Linux Device Driver Interview Questions

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

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

Explain about the notifier chains.
Solution
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.
Solution

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

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?

Operating System Interview Questions

Kernel Synchronization

What are the differences between mutex and semaphore?

What is a race condition?

How to avoid a race conditon?

What is a critial 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 inheritence?

Microprocessor and Embedded Hardware Interview Questions

Memory Management Unit and Virtual Memory

What is an MMU and what is it used for?

What is TLB and what is it used for?

What is Virtual Memory?

What are the advantages of Virtual Memory?

Cache

What is cache and where is it physically located?
What is cache coherency?
What is cache hit and cache miss?
There are two basic writing approaches:

- Write-through - Write is done synchronously both to the cache and to the backing store.
- Write-back (or Write-behind) - Initially, writing is done only to the cache. The write to the backing store is postponed until the cache blocks containing the data are about to be modified/replaced by new content.

What is TLB trashing and Cache trashing?

The term is also used when a small set of faster storage space, intended to be used to speed up access to a larger set of slower storage space, is accessed in a way that cancels out any benefits from the faster storage.

An example of this is cache thrashing, where main memory is accessed in a pattern that leads to multiple main memory locations competing for the same cache lines, resulting in excessive cache misses. This is most problematic for caches that have low associativity.

Quite similar is TLB thrashing, where the TLB is overrun by more requests than it can handle efficiently.
In a virtual storage system (an operating system that manages its logical storage or memory in units called pages), thrashing is a condition in which excessive paging operations are taking place. A system that is thrashing can be perceived as either a very slow system or one that has come to a halt.

What is a cache line?

How much amount of data is transferred at a time from main memory into cache?
Check out following link for more information:
http://en.wikipedia.org/wiki/Cache_(computing)
http://en.wikipedia.org/wiki/CPU_cache

What is associative memory or Content-addressable memory (CAM)?

Memory that is addressed by content rather than by address; content addressable is often used synonymously. An Associative memory permits its users to specify part of a pattern or key and retrieve the values associated with that pattern.

Unlike standard computer memory (random access memory or RAM) in which the user supplies a memory address and the RAM returns the data word stored at that address, a CAM is designed such that the user supplies a data word and the CAM searches its entire memory to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found (and in some architectures, it also returns the data word, or other associated pieces of data). Thus, a CAM is the hardware embodiment of what in software terms would be called an associative array.

Content-addressable memory (CAM) is a special type of computer memory used in certain very high speed searching applications. It is also known as associative memory, associative storage, or associative array, although the last term is more often used for a programming data structure.

References:
http://en.wikipedia.org/wiki/Content-addressable_memory

Explain about write through and write back cache.

What is write back read alloc and write alloc?

What is cache buffer?

What is set associative cache?

Explain about cache memory synchronization in a multicore system.

DMA (Direct Memory Access)

What is scatter-gather DMA?

C Programming Interview Questions

Bit manipulation

Write 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.

```
if((n) & (n-1))
```

```
 {n is a power of 2}
```

Program for finding whether a number is even or odd.

```
if((n) & (1))
 {odd}
else
 {even}
```

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);
}
```

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) ;
}
```

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) ;
}
```

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

```
unsigned int num_of_ones(unsigned int num)
{
unsigned int count = 0;
while (num != 0) {
num = (num) & (num-1);
count++;
}
return count;
}
```

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

```
enum {
EVEN,
ODD
} even_odd;

unsigned int check_even_odd_no_of_ones(unsigned int num)
{
```

```
if(num_of_ones(num) & 1)
return ODD;
else
return EVEN;
}
```

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

```
unsigned int first_lowest_bit_set(unsigned int num)
{
unsigned int count = 0;

while(num) {
count++;
if(num&1 == 1)
break;
num = num >> 1;
}

return count;
}
```

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

```
unsigned int first_highest_bit_set(unsigned int num)
{
unsigned int count = 0;

while(num) {
count++;
if(num&(1<<31) == 1)
break;
num = num << 1;
}

return count;
}
```

Write a function for reversing the bits in a number.

```
unsigned int bit_reverse(unsigned int n)
{
 unsigned int m = 0, i;

 for (i = 0; i < 32; i++) {
  m |= (n & 1) << (31-i);
  n >>= 1;
 }

 return m;
}
```

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

```
31                          0
```

```
------------------------
 |        |
 m        n

(num >> n) & ~(~0 << (m-n+1))
```

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

```
e.g.
  4 2
  | |
10101010 - num
11111111 ~0
11111100 (~0<<n)
00011111 (~0>>(31-m))

00011100 (~0<<n) & (~0>>(31-m))    --> mask

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

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

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

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

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

Write a piece of code for sizeof() implementation.

```
#define sizeof(a) ((char *)(&a+1)-(char *)&a)
```

Explain about ffs and ffz implementations in ARM linux.

```
/*
 * On ARMv5 and above those functions can be implemented around
 * the clz instruction for much better code efficiency.
 */

static inline int fls(int x)
{
    int ret;

    if (__builtin_constant_p(x))
       return constant_fls(x);

    asm("clz\t%0, %1" : "=r" (ret) : "r" (x));
    ret = 32 - ret;
    return ret;
}

#define __fls(x) (fls(x) - 1)
#define ffs(x) ({ unsigned long __t = (x); fls(__t & -__t); })
#define __ffs(x) (ffs(x) - 1)
```

```
#define ffz(x) __ffs( ~(x) )
```

Explain about container_of() and offsetof() implementations.

```
/**
 * container_of - cast a member of a structure out to the
containing structure
 * @ptr:        the pointer to the member.
 * @type:       the type of the container struct this is embedded
in.
 * @member:     the name of the member within the struct.
 *
 */
#define container_of(ptr, type, member) ({                      \
    const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
    (type *)( (char *)__mptr - offsetof(type,member) );})


#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
```

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

```
/* a ^ b */
c = 0;
for (x = 0; x <= 15; ++x) {
  c += c;
  if (a < 0) {
    if (b >= 0) {
      c += 1;
    }
  } else if (b < 0) {
    c += 1;
  }
  a += a;
  b += b;
}

/* a & b */
c = 0;
for (x = 0; x <= 15; ++x) {
  c += c;
  if (a < 0) {
    if (b < 0) {
      c += 1;
    }
  }
  a += a;
  b += b;
}
```

```
/* a | b */
c = 0;
for (x = 0; x <= 15; ++x) {
  c += c;
  if (a < 0) {
    c += 1;
  } else if (b < 0) {
    c += 1;
  }
  a += a;
  b += b;
}
```

Data Structures

Write a program for reversing a singly linked list?

```
struct node {
int data;
struct node *next;
}

void reverse(struct node **head)
{
struct node *t0, *t1, *t2 = NULL;

t0 = *head;

while(t0 != NULL) {
t1 = t2;
t2 = t0;
t0 = t0->next;
t2->next = t1;
}

*head = t2;
}
```

Write a program for a singly linked list (insert, delete, count, search etc functions).
Solution
Write a program for a doubly linked list (insert, delete, count, search etc functions).
Solution
Write a program for a circular singly linked list (insert, delete, count, search etc functions).
Solution
Write a program for a circular doubly linked list (insert, delete, count, search etc functions).
Solution
Write a program for binary tree implementation.
Solution

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

Solution

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

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

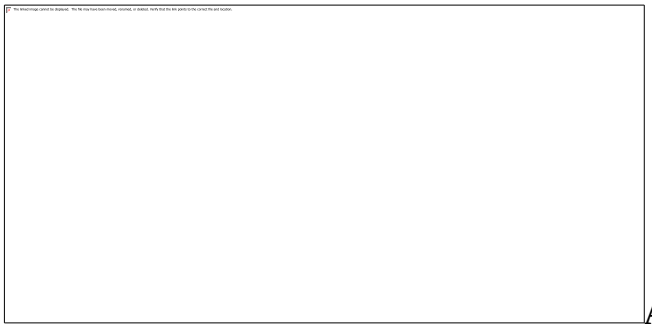
Write a c program for reversing a singly linked list.
Solution

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.
Solution
Write a c program to get the intersection point of two singly linked lists.
Solution 1 Solution 2

Sorting Techniques

Name some of the sorting techniques.
Bubble sort, Quick sort, Merge sort, Insetion sort, Quick sort etc
Write a program for Quick sort implementation.
Solution
Write a program for Bubble sort implementation.
Solution

Question: There are 2 link lists merging at some node. Find the node, where these 2 lists merge.



Answer: We should have one fact in mind that merging node is found by comparing the address (not value) of the nodes. Since we have 2 different lists, one way is to have 2 for loops and compare addresses of all the possible nodes. This solution is O(mn) where 'm' and 'n' are lengths of given lists.

If length of 2 lists are of equal length, we can traverse both the lists one node at a time and find the common node. But since lengths may not be same, we first find a node in bigger list, from where we have length same as that of smaller list. For this, first we need to find the length of both the lists, then move ahead the difference in bigger list and then match nodes in parallel.

Algorithm:

1. Find length (len1) of first link list.
2. Find length (len2) of second link list.

3. find diff = abs(len1 – len2)
4. Traverse the bigger list from the first node till 'diff' nodes. Now we have two lists, which have equal no of nodes.
5. Traverse both the lists in parallel till we come across a common node.

Code:

view plainprint?

```
1.  struct node* findMergeNode(struct node* list1, struct node* list2)
2.  {
3.    int len1, len2, diff;
4.    struct node* head1 = list1;
5.    struct node* head2 = list2;
6.
7.    len1 = getlength(head1);
8.    len2 = getlength(head2);
9.    diff = len1 - len2;
10.
11.           if (len1 < len2)
12.           {
13.              head1 = list2;
14.              head2 = list1;
15.              diff = len2 - len1;
16.           }
17.
18.           for(int i = 0; i < diff; i++)
19.              head1 = head1->next;
20.
21.           while(head1 !=  NULL && head2 != NULL)
22.           {
23.              if(head1 == head2)
24.                 return head1->data;
25.              head1= head1->next;
26.              head2= head2->next;
27.           }
28.
29.           return NULL;
30.         }
```

what is stored at /lib/modules ?	
Answer # 2	It contains all the kernel modules that needed to be loaded into kernel (booting etc). there will some .map, .dep (dependency files) files present. When the kernel needs a feature that is not resident in the kernel, the kernel module daemon kmod[1] execs modprobe to

load the module in.

You can see what modules are already loaded
into the kernel
by running lsmod, which gets its information by
reading the
file /proc/modules

what are the different ways the Linux can swich from User Space to Kernel Space & vice-versa

There are 2 situations when Linux can switch from user Space
to Kernel Space:-

1) by doing System calls
2) When interrupt comes (to handle interrupt)

Linux can switch from kernel Space to User space:-
1) process in kernel mode is preempted.
2) After completion of Interrupt handler / System call

The difference between fork(), vfork(), exec() and clone()

Fork : The fork call basically makes a duplicate of the current process, identical in almost every way (not everything is copied over, for example, resource limits in some implementations but the idea is to create as close a copy as possible).

The new process (child) gets a different process ID (PID) and has the the PID of the old process (parent) as its parent PID (PPID). Because the two processes are now running exactly the same code, they can tell which is which by the return code of fork - the child gets 0, the parent gets the PID of the child. This is all, of course, assuming the fork call works - if not, no child is created and the parent gets an error code.

Vfork : The basic difference between vfork and fork is that when a new process is created with vfork(), the parent process is temporarily suspended, and the child process might borrow the parent's address space. This strange state of affairs continues until the child process either exits, or calls execve(), at which point the parent process continues.

This means that the child process of a vfork() must be careful to avoid unexpectedly modifying variables of the parent process. In particular, the child process must not return from the function containing the vfork() call, and it must not call exit() (if it needs to exit, it should use _exit(); actually, this is also true for the child of a normal fork()).

Exec ：The exec call is a way to basically replace the entire current process with a new program. It loads the program into the current process space and runs it from the entry point. exec() replaces the current process with a the executable pointed by the function. Control never returns to the original program unless there is an exec() error.

Clone ：Clone, as fork, creates a new process. Unlike fork, these calls allow the child process to share parts of its execution context with the calling process, such as the memory space, the table of file descriptors, and the table of signal handlers.

When the child process is created with clone, it executes the function application fn(arg). (This differs from for, where execution continues in the child from the point of the fork call.) The fn argument is a pointer to a function that is called by the child process at the beginning of its

execution. The arg argument is passed to the fn function.

When the fn(arg) function application returns, the child process terminates. The integer returned by fn is the exit code for the child process. The child process may also terminate explicitly by calling exit(2) or after receiving a fatal signal.

what kind of information the linux driver modules (.ko) files has ?	
Answer # 1	kernel 2.6 introduces a new file naming convention: kernel modules now have a .ko extension (in place of the old .o extension) which easily distinguishes them from conventional object files. The reason for this is that they contain an additional .modinfo section that where additional information about the module is kept. Linux program modpost can be used to convert .o files into .ko files.

in one file global variable int i; is declared as static. In another file it is extern int i=100;
Is this valid ?

Re: what is the difference between user APIs and kernel system calls ?	
Answer # 1	Application Programming Interface (API) is a set of functions, objects, protocols or datastructures for the support of application development for developers/programmers. It is actually a kind of function definition which specifies how to make available of a specific service of the system/OS. The API's are available from library or from Operating system itself. Whenever a programmer need a specific service from OS, he/she can use appropriate API to do that. Processes in a system are run in different modes, process run in user mode have no access to the privileged instructions. If they want perform any privileged instructions or need of any services they request

	kernal for that service through System Calls. System calls are made by way of software interrupt. This is actually a request for the service whereas API is a function description that can be used by the programmer for his programs, its like a tool used to obtain a specific task in his programs.

What does exec family return?

Answer # 1	When successful exec will not return, it will start executing the new program However if there is an- error exec returns -1 and sets the errno to the appropriate value

diff b/w spinlock, seamaphores and mutex and where to use it.

What is atomic function / atomic variable ?

Answer # 1	atomic variables are the variables which can only be manipulated atomically using atomic APIs. Linux declares variable as atomic by using the type atomic_t. Basically used a way to achieve synchronization. an atomic operation is one which cannot be (or is not) interrupted by concurrent operations and cannot be broken up into smaller parts that could be performed by different processors. Atomic function is a function which is executed to completion without interruption. Atomic function can also be seen as a small critical section which is executed without interruption, locking.

. Where the History file can be located? 2. How will you harden the server? 3. Diff between Raid 1 and Raid 5? 4. What is the largest disk size can be used in LVM? 5. How will you remove a PV from lvm without any data loss? 6. What is the diff between ext3 and ext2 File system? 7. How we can use resize2fs, what is the purpose? 8. What is the purpose of LVM? Why it is used? 9. If the FS is in read-only mode, so we cannot create any file. How will you fix it? 10. How to create swap partition after OS installation? 11. What is the diff between ssh and telnet? 12. How to find out the dependency required for a package? .Diff between Active and passive FTP? 2.What is anacron? 3.Diff between yum update and yum install while doing for kernel? 4.root_squash and

no_root_squash does what? 5.What are the commands will you execute to find a new hardware? 6.How will you find out a lun allocated from SAN? 7.What is the main diff in CaT5 and CAT6 cable, except the 1/100 and 1/1000? 8.What is stale NFS and How will you fix it? 9.What is kernel panic error? 10.How will you recover password and shadow file, in case both got deleted. Explain the steps? 11.Explain boot process 12.What is network bonding. Explain the steps? 13.What are the exit codes returned by FSCK? 14.What is LDOM? 15.Diff between block and character device?

1. Where the History file can be located?

A. the file .bash_profile stored in home directory of user.
To see that file vi .bash_profile and edit it.

2. How will you harden the server?

A. A Server-- it is weather in testing or production-- are primary targets for
the attackers. By taking the proper steps, you can turn a vulnerable box into
a hardened server.
How to secure SSH sessions, configure firewall rules, minimize software, listed below,
1. Encrypt Data communication
-- use scp, ssh avoid FTP, Telnet and Rlogin /rsh
2. Minimize Software to minimize vulnerability
-- use RPM pkg management / YUM utility to remove unwanted packages installed
3. One Network Service per System or Vm Instance
-- Run different network services on separate servers or vm instance.
For example, if an attacker able to successfully exploit software called
Apache flow, he/she get an access to entire server including other services
such as MYSQL, email server and so on.
4. Keep linux software and Kernel up to date.
-- Use yum update or up2date
some distros apt-get update
5. Security essentials like selinux
6. password authentication like password aging, restricting to user previous
passphrases, and locking user accounts after login failures.
7. Disable unwanted services using chkconfig --list | grep "3:on"

and many more. . .

3. Diff between Raid 1 and Raid 5?

A. RAID 1 is disk striping. no mirroring no parity. Minimum 2 disks required. If any
One disk fails all the data get lost.
RAID 5 is disk striping with parity. Minimum 3 disks required. if anyone disk fails
Data is safe, if two fails data get lost.

4. What is the largest disk size can be used in LVM?

A. Don't know exactly, think of 2TB or 8TB (Warning- Not genuine answer)

5. How will you remove a PV from lvm without any data loss?

A. by using pvremove command.

6. What is the diff between ext3 and ext2 File system?

A. ext3 is also same as the ext2, but journaling concept is introduced in ext3.
Compared to ext2, ext3 is slow. ext2 less secure compared to ext3. ext2 is less
Performance where as ext3 is very good performance.

