debugging?

8. Tell about SPI?

9. What are the different way of flashing binary into boards?

10. What is boot monitor code?

11. Tell me any issue in which you debugged and found that there is some issue in hardware?

12. pull up / pull down register?

13. Maximum size you can allocate using kmalloc?

14. What message you will get if memory is not availeble?

15. 1) What is ioctl?

16. 2) What happens if you pass a invalid user space address in an ioctl?

17. 2) What is Virtual memory?

18. 4) What are different partitions of Physical memory?

19. 5) Related to multimedia driver like camera and HDMI as i was working on that

20. 6) Device tree vs board file?

21. What is crash and opps?

22. How you can analyse crash dump?

23. What is static variable?

24. What is local static and global static with example?

25. What happens if we pass some wrong address from user space to kernel space?

26. linked list intersection identification

27. kmalloc and vmalloc

28. What is page fualt and how it is handled in linux kernel?

29. ARM modes?

30. How many irq lines are there in ARM?

Below is the interview questions asked in an interview in Vihaan Networks for the position of Embedded Engineer having skills of Linux Device Driver , Linux Kernel Programming, Embedded C programming , BSP, RTOS , Linux Kernel Debugging , I2C, SPI , Bootloader etc.

1. What is breakpoints?

2. How software and hardware breakpoints are implemented?

3. What is volatile constant ?

4. What is the need of volatile constant in embedded systems?

5. How you will analyze Linux Kernel crash dumps ?

6. How pagefault is handled in linux kernel?

7. What is sempahore and mutex?

8. What is the difference in semaphore and Mutex?

9. What is the difference b/w spinlock and semaphore?

10. When to use Semaphore and Mutex?

11. What is the difference between constant and macro?

12. What is difference between inline function and macro?

13. Questions Related to i2c

14. Question Related to DMA

15. Tell about Process address space of a c program?

16. Difference between local static and global static?

17. How compiler maintain the scope of local static or global static variable?

18. Why can't we go for sleep while handling interrupt?

19. How nested interrupts are handled in ARM?

20. Different modes in ARM ?

21. What are the benefits using FIQ?

22. Tell me the Linux kernel Booting sequence?

23. How you will find out the issues if your device is not booting?

24. What is reset vector

25. pull up and pull down register?

26. Edge triggered and level triggered Interrupt?

27. Maximum size you can allocate using kmalloc?

28. What message you will get if memory is not available?

29. What is the significance of spinlock on uniprocessor system?

30. Interrupt Handling

31.

1. Types of interrupts

1) Software interrupts

2) Hardware interrupts

. Edge Triggering

. Level triggering

. MII

3) exceptions

2. Initialization of interrupts

3. Interrupt Handling

Questions:

1) What happens if interrupt handler goes to sleep?

2) What happens if interrupt handler uses semaphores

3) How to protect variable between two interrupt handlers?

4) How to protect variable between task and interrupt handler?

32. Interrupt Handling

33.

1. Types of interrupts

1) Software interrupts

2) Hardware interrupts

. Edge Triggering

. Level triggering

. MII

3) exceptions

2. Initialization of interrupts

3. Interrupt Handling

Questions:

1) What happens if interrupt handler goes to sleep?

2) What happens if interrupt handler uses semaphores

3) How to protect variable between two interrupt handlers?

4) How to protect variable between task and interrupt handler?

Linux Useful command for performance

1) iostat

Report Central Processing Unit (CPU) statistics and input/output statistics for devices and partitions.

2) netstat

3) blktrace

4) dtrace

5) perf

6) stap

7) top or htop

8) pidstat

9) nostat

10) dstat

11) slabtop

12) dstat

13) free

14) nicstat

15) ip

16) sar

17) sysctl

What is cooperative multitasking and pre-emptive multitasking?

What kind of priority is maintained in linux ?

What is user preemption and kernel preemption?

How ISR knows that it interrupts process in kernel mode or another interrupt (which enables further interrupts) - of course in kernel mode too

How does enabling and disabling interrupts from the kernel prevent race conditions?

How to copy MSB to rest of the byte?

How to use rotary encoder for counting?

Why is interrupt disabled between spin\_lock and spin\_unlock in Linux?

sleep-free kernel functions

Interrupt rate in modern computers

What does “interrupt hooking” mean?

