Examination 2

200 Points

**Name:** Rohan Chaudhury

**Email Address:** [rohan.chaudhury@tamu.edu](mailto:rohan.chaudhury@tamu.edu)

**UIN:** 432001358

**List of people I have worked with:** Abhishek Sinha, Rohit Sah, Shubham Gupta, Sherine Davis Kozhikadan

**Question 1. (10 points) Is it possible for one process to update a portion of a file while another process is trying to read it? How do processes deal with this situation to avoid indeterminate results?**

**ANSWER:**

**Yes,** it is possible for one process to update a portion of a file while another process is trying to read it. The outcome of this scenario depends on what the writer does.

“If the writer overwrites the existing file, then the reader will see the new content when the writer overtakes the reader, if ever. If the writer and the reader proceed at variable speeds, the reader may alternatively see old and new content.

If the writer truncates the file before it starts to write, the reader will run against the end of the file at that point.

If the writer creates a new file and then moves the new file to the old name, the reader will keep reading from the old file. If an opened file is moved or removed, the processes that have the file opened keep reading from that same file. If the file is removed, it actually remains on the disk (but with no way to open it again) until the last process has closed it.

Unix systems tend not to have mandatory locks. If an application wants to ensure that its writer component and its reader component don't step on each other's toes, it's up to the developer to use proper locking. There are a few exceptions where a file that's open by the kernel may be protected from writing by user applications, for example, a loop-mounted filesystem image or an executable that's being executed on some UNIX variants.”

“Consistency semantics represent an important criterion for evaluating any file system that supports file sharing. These semantics specify how multiple users of a system are to access a shared file simultaneously. In particular, they specify when modifications of data by one user will be observable by other users. These semantics are typically implemented as code with the file system. Following are several prominent examples of consistency semantics.

**A. UNIX Semantics**

The UNIX file system uses the following consistency semantics:

• Writes to an open file by a user are visible immediately to other users who have this file open.

• One mode of sharing allows users to share the pointer of the current location into the file. Thus, the advancing of the pointer by one user affects all sharing users. Here, a file has a single image that interleaves all accesses, regardless of their origin.

In the UNIX semantics, a file is associated with a single physical image that is accessed as an exclusive resource. Contention for this single image causes delays in user processes.

1. **Session Semantics**

The Andrew file system (OpenAFS) uses the following consistency semantics:

• Writes to an open file by a user are not visible immediately to other users that have the same file open.

• Once a file is closed, the changes made to it are visible only in sessions starting later. Already open instances of the file do not reflect these changes.

According to these semantics, a file may be associated temporarily with several (possibly different) images at the same time. Consequently, multiple users are allowed to perform both read and write accesses concurrently on their images of the file, without delay. Almost no constraints are enforced on scheduling accesses.

1. **Immutable-Shared-Files Semantics**

A unique approach is that of immutable shared files. Once a file is declared as shared by its creator, it cannot be modified. An immutable file has two key properties: its name may not be reused, and its contents may not be altered. Thus, the name of an immutable file signifies that the contents of the file are fixed. The implementation of these semantics in a distributed system is simple because the sharing is disciplined (read-only).”

Reference: [1] <https://unix.stackexchange.com/questions/41668/what-happens-when-you-read-a-file-while-it-is-overwritten>

[2] Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin, pg -608, 609, 15.7 Consistency Semantics

Question 2. (10 points) Why does a paged memory environment pose a problem for most DMA devices?

**ANSWER:**

Direct memory access (DMA) is a process in which memory operations can be sped up by allowing an I/O (input/output) device to transmit or receive data directly to or from the main memory while bypassing the CPU. DMA transfers solve the problem of the CPU being occupied for the duration of the transfer. The CPU starts the transfer, then does other operations while the transfer process is running, and ultimately gets an interrupt from the DMA controller when the transfer is complete. Disk drives, graphics cards, network cards, and sound cards all use DMA.

**Issues:** Cache coherency issues might arise as a result of DMA. “If a CPU has both a cache and external memory, the data the DMA controller has access to (RAM) may not be updated with the proper data in the cache.”

“Because the DMA controller is decoupled from the CPU, it has no idea about any address translation the CPU might be doing and operates directly on physical addresses. When real-mode software attempts a read or write from an address in expanded memory, it will not be able to calculate the physical address correctly and the EMM (expanded memory manager) needs to intervene.”