7. How we can use resize2fs, what is the purpose?

A. resize2fs is only for ext2 filesystem but not ext3.
first unmount the partition
#umount /dev/sda1

#tune2fs -O ^has_journal /dev/sda1 #to remove journal from /dev/sda1

#e2fsck -f /dev/sda1

#resize2fs /dev/sda1 600M #resize the partition

8. What is the purpose of LVM? Why it is used?

9. If the FS is in read-only mode, so we cannot create any
file. How will you fix it?
A. LVM is a mechanism use for providing specality of extending (or) reducing
the sizes of an existing partition.

10. How to create swap partition after OS installation?
A. swap can be created in two ways after the installation,
1. fdisk command
2. create a swap file using dd command

after creating swap file or file system

#mkswap /dev/sda10
#swapon /dev/sda10
#swapon -s #To see the swap devices

by using dd command

#dd if=/dev/zero of=/swap bs=1024 count=1

Which will creates the file size 1024(1GB).

#mkswap /swap
#swapon /swap
#swapon -s #to see the swap devices

11. What is the diff between ssh and telnet?
A. ssh is secured shell, allows the user to login remotely with more secured.
whereas telnet also same but authentications like passwords, transfers over a network
as text mode. so it is not good to use.

12. How to find out the dependency required for a package?
#rpm -qpR filename.rpm

Lists the dependency list of packages.
2.

Port number is already in use .. how to fix for apache..?

Answer # 1

Apache run on port 80. So you can check which service is using this port
ps -aux |grep :80

kill that process which is using port 80 .
pkill process name

Just restart the apache again.

or

YOu can chage the port no of apache in httpd.conf u can use 8080

Why do we have serial and parallel interface, which one was faster and why and when we should go for this interface.

Answer # 1

Both the interfaces have it own importance,
in serial the data is transmitted in sequence i.e is 1 bit
at a time for one clock cycle so to transfer 8 bits of info
you need 8 clock cycles in short it takes more time
comparatively.
where as in parallel 8 bits of transmission is transmitted
is in 1 clock cycle.
parallel transmissions normally used for tarnsmited data
from computer to printer(through LPT port)
parallel transmission is faster than serial transmission

What are the Advantages and disadvantages of script vs compiled program?

Answer # 2

while scripts have the advantages of
1) flexibility to change the of script
2) and being more portable.

compiled executables have the advantages of
1) less memory footprint
2) less execution time.

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

Answer # 1

kill <pid>
allows to kill the normal process.
where as

	kill -9 <pid> to kill normal process and also background process.

what is the difference between socket & port ?	
Answer # 1	port : a particular port number on a host socket: a host and a port like -> http port = 80 -> when you fire up google.com the socket on the server side is google.com:80 this is the server side socket on your local system where you are browsing the socket is yourip:port above 1024

what is the difference between socket & port ?	
Answer # 2	Port : Port no is the s/w address on a computer on the n/w. for instance smtp : 25 and pop : 109 and so on.... Socket : Socket is the communication path to a port. Basically Socket = ip + port, so Socket provides the access to the ip+port

Why bind system call is required in socket programming ? what is it;s Significance ?	
Answer # 1	bind system call assigns a name to the unnamed socket. Binding an address allows a process to register its address with the system. This makes it possible for other process to find it.

Inline expansion is used to eliminate the time overhead(excess time) when a function is called . It is typically used for functions that execute frequently. It also has a space benefit for very small functions, and is an enabling transformation for other optimizations.

Without inline functions, however, the compiler decides which functions to inline. The programmer has little or no control over which functions are inlined and which are not. Giving this degree of control to the programmer allows for the use of application-specific knowledge in choosing which functions to inline.

Comparison with macros

Traditionally, in languages such as C, inline expansion was accomplished at the source level using parameterized macros. Use of true inline functions, as are available in C99, provides several

benefits over this approach:

- In C, macro invocations do not perform type checking, or even check that arguments are well-formed, whereas function calls usually do.
- In C, a macro cannot use the return keyword with the same meaning as a function would do (it would make the function that asked the expansion terminate, rather than the macro). In other words, a macro cannot return anything which is not the result of the last expression invoked inside it.
- Since C macros use mere textual substitution, this may result in unintended side-effects and inefficiency due to re-evaluation of arguments and order of operations.
- Compiler errors within macros are often difficult to understand, because they refer to the expanded code, rather than the code the programmer typed. Thus, debugging information for inlined code is usually more helpful than that of macro-expanded code.
- Many constructs are awkward or impossible to express using macros, or use a significantly different syntax. Inline functions use the same syntax as ordinary functions, and can be inlined and un-inlined at will with ease.

Many compilers can also inline expand some recursive functions; recursive macros are typically illegal.

Bjarne Stroustrup, the designer of C++, likes to emphasize that macros should be avoided wherever possible, and advocates extensive use of inline functions.

Can the value of a constant be modified?

Well, Yes. This is because of the fact that the keyword ‘const’ puts a restriction only on the name of the variable. This means that the value of the variable ‘count’ cannot be changed through its name but through any other way that does not involve the variable name, yes the value can be changed. For example, we can use pointers to achieve the same. Following is a proof of concept for this :

```
#include

int main(void)
{
const int count = 0;
int *ptr = &count;
*ptr = 6;
printf("%d¥n", count);
return 0;
}
```

simple rules about postfix ++, prefix ++ and * (dereference) operators
1) Precedence of prefix ++ and * is same. Associativity of both is right to left.
2) Precedence of postfix ++ is higher than both * and prefix ++. Associativity of postfix ++ is left to right.
++*p -> The expression **++*p** has two operators of same precedence, so compiler looks for assoiativity. Associativity of operators is right to left. Therefore the expression is treated as **++(*p)**.
The expression ***p++** is treated as ***(p++)** as the precedence of postfix ++ is higher than *.
The expression ***++p** has two operators of same precedence, so compiler looks for assoiativity. Associativity of operators is right to left.Therefore the expression is treated as ***(++p)**.

What is a context switch?
a) Kernel switches from executing one process to another.
Pid of init is 1.
What is the default maximum number of processes that can exist in Linux?
32768

Parent process id of a deamon process is_
The process which terminates before the parent process exits becomes
Zombie
What is the use of strace command?
Answer:strace can be used to check the system calls called by the program. So, this can be used for debugging and benchmarking purposes

If one of the thread in multithreaded process is blocked on an I/O, which of the following is true?
a) The entire process with block if their is no kernel supported threads

1. Each process has unique
a) fd table...
b) file table
c) inode table
d) data block table

2. File descriptor table indexes which kernel structure?
a) struct file...
b) strruct fs_struct
c) files_struct
d) struct inode
3 . Switch table is used by
a) device special file....
b) directory file
c) fifo
d) link file.
View Answer

4 . What is the use of fcntl function?
a) locking a file
b) reading the file descriptor flag
c) changing the file status flag
d) all the above....

5. Which function can be used instead of the dup2 to duplicate the file descriptor?
a) read()
b) open()
c) stat()
d) fcntl()
6.
dup2(i,0) -> 7. Which function can be used instead of the dup2 to duplicate the file descriptor?

Which signal is generated when we press control-C?
a) SIGINT
Which signal is generated when we press ctrl-Z?
SIGSTOP
Which of the following signal cannot be handled or ignored and why?
Sigkill.
When real interval timer expires which signal is generated?
SIGALARM
Default action of SIGSEGV is core dumped.

System V IPC common attributes are
a) key
b) id
c) owner
d)all ....

Message queues are created in kernel space and fifo in file system.
Pipe size is 4kb and named pipe is 64kb.
Message queue takes size from kernel and size is 8kb.
With recent kernels (>= 2.6.35), you can change the size of a pipe with

```
fcntl(fd, F_SETPIPE_SZ, size)
```

where `size` is a `long`. The maximum size is in `/proc/sys/fs/pipe-max-size`.

And shared memory on ram so maximum size of shared memory may be 4gb -1 by setting SHMMAX=(4*1024*1024*1024=4GB)in 32 bit system.
The size of the shared memory is only 16KB on most Nvidia GPUs

pmap -d :

```
mapped: 206672K    writeable/private: 4352K    shared: 128K
```

You can see that the shared memory size in pmap is much smaller than top.

To see all shared memory settings, execute:

```
$ ipcs -lm

ulimit -a use to see limits

ulimit -s to change stack size

kmalloc vs vmalloc
```

I've googled around and found most people advocating the use of `kmalloc`, as you're guaranteed to get contiguous physical blocks of memory. However, it also seems as though `kmalloc` can fail if a contiguous **physical** block that you want can't be found.
What are the advantages of having a contiguous block of memory? Specifically, why would I need to have a contiguous **physical** block of memory in a *system call*? Is there any reason I couldn't just use `vmalloc`?
Finally, if I were to allocate memory during the handling of a system call, should I specify `GFP_ATOMIC`? Is a system call executed in an atomic context?

> GFP_ATOMIC
> The allocation is high-priority and does not sleep. This is the flag to use in interrupt handlers, bottom halves and other situations where you cannot sleep.
>
> GFP_KERNEL This is a normal allocation and might block. This is the flag to use in process context code when it is safe to sleep.

Vmalloc provide contagious virtual address but physical address could be differant.
max size of a single kmalloc is up to 4 MB.
Kmalloc is limited in size so for larger memory use vmalloc.

Difference between system calls and Library functions

As a user perspective both system calls and library functions are almost similar. In implementation point of view it's different. For instance the malloc() library function calls the sbrk() system call functions in C program to allocate the memory. If we want to have the better handling of memory functions we can use the sbrk system call function to write our own library functions. Library function lies in user process and system calls lies in kernel, so whenever we use the library function it calls the system function to perform the action. In other words system calls can't be changed but the library functions can be re-written based on our convenience. The other major distinction is system calls lies in kernel level and library functions lies in user level. So whenever we invoke the library functions, this will internally call the functions which reside in the kernel level.

Three main functions used for the control of process are

- **Fork** – used to create new process, which returns two values, one it will return process id of newly created process to it s parent and zero to the child process and the child process can get the parent process id at any time.
- **Exec** - used to replace the newly created process with new program And
- **waitpid** – parent process has to wait , until the child process completes its execution. If parent process is not waiting for its child process, then the parent process will become an orphan process and similarly if parent is not taking care of the child process which has finished its execution then the child process will become zombie process

Qualcoum interview question.

1.As kernel can access user space memory, why should copy_from_user is needed?

copy_from_user does not page fault unlike other kernel or user code sections when an attempt happens to access memory addresses passed by userspace. This allows it to gracefully handle exception instead of kernel panic.

If kernel directly accesses the user data structure, system will panic if it's not a valid address(eg NULL pointer). To avoid this situation, copy_from_user(..) is used. This function will fail, if proper user address is not provided and does not bring down the system.

2.what are the best synchronization techniques used in linux kernel?

for simple counter variables or for bitwise ------->atomic operations are best methods.
atomic_t count=ATOMIC_INIT(0); or atomic_set(&count,0);
atomic_read(&count);
atomic_inc(&count);
atomic_dec(&count);
atomic_add(&count,10);
atomic_sub(&count,10);

spinlocks are used to hold critical section for short time and can use from interrupt context and locks can not sleep,also called busy wait loops.
fully spinlocks and reader/writer spin locks are available.
spinlock_t my_spinlock;
spin_lock_init(&my_spinlock);
spin_lock(&my_spinlock);

// critical section

spin_unlock(&my_spinlock);

Spinlock variant with local CPU interrupt disable
spin_lock_irqsave(&my_spinlock, flags);

// critical section

spin_unlock_irqrestore(&my_spinlock, flags);
if your kernel thread shares data with a bottom half,
spin_lock_bh(&my_spinlock);

// critical section

spin_unlock_bh(&my_spinlock);

if you have more readers than writers for your shared resource
Reader/writer spinlock can be used
rwlock_t my_rwlock;

rwlock_init(&my_rwlock);

write_lock(&my_rwlock);

// critical section -- can read and write

write_unlock(&my_rwlock);

read_lock(&my_rwlock);

// critical section -- can read only

read_unlock(&my_rwlock);

mutexs are used when you hold lock for longer time and if you use from process context.
DEFINE_MUTEX(my_mutex);
mutex_lock(&my_mutex);
mutex_unlock(&my_mutex);

the 2 main ones are spinlock (looping until we get the resource) and mutex (sleeping untill we get the resource)

but there are implicit ways too:
- putting the synchronization code in interrupt and calling it
- putting the synchronization code in kernel and calling it
- using a queue or pipes (which are themselves syncronized)

3.how function pointers are shared across different processes? using which iPCs?

two processes can not share function pointers.

Both process has a different virtual address space. If you send address of a function from process A to process B, in process B address from A does not make any sense. The memory mapping in process A will be different from that in process B. If you want to do so u need to know the memory mapping of process A and according to that you need to deference the memory in process B.
if you want to use functions in two processes make library for that functions

and use that library in your processes.

4.is linux kernel a process / thread/ something else?

Linux Kernel is a passive component of the OS. It does not execute, neither it is a process/thread. It itself has many subsystem and could be called with system cal API/Interrupt that helps in executing the user space process in system space for more privileged access, either to I/O or any subsystem.

A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your computer. It is responsible for interfacing all of your applications that are running in "user mode" down to the physical hardware, and allowing processes, known as servers, to get information from each other using inter-process communication (IPC).

5.write a program with 2 threads. one thread should print even and other should print odd numbers in sequence. how would you make it SMP safe?

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>

int glob = 0;
sem_t mutex1;
sem_t mutex2;

void function1(void)
{
static int count1 = 0;
while(count1 < 100){
sem_wait(&mutex1);
glob ++;
printf("%d \n", glob);
sem_post(&mutex2);
count1++;
}
}


void function2(void)
{
static int count1 = 0;
while(count1 < 100){
sem_wait(&mutex2);
glob ++;
printf("%d \n", glob);
sem_post(&mutex1);
count1++;
}
}


main()
{
int rc1, rc2;
```

```
pthread_t thread1, thread2;


sem_init(&mutex1, 0, 1);

sem_init(&mutex2, 0, 0);

/* Create independent threads each of which will execute functionC */

if( (rc1=pthread_create( &thread1, NULL, &function1, NULL)) )
{
printf("Thread creation failed: %d\n", rc1);
}

if( (rc2=pthread_create( &thread2, NULL, &function2, NULL)) )
{
printf("Thread creation failed: %d\n", rc2);
}

/* Wait till threads are complete before main continues. Unless we */
/* wait we run the risk of executing an exit which will terminate */
/* the process and all threads before the threads have completed. */

pthread_join( thread1, NULL);
pthread_join( thread2, NULL);

