|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | Prepare   1-Linked List all possible Q's <https://www.geeksforgeeks.org/recently-asked-interview-questions-in-product-based-companies/#LinkedList> -> solve live     * 1. Look at all programming Intervoew Q's -> stack,tree, array, string , Bit magic , searching and sorting  <https://www.geeksforgeeks.org/recently-asked-interview-questions-in-product-based-companies/>     <https://www.geeksforgeeks.org/top-20-linked-list-interview-question/>  <https://practice.geeksforgeeks.org/explore/?company%5B%5D=Adobe&page=1&company%5B%5D=Adobe>  <https://practice.geeksforgeeks.org/explore/?company%5B%5D=Qualcomm&page=1&company%5B%5D=Qualcomm>    Memcopy: <https://www.geeksforgeeks.org/write-memcpy/>    Decimal to binary: <https://www.geeksforgeeks.org/decimal-equivalent-of-binary-linked-list/>     * 1. <https://practice.geeksforgeeks.org/courses/Amazon-Test-Series> online test series.        * 1. C -> **Bitwise, Strin**g, arrays, storage classes(extern and static), pointers , function pointers, array of function pointers, struct union, **macros**, stack, queue , debugging GDB   Sorting and searching technoquies  Sorting : Bubble, selection, insertion, merge  Search: Binary,  String: strstr, reverse word of a line  malloc - internel operation <https://stackoverflow.com/questions/7202069/how-can-i-know-the-allocated-memory-size-of-pointer-variable-in-c> bare metal alloc:       * 1. Linux system Programming : Multi-Threading , IPC (shared memory, Message Queues, fifo n pipe)   Semaphore, mutex, process states, system calls, kill, FM permissions(umask,chmod,chown) **MemoryManagement** …       * 1. Linux Internal and device driver : learn API  1- Char device driver - minor, minor, fops, device file   2- Ioctl  3- Interrupts -> tasklet, softirq, workqueue (all API's and flags )  4- Kthread -> programming , API, return value  5- Synchronization : Spinlock, mutex, semaphore and their variants (all API)  6- scheduling  7- memory management -> Paging, swapping , Demand Page, Page fault, TLB, page table  -> check interview Q's in gfg  Priority inversion       * 1. I2C, SPI, UART   2. Project   + Shutdown   + Kernel Upgrade- kdump ,dtbo, boot, make points of all the things -><https://www.dedoimedo.com/computers/crash-analyze.html#mozTocId877838>     9-= Platform driver notes, how a device get detected when inserted ?  10 Device tree - dts dtsi, dtbo   * + GPIO   + Table      Description automatically generated   + Static Analysis Issues   1. Variable initialization   2. Null pointer dereference   3. Array initialization   4. Array boundary check   5. Memset to pointer -> memset(&dest, 0, size)   6. Correct return value and type      * 1. Y2038:   The start date for computer calendars is January 1, 1970, and the idea is they \*theoretically\* count down in seconds to infinity. However, 32-bit processors are only physically capable of counting up to 2,147,483,647. So the idea is that, at **03:14:07 UTC On Jan 19, 2038**, these processor will reach max capacity.    Changes:  **struct** timespec64 ts  *Timeval*  Timespec  Gettimeofday  Settimeofday  Lock\_gettime  Tm  Utimes  Mktime  Ctime  Stat  Fstat  Ffstat  Utime  timeb |
|  |  | <https://www.geeksforgeeks.org/c-program-to-find-decimal-equivalent-of-binary-linked-list/>  **C Program To Find Decimal Equivalent Of Binary Linked List**    *From <*[*https://www.geeksforgeeks.org/c-program-to-find-decimal-equivalent-of-binary-linked-list/*](https://www.geeksforgeeks.org/c-program-to-find-decimal-equivalent-of-binary-linked-list/)*>* |  |  |  |
|  |  |  |  |  |  |