Also, “Most DMA controllers use physical memory addresses for their transfers. Using physical addresses requires the operating system to convert the virtual address of the intended memory buffer into a physical address and write this physical address into the DMA controller’s address register. An alternative scheme used in a few DMA controllers is to write virtual addresses into the DMA controller instead. Then the DMA controller must use the MMU to have the virtual-to-physical translation done. Only in the case that the MMU is part of the memory (possible, but rare), rather than part of the CPU, can virtual addresses be put on the bus.”

So, in paged systems, the DMA will require the MMU to translate the virtual addresses to physical addresses. MMU resides in the CPU and the DMA will now have to include it to do its operation but the main purpose of DMA was to not include the CPU and to directly access the memory. This purpose would now not be possible in paged systems.

Reference:

[1] <https://www.techopedia.com/definition/2767/direct-memory-access-dma#:~:text=Direct%20memory%20access%20(DMA)%20is,a%20DMA%20controller%20(DMAC)>.

[2] <http://www.os2museum.com/wp/386-memory-managers-and-dma/>

[3] <http://gauss.ececs.uc.edu/Courses/c4029/lectures/dma.pdf>

[4]  Andrew S. Tanenbaum - Modern Operating Systems, Pg 346, 5.1.4 Direct Memory Access

Question 3. (10 points) Differentiate between physical device drivers and virtual device drivers.

ANSWER:

**Physical Device Driver:**

1. “A physical device driver is a driver for a specific piece of hardware.”
2. “These drivers are written by the device vendor. Physical device drivers (PDD), ensure that the operating system can execute the device-specific functions of the peripheral device.”
3. “All peripheral devices and plug-in cards need such a driver to be able to execute the functions. There are drivers for keyboards, printers, monitors and computer mouse and for any other peripheral device, as well as for communication, audio, video or graphics cards.”
4. “The drivers are provided by the manufacturing companies with the peripheral, or they can be downloaded from the Internet. In many cases, the operating system recognizes the connected hardware and can access its own device driver library. This hardware is plug-and-play devices.”
5. “Driver software is dependent on both the operating system and the current device version, and its functionality is constantly updated.”
6. “To keep the user abstracted from the underlying hardware, the logical device driver which is a high-level abstraction of how a device behaves communicates with the physical device driver on the user's behalf. There are various sorts of printers, for example, and each one usually comes with its drivers. The user communicates with the logical driver through the standard printer API, and the logical driver handles all communication with the physical driver. The physical device driver vendor is responsible for ensuring that their driver works with the logical device driver model.”

**Virtual Device Driver:**

1. “Virtual device drivers represent a particular variant of device drivers. They are used to emulate a hardware device, particularly in virtualization environments, for example when a DOS program is run on a Microsoft Windows computer or when a guest operating system is run on, for example, a Xen host.”
2. “Instead of enabling the guest operating system to dialog with hardware, virtual device drivers take the opposite role and emulate a piece of hardware, so that the guest operating system and its drivers running inside a virtual machine can have the illusion of accessing real hardware. Attempts by the guest operating system to access the hardware are routed to the virtual device driver in the host operating system as e.g., function calls.”
3. “The virtual device driver can also send simulated processor-level events like interrupts into the virtual machine.”
4. “Virtual devices may also operate in a non-virtualized environment. For example, a virtual network adapter is used with a virtual private network, while a virtual disk device is used with iSCSI. A good example of virtual device drivers can be Daemon Tools. There are several variants of virtual device drivers, such as VxDs, VLMs, and VDDs.”

“Device drivers are basically a set of modules/routines so as to handle a device for which a direct way of communication is not possible through the user's application program and these can be thought of as an interface thus keeping the system small providing for minimalistic of additions of code, if any.

**Physical device drivers** can’t perform all the logical operations needed in a system in cases like IPC, Signals, and so on.

The main reason for having **virtual device drivers** is to mimic the behavior of certain hardware devices without them actually being present and this could be attributed to the high cost of the devices or the unavailability of such devices.

These basically create an illusion for the users as if they are using the actual hardware and enable them to carry out their simulation results.

Examples could be the use of **virtual drivers** in case of Network simulators, also the support of virtual device drivers in case a user runs an additional OS in a virtual box kind of software.”