exit(0);
}
```

7.how a function from one user process can be called in other user process?

shared memory

8. *1. explain device tree concepts in linux.*

Device tree is a data structure that describes the hardware and is passed to the kernel at boot time.

In order to remove same code across different board the device trees are used. These are structure that define the available devices on a platform. These are created from a board definationts file to a binary file which is loaded in memory by the bootloader. The kernel populates the device tree structure based on this binary. Thus different boards can be supported without recompiling the kernel

just explain sysfs that is enough.

9.what is diffarence between semop and semctl?
we can change the semaphores using both apis?what is the diffarence?

To use semaphore for locking and unlocking semop api will be used. To control(delete, info, etc) semaphore semctl api will be used.

10.how to manage memory in linux kernel?

linux kernel divides memory into two are ways.
1.physical memory management.
2.virtual memory management.
buddy system manages physical memory
virtual memory is managed by process address space.

11.what is bus error? common causes of bus errors?

The first thing that needs to be addressed is: What is a bus? A bus is a communication unit that allows the CPU to interact with peripherals, there are different type of buses such as PCI, I2C, MDIO, Memory Buses, etc. Normally each bus would have its own protocol for transmitting data across devices, for example in the case of PCI you can have timeout errors or windows errors (data is directed to unknown addresses/devices). In memory, bus errors would refer to alignment but other errors could be attributed to physical HW problems such as faulty connections. Other type of bus errors could be single and multiple bit errors, this could be addressed by using ECC memory.

Adding to the above, bus error can also be generated while accessing some register memory whose access is disabled. For example AHB-lite (Advanced High performance Bus) feature for arm

12.what is memory leak ?

When Dynamically allocated memory is not freed after its use, it leads to memory leak.
It is a very sensitive issue in embedded programming

*Char *ptr1= malloc(sizeof(5));*

*Char *ptr2= malloc(sizeof(6));*

ptr1=ptr2;
free (ptr1); / now it frees the ptr2 allocation */*
here the 5 allocated is memory leak which can be freed as we dont have the pointer of that memory location.

13.which process / directory is responsible for the kernel is decompresed during boot up ?

/boot/vmlinuz

or

The decompressing of kernel may depend on architecture. For example in x86, the bootloader usually does the decompressing of kernel, whereas in arm architecture, there are several assembly scripts (like head.S, start.S etc) causes calling to decompress_kernel() (arch/arm/boot/compressed/misc.c) function which in turn calls do_decompress(), which is specific to the machine type.

14.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.

In kernel context, we should loop (although not recommended) through task_struct list and check whose pid matches and then check the flags of it. If its CLONE_VM | CLONE_FS | CLONE_FILE | CLONE_SIGHNDL - it means these resources are shared, so that is a thread. Else, if its only SIGCHLD, its a process.

check in task_struct:: if pid and tgid are same then its a process else its a thread.

15.What is the difference between kill-6 and kill -9

SIGKILL 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.

16.how we can modify the default behaviour of setsockopt() parameters

by preloading a .so with modifed setsockopt()

17.describe the steps to add a new global sysctl() or fnctl() paramater

sysctl is used to modify kernel parameters at runtime. The parameters are listed under /proc/sys/. Procfs is required for sysctl support in Linux. You can use sysctl to both read and write sysctl data. You must login as the root user to use any one of the following command.
Method # 1: Setting value via procfs
You can use standard echo command to write data to variables:
 # echo "value" > /proc/sys/location/variable
Method # 2: Temporary on the command line
Use sysctl command with -w option when you want to change a sysctl setting:
sysctl -w variable=value
Method # 3: Configuration file /etc/sysctl.conf
This is recommended way. First open /etc/sysctl.conf file
 # vi /etc/sysctl.conf
Now add value:
variable = value
Close and save the changes. Type the following command to load sysctl settings from the file /etc/sysctl.conf file:
 # sysctl -p

Set the socket to non-blocking mode using:- fcntl(sockfd, F_SETFL, flags | O_NONBLOCK)
To unset the O_NONBLOCK flag:- fcntl(sockfd, F_SETFL, fcntl(sockfd, F_GETFL) & ~O_NONBLOCK);

18.How can make Mutex global to All CPUs in your Board? Because spinlock is global to all CPUs in SMP system. Just make Mutex work like spinlock?

Well a mutex may sleep and spinlock cannot, so u need to first do some magic to make a process trying to acquire a mutex not sleep and spin

Inside spinlock implementation, first it disables kernel preemption only when ur kernel built with kernel preemption. (I Read).

19.

What are upper half and bottom half in device drivers.Why are they used?

upper half and lower half are the terms related to interrupt handler and hardware devices only via drivers raise interrupts in this context , they asked u in form of device driver. Each device has to register its interrupt handler (which is the part of device driver. Now for devices which raise interrupt frequently or in case when same interrupt handler is used via devices so to increase the service performance driver designers design the handlers in such a way that whenever an interrupt occurs the OS does the most important part of handler "upper half" to respond to interrupt,create a data structure containing device specific data called "lower half" for later processing when CPU becomes available. This way interrupt handlers can be used in case when the interrupt raise frequency is high.

In real time most of work is done in upperhalf for deterministic behaviour and in gpos this was dome in bottom half . upperhalf are written in ISR and bottom half is via tasklets or kernel thread,

workques .

hardware devices interact with kernel through sending signals to kernel.this is called interrupt.after receiving interrupt kernel must do corresponding actions by interrupt handlers called interrupt service routines.(ISR).
if ISR not preemtable and interrupts on the current processor or on all processors are disabled. now if ISR takes longer time to finish its work,system's interactive response will be degraded.
So to improve this ,total work is divided into two parts/halves.
1.top half-interrupt handler
2.bottom half-softirq,tasklet,workqueue.

In the top half , ISR will do most important time critical work like receiving data from network device or sending data to network device .
In the bottom half ,remaining data processing work like passing data to upper layers of newnetwork protocols by one of these like softirqs or tasklets or workqueues.
for most device drivers tasklets are good choice.
for very very time critical works, like network data or blocl data softirqs are used.
in the process context workqueues are used.

20.Differenced between fork(),pthread() system calls.What is vfork().What is a file system how does kernel stores the data structure with respect to file system,he hinted to super blocks I was not much aware of super blocks etc in a file system,then I attempted for VFS kind of thing.

my opinion regarding the first qstn:
fork() - it does call clone() inside it but with the flag as SIGCHLD i.e. clone(SIGCHLD, 0). So, none of the resources are shared. But, copied to child on COW basis.

pthread() - is a POSIX wrapper for clone() system call to create a thread, in this case clone is called like clone(CLONE_VM | CLONE_FS | CLONE_FILE | CLONE_SIGHNDL). So, address space, open files, file system handlers, signal handlers are shared.

vfork() - is also calling clone() similar to fork(), only difference is process address space is not copied to child and execution of parent is suspended till the execution of child. They used it before COW was introduced.

Interesting thing is all of them are calling clone() because linux kernel actually does not differentiate between process and thread except the flags - all are tasks.

22.What does the following command do in unix:
>ls a/b/c/./e/../f/./../g/./../h
Consider a,b,c....g,h to be folders?

List of a/b/c/h
the directory tree is
a
|-b
...|-c
.....|-e
.....|-f
.....|-g
.....|-h

23.Command to sort the file contents in Linux?

ls -al | sort -r -n -k2

here k2 its signifies the column of sorting .

24.Content Based Search (Have a file with a word say "Mumbai". Do not remember the File Name or the File Path. Search for the string)?

fgrep "mumbai" FILE path

*grep -ir <search_string> **

25.Write code for page fault handler in Linux (I have a project on this so may be I was asked)?

I don't think, it is about invalid reference. It is about Page table entry. If process is trying to write in text segment, That's protection fault, Kernel sends SIGSEGV to process. If it just need a valid page, which entry is not present in Page table, that's Page fault.

26.Say there is an operation like a=b; Its the statement for which you want to avoid concurrency. (Say SMP or preemption or what ever)
With out using locks(spin,semaphore,mutex etc. etc.) how would you make this statement protected?

Ans: atomic operations (atomic assign etc.)

disable all interrutp when this statement get executed in case of uniprocessor.

27.

How would you debug kernel code.?

printks ! Simple and very convinient. KDB and kernel probes can also be used.

Or

kernel code can be debugged various methods some of them
1) UML (User mode linux) - Best method but lacks Device Driver support
2) KGDB (Kernel gnu debugger.. via serial,network, also we can use vmware etc)... Best needs two systems
3) kdb (Builtin kernel debugger)... Source code debugger is not possible
4) kdump tools (for Kerenel core dump etc) similar like UM

28.How would you do benchmarking (compare the performance) in a device driver code? Apart from timing or time is there any other standard way? He basically meant comparing Programmed I/O and DMA. (Leave security etc. only performance comparison)?

we can use the struct timeval and timespec timers inside the printk to see the transfer time of bulk data using the I/O map or DMA transfer

or jiffies

29.How would you handle sleeping or blocking instructions in an Interrupt Service Routine(if unavoidable) or basically if the length of ISR is long?

Ans: Tasklets and Workqueues. PLEASE LET ME KNOW IF THERE ARE ANY OTHER.

SoftIRQs and tasklets cannot use any blocking instructions as they run in an interrupt context. Workqueues can only be used from a process context. So, I guess if the ISR is long, it can wake up a kernel thread, and let the kernel thread do the dirty work i.e. use blocking instructions.

Or

Sleep or blocking can not be allowed in ISR(Interrupt context), Tasklets(soft interrupt context) But allowed in WorkQueues(kernel context).

30.What is virtual memory ? What is the pre-requisite in hardware for supporting virtual memory ?

The pre-requisite is that the CPU should have a MMU to support virtual to physical address translation.

31.

System call? What happens in low level.

System calls are for kernel service.
Using 0x80 INT, it jumps to kernel context.
For this u shd have asmlinkage pre-knowledge is required.
While implementation, u need to increment macro count from particular header file then define ur call and u need to put in corresponds...location

32.

Signal handling mechanism.

Process PA can send signal to Process PB. But it PA can get that signal from kernel only.
For that kernel developers kept some mechanism inside like

"find_task_by_pid" call needs PID of process to which signal to be sent.
Then this call return task_struct instance.
"send_sig_info" call needs the aboce task_struct and signal name.
The above these two methods implemented for signal handling mechanism.

33.Entry points into kernel?

System Call, Synchronous Exceptions(Divided by Zero..etc)also called as traps, Asynchronous Exceptions(Interuprts).

Or

Procfs , Kernel module , Interrupts

34.How do you debug?

Gdb.

35.How would sleep in a kernel?

By making a call to function :
*thread_switch(S_SLEEP /*State*/, wchan *wc /*wait channel*/)*

This would put curthread inside the kernel to sleep by putting it on sleep queue and changing its state from runnable to sleep.

or

wait_queue_timeout are best way to to sleep inside kernel

or

why do we need sleep in the kernel?
Kernel is like a machine which run everything , if kernel sleep , other thread won't get a chance to run , so the whole system will stop , unless you want to process interrupt.

I guess if in kernel you want this thread to sleep, you can do a context switch to let the current thread sleep, then you can let other thread run.

if you really want to sleep in kernel, you can disable interrupt, so that the scheduler won't have a chance to work , you will be able to sleep as long as you want with a while loop.

36.Difference between semaphore and mutex.?

An ownership kind of thing is associated with a mutex. The thread which takes a lock,can release a lock.It is used to mutual exclusion on the same block.

Semaphore on the other hand, acts as a signaling mechanism. A thread can enter into critical section after receiving signal from some other thread.
In short, a lock can be released by some other thread apart from the thread which has acquired it.

37.Mutex:

Is a key to a toilet. One person can have the key - occupy the toilet - at the time. When finished, the person gives (frees) the key to the next person in the queue.

Officially: "Mutexes are typically used to serialise access to a section of re-entrant code that cannot be executed concurrently by more than one thread. A mutex object only allows one thread into a controlled section, forcing other threads which attempt to gain access to that section to wait until the first thread has exited from that section."
Ref: Symbian Developer Library

(A mutex is really a semaphore with value 1.)

Semaphore:

Is the number of free identical toilet keys. Example, say we have four toilets with identical locks and keys. The semaphore count - the count of keys - is set to 4 at beginning (all four toilets are free), then the count value is decremented as people are coming in. If all toilets are full, ie. there are no free keys left, the semaphore count is 0. Now, when eq. one person leaves the toilet, semaphore is increased to 1 (one free key), and given to the next person in the queue.

Officially: "A semaphore restricts the number of simultaneous users of a shared resource up to a maximum number. Threads can request access to the resource (decrementing the semaphore), and can signal that they have finished using the resource (incrementing the semaphore)."
Ref: Symbian Developer Library

39.Write a pseudo code for page fault handler.?

40.What is an ISR? What are the basic operations there?

An interrupt handler, also known as an interrupt service routine (ISR), is a callback subroutine in an operating system or device driver whose execution is triggered by the reception of an interrupt

Step in executing an Interrupt:

1) It finish the instruction it is executing and saves the address of the next instruction (PC) on the stack.

2) It also saves the current status of all the interrupt internally.

3) It Jumps to a fixed location in memory called the interrupt vector table that holds the address of the interrupt service routine.

4) The microcontroller gets the address of the ISR from the interrupt vector and jumps to it. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine.

5) Upon executing the RETI instruction ,the microcontroller returns to the Place where it was interrupt.

Or

x86 intepretation
1 - interrupt occurs (from PCI, keyboard controller, whatever...)
2 - save current context (unless interupt is masked, then do nothing end of story)
3.- Lookup address in interrupt descriptor table, jump to it.
4 - do stuff...
5 - Clear interrupt (maybe a legacy thing)
6 – return

42.Explain about what you cannot do in a ISR.

ou can't use sleeping functions inside irq handler .
Write less instruction in handle

43.Various locking mechanisms?

mutex , semaphore, spin lock

44.Write code for any pointer function.

45.Macros vs Functions?

46.code to find out the fist set MSB bit.

47.Find the angle between min and hour hands of a clock.

```
int angle( int hr, int min)
{
return abs ( min*6 - (hr*30 + min*0.5) )
}
```

48.How do you improve performance of kernel code.

By seting kernel variable.

49.Given a matrix (2D array). How would you rotate it 180 degrees? Clue: Swapping rows with columns rotates it by 90 degrees.

```
for(i=0;i<row_total;i++)
{
  for(j=0;j<column_total;j++)
  {
    c[i][j] = a[i][j];
  }
}
for(i=0;i<row_total;i++)
{
  for(j=0;j<column_total;j++)
  {
    a[i][j] = c[row_total-1-i][colum_total-1-j];
  }
}
```

Guys , we are talking about 180 , not 90
say :
a:
0,1,2,3
4,5,6,7

after switching180, it would be:
3,2,1,0
7,6,5,4
so it is basically just a reverse of ever row.

50.Why pass double pointers to functions? How would you declare the prototype of a function that takes double pointer?

51.How can one change the value of the Kernel Variables during the runtime?

Depends on what 'variables' the interviewer refers to. In the case of devices, they can be accessed through the /sysfs Virtual file system. With respect to processes, certain parameters can be changed through /prof VFS.

use sysctl -a to see available parameters.

Or

Sysfs. procfs, configfs, debugfs

52.Unix script to find the specific string from the given file?

Grep

53.how to find a child process in unix?

Pstree.

54.What is cache ? How it is used and mapped the physical address cache and virtual address cache ?

Cache is a component that improves performance by transparently storing data such that future requests for that data can be served faster.

The data that is stored within a cache might be values that have been computed earlier or duplicates of original values that are stored elsewhere. If requested data is contained in the cache (cache hit), this request can be served by simply reading the cache, which is comparably faster. Otherwise (cache miss), the data has to be recomputed or fetched from its original storage location, which is comparably slower. Hence, the more requests can be served from the cache the better the overall system performance is.

Mapping between physical address and cache:-

Main Memory Cache Memory
---------- |Index Data | ---------- ---------- |Index Tag Data | |0 xyx |<------- ---------- ---------- | |-----|0 2 pas | |1 sdp | | | ---------- ---------- |---|-----|1 0 xyx | |2 pas |<----------| ----------

---
|3 eqw |

| |

55.User application will block on IOCTL, in kernel untill interrupt arrives this IOCTL will be blocked. When interrupt comes IOCTL should retun. Desing this mechanism.

he is simply telling you to implement a some mechanism such as waitqueues

56.What is IOCTL, how it is used in user and kernel driver code ?

IOCTL is system call where from user space u can call like:
ioctl(fd, <CMD>, <Addr of buffer>)
But u need to define IOCTL call for that particular FD in kernel.

Or

ioctl exposes mechanism to implement device specific system calls. Such as if you want to implement speed change of motor, you can implement it in ioctl call with specific flag as switch. Same flag is used by application for making ioctl system call to request for motor speed change.

best example is selection of wifi connection on your laptop , it internally uses ioctl.

58.What is difference between wake_up ( ) and wake_up_interruptible ( ) APIs in linux kernel ? When should should use which one, how it should be decided ?

wake_up - wakes exactly one exclusive sleeping process in TASK_INTERRUPTIBLE or TASK_UNINTERRUPTIBLE state from the wait queue
wake_up_interruptible - wakes only one exclusive sleeping process in TASK_INTERRUPTIBLE from the wait queue

Wake_up_interruptible might be used in semaphore which wakes up only one process waiting for the signal from the wait queue. If you need to wake up only processes which are waiting on an event, then wake_up_interruptible must be used.
To generally wake up all the processes to some activity such as device I/O then wake_up has to be used.
I might be wrong, but this what I understand.

Or

wake_up_interruptible() can wake only those processes which are put to sleep using wait_event_interruptible(), i.e. the processes which are put to sleep with its state as TASK_INTERRUPTIBLE.

wake_up() wakes up all the processes which is waiting for that event.

59.what is difference between sleep_on ( ) and interruptible_sleep_on ( ) APIs in linux kernel ? When should should use which one, how it should be decided ?

The interruptible_sleep_on( ) function is identical to sleep_on( ) , except that
it sets the state of the current process P to TASK_INTERRUPTIBLE instead of TASK_UNINTERRUPTIBLE so that P can also be awakened by receiving a signal.

60.Differentiate between threads and processes
What are spin locks, are they better then mutex ?

Spin lock is hardly even better than mutex, since its mostly implemented as a while loop with condition check. Now two processes or threads can check the same condition together and enter, which will be a race condition. Obviously lot of waste of CPU cycles.
Mutex however puts a process which was unable to acquire a lock put to a queue and wake one up upon unlocking

or

Spinlocks are efficient if threads are only likely to be blocked for a short period of time, as they avoid overhead from operating system process re-scheduling or context switching. For this reason, spinlocks are often used inside operating system kernels. However, spinlocks become wasteful if held for longer durations, both preventing other threads from running and requiring re-schedulin

61.How to write a perticular value on real hardware register or hardware address in device driver programming ?

It all depends on how the real hardware is connected to the device driver module (like I2C bus etc). If the device driver can access the hardware register (like a I/O mapped memory) we have to type cast the memory of the register address to the pointer of the data type it stores then write to that pointer address. Ex:
**(uint16 *)0x1234 = (uint16) dat.*

Or

outb(base_address of register,data).

62.Given a url how do u download page from server. He was looking for some command to do that in windows when I mentioned u can use wget in linux. How do u get all pages of that url. How do u eliminate cycles and same pages?

telnet <host> 80
GET <URL>

63.How SpinkLocks work in SMP and UP architectures.

Its more about how spin locking work in terms of disabling and re-enabling interrupts..
On SMP, spin_lock_irqsave() and spin_unlock_irqrestore is used.. it disables and re-enables the LOCAL interrupts..i.e. interrupts on the same CPU on which lock is help...so
other CPUs can continue serving the interrupts on them..

64.Asked me what happens when a interrupt is received,told the standard answer but he asked me what is contained inside an interrupt handler

May be he expects one of following things:
1. Interrupt code shouldn't have any lock or other things which may delay/block execution of ISR.
2. Basically ISR could be either a kernel function (in case of system call) OR a function in the device driver code.
3. Minimum amount of work will be done immediately and most of the work will be assigned to the kernel subsystem such as different queues.
4. The ISR must save the context and restore it again.

65.Replace a word "Old" with word "new" in all the files of the directory. Solution can be a linux script also?

*sed -i 's/Old/new/g' *.txt*

or

*perl -pi -e 's/Old/new/g;' *.txt*

66.In a particular directory, list the 15 recently modified files. ls -ltr | tail -15.?

find / -type f -mtime -15 > output.txt
This command will find all the files that have been modified in the last 15 days and will store it in output.txt

*67.Can Linux be compromised? How secure is Linux? How would you compromise a *NIX system?*

68.In a given text file, replace all instances of "a savings of X%" with "(X% off)", where 'X' represents an actual percentage