What is the exact definition of 'process preemption'?

nested interrupt is allowed in linux, but the size of interrupt stack is limited,

is there any chance that too many nested interrupts cause stack overflow which crashes the whole system!

How to debug GPIO interrupt functions?

Timer interrrupts vs Dummy loops ?

How does MSI interrupt work in linux driver?

Which processor an interrupt will be generated on?

ARM interrupt service procedure vs function call procedure

Retrieving pid of tasklet

How do I write to a \_\_user memory from within the top half of an interrupt handler?

Why can interrupt handler has sleep functionality?

Are the Interrupt Stack and the Kernel Stack the same stack?

Why interrupt handler cannot use user stack ?

What is a user thread and a kernel thread?

Number of kernel threads = cores?

Maximum number of threads per process in Linux?

cat /proc/sys/kernel/threads-max

There is also a limit on the number of processes (an hence threads) that a single user may create, see ulimit/getrlimit

Linux implements max number of threads per process indirectly!!

number of threads = total virtual memory / (stack size\*1024\*1024)

ulimit -s

how it differs from user thread ?.

what is the relationship between the two threads ?.

how can i implement kernel threads ?.

Why must all user threads be mapped to a kernel thread?

What kind of hardware do the newer kernels support?

How do I identify which interrupt line shown in /proc/interrupts list is shared?

How does the Linux kernel handle shared IRQs?

What is the relationship between fork() and pthread\_atfork() in Linux?

What is the relationship between System Call and Software Interrupt in Linux?

How does the fork() system call in Linux work?

user/kernel interface (system calls, procfs/sysfs, ioctl) :

synchronization between contexts (how would you synchronize access to a shared memory area used from an interrupt handler and a workqueue on a SMP preemptive kernel).

memory allocation (kmalloc vs vmalloc)

are you familiar with the Linux kernel development process? Do you have any patch accepted in the mainline?

What are good ways to debug memory corruption in C under Linux?

How do I debug a kernel module in which a NULL pointer appears?

How to let kernel not run the printk with KERN\_DEBUG

How do I include the device tree with my kernel?

How to attach file operations to sysfs attribute in platform driver?

How size of virtual memory is decided and calculated in Linux ?

How to remove packet from UDP read buffer in kernel?

How can I bind my own serial device with a driver?

How does Linux Kernel know where to look for driver firmware?

How to create a device in /dev automatically upon loading of the kernel module for a device driver?

sudo mknod -m 0666 /dev/msio c 22 0

How to use netlink socket to communicate with a kernel module?

How to implement a Linux Device Driver for Data Acquisition Hardware?

How to find owner socket of sk\_buff in Linux kernel?

Why Socket Connection Blocked and TCP Kernel Keeps Retransmitting [ACK] packets

C Programming Interview Questions:

Volatile variable in-depth (definition, use case, impact, when to avoid, etc)

Priority inversion, priority inheritance, priority ceiling

Difference btwn Process and Thread

What is Process preemption

call back functions?

How can you avoid accessing of an array beyond its limits?

If passing name is an call by value, then array name is passed is it also call by value?

how can you determine whether your memory is in protected or un-protected mode

C program to swap every 2 bits in a 8 bit binary number

Write a program to find how many bit to toggle in 2 binary numbers so that they become equal

Write a program for to get the number of bits toggle in 2 binary numbers and toggle them to make the numbers equal

How you Decide the stack size for the function or thread

Memory Layout

How to decrease the time of booting processes

What is the functionality of PROBE function

How to detect whether a device is not detected?

How to find if their is in repeating node in linked list

Find a word from string

A thread is created by processes, how the process comes to know the completion of that created thread

In a big array consists of 1's & 0's ,write an efficient programe to keep all 1's to right side and 0's to other side

What is Thrashing

Deadlock - example

MM - Segmentation, paging, swapping

Paging vs swapping

Different segments in a program

Does linux use segmentation

What is DMA. Modes - cycle stealing/burst (blk transfer)/transparent

Cache coherency during dma. Which component handles it

Physical Virtual and logical addresses

difference between exceptions and signals and interrupt.

In a Process Address Space what if stack or heap collide?