Reference:

[1] <https://www.techopedia.com/definition/4801/virtual-device-driver-vxd>

[2] <https://www.geeksforgeeks.org/device-driver-and-its-purpose/>

[3] <https://stackoverflow.com/questions/42367163/what-is-difference-between-physical-driver-and-virtual-driver>

[4] <https://www.pcds.co.in/iqa/why-do-we-need-virtual-device-drivers-when-we-have-physical-device-drivers.php>

[5] <https://www.itwissen.info/en/physical-device-driver-PDD-120784.html#gsc.tab=0>

**Question 4. (10 points) Why do variable length records incur more storage overhead than fixed length records?**

**ANSWER:**

A file that stores data is often thought of as being made up of records, each of which is made up of fields. There must be a means to discover the margins between adjacent records and the borders between adjacent fields within each record in order to read data from a file and interpret it appropriately. “Fixed length data files can be created with record lengths in the range from 5 bytes to 65,535 bytes. Variable-length data files are created with a defined minimum record length. The minimum record length specifies the smallest data record that will be added to the file. Conceptually, this minimum record length corresponds to the fixed-length fields that may be part of your record structure.” Now,

1. “Variable-length records require a little more processing and some extra code which is the cause behind incurring more storage overhead. Files of variable-length records require additional information to be stored in each record, such as **separator** **characters** and **field types**. This is not required in fixed-length records where we know the starting position and size of each field which are fixed.”
2. The following methods are often used to designate field boundaries in variable length records:
   1. Making use of delimiters: Between each pair of adjacent fields, make use of an additional delimiter character (or string, if required) by appending it after each field.
   2. Making use of indicators of field length: Taking the lengths of the record's variable length fields and storing them within each record. These lengths can be placed at the start of the record or one after the other before the relevant field.

A similar approach is required for inter-record boundaries, which necessitates the inclusion of the record length or the use of a delimiter within the entire record itself. This extra information within the records requires some extra space.

These are the main reasons for variable-length records incurring more storage overhead than fixed-length records.

Reference:

[1] <https://docs.faircom.com/doc/ctreeplus/30557.htm>

[2] <https://fac.ksu.edu.sa/sites/default/files/lecture3-disk_storage_basic_file_structures_and_buffer_management_0.ppt#:~:text=In%20a%20file%20of%20fixed,separator%20characters%20and%20field%20types>.

[3] <https://www.cs.scranton.edu/~mccloske/courses/cmps340/file_record_storage.html>

Question 5. (10 points) On your assigned account on the Texas Cyber Range: implement password aging on your account. Copy and paste the screen output from your VM to illustrate you have implemented aging and describe and justify the policy you selected.

ANSWER:

I have used chage command to implement password aging for the account ‘user’.

The chage command is used to update the expiration information of a user's password. The chage command modifies the time duration between password changes and the last password change date. The system uses this information to determine when a user's password needs to be changed.

Following are the justifications behind the policies and commands that I have used:

1. chage -l user: This command is just used to view the account aging information of the account ‘user’.
2. sudo chage -d 2022-04-17 user:

Details from the user manual of chage command :

The option “-d, --lastday LAST\_DAY is used to set the number of days since January 1st, 1970 when the password was last changed. The date may also be expressed in the format YYYY-MM-DD”

* I have used the command sudo chage -d 2022-04-17 user to set the last password changed date to 2022-04-17 (of the account ‘user’), i.e., the day on which I implemented the password aging policies since I wanted to set another policy which considers the maximum number of days during which a password is valid using the -M command which counts the number of days from the date set here. Details of the command are provided in the next bullet point.

1. sudo chage -M 90 user:

Details from the user manual of chage command :

The option “-M, --maxdays MAX\_DAYS is used to set the maximum number of days during which a password is valid. When MAX\_DAYS plus LAST\_DAY is less than the current day, the user will be required to change their password before being able to use their account. This occurrence can be planned for in advance by use of the -W option, which provides the user with an advance warning. Passing the number -1 as MAX\_DAYS will remove checking a password's validity.”

* I have used the sudo chage -M 90 user command to set the maximum number of days during which the current password is valid for the account ‘user’. I have selected 90 days or approximately 3 months as the time limit from the last password changed date for which the current password will be valid after which the user will have to change their password in order to use their account. Password thefts are quite common and so in order to better secure our accounts, we should change our passwords periodically after every few months. That is why I have implemented this policy.

1. sudo chage -W 10 user:

Details from the user manual of chage command :

The option “-W, --warndays WARN\_DAYS is used to set the number of days of warning before a password change is required. The WARN\_DAYS option is the number of days prior to the password expiring that a user will be warned their password is about to expire.”

* I have used sudo chage -W 10 user to set the number of days that the user will be warned before a password change is required as 10. This policy is implemented so as to give the user a warning and a fair amount of time to be able to change their password before their current password becomes invalid.

1. sudo chage -E 2023-04-25 user:

Details from the user manual of chage command :

The option “ -E, --expiredate EXPIRE\_DATE is used to set the date or number of days since January 1, 1970 on which the user's account will no longer be accessible. The date may also be expressed in the format YYYY-MM-DD (or the format more commonly used in your area). A user whose account is locked must contact the system administrator before being able to use the system again. Passing the number -1 as the EXPIRE\_DATE will remove an account expiration date.”