```
sed /s/″a savings of X%″/″a savings of X% off″/g
```
or

sed 's/a savings of (\d+)%/a savings of \1% off/g'

69.How do you debug a process which is consuming 90% of CPU?

do a kill -6 to get a core dump and debug the core

or

kill -6 is one solution as one user mentioned

Set conditional breakpoints by having counters at multiple places in the suspected process. Based on one counters hitting a certain count you can break and look into the possible buggy code area

You can have throttled printf mechanism based on the above counter logic

A normal process executing in user mode cannot consume 90% of CPU as the process will get switched out after it consumes its time share slot.

If the process is a Real Time process and has a bug in its code, inspite of getting preempted after its timeshare, it could get the cpu right back if it is the highest priority process. So, it may show 90% accumulated cpu usage over time, but cannot consume 90% in one continuous grab of cpu time.

The only way for the cpu to be 90% consumed by one process is some sort of kernel problem. Usually, there is some sort of an infinite loop OR a very long running process that is not preempting and giving up the CPU.

One way to debug such problems is to use kernel tracers. For example, refer to ktracer on HPUX. This tool traces kernel procedure calls, arguments and timing. So, if there is a loop and a particular stack trace is being repeatedly executed leading to a loop, we can see it in ktracer output.

A similar tool is Solaris Dtrace

70.-what are fork exec,IPC.

fork creates a new process, which is a copy of the process which called fork. exec replaces the current process with the executable given as parameter. For fork/exec combination is the only way to start a different executable.

IPC = inter-process communication, the OS provides pipes, shared memory, even sockets

or

when is fork called then it is create a new process and exec is replaced from one process to another but another process has same PID.

and IPC is a mechanism which used to synchronization and communication between related and unrelated process(Shared memory,message queue,)

71.how can you periodically print the cpu usage of a given process id in Linux?

We can use the top command by using -p option.
first grep by using ps -ef | grep "processname" and then use the pid.
or
top -p pid

or

cat /proc/cpuinfo

72.After fork, does new process get file handles and locks?

No, file locks set by the parent process shall not be inherited by the child process. Also the child process shall not inherit any address space memory locks established by the parent process via calls to mlockall() or mlock().

Or

The child process has its own copy of the file decriptors for the parent process. Each of the file descriptors for a child process refer to the same open file description with the corresponding file descriptor of the parent process.
The child process does not inherit file locks set by the parent process.

73.In unix, how do you kill all child processes of a particular parent?

pkill -P <PPID>

74.In unix, what is a defunct process ?

Its a zombie. A child process that is dead but its entry in the process table hasnt been removed so that the parent process can get the exit status information of this child.
Use ps v -u username and check under STAT column with Z value.

75.Can u have a floating point operation inside a interrupt handler?

He asked me this question after he asked me whether there can be a print message inside a interrupt handler.

Well.. first of all, why would kernel need a floating point operation.
Secondly, if it ever requires, I don't see any hurdles in it specially if those operations are trivial and non-blocking.
Why would interrupt handler get blocked? Its just that it has to handle all possible errors/exceptions including unhandled one, and probably take care to save/restore FPU context.
The best alternative would be using 'software fp operations' through compilers - if one is not sure about restoring FPU state.

Or

The ISR code does not run in the normal task context. It has no task control block and all ISR's share a single stack. Because of these differences there are restrictions to the type of routines that can be used in the ISR.

ISR's should not invoke functions which may cause ``blocking'' of the caller. For example, semTake. malloc and free cannot be used because they call functions which may cause blocking and thus all

creation and deletion functions are forbidden since they use malloc and free. An ISR must not perform I/O through the VxWorks I/O system. A call to a device driver may block the system if the caller needs to wait for the device. However, the VxWorks pipe driver has been designed to permit writes by interrupt service code.

The best way to print out messages from an ISR is to use the function logMsg or other functions provided by the library logLib. ISRs should not use floating point instructions since these registers are not saved on entry to the ISR. If floating point instructions are to be used the registers must be saved using the functions in fppALib. However, floating point operations are time intensive and should be avoided in ISRs.

1. *DMA controller.*
2. Cache coherency.- MESI /MSI protocol
3. Cache coherency mechanism.
4. Interrupt handler.
5. what happens when function1 calls function2 with it.(like where does the linkage register stuff get stored..and resume execution)
6.Can u have reentrant code inside interrupt handler. (NO)
7.What will happen/can u have printf/printk inside an interrrupt hancler (i think he wanted me to say no.. but I did not know the reason)
8.context switch.. when do u need it.
9.what does a interrupt handler take in as input... and what does it return.(it does not accept or return anything)

10. what is the difference between ISR and interrupt handler.(Both are the same)

11.how to search a book in one million books.

12.How to check whether a linked list is circular.

1. *DMA controller.*
Its a piece of intelligent hardware that helps transfer of chunk of data from/to a device/memory. This allows the CPU to do other job during the transfer. This is the theoretical answer one can give but in reality there can be many specifics associated with a DMA controller and associated transfer.

2. Cache coherency.- MESI /MSI protocol
MESI - Cache Coherency protocol used in x86 processors. M- Modified, E - Exclusive, S - Shared , I - Invalid
MSI - Message Signalled Interrupt - Again specific to x86 (latest processors). Doing a mem write in a specific memory mapped region, cause a x86 interrupt

3. Cache coherency mechanism.
MESI protocol is used to maitain cache coherency between processors and memory controllers

4. Interrupt handler.
A function registered to service a interrupt of a device

5. what happens when function1 calls function2 with it.(like where does the linkage register stuff get stored..and resume execution)
This is again processor specific stuff. On a x86 processor, its based on "ebp" register which holds the linkage across stack frames. The return address of the function will be the next address to "ebp"

in the current stack frame.
In other processors the concept remains same but the designated registers differ. This is defined in ABI (Architecture Binary Interface) of the processor

6.Can u have reentrant code inside interrupt handler. (NO)
Yes. Interrupt code should be re-entrant coz the interrupt can be nested. This can be achieved using spin_lock_irqsave if you access global variables within the handler

7.What will happen/can u have printf/printk inside an interrrupt hancler (i think he wanted me to say no.. but I did not know the reason)
You can have printks in interrupt handlers but you will loose interrupts and the purpose of debugging is lost.

8.context switch.. when do u need it.
When a current executable moves from "running" state to any state other than running like pending, sleeping state, we need a context switch to run the other thread/process/executable which is in ready-to-run state

9.what does a interrupt handler take in as input... and what does it return.(it does not accept or return anything)
Interrupt handler does not return anything

10. what is the difference between ISR and interrupt handler.(Both are the same)
Both are same

11.how to search a book in one million books.
Its more of one's opinion rather than a one specific answer

12.How to check whether a linked list is circular.
As others mentioned, have two pointers moving faster and one pointer moving one after other. If the slow pointer meets any of the faster pointer, then the list is circular. If the list is not circular, either the fast or slow pointer will hit a NULL and we can use that as a condition to exit the function

1. *which kernel were u working? debugging techniques? challenges?*

2. a mad user tries to allocate 1 gb memory using calloc.
but the program fails after allocationg about 800mg(appx. i dont remember). Tell me what could have gone wrong?

3.
We know disabling interrupts works only if it is single processor(i.e local disabling of interrupts).

Consider this case where we have a SMP(symmetric multi proc) the processor. Processor-1 wants to perform some critical operation so it disables all the interrupts.

What will happen when processor-2 throws an interrupt.

calloc beasically puts zeros when allocation memory. The program crashes because it tries to pu in 0s to all the page table entries.

So the page table is 4096 bytes(somthing like that).. so wehn all the pages are filled that when the program crashes... It does not crash after allocatiing all the 1gb.

3. Answer: P1 will be interrrupted. So the critical operation might have a problem.

Basically when critical operation needs to be done.. have spin locks.. which exist over both p1 and p2.

but again using spin locks has to be done carefully bcoz... they might end up getting into a deadlock(i dint understand how. kindly someone explain).

Deadlock can be avaided if.. we have a interruot handler.. in p1 which disables the interrupt handler and then graps the spin lock..

these things are complex.. and I have made it even more cryptic... the problem is that i did not completely understand what the interviewer was talking..(my first interview in life)... SO if people could elaborate on these lines... it would be helpful for me and the

76.How one can measure time spent in context switch.

I will add one thing about this. Context switch has three parts:
+ picking a new thread/process to run: means looking at the scheduler queues to pick up the best process to run. If there is one better than the current one, pick it.
+ saving the context of the current thread/process
+ restoring the context of the newly picked process

One could definitely save the timer ticks value before the first step and save the timer value after the last step. This will give the time for one context switch.

The interesting point to note is that this value has to be stored in global memory. If the value is stored in the Uarea or any per-thread storage, then after the context switch, that address is no longer accessible.

The other interesting point is what is the variable part of the context switch here. Depending on how many scheduler queues need to be checked, the time to search and pick the next best process might vary. And, if the memory location where we save and restore from is NUMA memory, then the memory saving time can also vary.

Or

Context switch only occurs in 3 situations (exception, interrupt, and system call). It looks like we can write a system call to calculate this. I'm just talking out of my ass. Any comments?

->Consider the statement
result = a ? b : c;
Implement the above statement without using any conditional statements.

```
a && (result = b, 1) || (result = c, 0)
```

->Write a multi threaded C code with one thread printing all even numbers and the other all odd numbers. The output should always be in sequence
ie. 0,1,2,3,4....etc

->Write a piece of code to find out if the system is x86 architecure of Sparc

use-> uname -a | grep -q "x86" in /bin/sh

->In Linux, we use virtual address.So each process will think it has 4 GB memory space even if the real memory is only 2GB. Now suppose we do not have MMU and programmer use real physical address in their program. We only have small size of physical memory. How can we design the system?

You can design the system in 2 ways :

1. Implement the virtual memory yourself and use File IO operations if and when you need more memory than existing memory. You need to swap some of your data to a File to create space for new Objects.

2. You try to clean memory just like a garbage collector does and remove unwanted objects from time to time. This is inferior to 1 and you run the risk of your program crashing because of lack of memory.

You can use a combination of 1 & 2.

->The interviewer asked the following question.

```
char *s = "Hello";
printf("%s",s);
printf(s)
```

The second print statement crashes sometimes. Why

Actually, when we assigns
*char * s= "Hello".*
's' starts pointing in read only segement where the "Hello" is.And when you do printf(s), IMO printf alwyas sees first argument as format string which it can change. And as he tries to change or format that argument you will get an access vilolation as you are writing to read only segment.

->Given an int, write code to return the number of bits that are 1 in O(m) time, where m is the number of bits that are 1.

```
public static int getNOnes(int n)
        {
                int result = 0;
                while(n>;0)
                {
                        n = n&(n-1);
                        result++;
                }
        return result;
        }
```

*->Assuming you have three N bit unsigned integers a, b and c, what is the min number of bits you would need to store the result of a * b + c?*

Another easy way to understand this is : consider N to be 8 bits or 1 byte... An unsigned int of 1 byte can have a max value of 255. Now applying the given equation with the maximum values possible:

*a*b + c = 255*255 + 255 = 65280*

we know unsigned 8 bit value max is 255 . Similarly unsigned 16 bit value is 65536 and 65280 is less than 65536. Hence we need 16bits for storing the result or 2N bits.

->Given a char array find possible soln like :
eg for "abc"
ans: {a,ab,abc.b,bc,c}

char[] function(char[])

```
#include <stdio.h>
#include <string.h>

void main() {
        char string[10] = "abc";

        int i,j,k;
        for(i=0;i<strlen(string);i++) {
                for(j=0;j<(strlen(string)-i);j++) {
                        for(k=0;k<=j;k++)
                                printf("%c", string[i+k]);
                        printf(",");
                }
        }
}
```

->

Given an integer array find the longest subarray cantaining consequitive nos.

eg {4,5,34,33,32,11,10,31}
ans is {31,32,33,34}

int[] function(int[])

```
#include<stdio.h>
#include<conio.h>
int arr[] = {5,2,7,9,3,6,8,1};
int max=0, max_in,count;
int longest(int);
int main()
{
int i;
for(i=0;i<8;i++)
{
count = 1;
longest(arr[i]);}
for(i=0;i<max;i++)
{
printf("%d\t",max_in);
max_in++;
}
getch();
return 0;
```

}

int longest(int in)
{
int i,in_1=in;
for(i=0;i<8;i++)
{
if(arr[i]==in+1)
{
longest(in+1);
count=count+1;
break;
}
}
if(count>max)
{
max=count;
max_in = in_1;
}
return 0;
}

->Give output for the following code

```
#include<stdio.h>

void main()
{
    int i = 5;
    printf("%d¥n",i++ + ++i);
    printf("%d¥n",i++ + ++i + i++ + i++);
    printf("%d¥n",++i + i++ + ++i + i++);
}
```

Please give the output with os and compiler u used with proper explanation. Getting unexpected answers..... help !!!
This is old question posed by senior student

i got answer as
12
35
52
but
When i ran this program on machine i shocked to get the answer 12,32,50.
i am writing here what concept i used.
If we do some operation in printf statement on same element then we have to use stack.
so in first printf statement:-
structure of stack is
++i (top of stack) => 6
i++ => 6
and after the execution of this value of i is 7 and i++ + ++i is 6+6 = 12

in second printf statement

structure of stack is
i++ <- (top of stack) = > 7
i++ =>8
++i => 10
i++ (lower index of stack) => 10
and after the execution of this value of i is 11 and i++ + ++i + i++ + i++is 10 + 10 + 8 + 7 = 35
in third printf statement
structure of stack is
i++ <- (top of stack) = > 11
++i =>13
i++ => 13
++i (lower index of stack) => 15
and after the execution of this value of i is 15 and ++i + i++ + ++i + i++ is 15 + 13 + 13 + 11 = 52

but When i ran this program on machine i shocked to get the answer 12,32,50.
i dont know what concept they used ..most probably side effect..
let me know if some have answer of this question with proper explanation.
thanks in advance ..!!!

->swap every two bits in an unsigned char .. eg swap bits at 0 and 1st position, swap 2nd and 3rd position, swap 4th and 5th position etc ..

```
unsigned char swap_bits(unsigned char ch)
{
    return ((ch & 0xAA)>>;1 ) | ( (ch & 0x55) << 1);
}
```

-> Given two numbers "a" and "b" and an average formula (a+b)/2. Find one condition where it wont work. Also, give solution to it

```
int computeAvg(int a, int b)
{
      return (a >> 1 + b >> 1 + (a&;1 + b&;1) >> 1);

}
```

->In a multi-threaded process,If one thread is busy on I/O will the entire process be blocked?

It WILL be blocked if the thread is implemented as a user level thread and will NOT be blocked if it is implemented as a kernel level thread.

->Assuming there's no Array data structure in C, how would you implement it?

to implement integer array of size array_size
*int *p = (int *) malloc (sizeof(int) * array_size);*

now you can assign values using p[i] = something // i =0 to array_size-1
and and access them using same;

->given two integers and two bit positions. Set the first integer between the two bit positions to be that of the second integer.

int replace_bits(int a, int b, int x, int y)
{

int mask = ((1 << (y - x + 1)) - 1) << x;
// Clear a and replace with that of b
return ((a & ~mask) | (b & mask));
}

or

s => start bit
e => end bit
first => first value
second => second value

Now

bits = e - s + 1;
*mask =((~0) << ((sizeof(int) * 8) - bits)) >> ((sizeof(int) * 8) - bits - s)) ;*
(first & ~mask) | (second & mask)

->Difference between constant char pointer and constant pointer to a char.

*const char pointer: const char **

The chars the pointer points to cannot be modified.

*const pointer to char: char * const*

The address the pointer points to cannot be modified.

->Difference between array and linked list

The difference is the way we access the data , we can access the data randomly in array using the index, but in linked list we have to loop sequentially to get into the data...
However, the linked list is more efficient in saving memory in the way they use pointer to save the address of the next node (ie don't have to allocate memory for all the data that we dont' use or not use yet.

->int error()
{
*char *buf=(char *) malloc(10,sizeof(char));*
return buf;
}

whats the error....?

warm up question

->example of volatile usage.

The usual four examples of volatile objects are:

1. an object that is a memory-mapped I/O port
2. an object that is shared between multiple concurrent processes
3. an object that is modified by an asynchronous signal handler
4. an automatic storage duration object declared in a function that calls setjmp and whose value is changed between the call to setjmp and a corresponding call to longjmp

*->aligned malloc, asked me exactly y do u allocate extra space for the void *. i tried to explain why,*

he wasnt listening. in the end he told me what i was trying to tell him