Delete a particular node in a linked list having argument passed as address of a node.

without Header pointer known and other implementation with header pointer known

Write program to Insert element in static array

what all Debugging techniques you know?

Determine the minimum stack size required for given prog by end of main function?

how to debug your driver if u had any buffer issues

What is shared lib and static library? And how to create it ?

Difference between #define and CONST, which one will you chose in programming?

Difference between typedef and #define?

can we add two pointers

tell me the logic how can you find depth of a stack?

Can Volatile be applied to pointer?

why size of data types is restricted?

difference b/w macro and a constant variable

malloc vs calloc

Memory barriers; Why are they required

struct alignment and packing.

Pass by value and ref

size of void pointer

What is re-entrancy

What is structure padding?

How to write own malloc call

What is the difference between inline and macro?

Write program toreset bit in register if address of register is given

reset (int address, int bit)

How to know size of memory allocated by malloc using pointer?

Diff between mutex and semaphore? Can semaphore be used for data synchronisation purpose?

http://cinterviewquestionandanswer.blogspot.in/2014/01/c-interview-question.html

Bitwise operation :

Bit manipulationWrite the logic for setting nth bit.

Write the logic for clearing nth bit.

Write the logic for toggling nth bit.

Write the logic for setting nth to mth bits, where n > m.

Write the logic for clearing nth to mth bits, where n > m.

Write the logic for toggling nth to mth bits, where n > m.

Program for finding number of 1s and 0s in a 32-bit number.

Program for finding whether a number is power of 2 or not.

Program for finding whether a number is even or odd.

Write a function to swap even bits with consecutive odd bits in a number.

Write a function to swap odd bits in a number.

Write a function to swap even bits in a number.

Write a function to find out the number of 1s in a number.

Write a function to check whether the number of 1s present in a number are even or odd.

Write a function for finding the first lowest bit set in a number.

Write a function for finding the higest bit set in a number.

Write a function for reversing the bits in a number.

Write the code for extracting nth to mth bits, where n < m.

Write the code for toggling nth to mth bits, where n < m.

Write the code for setting nth to mth bits, where n < m.

Write the code for clearing nth to mth bits, where n < m

Write a piece of code for sizeof() implementation.

Explain about container\_of() and offsetof() implementations.

How to implement bit-wise operations without using bit-wise operators?

Check if a number is multiple of 9 using bitwise operators

Count strings with consecutive 1’s

Gray to Binary and Binary to Gray conversion

Find the maximum subset XOR of a given set

Given a set, find XOR of the XOR’s of all subsets.

Sum of Bitwise And of all pairs in a given array

Find Next Sparse Number

Find the maximum subarray XOR in a given array

Find XOR of two number without using XOR operator

Write a program to add one to a given number. You are not allowed to use operators like ‘+’, ‘-‘, ‘\*’, ‘/’, ‘++’, ‘–‘ ?

Multiply two integers without using multiplication, division and bitwise operators, and no loops

Check if a number is a power of another number

Check perfect square using addition/subtraction

Count numbers having 0 as a digit

Number of perfect squares between two given numbers

Write an Efficient C Program to Reverse Bits of a Number

Calculate square of a number without using \*, / and pow()

Linklist:

Data Structures Write a program for reversing a singly linked list?

Write a program for a singly linked list (insert, delete, count, search etc functions).

Write a program for a doubly linked list (insert, delete, count, search etc functions).

Write a program for a circular singly linked list (insert, delete, count, search etc functions).

Write a program for a circular doubly linked list (insert, delete, count, search etc functions).

Write a program for binary tree implementation.

You are given a pointer to a node (not the tail node) in a singly linked list. Delete that node from the linked list.

How to check whether a linked list is circular or not?

How would you find a loop in a singly linked list?

Write a c program for reversing a singly linked list.

Given two singly linked list, find if they are intersecting. Do this in single iteration. Also find the intersecting node in O(n) time and O(1) space. By intersection I mean intersection by reference not by value.

Write a c program to get the intersection point of two singly linked lists.