* I have used the command sudo chage -E 2023-04-25 user to set the date as 2023-04-25 on which the account “user” will no longer be accessible. This policy is implemented as these accounts are provided to students for a defined period of time (I have assumed that period as ~1 year) for their class purposes after which they will no longer be needing these accounts. So, their access to these accounts is removed after that period of time, and in case they are in need of access to these accounts they can contact the system administrator to regain access.

1. sudo chage -I 5 user:

Details from the user manual of chage command :

The option “-I, --inactive INACTIVE is used to set the number of days of inactivity after a password has expired before the account is locked. The INACTIVE option is the number of days of inactivity. A user whose account is locked must contact the system administrator before being able to use the system again. Passing the number -1 as the INACTIVE will remove an account's inactivity.”

* I have used the command sudo chage -I 5 user to set the number of days of inactivity after a password has expired before the account is locked as 5. This policy is implemented so as to give the user some amount of time before their account is locked after their password has expired.

Following are the required screenshots showing the implementation:

Text

Description automatically generated

Text

Description automatically generated

Reference:

[1] The user manual of chage command viewed using “man chage” command

**Question 6. (10 points)** **Regarding spin locks: in what way can spinning be useful in a multiprocessor? Why is spinning not useful in a uniprocessor system?**

**ANSWER:**

“A spinlock is a lock that causes a thread trying to acquire it to simply wait in a loop (‘spin’) while repeatedly checking whether the lock is available. Since the thread remains active but is not 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 (the one that holds the lock) blocks or ‘goes to sleep’.”

**“**When one uses regular locks (mutexes, critical sections, etc), the operating system puts our thread in the WAIT state and preempts it by scheduling other threads on the same core. This has a performance penalty if the wait time is really short because our thread now has to wait for a preemption to receive CPU time again. Spinlocks don't cause preemption but wait in a loop (‘spin’) till the other core releases the lock. This prevents the thread from losing its quantum and continues as soon as the lock gets released. The simple mechanism of spinlocks allows a kernel to utilize it in almost any state.**”**

As a result, spinlocks are only suitable **for SMP (symmetric multiprocessor) systems** in which the current spinlock owner can continue to operate on a separate processor. Other processes run on different processors in a multiprocessor system, changing the program state which leads to releasing the initial process from the spinlock. “Spinlocks are helpful in multiprocessor systems where a thread can run in a busy-loop (for a brief time) instead of being put in a sleep queue.”

“On multiple CPUs, spin locks work reasonably well (if the number of threads roughly equals the number of CPUs). The thinking goes as follows: imagine Thread A on CPU 1 and Thread B on CPU 2, both contending for a lock. If Thread A (CPU 1) grabs the lock, and then Thread B tries to, B will spin (on CPU 2). However, presumably the critical section is short, and thus soon the lock becomes available, and is acquired by Thread B. Spinning to wait for a lock held on another processor doesn’t waste many cycles in this case, and thus can be effective”

**In a uniprocessor system:** Since the condition that would release a process out of a spinlock can only be attained by executing a different process, spinlocks are not suited for single-processor systems. Other processes do not receive the chance to create the program condition necessary for the first process to advance if the first process does not yield the processor. Other processes run on other processors in a multiprocessor system, changing the program state and freeing the initial process from the spinlock.

“For spin locks, in the single CPU case, performance overheads can be quite painful; imagine the case where the thread holding the lock is pre-empted within a critical section. The scheduler might then run every other thread (imagine there are N − 1 others), each of which tries to ac- quire the lock. In this case, each of those threads will spin for the duration of a time slice before giving up the CPU, a waste of CPU cycles.”

Reference:

[1] <https://en.wikipedia.org/wiki/Spinlock>

[2] <https://stackoverflow.com/questions/1957398/what-exactly-are-spin-locks>

[3] <https://practice.geeksforgeeks.org/problems/why-spinlocks-are-not-appropriate-for-single-processor-systems-yet-are-often-used-in-multiprocessor-systems>

**Question 7. (10 points) Why is the TLB flushed after a context switch?**

**ANSWER:**

“A translation lookaside buffer (TLB) is a memory cache that stores the recent translations of virtual memory to physical memory. It is used to reduce the time taken to access a user memory location.”