```
memalign(size, align)
{
     total_size = size + align
     ptr = malloc(total_size)
     ptr += (char *)ptr + (align - (uint)ptr % align)
     return ptr
}
```

->os concepts involving deadlocks, semaphores, spinlocks, sleeping in the kernel (process and interrupt context), softirq's
interrupts, interrupt handling, few virtual memory questions

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

A semaphore is a variable or abstract data type that provides a simple but useful abstraction for controlling access by multiple processes to a common resource in a parallel programming environment

In software engineering, a spinlock is a lock where the thread simply waits in a loop ("spins") repeatedly checking until the lock becomes available. Since the thread remains active but isn't performing a useful task, the use of such a lock is a kind of busy waiting. Once acquired, spinlocks will usually be held until they are explicitly released, although in some implementations they may be automatically released if the thread being waited on (that which holds the lock) blocks, or "goes to sleep".

softirq stands for software interrupt

->find if 2 strings are anagrams of each other

->strucure padding, why network packets are not padded, how to avoid padding (i told him the __attribute(packed)__ pragma)

#pragma pack(push,1)

structure

#pragma pack(pop)

->implementation of the sizeof operator, i told him its an operator, he said no. he wanted me to write the code how it works.

*#define my_sizeof(type) *
*({typeof(type) x; (char *)(&x+1)-(char*)(&x);})*

->How to find out the intersection of a link-list without using flags or modifying the link list

It can be done in this way
1. Find out the length of both lists.
2. Subtract small one from longer - This will bring both in same point to traverse
3. Now start traversing both & check for pointers. If they both are same at any point of time, they are intersecting. Else they are not

->Rotation of a 32 bit number

```
// Rotate an integer n left by k
int rotLeft32(int n, int k)
{
  return (n << k) | ((unsigned int)n >> (32 - k));
}
```

->When would you use a hash table? Specific situations were asked

->write a code to find the endianness of the system

```
int main()
{int n=1;
char *c=&n;
if(c)
printf("Little endian");
else
printf("Big endian ");
return 0;
}
```

->given a pointer to a single node of singly linked list, with no other details (dont know where the head is) and its not a circular list, how would you remove only that node from the list

```
void delNode(Node **p)
{
Node *temp = *p;
*p-> data = temp->next->data;
*p->next = temp ->next;
free(temp);
}
```

->How would you do benchmarking (compare the performance) in a device driver code? Apart from timing or time is there any other standard way? He basically meant comparing Programmed I/O and DMA. (Leave security etc. only performance comparison)

we can use the struct timeval and timespec timers inside the printk to see the transfer time of bulk data using the I/O map or DMA transfer.

Or jiffies

->Is sizeof() a macro or a function? I said macro... asked why not a function (leave alone the inline advantage of a macro)

Ans: In C there is no overloading. We send various 'Types' of parameters to sizeof. Macros dont do type checking. Hence Macros!

Or

sizeof is neither a macro nor a function. Its a unary operator like ! or ~. Its implemented by(in) compiler.

->What is Memory Alignment?
Can Two Processes have the same Virtual Address?
Difference between a MUTEX and a SEMAPHORE?
Does Malloc give a memory aligned address?

What is MMU made up of? (with reference to data structures

memory alignment is aligning variable or user defined variable acc. to processor's memory addressing strategy.
yes,two processes can have same virtual address.
Semaphores can suffer with priority inversion where as mutex cant. secondly, semaphores can have no deadlock if function is called recursively, but mutex can suffer with deadlock in that case. Thirdly, in multithreaded environment, any thread can signal the semaphore where as in mutex who have taken the lock can only unlock it.
Yes,Malloc gives a memory aligned addresses.
MMU is memory management unit which assign virtual addresses to processes instead of physical addresses and take care of paging concept of processes.

Or

2 processes cannot have same virtual address.. process have independant address space... but 2 virtual pages of different process can be mapped to same physical pae frame with read-only permissions

->Program to reverse the linked list!

```
void reverse(Node **head) {
 Node *current = *head;
 Node *next;
 Node *temp;
 if (head == NULL || *head == NULL || (*head)->next == NULL)
  return;
 temp = current->next;
 current->next = NULL;
 while (temp != NULL) {
  next = temp->next;
  temp->next = current;
  current = temp;
  temp = next;
 }
 *head = current;

}
```

->Write a C program to interchange the nodes of a linked list. Consider the linked list 1->2->3->4->5. You should get the o/p as 2->1->4->3->5. The pointers should be exchanged not just the data!

I used 3 pointers. Not sure if you can do it with two. If anyone has any suggestions please let me know! handles even # of nodes, odd # of nodes, 0, 1, 2 nodes.

```
void swap_every_two(node **head) {
  node *current = *head;
  node *next = NULL;
  node *prev = NULL;
  if (head == NULL)
    return;
  if (*head == NULL) /* no elements */
    return;
```

```
  if ((*head)->next == NULL) /* only one element */
    return;
  *head = current->next; /* update the head pointer */
  next = current->next;
  current->next = next->next;
  next->next = current;
  prev = current;
  current = current->next;
  while (current != NULL && current->next != NULL) {
    next = current->next;
    current->next = next->next;
    next->next = current;
    prev->next = next;
    prev = current;
    current = current->next;
  }
```

->Implement a MACRO(i,j,k) where i is a Hex number, j is the bit position and k is 0/1. So based on k, jth bit in i should be replaced. (Not no IF stmts allowed nor For loops in a macro)

#define MACRO(i,j,k) i = i&(((((i>>j)<<1)|k)<<(j-1))|(~(1<<j)))

->If we add a signed integer and a unsigned integer, will the result be signed or unsigned? What I guessed was signed, but after searching online for a while, looks like I was wrong. Can anybody give more detailed explanation?

Signed integer max 2^31, Unsigned 2^32. You are asking for trouble if unsigned number is big.

->What does the Static key word used in C qualify?

Static keyword results in only one instance of the variable.the memory for the is allocated at compile time

->given a structure,
struct {int a, float b, char c } x,y;
what are the 4 ways to copy contents of x into

4 ways I can think of are following:
1) y = x;
2) y.a=x.a, y.b=x.b, y.c=x.c
3) memmove(&y,&x,sizeof(x));
4) memcpy(&y,&x,sizeof(x));

->What is difference between wake_up ( ) and wake_up_interruptible ( ) APIs in linux kernel ? When should should use which one, how it should be decided ?

wake_up - wakes exactly one exclusive sleeping process in TASK_INTERRUPTIBLE or TASK_UNINTERRUPTIBLE state from the wait queue
wake_up_interruptible - wakes only one exclusive sleeping process in TASK_INTERRUPTIBLE from the wait queue

Wake_up_interruptible might be used in semaphore which wakes up only one process waiting for the signal from the wait queue. If you need to wake up only processes which are waiting on an event, then wake_up_interruptible must be used.
To generally wake up all the processes to some activity such as device I/O then wake_up has to be used.
I might be wrong, but this what I understand.

->how to find the msb in smallest no. of steps

*a & (((sizeof(a)*8)-1)<<1)*

->Addition of two signed characters, what happens the carry bit, and to the signed bit. Can the carry bit be over written on a signed bit

Yeah when an overflow occurs the sign bit is overwritten by the carry bit.

```
char a= 127;
a++;
printf("%d",a);
```

The output of the above code would be -128

->

What is the most efficient way of representing a number in the multiple of 256? (Clue - How do you clear a bit?) - Anyone has an answer to this

```
inline unsigned int mod256(unsigned int _i)
{
_i &= (~255);
_i >>= 8;
return _i;
}
```

->Have you ever called any device's function?

Compute Unified Device Architecture is an easy way to use GPU for General Purpose Programming. No graphics knowledge is required to use CUDA for doing a program using GPU. A CUDA program is almost same as a C program but have some additional features. In CUDA a function can be run in many threads by giving a execution configuration while calling a function. There are 3 kinds of functions in CUDA. A device function which can be only executed in the device and called from device, a global function which can be called from host(CPU) using some configuration and gets executed in device, and a pure host function which must be executed in the CPU only.

There are some additional specifiers to distinguish the function type.

__device__ - if a function is suffixed with __device__ it becomes a device function which can only be executed at device and which can only be called from a function that executes on device.
__host__ - These functions are the normal C functions that can be executed on the host(CPU)
__global__ - A function suffixed with __global__ can be called from CPU. But for calling this function the execution configuration must be mentioned. The execution configuration decides how many threads and blocks have to be made for executing this function.

Also there are different kinds of memory,

__shared__ - if this is prefixed that memory becomes a shared memory and it can be shared across threads. This is the fastest memory.
__constant__ - a memory to which we can write from host only.
__device__ - a memory to which we can write from both device and host.

Each thread will have a thread ID and block ID to know which area of the data need to be processed by this thread. This is the tricky area where all the performance improvement lies.

->How would you reverse bits of an integer in an optimized way suitable for an embedded system?

use lookup table to store reversed bits of every possible combination in a 8-bit field, so your lookup table is O(1) in memory with 256 elements. then:

```
int input,result;

result=((table[input&0xff]<<;24)|(table[(input>>;8)&0xff]<<;16)|(table[(input>>;16)&0xff]<<;8)|(
table[(input>>;24)&0xff]));
  return result;
```

->What is the difference detween ISR & function call

function calls are done in the program

ISR are the special function that are excuted on particular interrupts..
it mit be s/w or h/w.
u can call function as F()
but isr cannot be involed like that its based on the signal(s/w / h/w)

ISR is not function.
ISR never accept or return any value !!!!

ISR execution not necessarily causes a context switch, it causes a mode switch to kernel mode from user mode. After the execution of the ISR kernel may decide to continue with the same process or make a context switch.

an ISR flow usually consist of:
- Save ALL registers and system states
- Set all registers needed by the ISR, including the data and stack pointers.
- Process the event
- Restore ALL registers
- Return to the interrupted process

or

ISR:
Asynchronous event that can occur any time during the execution of the program
Saves the PC, Flags and registers on the stack and disables all the interrupts and loads the address of the ISR
ISR cannot have arguments that can be passed to it
Cannot return values
Enables the interrupts
Generally small as they are taking the time of some other process
Some of ISR have have their own stack

Fucntion:
Occurs when ever there is a function call
Saves the PC and registers on the stack
Can have arguments
Can return values
No restriction on the size and duration of execution

->what type of values cant be used in switch case can we use alphabets,floats,strings

*switch case allows only integer constant or constant expression like case 1,case 2+3*4 etc*

->where is the memory map for the process stored and is it in hardware or in software

Memory of a process is nothing but the virtual address space of that process. So the map of it, as in which physical page it resides, is stored in Page Table (single level or multi-level). Page Table is software, a part of the kernel.

Or

Pagetable can be stored in a hardware registers..the address of this register is stored in PCB block...if the pagetable is very big then we can store the pagetable in main memory..a pointer called page table base register points the address of the pagetable.

->explain what is done in compilation phase and what is done in linking phase and where does assembler comes into picture if at all ?

Compilation phase : Produces assembly code[Instruction sets defined for the Core].
Assembler : Convert assembly instructions into opcode-offset format ..object file[usually a relocatable code].
Linker : Link different object file via resolving references and consolidate them into an executable file.

Or

Normally the C's program building process involves four stages and utilizes different 'tools' such as a preprocessor, compiler, assembler, and linker.
At the end there should be a single executable file. Below are the stages that happen in order regardless of the operating system/compiler and graphically illustrated in Figure w.1.
Preprocessing is the first pass of any C compilation. It processes include-files, conditional compilation instructions and macros.
Compilation is the second pass. It takes the output of the preprocessor, and the source code, and generates assembler source code.
Assembly is the third stage of compilation. It takes the assembly source code and produces an assembly listing with offsets. The assembler output is stored in an object file.
Linking is the final stage of compilation. It takes one or more object files or libraries as input and combines them to produce a single (usually executable) file. In doing so, it resolves references to external symbols, assigns final addresses to procedures/functions and variables, and revises code and data to reflect new addresses (a process called relocation)

->what will happen if process without any children executed thread_join

since we haven't created the thread and we are asking the parent to wait for it...it will do nothing imo...i tried it on linux

or

thread_join is an API which causes the calling thread to wait till a particular thread be executed.

Let say we have two threads available in a process.
1- Main thread : - Generates another thread
2- Both threads are independent and start executes in parallel.
3- If let say Main thread get CPU and it doesn't execute "join" API , it might cause process to exit without waiting for the newly generated thread completes its execution.

```
int main()
{
create_thread(); //We have 2 threads now main thread + child thread;
main thread completes the execution;
```

//thread_join(); wait for child thread execution
exit(0);
}

->WHY CONSTRUCTORS NEED NOT TO BE DECALRED AS VIRTUAL

The purpose of the Constructors is to construct an object of that very type. Declaring it virtual is meaningless because by the time a constructor is called, compiler knows the exact type of the object to create.
It is the same reason that virtual functions called from within a constructor bypass the virtual call mechanism. Because within a CTOR, the exact object type is known, compiler can directly call the correct version without having to go thru the vfn table.

->constants defined through $define are placed in what type of memory and do they follow block scope will o/p of print be 1

```
void main()
{
funct();
printf("%d",x)
}

funct()
{
#define x 1
}
```

It will give a syntax error as "x undeclared" .. The preprocessor replaces all the statements followed by the #define statements within the scope..

This will work without any error-

```
void main()
{
#define x 1
funct();
}

funct()
{
printf("%d",x);
}
```

whereas this will give an error-

```
void main()
{
funct();
printf("%d",x);
}

funct()
{
#define x 1
}
```

#define is a preprocessor. it doesn't take any area inside the program memory. Before compilation it replace with the actual value inside the code.

->ell some of applications of buffer?

What is D-Word, Q-word?

Buffers are used to speed up the operations. They reduce delay

D word is double word 32
Q-word is Quad word. 64

->how to compare 2 floating point no.s

```
int compareFloats(float f1, float f2) {
    char *b1;
    char *b2;

    b1 = (char*) &f1;
    b2 = (char*) &f2;

    int ix=0;
    for (ix; ix< sizeof(float); ix++, b1++, b2++) {
         if (*b1!=*b2)
             return 0;  // not same
    }
    return 1;  // same
}
```

->In a computer which has a word size of 4 bytes, how do we read just a single byte?

By using character pointer.

```
#include <stdio.h>
int main()
{
int x=256;
char *p=&x;
printf("%d",*p++);
printf("%d",*p++);
printf("%d",*p++);
printf("%d",*p);
}
```

```
->Provide an example of synchronous and asynchronous signals
```

synchronous: SIGFPE, SIGSEV ,asynchronous: SIGINT.in UNIX all signals are asynchronous

->What signal is generated on divide by 0?

SIGFPE is the signal generated and the code generated is FPE_INTDIV in case of integer divide by zero and the code generated is FPE_FLTDIV in case of floating point divide by zero.

->What do we mean by 64 bit architecture? What are the advantages and disadvantages of 64 bit architecture?

 # 64-bit refers to the number of bits that make up the structure of various parts of a processor's architecture. These structures usually include the data bus, address bus, and internal registers.

 # Memory-mapped files are becoming more difficult to implement in 32-bit architectures, especially

due to the introduction of relatively cheap recordable DVD technology. A 4 GB file is no longer uncommon, and such large files cannot be memory mapped easily to 32-bit architectures; only a region of the file can be mapped into the address space, and to access such a file by memory mapping, those regions will have to be mapped into and out of the address space as needed. This is a problem, as memory mapping remains one of the most efficient disk-to-memory methods, when properly implemented by the OS.

 # Some programs such as data encryption software can benefit greatly from 64-bit registers (if the software is 64-bit compiled) and effectively execute 3 to 5 times faster on 64-bit than on 32-bit.

 # The main disadvantage of 64-bit architectures is that relative to 32-bit architectures the same data occupies more space in memory (due to swollen pointers and possibly other types and alignment padding). This increases the memory requirements of a given process and can have implications for efficient processor cache utilization. The ability of 64-bit applications to access a larger physical and virtual memory space means that address tables are larger and can result in a larger data-transaction overhead. As a result, small or repetitive tasks may run marginally slower than in a 32-bit environment.

->From which type of memory does shared segment come?

Heap segment.

->What happens when the following piece of code is executed?

*Char *ptr;*
*While(1) ptr = malloc(1024*1024);*

Malloc would simply start failing when it can no longer allocate more memory. The program will continue to run...

or

stack overflow cant occur malloc uses heap :)

->In execution is it possible for an old process to get control back?

yes its very much possible, when a process that did an interrrupt event, will get the control back after the interrupt request has been finished.

->In byte addressable memory, what is the smallest size of data accessible? Is it byte? what about bool?

Yes a bolean use 1 Byte.

->Explain what is meant by synchronous and asynchronous bus.

synchronous bus include clock bus in control lines.A fixed protocol for communication.It involves very little logic and can run very fast.disadv:Every device must be on the same clock rate.

Asynchronous bus is not clocked and can accomodate with wide range of devices.disadv:
It require handshaking protocol.

->is it possible signals to be send in hardware and what is the example simillarly example of software and hardware interrupt ?

in C++
signal() can be used to register a handler, raise() can be used to raise a signal.

Signals:

SIGABRT (Signal Abort) Abnormal termination, such as is initiated by the abort function.

SIGFPE (Signal Floating-Point Exception) Erroneous arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with a floating-point operation).

SIGILL (Signal Illegal Instruction) Invalid function image, such as an illegal instruction. This is generally due to a corruption in the code or to an attempt to execute data.

SIGINT (Signal Interrupt) Interactive attention signal. Generally generated by the application user.

SIGSEGV (Signal Segmentation Violation) Invalid access to storage: When a program tries to read or write outside the memory it is allocated for it.

SIGTERM (Signal Terminate) Termination request sent to program

->if we declare some auto/global variables inside a particular thread will they be visible to another thread too (remember threads share same memory space) ?

thread has its own stack, so auto declared inside thread should not be visible outside

if you mean a static variable as a global variable inside the thread, imo it should not be visible outside the thread.

Or

AS per me static variables declared inside thread are visible to other threads since all threads have the same address space

or

Auto variables are stored on stack, whereas global and static variables are stored in data segment. Threads share all segments expect stack. So they share the auto variables, but not the global or static.

->if thread allocate something dynamically will it be visible to other threads

no,
it will not visible to other thread...

each thread has own stack area. and heap area also.

so only if you re initlize globle variable with in one thread it will visible to other thread...

or

Each thread has a private stack All threads share a common heap.

Dynamic memory allocated from heap.

SO yes it will be visible to others

->in exec since pid of the new process has the same pid as the old process after exec. So what happens heap stack and other memory space of the process ?

Clean up - cleared. Essentially it would be a new process with old id.

->pass array by value to a function

Syntax-wise, strictly speaking you cannot pass an array by value in C.

void func (int x); /* this is a pointer */*

void func (int x[]); / this is a pointer */*

void func (int x[10]); / this is a pointer */*

by using stracture we can do it...
eg:

```
struct A { int arr[2]; };
void func(struct A);
```

->explain nested interrupts

Interrupt Service Routine [ISR] getting interrupted. That is a nested interrupt.
Example - mouse interrupt is being interrupted by power switch interrupt.

->difference between software development cycle in rtos and normal os.

Not quite complete answer, but one of the difference will be while in the Design stage, more care will be taken to maintain the deterministic scheduling of a RTOS and also in the Testing stage, it would taken care of, the RTOS gives a strict deterministic time results.

->What is the differnce between embedded systems and the system in which rtos is running?

Embedded system use a software which should be able to run for a along time without human intervention. There are many embedded systems which don't even need an OS or any scheduling algorithm. embedded systems with RTOS are more sophisticated devices with scalability and deterministic behaviour.

->WHY padding increases systems performance

Padding is done to fast access the memory.Let us take ex of structure padding.
In it , structure members are aligned to be based on the memory pointer size.
If word size is 4 bytes then data should be read at the offset of multiple of 4.

->advantages and disadvantages of Memory mapped I/O and Input mapped I/o.

Memory Mapped I/O means using the memory addressing to interface with the peripherals, while I/O mapped I/O (also called Port mapped I/O) means using 8-bit port addresses to interface the peripherals.

->what is signal handler? In which space they are written user or kernel

default handler is written in kernel, user-defined handler is in user space, user define handlers can override the kernel ones

->explain what is saved in thread context switch and compare with process context switch

only stack,PC and registers are saved in a thread context switch..

->explain embedded systems properties of cellphone

Low memory footprints, low power consumption, scalable OS (RTOS),deterministic behaviour

->Explain the boot sequence ?

BIOS is given the control.
BIOS loads the first 512 bytes of the Hard disk in memory location 0x7c00. This is nothing but the boot loader located in the MBR.
BIOS passes CPU control to this location and execution begins. This bootloader loads the appropriate kernel image in memory and puts CPU in protected mode from real mode.

->Difference between Interrupt and a signal when to use each ?

Signal is a form of Inter process communication between in a multiprocessing environment. Interrupts are of two types- Software and Hardware. Both these forms stops what the CPU is currently doing and executes the interrupt service routine (handler).

example of Hardware interrupt : Trap, Interrupt from an I/O device.
Software Interrupt : System call, exception etc.

Or

signal is a form of interprocess communication .It is an asynchronous noticification sent to a process or specific thread.when a signal is sent the operating system interrupt the target process's normal flow of execution.
Interrupt is a signal that is generated by hardware or software that need 's immediate attention . an inetrrupt alert the processor to a high priority condition requiring the interruption of the current code that processor is executing .then processor responds by suspending it's current activity and execute a small program called interrupt handler to deal that event .

->if there r 2 processes with the same deadline in RTOS and we can run only 1 processes and both the processes are exactly same in everything like start time priority and all how will scheduler decide which process to run

What i have an approach in mind is that, as both the process have the same priority and stuff, the round robin or timeslice mechanism would come in action and a certain timeslice would be given to the processes and once they are executed, the processor will again look for a higher prioirty task and then execute it first and so on...
Please let me know if there are some other ideas for this

->after context switch processes which has interrupted other process starts running but how does os know which process to run after running current process. Simillarly in case of interrupt how does interrupt handler know where it should go back ?

i guess its the process control block[PCB] thats keeps getting pushed on top of stack for each interruption/switch by another process,,, so the moment one process is over the one sitting on TOP of the stack is picked and that is exactly the same process that was running when the request for serving another process come...and this applies recursively upto the last process left in the Stack...similar to function calls in programming languages like C/C++ etc...

->what do you mean by address space of a process what essentially it is ?

address space of a process is the memory allocated to the process by the OS for completing its operation. the process gets exclusive rights to that address space unless the process wants to share something with some other process...this address space as allocated by OS is essentially belongs to Virtual Memory that gets mapped to the real one....

Address space of a process essentially consists of -

1> Stack
2> Heap
3> Data segment
4> Code segment

->

What is the difference between process and thread context switch what all needs to be saved for process and thread context switch like do we need to save heap, basically she wanted me to list everything when i said registers she asked me to tell the name of the registers then she asked if process variables needs to be saved and so on ?

the mininum content required to restart a process where it left off before getting interrupted by another process, is the information that needs to be saved through PCB which essentially contains things mentioned clearly in the definition of Process Control Block@wikipedia: http://en.wikipedia.org/wiki/Process_control_block ....hope this helps.

->find max number of repetitions in an array.like {2,3,4,5,2,3,2} max repeat is 2. optimize it for space then in time.

Hash table wont be optimized by space... I think sorting is the technique, if we look for space optimization, and Hash table is the solution if time complexity is more improtant.

->basic questions like what is volatile , static and where to use them and why?

->write a macro to give offset of particular field in structure.Like struct abc { int a,int b,char c}. Write a macro like offset(abc , c) to find offset of c from top.

#define offset(abc,c) ((char)(&(abc.c)) – (char*)(&(abc)))*

->everse binary representation of number.

->I have a number in float like 2.5. Store this number in int in such a way that store 2 in bits 31-16 and 5 in 15-0.write a function to do that to convert it and vice versa.

should be correct up to rounding:

```
unsigned float2fixed(float x) {

unsigned res = (unsigned)floorf(x) << 16;
res |= ((unsigned)floorf(x * 65536.0f) & 0xffff);

}

float fixed2float(unsigned x) {

float res = (float)(x >> 16);
res += (float)(x & 0xffff) / 65536.0f;

}
```

->Let suppose there is a FIFO. Data is written into the FIFO at the speed of 4 ns. A maximum of 80 words /100 cycle are expected. The read port reads the data at the speed of 5ns. It

can read 80 words/ 80 cycles. What should be the depth of the FIFO so that we dont lose any data. Hint: Consider the previous cycle as well

*The worst case is that it can write 160 words consecutively. The time of writing is (160*4)ns. Reading is one word per cycle. So the number of reading is 160*4/5=128. Therefore, we need 160-128=32.*

->Detect loop in a singly linked-list

*..mplement atoi(char *p)*

```
int atoi(char *p)
{
int total = 0;
int i=0;
while(p[i] != '\0')
{
total *= 10;
total += p[i] - '0';
i++;
}
return total;
}
```

///What is DMA? Can user level buffer / pointer used by kernel or drivers?

///Write an aligned malloc & free function. Which takes number of bytes and aligned byte (which is always power of 2)

Ex. align_malloc (1000,128);

it will return memory address multiple of 128 of the size 1000.

aligned_free();

it will free memory allocated by align_malloc.

///

Write an printhex function which implements "%x" in printf without using printf.
Example :-

printhex(10) output is A .

```
char table[16] = { '0', '1', '2','3','4','5','6','7','8','9','A','B','C','D', 'E','F' };
void printhex(int input)
{
      int index = input%16;
      input/=16;
      if(input) printhex(input);
      putchar(table[index]);
}
```

///Find the Maximum of 2 numbers without using any if-else or any other comparison operator

Hmm... its basically how a multiplexer works . suppose 2 numbers are A and B .
Perform the operation A-B and extract the MSB .let it be k. Now if k =1 then A>B else B>A (2's complement)..

*return (1-k)*A+k*B. This will alyways return the maximum of A&B*

//What is the minimum number instructions required to swap Odd and even bits in an uinteger
Eg
10101010 should become 01010101

well ths soultion will be like
1) let the number be A = 10101111
2) A1 = A & 10101010 = 10101010
3) A2 = A & 01010101 = 00000101
4) A1 = A1 >> 1 = 01010101
5) A2 = A2 << 1 = 00001010
6) Answer = A1 | A2 = 01011111

so u require 5 instructions

///How will you determine if a loop exists in a link list?

```
function boolean hasLoop(Node startNode){
  Node slowNode = Node fastNode = startNode;