Find loop in linked list and remove the loop

implement Stack and Queue using Linked List

Repeatedly Delete N nodes after M nodes of a Linked list

Reverse every k nodes of a linked list

Reverse a Linked List using Recursion

Remove Duplicates from a Linked List

Print Linked List Elements in Reverse order

Merge a linked list into another linked list at alternate positions

Move last node to front in linked list

Swap every two nodes in a linked list

Frequency of a given number in a Linked List

Delete alternate nodes of a Linked List

Rotate linked list by K nodes

Reverse a singly linked list

Write a function to get the intersection point of two Linked Lists (Y Shape)

Write a program to detect loop in a Linked List

Insert nodes into a linked list in a sorted fashion

Write a C program to return the nth node from the end of a linked list

Find the size of a structure with out size of operator.

why stack is always in higher location.

what is the difference between global and static

have u debug any issues using jtag/ any other debugger.if yes, how can we get the call stack in the debugger.

Write a function to find whether machine is little endian or big endian.?

Write a program to find occurrence of particular key in given string?

write a program to move all 0's to one side and 1's on ther side of array?

Write a program to find largest element in an array?

Write a program to find second largest element from array?

Write a program to count total number of vowel,consonant present in given string?

Write a program to check weather string is palindrome or not?

Write a program to implement strncpy function using recursion

Write a program to implement strlen(), strcpy(),strncpy(), strrev(),strcmp() function"?

Write a program to find size of variable without using sizeof operator?

Can we increment the base address of array? Justify your answer.

What is difference between array and pointer?

Write a program to find weather machine is 32 bit or 64 bit?

What is memory leakage? How can we avoid it?

What is NULL , void, dangling pointer?

Can we use const keyword with volatile variable?

What does keyword const means?

Can we declare main() function as static?

Storage class static and extern and register in details.

Can we declare static variable in header file?

Where a const volatile variable is used?

1) what are the compilation steps? Explain

ans:- there are 4 compilation steps they are 1.preprocessor 2.compiler 3.assembler 4.linker.

1.preprocessor:-read the source code and resolve all preprocessor directives.

gcc -E first.c -o first.i

vim first.i

2.compiler:-transulates preprocessor code into assembly language.

gcc -S -v first.i -o first.s

vim first.s

3.assembler:-converts the assembler language in to binary language(or ) machine instructions.

gcc -C -v first.s -o first.o

objdump -D first.o

4.linker:-applies the build process i.e., the linker links all the relocatable binary files and libraries.

gcc first.o -o first

objdump -D first

write a program to print size of integer variable without using sizeof operator

write a program to find repeated character in a given string? code should be generic?

Implement string reverse program without using temporary buffer and strrev function?

how do you transfer data from userspace to kernel space?

What is generic system call that is used to transfer data from user space to kernel space?

write a program to copy a string from source to destination without using strcpy?

what is difference between mutex and semaphores

which one is faster array or linklist ?

how you cna tell whether your system is little or big indian ?

Write a program to implement memcpy() on your own

Difference between library call and a system call

Explain the meaning of Kernal.

Operating System Kernal - The kernel is the essential center of a computer operating system, the core that provides basic services for all other parts....

Operating System - What is a command interpreter?

Operating System - A command interpreter is the part of a computer operating system that understands and executes commands that are entered....

What is a daemon?

Operating System daemon - In Unix and some other operating systems, a daemon is a computer program that runs in the background, It is not under the direct control of a user.......

Basic functions of process management.

Operating System process management - The basic functions of the OS wrt the process management are : Allocating resources to processes,.....

What is a named pipe?

Operating System named pipe- Its a pipe that an application opens by name in order to write data into or read data from the pipe. ...

What is pre-emptive and non-preemptive scheduling?

In non-preemptive scheduling, a running task is executed till completion. It cannot be interrupted......

Operating System semaphore - What is a semaphore?

Operating System semaphore - A semaphore is a variable. There are 2 types of semaphores:Binary semaphores, Counting semaphores.....

Meaning of mutex - Binary semaphore vs. Mutex

Operating System mutex - A mutex and the binary semaphore are essentially the same. Both can take values: 0 or 1. However, there is a significant difference...

What are the different types of memory?

Operating System memory types - Cache Memory - This is a small amounts of memory used to speed up system performance.....

Operating System virtual memory - Explain the meaning of virtual memory.

Operating System virtual memory - Virtual memory is an approach to make use of the secondary storage devices as an extension of the primary storage of the computer......

What is RTOS?

What is RTOS? - A certain capability within a specified time constraint is guaranteed by an operating system called ‘real time operating system’.