1. “The TLB must be managed during a context switch. If the TLB entries contain an ASID (Address Space Identifier), they can remain in place; the operating system only needs to flush TLB entries if their ASID has changed meaning (e.g. because a process has exited). If the TLB entries are global, they must be invalidated when switching to a different context.”
2. “Each process has its own page table. The address of the page table is called the page table base register (PTBR). When we switch to a new process, we must update the PTBR to refer to the new process's page table (which is stored in the PCB). Because there is a new active page table, all entries in the TLB are no longer valid. Therefore the TLB must be flushed. As the new process runs, it will generate a large number of TLB misses until the pages it is actively using have been entered in the TLB. Handling TLB faults is expensive; this is one of the biggest overheads involved in context switching.”
3. “On an address-space switch, as occurs on a process switch but not on a thread switch, some TLB entries can become invalid, since the virtual-to-physical mapping is different. The simplest strategy to deal with this is to completely flush the TLB.”
4. “Flushing of the TLB can be an important security mechanism for memory isolation between processes to ensure a process can't access data stored in memory pages of another process. Memory isolation is especially critical during switches between the privileged operating system kernel process and the user processes – as was highlighted by the Meltdown security vulnerability. Mitigation strategies such as kernel page-table isolation (KPTI) rely heavily on performance-impacting TLB flushes and benefit greatly from hardware-enabled selective TLB entry management such as PCID.”
5. “Normally, entries in the x86 TLBs are not associated with a particular address space; they implicitly refer to the current address space. Hence, every time there is a change in address space, such as a context switch, the entire TLB has to be flushed.”
6. “If the TLB does not support separate ASIDs, then every time a new page table is selected with each context switch, the TLB must be flushed (or erased) to ensure that the next executing process does not use the wrong translation information. Otherwise, the TLB could include old entries that contain valid virtual addresses but have incorrect or invalid physical addresses leftover from the previous process. ”

Reference:

[1] <https://en.wikipedia.org/wiki/Translation_lookaside_buffer>

[2] <https://cs.stackexchange.com/questions/1088/what-happens-to-the-cache-contents-on-a-context-switch>

[3] <https://www.cs.cornell.edu/courses/cs4410/2015su/lectures/lec16-thrashing.html#:~:text=Context%20switching&text=Because%20there%20is%20a%20new,been%20entered%20in%20the%20TLB>.

[4] <https://stackoverflow.com/questions/28787964/flush-tlb-on-a-context-swtich>

[5] Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin, pg -366

Question 8. (10 points) Why does the operating system need to invalidate a page table entry in the TLB if a page is moved to secondary storage?

ANSWER:

“Translation Lookaside Buffer (TLB) is nothing but a special cache used to keep track of recently used transactions. TLB contains page table entries that have been most recently used. Given a virtual address, the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as index while processing page table.”

“There are a bunch of related operating system housekeeping-like tasks for which manipulating the valid bit is useful. For example:

1. Paging out memory requires detecting what pages have been recently used so that infrequently used pages can be paged out. One way to do this is to mark the page as invalid for a short time. If the page is used during that time, the operating system knows that it was accessed because a page fault occurred. (Modern Intel CPUs have a more efficient mechanism for this, but the valid bit works if you're using a CPU where that isn't supported.)
2. An operating system will often mark a dirty page (e.g. a modified page which represents a memory-mapped file) as invalid while it is being flushed to secondary storage. Any program which tries to modify a page that is in the process of being flushed should wait until the write is complete so that the data being written is in a known state.
3. Operating systems can use the validity bit to help debuggers. Many modern debuggers let you set a breakpoint that fires if a certain variable is read or modified. One way to implement this is through the page fault mechanism.

Basically, any time that it would be useful to handle a memory operation by trapping to the operating system is a time when it would be useful to mark a page as invalid.”

“The general concept behind demand paging is to load a page in memory only when it is needed. As a result, while a process is executing, some pages will be in memory, and some will be in secondary storage. Thus, we need some form of hardware support to distinguish between the two. The valid–invalid bit scheme can be used for this purpose. When the bit is set to ‘valid,’ the associated page is both legal and in memory. If the bit is set to ‘invalid,’ the page either is not valid (that is, not in the logical address space of the process) or is valid but is currently in secondary storage. The page-table entry for a page that is brought into memory is set as usual, but the page-table entry for a page that is not currently in memory is simply marked invalid. But what happens if the process tries to access a page that was not brought into memory? Access to a page marked invalid causes a page fault. The paging hardware, in translating the address through the page table, will notice that the invalid bit is set, causing a trap to the operating system.”