  while (slowNode && fastNode)
  {
    if (slowNode == fastNode) return true;
    slowNode = slowNode.next();
    fastNode = fastNode.next();
    if (slowNode == fastNode) return true;
    fastNode = fastNode == null? null : fastNode.next();
  }
  return false;
}
```

///What's wrong with this code:
unsigned int i;
for (i = 100; i >= 0; --i)
printf("%d\n",i);

//Describe in english what this code do?
((n & (n-1)) == 0)

No! This code reveals whether the integer is a Power of 2.

//Explain Virtual Memory. What is the TLB?

Virtual memory is very important because each program is gonna take 2 power 32 bytes of memory which 4GB and many programs doesnt use that much of memory physically. So we have RAM which will map the addresses to the physical address. So once a variable needs an address its gonna look into the page table and will get the physical memory page to the RAM. So pages will be going in and out and you feel that you have been actually given 4GB which you dont know that it hasn't allocated. So its gonna load only on demand into the virtual memory and its the memory managers duty to avoid two processes or programs virtual memories to map to the same physical address and there by corrupting each other accidentally. Now coming to TLB, TLB is translation Lookaside Buffer, Since we need to map the virtual address to physical address in the disk, its gonna look in the TLB cache to see where the corresponding physical address is present to the corresponding virtual address requested by the process.

Confused...you can actually google them and study a bit architecture to brush up these internal details.

Difference between Block and Character Devices

In Linux environment, or Linux OS will not consider the extension of File names . For instance extension of file has more significance only in Windows operating system. That is filename.bat or filename.exe has more significant

what is the difference between block and character devices ?

when the user writes some thing the information is saved and written in chunks and assume that every time if you want to access the device it will be a time consuming task. Instead we can use a buffer where it can store the latest information. Devices which uses buffers are called block devices and which are not using buffer cache are called character devices

Difference between kill and raise Functions

Kill function send a signal to process or group of process whereas raise function is used to allow a process to send a signal to itself.

Difference between Alarm and pause functions

Alarm functions allow us to set a timer that will expires at a specific time in future and if the timer expires, SIGALARM signal is generated. we need to handle this signal and if we are not handling it, the default action is to terminate the process
Pause functions are used to suspend the calling process until the signal is caught.

Signals that cant be ignored

Signals are the classic examples of asynchronous events and even though we catch or ignore the signal. There are two signals where the user cant catch those signals. **SIGKILL** and **SIGSTOP** are the two signals where the user cant ignore the signals and this allows the supervisor to control the status of any process.

Symbolic link vs Hard link in Linux

Links are more useful in Linux system and the same is created at different purpose. Symbolic links are created with 'ln -s' option and hard link without 's' option. In case of hard link the inode number is same and we can create as many aliases to the same location and the other important point here is the content will get destroyed only if we remove the entire alias.
The inode number for all the files linked through hard link will be same and in case of symbolic link the file is just linked to the source location. This is equivalent to the creation of short cuts created for windows software's

Important service and daeomns in linux
When Linux system is booting we could see a lot of services are coming up with OK or failed. every one wonders what is the nature of program and what could be the importance of this services. once Linux system is getting booted.. it reads the file called /etc/inittab which is very important and this is an important file where the init progam reads this file and follows the instructions. The services which are configured and needs to be started will be available in /etc/rc.d/init.d/ directory which has all the services or daemons. This services are simply Bash scripts and other important point is for new Linux users, this is an important place where all kinds of shell scripting is being used

Managing Disk space
There is lot of scenarios where we need to take care of the disk space and in the real world if we are not taking care of the disk space, there could be serious performance issue. For a process to perform well, it needs both memory space as well as disk space.
We can use the **disk free (df) command** to identify the amount of free space available and if we want to find our consumption of file directory rather than file system, probably we can use **disk usage (du) command.** And if we use du with (-s) option this will give the summary rather than moving to each directory.
Once we identified the disk space and utilization, probably we can find out the file which is consuming more space by using the **find command** with size option.
Once we identified the files which are having larger space, the same can be reduced by using the **compress and zip command**. We can use the extract the files by using the **uncompress and unzip command** that are compressed already

Command used for File Processing in Linux

We need to process the files such as logs or source codes for different purpose. Basic understanding of each command related to file processing is essential to reduce the effort that we put forth for the same.
Head and tail command is used to extract the top or bottom of the file contents. If we use head command, few lines from top of the file will be displayed and if we use tail command, bottom of the file content will be displayed. **tail** command can also be used more frequently to monitor the growth of file (with -f option)
tr is used to translate characters such as converting from

- lower to upper case and vice versa
- squeezing of spaces
- converting the delimiter and so on

grep command is used to find the matching patterns in a file. This can be used for

- To search the matching patterns with ignoring cases
- To count the matching patterns in file

- To print other than matching patterns and so on

Tips and Tricks with Shell Programming.

Every one wants to become master in Linux. what actually need is to understand the shortcuts and needs patience to understand how each command works. Lets take a simple example of how ls command works. That will give a better idea of the system calls involved and also will give a clear idea of how the inode number works.

What is the importance of parameter $?
Parameter $? stores the exit status of the last command

When I can use the parameter $?
parameter $? can be used in scenarios where we want to execute some commands based on the status of the last command
The other scenario is if we want to display the status of the last executed command

How I can increase the size of the history command
$HISTSIZE is the built in variable which can be altered to increase/decrease the size of the history command

What are the operators used as logical and conditional execution in bash shell programing
$$ is used as Logical and operator
|| is used as Logical or operator

How to identify the process id of last background job
$! can be used

How to use the command line arguments in shell programming
if we execute the shell programming as
shell_name arg1 arg2 arg3 arg4
for_instance
run_program 12 13 14 15

Here
$0 is run_program, which is the program name
$1 is 12
$2 is 13
$3 is 14
$4 is 15
$# is 4 which is total number of command line arguments

What is the purpose of wait command in Linux

Wrong command is used to verify whether any process is running in background in Linu x environment. We should not confuse with the wait() system call. This is a wait shell command which can be used along with process id or without process id.

What is the difference of using wait with and without process id ?
wait with process id will wait for only the particular process id to get completed. wait will wait for all background process to get completed. if the user wants to ensure that there is no process running

in background we can run the wait command

Basic commands used in Linux

- **grep** print lines matching a pattern
- **wc** print the number of newlines, words, and bytes in files
- **sort** – sort lines of text files
- **su** – change user ID or become super user
- **passwd** – update a user's authentication tokens(s)
- **who** – show who is logged on
- **tar** – to archive a file
- **du** – estimate file space usage
- **cat** – displaying and creating files
- **echo** – display a line of text
- **rm** -remove files or directories
- **find** – search for files in a directory hierarchy
- **history** – prints recently used commands
- **cp** – copy files and directories
- **mv** – move (rename) files
- **man** shows all information about the command
- **sed** stream editor for filtering and transforming text
- **awk** pattern scanning and processing language
- **bg** – make a foreground process to run in background
- **fg** – to make background process as foreground process
- **jobs** – displays the names and ids of background jobs
- **chmod** – change file access permissions
- **chown** – change file owner
- **chgrp** change the file group
- **sort** – sort lines of text files
- **kill** – to terminate the process
- **ps** – to view the process status
- **split** – splitting of a file into multiple files
- **lp** – printing of a file
- **more** paging output

How do I can schedule the Jobs in Linux environment

Scheduling of jobs

Scheduling of jobs is needed if the user wants to run the program at some time later or if he wants to run the program periodically

what are the command that are available to schedule the jobs ?
at command which is used to schedule the job once at later point of time
crontab is used to schedule the jobs periodically
batch command is used to schedule the job when the CPU utility is low

which command can be used to schedule the jobs (at or cron) and how I can select it ?
Again this is depends on the purpose. If I wants to run the job periodically then my choice should be crontab and if I want to run the job only once, then I can prefer running the command with at

How do I control the scheduling of jobs with at command ?
at -l command will let you know the jobs scheduled to run at later stage
atrm command will allow you to remove the scheduled jobs based on job id

what will be the reason if there is some issue with crontab or at command ?
there could be multiple reasons for the above issue
there could be problem with syntax and more preferably we have to verify whether the two important process which are taking care of scheduling the jobs is running or not. crond and atd process are more important process which does all the work

How to control the process in foreground and background

A program in execution is called process. each process will have unique process id.

How do I differentiate the process is running in foreground/background ?
if the user needs to wait for the result of the program, then the job is running in foreground. This also means if the user log out the terminal, then the job will get killed. In case of back ground jobs , when the user runs the program, he will run with (&) symbol where he will get the job id of the process running in background.

How I can run the programs in background ?
program_name arguments_input &
or
nohup program_name arguments_input &

How do I know how many jobs are running in background ?
jobs command will allow the user to know the jobs running in background with job id

How do I can move the process from foreground to background ?

1. run the process in foreground
2. press control+z, the process will get stopped
3. and run immediately with the command bg

How do I move the process from background to foreground ?

run the jobs command and view the job id of the program that we want to move to foreground.
fg %job_id
fg %starting name of process
will allow the process to move from background to foreground

HP Interview Questions

1. Consider a scenario where the same program is executed twice in different terminal. if that is the case, please elaborate whether the above process will share the text region or not?
2. How do you give maximum information to the user to debug the core file?
3. Explain how the sigbus signal is generated?
4. Write a 'C' program to simulate a stack.

Yahoo Interview questions

1. What is meant by MSB and LSB ?
2. How to you find your system is little endian or bigendian ?
3. How to you find your system is little endian or bigendian through C/C++ program ?

4. How do you implement the Hash array technique of PERL through C program ?
5. What is meant by Singleton ?
6. How do you implement the singleton especially what type of constructor we used to implement the same

Unix/Linux Interview Questions

what is zombie process?

if child process dies and the associated parent process still exist, then we call the child process is zombie process.

What is orphan process ?

The exit() system call terminates the execution of a process. All open file descriptors are closed. If the exiting process had any child process, they no longer have a parent and as a result the kernel sets their parent-process-IDs to one, which is the process-ID of the init process. These child processes are called *orphans* and init is said to have adopted them

How Protocol Works ? Interview Questions

1. How FTP works?
FTP uses two separate TCP connections and one is called as command channel (server uses port 21) and other is called as data channel (Server uses port 20) and client always uses port greater than 1023 for both command and data channel.

2. How TCP works?
TCP working principle is based on three way handshaking between Host-A and Host-B. Host-A transmits SYN (synchronizing bit of where Host-A will transmit) and in order to acknowledge this Host-B will transmit ACK with what type of SYN bit Host-B is going to transmit and in order (No more data from the sender bit) will be transmitted to indicate that no data are going to be transmitted between the hosts.

3. Is TCP sends the data as continuous stream of bytes or packets?
TCP sends the data as continuous stream of bytes not as individual packets. With help of sequence number and acknowledgment number in TCP header, TCP takes care of maintaining the sequence in which the bytes send and received.

4. What are the two types of DNS n/w activities?
a. Lookups (when client request the information from the server such as IP-address of the host name or the host-name of the IP-address etc. UDP/TCP)
b. Zone transfer (occurs when secondary server trying to request the primary server that it knows about the zone information-TCP)

5. How SIP works ?
SIP is application layer protocol borrows most of the syntax from HTTP and SMTP. It is a signaling protocol used to transmit and control Video/Voice over IP protocol. SIP needs user agent to be more intelligent which can handle SIP signals from SIP client to SIP server and it can work on any one of the following underlying protocol

1. **User datagram protocol (UDP)**
2. **Transmission Control Protocol (TCP)**
3. **Stream Control Transmit protocol (SCTP)**

Linux Kernel Programming

Linux-kernel can be distinguished in to various subsystems

Process subsystem
This subsystem supervise the process manipulation, such as

- Creation of process (system call fork, which is used to create new process in Unix file system),
- Transformation (loading another program into a process with system call exec. After calling exec we can't return to the parent process),
- Synchronization of the process(parent waiting for child with system call wait) and
- Termination of the system call or process (system call exit).

File subsystem
This subsystem, also called the Virtual File system Switch (VFS) manages the manipulation with files. Since files are stored on different types of file systems, the VFS-layer contains file system-specific code for each supported type.
Memory management
This subsystem manages the distribution of memory pages and takes care of freeing up memory as soon as the quantity of free memory is too small.
Device drivers
Consider device drivers as the layer between the generic kernel and the specific hardware. A device driver (character/Block/network) is a predefined set of subroutines or functions which manages a particular type of hardware-controller or device combination.
Socket subsystem
This subsystem deals with the network protocol families (e.g. the INET-family supporting the TCP/IP protocol stack

Linux kernel Programming Interview questions

What is meant by system call?
Systems calls run in kernel mode on the user's behalf and are provided by the kernel itself.
What is the command to install the module in Linux?
Insmod (install – module) is the command to install the Module
What is the command to remove the module which are already installed
rmmod (remove-module) is the command to remove the module which are already registered
What is a kernel Module?
Kernel Modules are bits and pieces of code that can be loaded and unloaded in to Kernel based on demand or user request. The idea is simple that the kernel size will be small and modules are added/removed without rebooting the kernel
What is the difference between programs and Kernel Modules?
A program is compiled and an executable version where it has the main () function and it perform the task and comes for the halt. In case of modules it has module_init()/module_exit() function and it just initialize the modules and the rest is taken care by kernel.
What is the difference between user space and kernel space?
What is meant by Major number and Minor number?
If kernel wants to know what type of driver it needs to use, then it has to see the major number. for instance if u have two floppy in the same PC, the two floppies will have the same major number and to differentiate between the two minor numbers are used, which will be used by the device driver.

Hence Major numbers are being used by the kernel and the minor numbers are being used by the device drivers

What is the function used to register a device?
Can we uninstall the devices which are registered already?
What is the importance of /proc File system ?
What is the difference between system call and Library Call?
What is the disadvantage of making the kernel as monolithic?
What is the difference between printf() and printk() ?
Difference between Block and Character device drivers
How to distinguish various subsystems of Linux Kernel

Linux - File Sytem

Linux file system on disk, independent of its type of file system always contains the following classification.

Super block

The super block portrays the characteristics of the file system, such as the

- Type (magic number)
- Amount of inodes (total and free)
- Amount of data blocks (total and free)
- File system state, etc.

Inode

An inode portrays the uniqueness of one file, such as the

- Type (regular, directory, symbolic link, ...)
- Size in bytes
- Time-stamps (last modification/access/...)
- Permissions of file
- Owner of file
- References to the blocks containing the data, etc.

The *name* of a file however is not stored in the inode. Every inode has a unique sequence-number, starting with 1. By convention inode #2 is the root-directory inode.

Data blocks

The data blocks portray the data or contents of the file.

Debugging with GDB GNU Source Level Debugger

Programs such as C and C++ needs the debugger to debug the issues and we need some tool to identify the issues. Even though there are different tools available to debug the issues most common tool used by every one is GDB.

GDB is free software available in almost all POSIX system.

List of features are

- Free software
- Debug the core
- Debug the running Process
- Examine the symbol table
- altering execution of process

How to run the gdb program

- gdb Program core

- gdb Program process-id

Interview Tips

This page will give you some of the points to be noted on how to get into MNC like WIPRO, CTS, TCS, Tech Mahindra, INFOSYS,etc..

1. Ensure your resume is up to-date with brief summary of your experience
2. Upload the resume in job sites
3. Frequent update of resumes in all job sites
4. Resume updates has to done in all job sites and a single resume has to be available at all time
5. Ensure you have updated your contact number is updated in all job sites
6. Better to have telephonic round of interview during first level . This will reduce the time
7. Ensure you go-through the company web site and understand the technology and domain of the company
8. If you are not searching for job, please hide your resume in all job portal. This will allow you to get a fresh search when you require for a job
9. Ensure that you stay in a company for at least a year and if you feel that you are growing with the company its better to stay in the same company for at least 5 year of time

Threads

In computer Programming, if we want to have communication between the different process normally we may use **IPC (Inter Process communication)** techniques such as message queues,semaphore and shared memory. if we want to execute series of instructions within the program then we must be proficient in creation/managing of threads.Thread libraries communicate with LWPs (Light weight Process) to schedule threads. LWPs are also sometimes referred to as kernel thread. Thread process contain

- Thread creation (pthread_create() function)
- Thread termination
- Synchronization using mutex (joins, blocking), controlling the serialization of data by avoiding the same data is updated concurrently by different threads. The concept is being simple. Before accessing the data, each thread will lock it and once the job is over again it will unlock it.
- Scheduling of threads
- Data managing and
- Thread communication.

A thread is a progression of instruction that can be executed in parallel with other threads. This type of thread communication can be well established by using high level language such as "***java"*** and "***C" Programming language*** using "***pthreads Concept"***
All threads within a process will share the same address space. This is the main advantage in terms of threads vs IPC (Inter process communication). In case of IPC each process will have different address space which leads to the communication of Semaphores, Shared memory and message queue concept being used.
A thread does not preserve a catalog of created threads, nor does it know the thread that produced it. This is up to the user to preserve the list of threads created.

Threads in the same process will share the subsequent resources

- Process instructions
- Data
- File descriptors

- Signals and signal handlers (advantage over IPC, which reduces users time in writing separate signal handlers for each threads)
- Present/current working directory
- User and group id

Apart from sharing the common resources between each threads in a process, Each thread will have the subsequent unique features

- Unique Thread ID
- Set of registers,
- Stack pointer
- Stack for local variables and return addresses
- Masking the signals
- Priority (Priority of thread can be varied and the user can give importance)

The unique benefit here is threads are easy to handle within a process and communication between the threads is happening between a single process whereas in case of IPC these communication is happening between different set of process
Mutexes are needed whenever we use threads to avoid data in-consistency in case of race condition.

Event Driven Programming

Event Driven Programming
Event-driven programming or event-based programming is a programming pattern in which the flow of the program is determined by events.
For instance- sensor outputs or user actions (mouse clicks, key presses) or messages from other programs or *threads.*