Difference between hard real-time and soft real-time

Hard real-time and soft real-time - Critical task completion on time is guaranteed by a hard real time system.

What type of scheduling is there in RTOS?

Type of scheduling in RTOS - The tasks of real time operating system have 3 states namely, ‘running’, ’ready’, ‘blocked’.

What is interrupt latency?

What is interrupt latency? - The time between a device that generates an interrupt and the servicing of the device

What is priority inheritance?

What is priority inheritance? - Priority inversion problems are eliminated by using a method called priority inheritance.

What is spin lock?

What is spin lock? - In a loop a thread waits simply (‘spins’) checks repeatedly until the lock becomes available.

What is an operating system? Functions of an operating system - operating system

An operating system is an interface between hardware and software. OS is responsible for managing and co-ordinating the activities of a computer system.......

What is paging? Why paging is used? - operating system

OS performs an operation for storing and retrieving data from secondary storage devices for use in main memory........

Difference between a process and a program - operating system

A program is a set of instructions that are to perform a designated task, where as the process is an operation........

Physical memory and virtual memory - operating system

Physical memory is the only memory that is directly accessible to the CPU.......

Difference between socket and pipe - operating system

Sockets: Socket is a part of OSI layer model. Communication among different layers is performed through sockets......

Difference between THREAD, PROCESS and TASK - operating system

A program in execution is known as ‘process’. A program can have any number of processes. Every process has its own address space........

Differentiate between RAM and ROM - operating system

RAM: - Volatile memory - Electricity needs to flow continuously - Program information is stored in RAM.......

What is DRAM? In which form does it store data? - operating system

DRAM – Dynamic Random Access Memory. One of the read / write memory. DRAM is cheap and does the given task.......

What is cache memory? Explain its functions - operating system

Cache memory is RAM. The most recently processing data is stored in cache memory. CPU can access this data more quickly than it can access data in RAM.......

Differentiate between Complier and Interpreter - operating system

The program syntax is checked by the compiler; where as the keywords of the program is checked by the interpreter......

Describe different job scheduling in operating systems

Job scheduling is an activity for deciding the time for a process to receive the resources they request. First Come First Served: In this scheduling, the job that is waiting for a long time is served next........

What is a Real-Time System? - operating system

Real-time system is the study of hardware and software, which are subject to the operational deadlines from event to the system response......

What do you mean by deadlock? - operating system

Dead lock is a situation of two or more processes waiting for each other to finish their tasks. In this situation no progress or no advancement is made.......

Difference between Primary storage and secondary storage - operating system

Primary memory storages are temporary; where as the secondary storage is permanent.......

What are the different types of Kernel?

Kernels are basically of two types:

a. Monolithic Kernels - In this architecture of kernel, all the system services were packaged into a single system module which lead to poor maintainability and huge size of kernel.

b. Microkernels - They follow the modular approach of architecture. Maintainability became easier with this model as only the concerned module is to be altered and loaded for every function. This model also keeps a tab on the ever growing code size of the kernel.

What are the disadvantages of Microkernels?

Following are the main disadvantages of Microkernels. Usually these disadvantages are situation based.

a. Larger running memory footprint

b. Performance loss due to the requirement of more software for interfacing.

c. Difficulty in fixing the messaging bugs.

d. Complicated process management.

What is a daemon?

- Daemon - Disk and execution monitor, is a process that runs in the background without user’s interaction. They usually start at the booting time and terminate when the system is shut down.

How would you identify daemons in Unix?

- The name of daemons usually end with 'd' at the end in Unix.

- For e.g. httpd, named, lpd.

What do you mean by a zombie process?

- These are dead processes which are not yet removed from the process table.

- It happens when the parent process has terminated while the child process is still running. This child process now stays as a zombie

What is context switching?

- Context is associated with each process encompassing all the information describing the current execution state of the process

- When the OS saves the context of program that is currently running and restores the context of the next ready to run process, it is called as context switching.

- It is important for multitasking OS.

What is synchronization? What are the different synchronization mechanisms?

Synchronization means controlling access to a resource that is available to two or more threads or process. Different synchronization mechanisms are:

- Mutex

- Semaphores

- Monitors

- Condition variables

- Critical regions

- Read/ Write locks

What is the basic difference between pre-emptive and non-pre-emptive scheduling.