The invalid bit of a page table indicates whether the page is available in memory or storage. The invalid bit helps with demand paging since when an invalidated page is being tried to access, it triggers a trap, which results in the required page being fetched into memory. A page is moved to the secondary storage when it is not used recently or is unwanted. Now, TLB only contains the entries of the page table that have been used most recently. So if a page is moved to the secondary storage then it has not been used recently and so TLB does not need to keep track of that page. That is why the operating system “needs to invalidate a page table entry in the TLB if a page is moved to secondary storage”.

Reference:

[1] <https://www.geeksforgeeks.org/translation-lookaside-buffer-tlb-in-paging/>

[2] <https://www.tutorialspoint.com/operating_system/os_memory_management.htm>

[3]<https://eng.libretexts.org/Courses/Delta_College/Operating_System%3A_The_Basics/07%3A_Memory/7.6%3A_Paging>

[4] <https://cs.stackexchange.com/questions/66541/what-is-the-need-for-valid-invalid-bit-in-paged-memory-technique>

[5] Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin, pg -393, 10.2.1 Basic Concepts

Question 9. (10 points) Suppose a block mapping system represents a virtual address v = (*b*, *d*) using 32 bits. If block displacement *d* is specified using *n* bits, how many blocks does the virtual address space contain? Discuss how setting *n* = 6, *n* = 12 and *n* = 24 affects memory fragmentation and the overhead incurred by the mapping function.

ANSWER:

Virtual Address: v = (b, d)

Where,

b is the Virtual Page Number field in the virtual address space and d is the block displacement.

Now, v is a 32 bit address where the block displacement is using n bits.

So, (32-n) bits is the value of Virtual Page Number field and,

There would be 232-n number of blocks in the virtual address space. (Answer)

For n=6:

We have 232-6=226 = 67108864 blocks in the virtual address space with 26 as offset, that is, we have the page size as 26= 64bytes.

So in this case, the block/page size is small and so internal fragmentation is minimal. However, since the number of blocks in the virtual address space is very large so this would be impossible to implement as the block mapping table would consume too much memory.

For n=12:

We have 232-12=220 = 1048576 blocks in the virtual address space with 212 as offset, that is, we have the page size as 212= 4096bytes.

So, in this case, the page size and the number of blocks in the virtual address space are quite moderate. Internal fragmentation is more than in the n=6 case but the number of blocks in the virtual address space is lesser than in the n=6 case which makes this implementation practically feasible (as the block mapping table consumes less memory than in the n=6 case) and the most commonly used.

For n=24:

We have 232-24=28 = 256 blocks in the virtual address space with 224 as offset, that is, we have the page size as 224= 16777216bytes.

So, in this case, the page size is very large and so the internal fragmentation is also high. But since the number of blocks is not very large (=256), so not much memory would be consumed by the block mapping table.

So, we can see that, as the bits for offset increase, there is an increase in the page size and so an increase in internal fragmentation. However, with an increase in the number of bits for offset, there is a decrease in the number of blocks in the virtual address space and so the memory consumed by the block mapping table also decreases. “Larger page sizes lead to a large amount of wasted memory, as more potentially unused portions of memory are loaded into main memory. Smaller page sizes ensure a closer match to the actual amount of memory required in an allocation. As an example, assume the page size is 1024 KiB. If a process allocates 1025 KiB, two pages must be used, resulting in 1023 KiB of unused space (where one page fully consumes 1024 KiB and the other only 1 KiB). A page size that is too small introduces a larger page table entry increasing the lookup time and additional overhead of the TLB (Translation lookaside buffer) when addressing. Excessive page size wastes memory space causes memory fragmentation and reduces memory utilization. The above two factors were fully considered when designing memory page sizes in the last century, and a 4KB memory page was finally chosen as the most common page size for operating systems.”

Reference:

[1] <https://en.wikipedia.org/wiki/Page_(computer_memory)>

[2] <https://www.sobyte.net/post/2021-12/whys-the-design-linux-default-page/>

[3] <https://www.clear.rice.edu/comp425/slides/L31.pdf>

**Question 10. (10 points) Outline the operation of AES as defined in FIPS 197. In your outline, list each step in the execution of the algorithm. Explain in detail why or why not AES may be considered a Feistel Network. (Note: Simply cutting and pasting will likely result in a disappointing score.)**

**ANSWER:**

**🡪**The AES (Advanced Encryption Standard) Encryption algorithm which is also known by the name of Rijndael algorithm is a 128-bit symmetric block cipher algorithm.

🡪 These individual blocks in this algorithm are transformed using keys of 128, 192, and 256 bits. After encryption of these blocks, it combines them to generate the required ciphertext.