|  |  | Juniper N/w Interview Q's  <https://www.geeksforgeeks.org/tag/juniper-networks/>  <https://practice.geeksforgeeks.org/explore/?company%5B%5D=Atlassian&company%5B%5D=Juniper%20Networks&page=1&company%5B%5D=Atlassian&company%5B%5D=Juniper%20Networks>  <https://www.geeksforgeeks.org/reverse-words-in-a-given-string/>   <https://www.geeksforgeeks.org/total-area-two-overlapping-rectangles/>  Memory leak – how to find leak during system start up    *From <*[*https://www.geeksforgeeks.org/juniper-interview-experience-set-5-sde111/*](https://www.geeksforgeeks.org/juniper-interview-experience-set-5-sde111/)*>*  <https://www.geeksforgeeks.org/merge-two-sorted-arrays/>   <https://www.geeksforgeeks.org/insertion-sort/>   <https://www.geeksforgeeks.org/why-quick-sort-preferred-for-arrays-and-merge-sort-for-linked-lists/>   <https://www.geeksforgeeks.org/toggle-odd-bits-number/>   <https://www.geeksforgeeks.org/write-memcpy/>  copy\_to\_user() and copy\_from\_user() do a bit of magic to ensure that it only copies from/to userspace, and then only from valid address ranges. If the userspace provided an invalid pointer and/or length, the functions will return the number of bytes that could not be copied. In real life, you check if they return zero, and if not, you return -EFAULT to indicate the pointer/length was bogus.    memcpy() does nothing of that sort. If the user provides a cleverly crafted pointer, memcpy() will happily copy kernel data; it does absolutely no sanity checks. copy\_to\_user() and copy\_from\_user() do all t**he necessary checks and preparations.**    In kernel sources, pointers received from userspace are usually annotated \_\_user. This is just make sure you (the kernel programmer) remembers to use copy\_to\_user()/copy\_from\_user() to reference the data pointed at.    The simple rule is: Whenever the pointer is specified by the userspace process, either indirectly or directly, you use copy\_to\_user()/copy\_from\_user(). You only use memcpy() with pointers internal to the kernel that are never supplied to userspace.    *From <*[*https://www.linuxquestions.org/questions/linux-kernel-70/memcpy-and-copy\_to-from\_user-930170/*](https://www.linuxquestions.org/questions/linux-kernel-70/memcpy-and-copy_to-from_user-930170/)*>* | | |  |
|  |  |  |  |  |  |
|  | Points taken from GFG interview exp     * 1. Deadlock, situation for deadlock, min thread/resource required for dead lock, how to overcome from deadlock, critical section      * 1. Difference between message passing and shared memory   2. WAP for power of two in C ->    if a number n is a power of 2 then bitwise & of n and n-1 will be zero.    <https://www.geeksforgeeks.org/c-program-to-find-whether-a-no-is-power-of-two/#:~:text=Another%20solution%20is%20to%20keep,is%20a%20power%20of%202>.     * 1. Bit map implementation / get empty position in 2000 bits / why char, why not int or uint      * 1. Swap nibbles aabbccdd -> ddccbbaa ->      * 1. Array of pointers and pointer to an array.   2. Little endian and Big Endian   Htopns ->   * + htons() converts host byte order to network byte order.   + Network byte order is Big-Endian and host byte order can be either Little-Endian or Big-Endian.   + On a Little Endian system htons() will convert the order of a multi-byte variable to Big-Endian.   + What will htons() do in case if the host byte order is also Big-Endian?   + Explain stack?   + What is segmentation fault / core dump ? Unauthorized access of memory, accessing freed memory, scanf issue.   How do you debug this senario ? Using GDB;   * 1. TOP:   + It provides a real time view of running system.   + Shows currently running list of proccess and managed by linux kernel   + Shows utilization of CPU , memory of each process and overall as well   + Display specific user process : top -n process\_name   + Count set bits in an integer   + Merge two sorted array and two sorted Linked List   + Decimal equvalent of binary LL | | |  |  |
|  |  |  |  |  |  |