Pre-emptive scheduling allows interruption of a process while it is executing and taking the CPU to another process while non-pre-emptive scheduling ensures that a process keeps the CPU under control until it has completed execution.

Is non-pre-emptive scheduling frequently used in a computer? Why?

No, it is rarely used for the reasons mentioned below:

- It can not ensure that each user gets a share of CPU regularly.

- The idle time with this increases reducing the efficiency and overall performance of the system.

- It allows program to run indefinitely which means that other processes have to wait for very long.

Explain condition variable.

- These are synchronization objects which help threads wait for particular conditions to occur.

- Without condition variable, the thread has to continuously check the condition which is very costly on the resources.

- Condition variable allows the thread to sleep and wait for the condition variable to give it a signal.

What are read-write locks?

- Read - write locks provide simultaneous read access to many threads while the write access stays with one thread at a time. They are especially useful in protecting the data that is not frequently written but read simultaneously by many threads.

- They are slower than mutexes.

What is a deadlock?

- It is a condition where a group of two or more waiting for the resources currently in use by other processes of the same group.

- In this situation every process is waiting for an event to be triggered by another process of the group.

- Since no thread can free up the resource a deadlock occurs and the application hangs.

What are the necessary conditions for deadlock to occur?

a. At least one resource should be occupied in a non-sharable condition.

b. A process holding at least one resource is waiting for more resources currently in use by other processes.

c. It is not possible to pre-empt the resource.

d. There exists a circular wait for processes.

Name the different types of memory?

a. Main memory also called primary memory or RAM

b. Secondary memory or backing storage

c. Cache

d. Internal process memory

Differentiate between logical and physical address.

- Physical addresses are actual addresses used for fetching and storing data in main memory when the process is under execution.

- Logical addresses are generated by user programs. During process loading, they are converted by the loader into physical address.

When does page fault error occur?

- It occurs when a page that has not been brought into main memory is accessed.

Explain thrashing.

- In virtual memory system, thrashing is a high page fault scenario. It occurs due to under-allocation of pages required by a process.

- The system becomes extremely slow due to thrashing leading to poor performance.

Explain thread.

- It is an independent flow of control within a process.

- It consists of a context and a sequence of instructions for execution.

What are the advantage of using threads?

The main advantages of using threads are:

a.) No special communication mechanism is required.

b.) Readability and simplicity of program structure increases with threads.

c.) System becomes more efficient with less requirement of system resources.

What are the disadvantages of using threads?

The main disadvantages of using threads are:

- Threads can not be re-used as they exist within a single process.

- They corrupt the address space of their process.

- They need synchronization for concurrent read-write access to memory.

What is a library?

It is a file which contains object code for subroutines and data to be used by the other program.

What are the different types of scheduling algorithms?

The scheduling algorithms decide which processes in the ready queue are to be allocated to the CPU for execution. Scheduling algorithms can be broadly classified on the basis of:

- Preemptive algorithms

- Round Robin Scheduling

- Shortest Job First Scheduling (can be both)

- Priority Scheduling (can be both)

- Non-preemptive algorithms

- First Come First Served Scheduling

Non-Preemptive algorithms: In this type of scheduling once a CPU has been allocated to a process it would not release the CPU till a request for termination or switching to waiting state occurs.

Preemptive algorithms: In this type of scheduling a process maybe interrupted during execution and the CPU maybe allocated to another process.

Why is round robin algorithm considered better than first come first served algorithm?

The first come first served algorithm is the simplest scheduling algorithm known. The processes are assigned to the CPU on the basis of their arrival time in the ready queue. Since, it is non-preemptive once a process is assigned to the CPU, it will run till completion. Since a process takes the CPU till it is executed it is not very good in providing good response times. It can make other important processes wait un-necessarily.

On the other hand, the round robin algorithm works on the concept of time slice or also known as quantum. In this algorithm, every process is given a predefined amount of time to complete the process. In case, a process is not completed in its predefined time then it is assigned to the next process waiting in queue. In this way, a continuous execution of processes is maintained which would not have been possible in case of FCFS algorithm

Explain how a copying garbage collector works. How can it be implemented using semispaces?