🡪It is based on the substitution-permutation network or SP network for short. It consists of a sequence of interconnected processes, such as substituting inputs for specified outputs (substitutions) and bit shuffling (permutations).

**Properties of AES:**

1. **SP Network:**

Instead of a Feistel cipher structure, such as in the case of the DES algorithm, AES uses an SP network structure.

1. **Key Expansion:**

During the initial stage, it just takes a single key, which is then enlarged to many keys utilized in individual rounds.

1. **Byte Data:**

Instead of bit data, the AES encryption method operates on byte data. As a result, throughout the encryption process, AES treats these 128-bit block sizes as 16 bytes.

1. **Key Length:**

The number of rounds to be performed is calculated by the length of the key used for encryption. For example, there are 10 rounds in a key of 128-bit, 12 rounds in a key of 192-bit, and 14 rounds in a key of 256-bit.

**Procedure in which AES works:**

* AES functions by delivering altered data in a certain order between several stages.
* Each block in the algorithm is 16 bytes long, and the data is held in a single block by a 4x4 matrix, and each cell contains a single byte of data.
* A state array is the matrix seen in the figure below.
* The original key is extended into keys of (n+1) in number, where n is the number of rounds in the encryption procedure to be followed.
* For example, the number of rounds for a key of 128-bit is 16, and the number of keys to be created is 10+1, for a total of 11 keys.

A screenshot of a game

Description automatically generated with low confidence

**The steps that are followed in AES are shown in the image below.**

Diagram

Description automatically generated

* The steps shown are to be performed in sequential order for each block.
* It combines the constituent blocks together to generate the final ciphertext after successfully encrypting them.
* The steps in the method are described as follows:

1. **Add Round Key:**
2. We use an XOR function to combine the block data in the state array with K0 (the first key created).
3. The produced state array is sent to the following phase as input.

A picture containing text, shoji, crossword puzzle, clock

Description automatically generated

1. **Sub-Bytes:**
2. This stage divides each byte of the corresponding state array into two equal partitions and transforms it to hexadecimal.
3. These are the rows and columns.
4. New entries for the final state array are produced by mapping these rows and columns with an S-Box (substitution box).

A picture containing graphical user interface

Description automatically generated

1. **Shift Rows:**

This option switches the row items around. In this step:

1. The very1st row is skipped
2. The components present in the 2nd row is moved by 1 position to the left.
3. The components in the third row are moved 2 places to the left
4. Elements in the last row are moved to the left by 3 positions.

A picture containing text, shoji, crossword puzzle, clock

Description automatically generated

1. **Mix Columns:**

In this step:

1. A new column for the next state array is obtained by multiplying a constant matrix by every column in the present state array.
2. The state array for the following step is obtained after multiplying all the columns with the same matrix.
3. This stage is not to be completed in the final round.

Diagram, table

Description automatically generated with medium confidence

1. **Add Round Key:**

The key in the round is XOR’d with the corresponding state array produced in the step prior to this.

Now:

1. If this round is the final round: Then the corresponding state array obtained as a result now becomes the corresponding ciphertext for the specified block
2. Otherwise, it is passed on as the new state array input for the following round.

A picture containing text, electronics

Description automatically generated

These are the basic steps of the AES Encryption algorithm.

**AES may not be considered a Feistel Network for the following reasons:**

A Feistel network utilizes a succession of rounds to divide the input block into two sides, permutes one side using the other side, and then it swaps the 2 sides as is shown in the diagram below.

Diagram

Description automatically generated

This is something that AES does not perform. The entire state must be permuted while performing a round in AES. AES is made up of the following steps: Sub Bytes, Shift Rows, Mix Columns, and Add Round Key, and all of these steps operate in a different manner than in a Feistel network:

1. In the **Sub Bytes** step, byte-by-byte substitution is done from a constant table and the value of one byte does not affect the permuted value of another byte.
2. In the **Shift Rows** step, only four-byte words are permuted at a single instance using only the specified four bytes and in this permutation process, no other word’s byte is used.
3. In the **Mix Columns** step, the four-byte words are permuted at a time with the usage of only these four bytes, and in the mentioned permutation process no other byte from any other word is used.
4. The **Add Round Key step** is basically a permutation that utilizes the “derived” round key. The value of one byte does not affect the permuted value of another byte.

Just the Shift Rows and Mix Columns stages enable a byte to impact "the permutation of any other bytes in the state," and in both of those stages, a specific byte can only influence the permutation of other bytes when it is undergoing permutation itself. None of this corresponds to a Feistel network's "divide the block into A and B and use A to permute B" method, so AES cannot be considered a Feistel Network.