Mouse clicks or key presses from the external user for any UI based applications.
Communication between two defined Processes through threads (POSIX) .Here process indicates the program in execution and for each process we will have a defined process id.
Event-driven programming can also be distinct as an application architecture method in which the application has a main section which is clearly separated down to two segments: the first one is event selection (or event detection), and the next one is event handling. In embedded systems the similar could be achieved using interrupts instead of a continually running main section; in that case the event detection resides entirely in hardware.

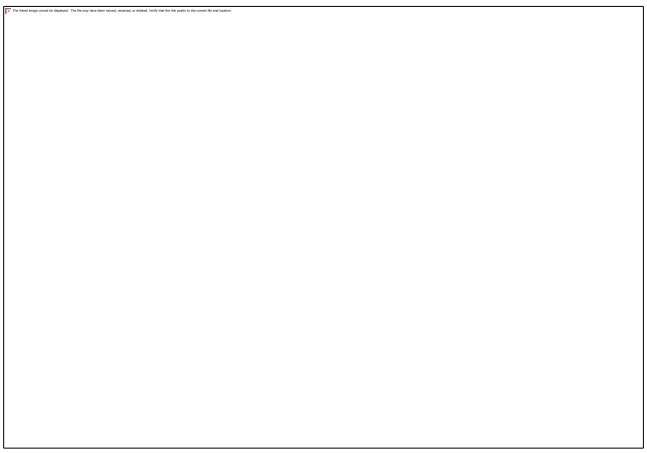
Shell Programming Language

In Linux system, we have different shell programming being used. The shell which is an layer over the kernel and the user directly speaks with the shell and in turn shell interacts with the kernel and send the response back to the user. The shell acts like a command line interpreter which acts like an user interface to the user.

Different type of shells are available in Linux Operating system
a) **csh (C shell)** The C Shell was developed by Bill Joy for Berkley Software Distribution. The original shell was the Bourne shell; ***sh***. POSIX platform will either have the Bourne shell (***bash***), or a Bourne compatible shell available. *sh* has very good features for controlling input and output, but is not well suited for the interactive user. To meet the interactive user audience C shell, ***csh***, was written and is now found on most, but not all, POSIX systems.
Features of *csh*

- Uses C type programming syntax, the language ***UNIX*** operating system is written in, but has a more awkward input/output implementation.
- Job control methodology, so that you can reattach a job running in the background to the foreground.

- History feature which allows you to modify and do again previously executed commands.



b) ***bash*** **(Bourne again shell).** The Bash shell was developed by "Stephen Bourne at Bell Labs"
c) ***ksh*** **(Korn Shell)** The Korn Shell was developed by "David Korn at Bell Labs".

The above picture gives the information related to the list of service used to start/stop/restart/status all type of servers which are programmed with shell programming.

Deadlock

Deadlock occurs when two process are each waiting for a resource that the other has locked. The potential for deadlock exists if a process that controls a locked region is put to sleep when it tries to lock another region that is controlled by a different process

TCP/IP Reference Model

When compared to OSI Model, TCP/IP has four layers
a)**Host to Network Layer** (Physical +Data link)
b)**Internet Layer** (Internet)
Internet layer has
IPV4
IPV6
c)**Transport Layer**
Two end to end protocols defined in transport layer are
User Datagram Protocol and
Transmission control Protocol
d)**Application Layer**
The important protocols in Application layer are
SNMP (Simple Network Management Protocol)
RTP (Real time transfer Protocol)
RTCP (Real time Control Protocol)
SMTP (Simple Mail Transfer Protocol)
Telnet (Virtual Terminal Protocol)
FTP (File Transfer Protocol)
SIP (Session Initiation Protocol)
DNS (Domain Naming Service)
HTTP (Hyper Text Transfer Protocol)

When compared to OSI Model Presentation and session layer are not present in TCP/IP model. since they are little use to Application layer and from the experience of OSI model the above two layers are not present in OSI Model

OSI Model

Open system Interconnection Model (OSI) Model which is applicable to systems which are open for communication with other systems.

OSI Model has seven Layers. They are

1. **Physical layer** is concerned with transmitting raw bits over a communication channel
2. **Data link layer** is concerned with error correction deduction and recovery
3. **Network layer** is concerned with controlling the operation of subnet
4. **Transport layer** is true end to end layer from source to destination
5. **Session layer** allows user on different machines to establish sessions between them. The main service is token management and synchronization
6. **Presentation layer** is concerned with syntax and semantics of the information transmitted
7. **Application layer** - all virtual terminal software and the file transfer are handled by application layer

Strace and Truss - Diagnostic tool

strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them. Students, hackers and the overly-curious will find that a great deal can be learned about a system and its system calls by tracing even ordinary programs. And programmers will find that since system calls and signals are events that happen at the user/kernel interface, a close examination of this boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions.

Use **ps -aef** to identify the process and use **strace -p** process_id to identify the low level commands of the respective process

strace is the diagnostic tool used in linux.
truss is used to trace the system/library calls (not user calls) and signals made/received by a new or existing process. It sends the output to stderr

Find (To search for files/directory)

1. *To find all the files in a system regardless of the case*
To search the files throughout the system

find / -iname "fileName"

[shankar@dhcppc0 ~]$ find . -iname cute -type d
./archive/Cute
./archive/cute
In the above instance, find searches for both 'Cute and also 'cute'

To search the files from the specific path

find /usr -iname "fileName"

Here -iname will search for the files irrespective of case

2. To find all the files whose file size is zero

find / -empty
find ~ -empty

find . -empty

In the above instance '/' stands for to search from the '/' path
and '~' refers to search from home path.i.e if u have logged in as user, then the search path is '/home/user'
and '.' refers to the current directory

[shankar@dhcppc0 archive]$ find . -empty
./example.c
./section.c
./class.c
./sanjeev.c
./Makefile
[shankar@dhcppc0 archive]$ ls -l
total 0
-rw-rw-r-- 1 shankar shankar 0 2009-06-13 14:50 class.c
-rw-rw-r-- 1 shankar shankar 0 2009-06-13 14:50 example.c
-rw-rw-r-- 1 shankar shankar 0 2009-06-13 14:51 Makefile
-rw-rw-r-- 1 shankar shankar 0 2009-06-13 15:02 sanjeev.c
-rw-rw-r-- 1 shankar shankar 0 2009-06-13 14:51 section.c

3. ***To list all the files that are greater than Kb***

[shankar@dhcppc0 archive]$ find ~ -size +2500k | more
/home/shankar/qt4-x11-4.3.2-4.fc8.i386.rpm
/home/shankar/.local/share/Trash/files/xine-lib-1.1.8.tar.gz
/home/shankar/.local/share/Trash/files/flash-plugin-9.0.48.0-release.i386.rpm
/home/shankar/.local/share/Trash/files/skype-1.4.0.118-fc5.i586.rpm

4. To find the file size greater than Mega bytes

[shankar@dhcppc0 archive]$ find ~ -size +25M | more
/home/shankar/jabbin-2.0beta.tar
[shankar@dhcppc0 archive]$ ls -lh /home/shankar/jabbin-2.0beta.tar
-rw-rw-r-- 1 shankar shankar 31M 2007-12-26 06:48 /home/shankar/jabbin-2.0beta.tar

5. To find the size equivalent to 250K
find ~ -size 250k
/home/shankar/Projects/iptables/autom4te.cache/output.1
[shankar@dhcppc0 archive]$ ls -lh /home/shankar/Projects/iptables/autom4te.cache/output.1
-rw-rw-r-- 1 shankar shankar 250K 2008-01-09 03:23
/home/shankar/Projects/iptables/autom4te.cache/output.1

6. *To list all the files that are less than 500b*

find / -size -500b

7.. *To find the directory named 'cute' in the given path*

find . -name "cute" -type d

[shankar@dhcppc0 ~]$ find . -name cute -type d
./archive/cute

Linux Operating System

Operating system which makes the hardware live and which acts as an interface between the hardware and user. Linux operating system is multi user , multi tasking operating system.
multi user operating system
Linux is multi user operating system. more than one user can simultaneously login to the same system
system administrator is having privilege to create users depend upon the needs. person who is logging with root is called as administrator and admin user can give privilege to other users depend upon the needs
multi tasking operating system
More than one task can be performed at same time in linux Operating system. one task can be run in the background and at the same time another process can be run in the foreground
In default Linux has the hierarchial file system and each directory is meant for specific purpose and we can see the purpose and list of files in detail
bin -binary and executable files can be found here

1. boot -bootable images will be here
2. dev-device file information will be available here
3. etc password and host information will be available here
4. home user's will be alloted space here
5. lib library information will be available here
6. lost+found
7. media
8. misc
9. mnt mountable cdrom and floppy information
10. net
11. opt
12. proc
13. root -Admin user's directory
14. sbin
15. selinux
16. srv
17. sys
18. tmp
19. usr
20. var

1 What POSIX stands for ?
Portable operating system compatible to UNIX operating system

2. What is the purpose of POSIX or why they use this name ?
POSIX is used to represent that operating system kernel is similar to unix. In other words once user is comfortable with UNIX operating system he may easy to operate in Linux operating system and vice versa

3. How to search for a file in Linux system ?

User can use the find command.

For example: Find / -name "*.c"

This command will search for all the .c files in Linux system here '/' stands for path . User can replace this path based on his convenience

Snippet of Command output

4. How to search for a directory in Linux system ?

User can use the same command by including the type option as -d.

Snippet of Command output

[root@dhcppc0 ~]# find / -name "tmp" -type d | more

File Types in POSIX Environement

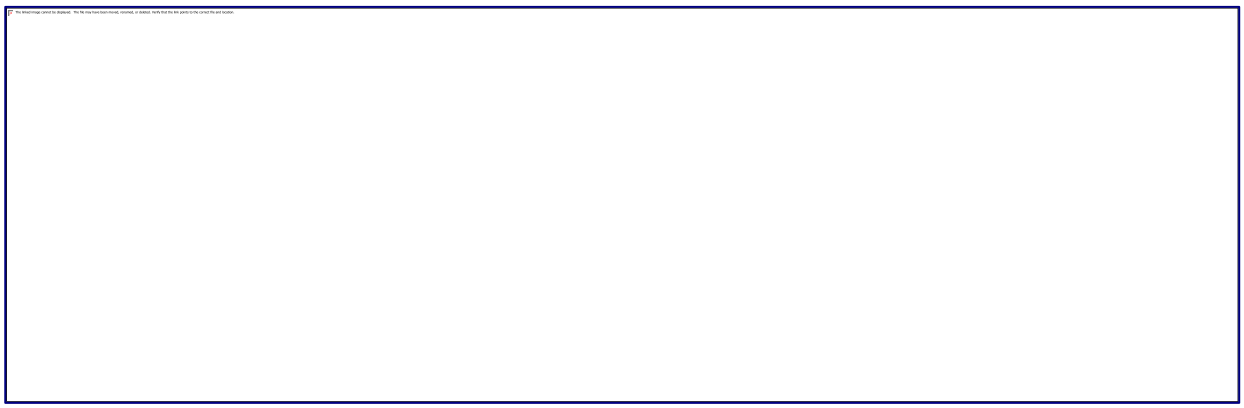
1. Regualar file
2. Directory file
3. Character specail file
4. FIFO
5. Block special file
6. Socket
7. Symbolic Fil

Linux Interview Question and Answer I

Posted: June 12, 2013 in Interview, Model I

0

Rate This



The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

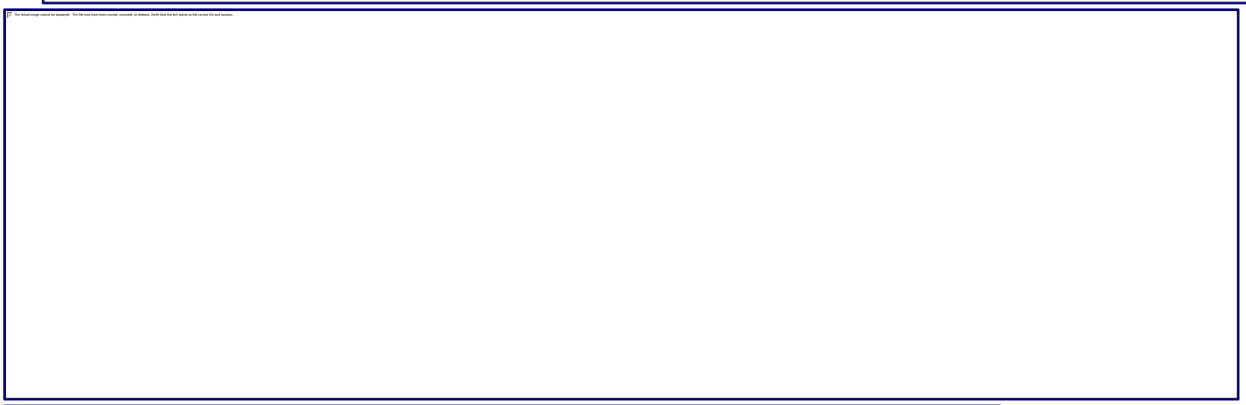
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

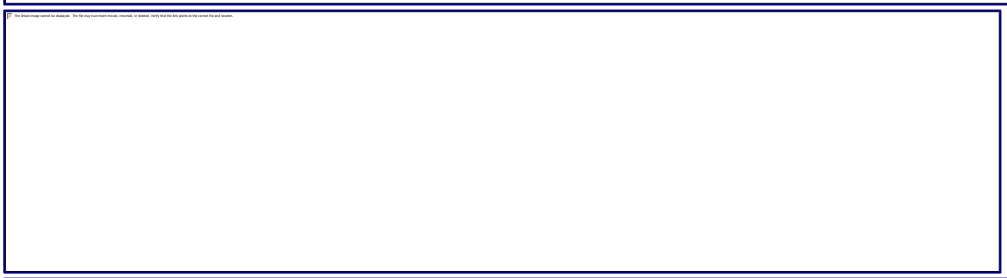
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.





Linux Interview Questions and Answers

Posted: June 12, 2013 in Interview, Model III

1

1 Vote

Real Time Based Questions in Redhat Linux

1. Linux Boot Process
2. Linux Partition
3. Linux File System Hierarchy
4. Important Configuration files Linux
5. /Proc
6. Server configuration file s and Packages
7. Monitoring Commands
8. Port Numbers In Linux

9.Linux Logs

Linux Boot Process (Startup Sequence)

Linux Partition for OS installation

/

/boot

Swap

/home

/var

/etc

Linux Directory Structure/ File System

Linux Server and Port number

20 – FTP Data (For transferring FTP data)

21 – FTP Control (For starting FTP connection)

22 – SSH(For secure remote administration which uses SSL to encrypt the transmission)

23 – Telnet (For insecure remote administration

25 – SMTP(Mail Transfer Agent for e-mail server such as SEND mail)

53 – DNS(Special service which uses both TCP and UDP)

68 – DHCP

69 – TFTP(Trivial file transfer protocol uses udp protocol for connection less transmission of data)

80 – HTTP/WWW (apache)

88 – Kerberos

110 – POP3(Mail delivery Agent)

123 – NTP(Network time protocol used for time syncing uses UDP protocol)

137 – NetBIOS(nmbd)

139 – SMB-Samba(smbd)

143 – IMAP

995 – POP3s

161 – SNMP(For network monitoring)

389 – LDAP(For centralized administration)

443 – HTTPS(HTTP+SSL for secure web access)

636 – ldaps(both tcp and udp)

873 – rsync

989 – FTPS-data

990 – FTPS

993 – IMAPS

2049 – NFS(nfsd, rpc.nfsd, rpc, portmap) **2401 – CVS server**

3306 – MySql

1**. /proc Directories with names as numbers**

Do a ls -l /proc, and you'll see lot of directories with just numbers. These numbers represents the process ids, the files inside this numbered directory corresponds to the process with that particular PID.

Following are the important files located under each numbered directory (for each process):

- cmdline – command line of the command.
- environ – environment variables.
- fd – Contains the file descriptors which is linked to the appropriate files.
- limits – Contains the information about the specific limits to the process.
- mounts – mount related information

Following are the important links under each numbered directory (for each process):

- cwd – Link to current working directory of the process.
- exe – Link to executable of the process.
- root – Link to the root directory of the process.
-

2. /proc Files about the system information

Following are some files which are available under /proc, that contains system information such as cpuinfo, meminfo, loadavg.

/proc/cpuinfo – information about CPU,

- /proc/meminfo – information about memory,
- /proc/loadvg – load average,
- /proc/partitions – partition related information,
- /proc/version – linux version

Some Linux commands read the information from this /proc files and displays it. For example, free command, reads the memory information from /proc/meminfo file, formats it, and displays it.

To learn more about the individual /proc files, do "man 5 FILENAME".

- **/proc/cmdline** – Kernel command line
- **/proc/cpuinfo** – Information about the processors.
- **/proc/devices** – List of device drivers configured into the currently running kernel.
- **/proc/dma** – Shows which DMA channels are being used at the moment.
- **/proc/fb** – Frame Buffer devices.
- **/proc/filesystems** – File systems supported by the kernel.
- **/proc/interrupts** – Number of interrupts per IRQ on architecture.
- **/proc/iomem** – This file shows the current map of the system's memory for its various devices
- **/proc/ioports** – provides a list of currently registered port regions used for input or output communication with a device
- **/proc/loadavg** – Contains load average of the system

The first three columns measure CPU utilization of the last 1, 5, and 10 minute periods.
The fourth column shows the number of currently running processes and the total number of processes.
The last column displays the last process ID used.

- **/proc/locks** – Displays the files currently locked by the kernel
  Sample line:
  1: POSIX ADVISORY WRITE 14375 08:03:114727 0 EOF
- **/proc/meminfo** – Current utilization of primary memory on the system
- **/proc/misc** – This file lists miscellaneous drivers registered on the miscellaneous major device, which is number 10
- /**proc/modules** – Displays a list of all modules that have been loaded by the system
- **/proc/mounts** – This file provides a quick list of all mounts in use by the system
- **/proc/partitions** – Very detailed information on the various partitions currently available to the system
- **/proc/pci** – Full listing of every PCI device on your system
- **/proc/stat** – Keeps track of a variety of different statistics about the system since it was last restarted
- **/proc/swap** – Measures swap space and its utilization
- **/proc/uptime** – Contains information about uptime of the system
- /proc/version – Version of the Linux kernel, gcc, name of the Linux flavor installed.

Linux Log Files

1. **/var/log/messages** – Contains global system messages, including the messages that are logged during system startup. There are several things that are logged in /var/log/messages including mail, cron, daemon, kern, auth, etc.
1. **/var/log/dmesg** – Contains kernel ring buffer information. When the system boots up, it prints number of messages on the screen that displays information about the hardware devices that the kernel detects during boot process. These messages are available in kernel ring buffer and whenever the new message comes the old message gets overwritten. You can also view the content of this file using the dmesg command.
1. **/var/log/auth.log** – Contains system authorization information, including user logins and authentication machinsm that were used.
1. **/var/log/boot.log** – Contains information that are logged when the system boots
1. **/var/log/daemon.log** – Contains information logged by the various background daemons that runs on the system
1. **/var/log/dpkg.log** – Contains information that are logged when a package is installed or removed using dpkg command
2. **/var/log/kern.log** – Contains information logged by the kernel. Helpful for you to troubleshoot a custom-built kernel.
1. **/var/log/lastlog** – Displays the recent login information for all the users. This is not an ascii file. You should use lastlog command to view the content of this file.
1. **/var/log/maillog /var/log/mail.log** – Contains the log information from the mail server that is running on the system. For example, sendmail logs information about all the sent items to this file
1. **/var/log/user.log** – Contains information about all user level logs
1. **/var/log/Xorg.x.log** – Log messages from the X
1. **/var/log/alternatives.log** – Information by the update-alternatives are logged into this log

file. On Ubuntu, update-alternatives maintains symbolic links determining default commands.

1. **/var/log/btmp** – This file contains information about failed login attemps. Use the last command to view the btmp file. For example, "last -f /var/log/btmp | more"
1. **/var/log/cups** – All printer and printing related log messages

1. **/var/log/anaconda.log** – When you install Linux, all installation related messages are stored in this log file

1. **/var/log/yum.log** – Contains information that are logged when a package is installed using yum
1. **/var/log/cron** – Whenever cron daemon (or anacron) starts a cron job, it logs the information about the cron job in this file
1. **/var/log/secure** – Contains information related to authentication and authorization privileges. For example, sshd logs all the messages here, including unsuccessful login

/var/log/wtmp or **/var/log/utmp** – Contains login records. Using wtmp you can find out who is logged into the system. who command uses this file to display the information.

1. **/var/log/faillog** – Contains user failed login attemps. Use faillog command to display the content of this file.

Apart from the above log files, /var/log directory may also contain the following sub-directories depending on the application that is running on your system.

- **/var/log/httpd/** (or) **/var/log/apache2** – Contains the apache web server access_log and error_log
- **/var/log/lighttpd/** – Contains light HTTPD access_log and error_log
- **/var/log/conman/** – Log files for ConMan client. conman connects remote consoles that are managed by conmand daemon.
- **/var/log/mail/** – This subdirectory contains additional logs from your mail server. For example, sendmail stores the collected mail statistics in /var/log/mail/statistics file
- **/var/log/prelink/** – prelink program modifies shared libraries and linked binaries to speed up the startup process. /var/log/prelink/prelink.log contains the information about the .so file that was modified by the prelink.
- **/var/log/audit/** – Contains logs information stored by the Linux audit daemon (auditd).
- **/var/log/setroubleshoot/** – SELinux uses setroubleshootd (SE Trouble Shoot Daemon) to notify about issues in the security context of files, and logs those information in this log file.
- **/var/log/samba/** – Contains log information stored by samba, which is used to connect Windows to Linux.
- **/var/log/sa/** – Contains the daily sar files that are collected by the sysstat package.
- **/var/log/sssd/** – Use by system security services daemon that manage access to remote directories and authentication mechanisms.

Trouble shooting In Linux

Getting ram information

cat /proc/meminfo

cat /proc/meminfo | head -n 1

Another fun thing to do with ram is actually open it up and take a peek. This next command will

show you all the string (plain text) values in ram.

sudo dd if=/dev/mem | cat | strings

Getting cpu info

Sometimes in troubleshooting we want to know what processor we are dealing with along with how much cpu is currently being used by our OS and programs. We can do this with these two commands.

cat /proc/cpuinfo

top

Check the temperature of your CPU

Keeping a computer within a safe temperature is the key to maintaining a stable system.e

List PCI and USB devices

To list all the PCI devices in your system issues the following command:

lspci

For USB use:

lsusb

Check out how much hard drive space is left

df -h

See what hard drives are currently detected

It is often times helpful to know what hard drives are connected to a system and what name was given them in the Linux directory. This info allows us to mount the hard drive and manipulate it.

sudo fdisk -l

Installed Programs

Packages

Ever want to find all the packages that are installed on your system? You can find all the packages and also find out why they are on your system. You can even determine what packages depend on them if any.

Find all installed packages

dpkg –get-selections | less

Find out why a packages is installed and what depends on it

aptitude why packagename

Find out where the package stores all of its files

dpkg -L packagename

Kill a process

ps -A | grep ProgramName

kill 7207

Miscellaneous

Go to a terminal

Ctrl + Alt + f3

return with, Ctrl + Alt + f7

Show all network connections

There are many great network scanners and assessment tools available for Linux but **netstat** is a very easy to use often a first step in troubleshooting network issues. We will leave the rest of the network tools for a later article as there is so much to cover.

netstat

List all files that are currently open on the system

This command will allow you to see all the files that are currently open on your system. Limiting the directory or coupling this command with grep is often useful for finding files that are still open restricting the ability to unmount a device. **Lsof** will also ouput the process id or PID. You can then kill the process using the **kill** command above.

lsof

Keep an eye on something for awhile

The watch command will repeat a command at a set interval (default 2 seconds) and output the response. This is useful for watching directories that change, watching hard drives fill up when a lot of data is being transfered, or using it with **lsusb** to watch for USB devices being plugged in.

watch ls

watch df -h

Find where a binary is stored and its libraries

Often times when running a cron command you want to include the absolute path to the command. Sometimes I run scheduled PHP tasks. This can be acomplished by using the '**whereis**' command.

whereis php5

Logs

See if you have kernel boot issues

dmesg | less

For more logs just cd into the **/var/log** directory and start using, **cat**, **less**, **tail**, **grep**, **find** or any other tool to view and search.

Linux Network troubleshooting step by step

Here we go into the topic.

1- First check that your interface (Network adapter) is enabled or not using: **ifconfig**

2- To make sure there is no internal problem.

ping to the loop back address **ping 127.0.0.1**

if there is no response **service network restart** if same repeats check network settings again.

3- check cable problem from **ethtool eth0**

if everything is fine then last line will show

```
Link detected: yes
```

if link is not detected plug it or change the cable according to need and problem

4-check the gateway settings in

/etc/network and
/etc/sysconfig/network-scripts/ifcfg-eth0

And check DNS settings in

/etc/resolv.conf

system-config-network

netstat

route

service iptables stop

(only disable the firewall for testing and dont forget to turn it on)

check boot messages if eth card is detected at boot time or not

cat /var/log/dmesg | grep -i eth0

or

dmesg | grep -i eth0

to check table of network interfaces

netstat -i

for more advanced troubleshooting

lspci | less
or
lspci | grep ethernet

to check all PCI buses and devices connected to them

These are enough to troubleshoot if still problem persist try installing drivers , check kernel related problems , check is there any need to recompile the kernel etc