The copying garbage collector basically works by going through live objects and copying them into a specific region in the memory. This collector traces through all the live objects one by one. This entire process is performed in a single pass. Any object that is not copied in memory is garbage.

The copying garbage collector can be implemented using semispaces by splitting the heap into two halves. Each half is a contiguous memory region. All the allocations are made from a single half of the heap only. When the specified heap is half full, the collector is immediately invoked and it copies the live objects into the other half of the heap. In this way, the first half of the heap then only contains garbage and eventually is overwritten in the next pass.

How does reference counting manage memory allocated objects? When can it fail to reclaim objects?

Reference counting augments every object with a count of the number of times an object has been referenced. This count is incremented every time a reference to that object is made. Also every time a reference is destroyed the reference is decremented. This process is repeated till the reference count becomes zero. Once the reference count of an object reaches zero the object can be reclaimed. In this way, reference counting systems can perform automatic memory management by keeping a count in every object. Any object that does not have a reference count can be considered to be dead and that memory can be reclaimed.

The reference counting method can fail to reclaim objects in case of cyclic references. There are no concrete ways to avoid this problem and it is always suggested to create an architecture that does not use a circular reference.

What differences are there between a semaphore wait signal and a condition variable wait signal?

Semaphore wait signal:

- They can be used anywhere except in a monitor.

- The wait() function does not always blocks its caller.

- The signal() function increments the semaphore counter and can release a process.

- If the signal() releases a process, the released and the caller both continue.

Condition Variable wait signal:

- It can only be used in monitors.

- The wait() function always blocks its caller.

- The signal() can either release a process or it is lost as if it never occurred.

- On signal() releasing a process either the caller or the released continues but not both at the same time.

For a deadlock to occur what are the necessary conditions

In order for deadlocks to occur there are four necessary conditions:

- Mutual Exclusion: The resources available are not sharable. This implies that the resources used must be mutually exclusive.

- Hold and Wait: Any process requires some resources in order to be executed. In case of insufficient availability of resources a process can take the available resources, hold them and wait for more resources to be available.

- No Preemption: The resources that a process has on hold can only be released by the process itself voluntarily. This resource cannot be preempted by the system.

- Circular Waiting: A special type of waiting in which one process is waiting for the resources held by a second process. The second process is in turn waiting for the resources held by the first process.

Why is the context switch overhead of a user-level threading as compared to the overhead for processes? Explain.

This is due to the reason that a context switch implementation is done by the kernel. During this process the state information is copied between the processor and the PCB (process control block) or the TCB (thread control block). Since the kernel does not know anything about user-level threads, technically it is not possible for it to be a user level thread context switch. The user level scheduler can do some limited state copying on the behalf of a thread prior to the control being handed to that thread. But this copying of state information is smaller compared to that of a kernel-level process. Also the process does not involve going into the kernel mode with the help of a system call.

State the advantages of segmented paging over pure segmentation?

In broad terms paging is a memory management technique that allows a physical address space of a process to be non-contiguous.

Segmented paging has a certain set of advantages over pure segmentation such as:

- Segmented paging does not have any source of external fragmentation.

- Since a segment existence is not restricted to a contiguous memory range it can be easily grown and does not have to adjust into a physical memory medium.

- With segmented paging the addition of an offset and a base is simpler as it is only an append operation instead of it being a full addition operation.

When does the Belady's anomaly occur?

The Belady's anomaly is a situation in which the number of page faults increases when additional physical memory is added to a system. This anomaly arises in some algorithms that implement virtual memory. The virtual memory allows programs larger than the physical memory space to execute. An algorithm suffers from this problem when it cannot guarantee that a page will be kept when a small number of frames are available. An optimal algorithm would not suffer from this problem as it replaces the page not to be used for the longest time. The anomaly occurs when the page replacement algorithm will remove a page that will be needed in the immediate future. An optimal algorithm will not select such a page that will be required immediately. This anomaly is also stated to be unbounded.

What complications does concurrent processing add to an operating system?

There are various complications of concurrent processing such as:

- A time sharing method must be implemented to allow multiple processes to have an access to the system. This will involve the preemption of processes that do not give up CPU on their own i.e. more than one process may be executing kernel code simultaneously.

- The amount of resources that a process can use and the operations that it may perform must be limited. The system resources and the processes must be protected from each other.

- Kernel must be designed to prevent deadlocks between the various processes, i.e. Cyclic waiting or hold and waiting must not occur.

- Effective memory management techniques must be used to better utilize the limited resources.

For any number, we can check whether its ‘i’th bit is 0(OFF) or 1(ON) by bitwise ANDing it with “2^i” (2 raise to i).

1) The left shift and right shift operators should not be used for negative numbers

The left-shift and right-shift operators are equivalent to multiplication and division by 2 respectively

The & operator can be used to quickly check if a number is odd or even

Check if a number is multiple of 9 using bitwise operators

Count strings with consecutive 1’s

Gray to Binary and Binary to Gray conversion

Find the maximum subset XOR of a given set

Given a set, find XOR of the XOR’s of all subsets.

Sum of Bitwise And of all pairs in a given array

Find Next Sparse Number

Find the maximum subarray XOR in a given array

Find XOR of two number without using XOR operator

Write a program to add one to a given number. You are not allowed to use operators like ‘+’, ‘-‘, ‘\*’, ‘/’, ‘++’, ‘–‘ ?

Multiply two integers without using multiplication, division and bitwise operators, and no loops

Check if a number is a power of another number

Check perfect square using addition/subtraction

Count numbers having 0 as a digit

Number of perfect squares between two given numbers

Write an Efficient C Program to Reverse Bits of a Number

Calculate square of a number without using \*, / and pow()

Bitwise operation :

Bit manipulationWrite the logic for setting nth bit.

Write the logic for clearing nth bit.

Write the logic for toggling nth bit.

Write the logic for setting nth to mth bits, where n > m.

Write the logic for clearing nth to mth bits, where n > m.

Write the logic for toggling nth to mth bits, where n > m.

Program for finding number of 1s and 0s in a 32-bit number.

Program for finding whether a number is power of 2 or not.

Program for finding whether a number is even or odd.

Write a function to swap even bits with consecutive odd bits in a number.

Write a function to swap odd bits in a number.

Write a function to swap even bits in a number.

Write a function to find out the number of 1s in a number.

Write a function to check whether the number of 1s present in a number are even or odd.

Write a function for finding the first lowest bit set in a number.

Write a function for finding the higest bit set in a number.

Write a function for reversing the bits in a number.

Write the code for extracting nth to mth bits, where n < m.

Write the code for toggling nth to mth bits, where n < m.

Write the code for setting nth to mth bits, where n < m.

Write the code for clearing nth to mth bits, where n < m

Write a piece of code for sizeof() implementation.

Explain about container\_of() and offsetof() implementations.

How to implement bit-wise operations without using bit-wise operators?

Count Total Set Bits in All Numbers From 1 to N

Find the Maximum of Two Numbers Without Using if-else

Find the Element that Appears Once in arr[] = {10, 1, 10, 3, 10, 1, 1, 2, 3, 3}.

Count Number of Bits to be Flipped to Convert A to B

Find Position of the only Set Bit

Linklist :

Find loop in linked list and remove the loop

implement Stack and Queue using Linked List

Repeatedly Delete N nodes after M nodes of a Linked list

Reverse every k nodes of a linked list

Reverse a Linked List using Recursion

Remove Duplicates from a Linked List

Print Linked List Elements in Reverse order

Merge a linked list into another linked list at alternate positions

Move last node to front in linked list

Swap every two nodes in a linked list

Frequency of a given number in a Linked List

Delete alternate nodes of a Linked List

Rotate linked list by K nodes

Reverse a singly linked list

Write a function to get the intersection point of two Linked Lists (Y Shape)

Write a program to detect loop in a Linked List

Insert nodes into a linked list in a sorted fashion

Write a C program to return the nth node from the end of a linked list

Why Num&sizeMinusOne faster than num&(size-1)

string:

How to Print duplicate characters from String?

How to check if two Strings are anagrams of each other?

How to program to print first non repeated character from String?

how to reverse a string using recursion ?

how to found duplicate character in a string ?

How to convert numeric String to an int?

Remove duplicate character from a string ?

How to reverse words in a sentence without using library method?

implement your own strstr , strcpy, snprintf etc.. ?

Reverse an array without affecting special characters ?

Length of the longest valid substring ?