Reference:

[1] <https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf?msclkid=35335aa3bd0a11ec95a4997d0cf6348b>

[2] <https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption>

[3] <https://crypto.stackexchange.com/questions/10605/why-is-aes-not-a-feistel-cipher>

**Question 11. (10 points)** **Feistel Networks “Choice of permutation classifies algorithms of this class.” Outline in DETAIL the operation of the TwoFish algorithm (**[**http://www.schneier.com/twofish.html**](http://www.schneier.com/twofish.html)**) in terms of a Feistal Network and explain why or why not TwoFish is a Feistel Network algorithm.**

**ANSWER:**

“Twofish is a symmetric block cipher; a single key is used for encryption and decryption. Twofish has a block size of 128 bits, and accepts a key of any length up to 256 bits. (NIST required the algorithm to accept 128-, 192-, and 256-bit keys.)”

“**Twofish architecture**

Twofish consists of a number of building blocks, such as the following:

* **Feistel network.** A method of transforming any function (F function) into a permutation that forms the basis of many block ciphers.
* **S-boxes.**Table-driven, nonlinear substitution operations. Twofish uses four precomputed, key-dependent, bijective, 8-by-8-bit S-boxes. They are commonly used in block ciphers.
* **Maximum distance separable (MDS) matrices.** A common feature of Reed-Solomon error-correcting codes. Twofish uses single 4-by-4 MDS matrix over Galois field(28).
* **Pseudo-Hadamard transform (PHT).**Simple mixing operation that runs quickly in software. Twofish uses a 32-bit PHT to mix the outputs from its two parallel 32-bit g functions.
* **Whitening.** The technique of XORing key material before and after the first and last rounds, respectively. Twofish XORs 128 bits of subkey before the first Feistel round and after the last Feistel round.
* **Key schedule.**The means by which key bits are transformed into round keys that can be used by the cipher. Twofish has a complicated key schedule.

**How Twofish works**

The encryption process in Twofish includes the following steps:

1. In each round, two 32-bit words act as inputs into the F function.
2. Each word is broken up into 4 bytes, which are then sent through four key-dependent S-boxes. These S-boxes have 8-bit I/O.
3. The MDS matrix combines the 4 output bytes into a 32-bit word.
4. The two 32-bit words are combined using a PHT.
5. In the subsequent step, they are added to two round subkeys and XORed into the right half.”

“A Feistel network implements a series of iterative ciphers on a block of data and is generally designed for block ciphers that encrypt large quantities of data. A Feistel network works by splitting the data block into two equal pieces and applying encryption in multiple rounds. Each round implements permutations and combinations derived from the primary function or key.”

“As illustrated in the figure below, Twofish is a Feistel network since, in each round, half of the text block is sent through an F function, and then XORed with the other half of the text block, making it a crucial determinant of the permutation's final permutation.” This shows that one half's permutation is modified by the other, indicating that it is a Feistel cipher.

Diagram, schematic

Description automatically generated

Reference:

[1] <https://www.techopedia.com/definition/27121/feistel-network>

[2] <https://www.schneier.com/academic/archives/1998/12/the_twofish_encrypti.html>

[3] <https://www.techtarget.com/searchsecurity/definition/Twofish>

**Question 12. (10 points) Consider the issue of magnetic remanence on magnetic storage media. What is the current state of the art in recovering bit patterns from a disk by analyzing the remaining magnetic patterns on the media? Be detailed in your answer.**

**ANSWER:**

“Magnetic force microscopy (MFM) is a recent (state-of-the-art) technique for imaging magnetization patterns with high resolution and minimal sample preparation. The technique is derived from scanning probe microscopy (SPM) and uses a sharp magnetic tip attached to a flexible cantilever placed close to the surface to be analyzed, where it interacts with the stray field emanating from the sample. An image of the field at the surface is formed by moving the tip across the surface and measuring the force (or force gradient) as a function of position. The strength of the interaction is measured by monitoring the position of the cantilever using an optical interferometer or tunneling sensor.

Magnetic force scanning tunneling microscopy (STM) is a more recent variant of this technique which uses a probe tip typically made by plating pure nickel onto a pre-patterned surface, peeling the resulting thin film from the substrate it was plated onto, and plating it with a thin layer of gold to minimize corrosion, and mounting it in a probe where it is placed at some small bias potential (typically a few tenths of a nanoamp at a few volts DC) so that electrons from the surface under test can tunnel across the gap to the probe tip (or vice versa). The probe is scanned across the surface to be analyzed as a feedback system continuously adjusts the vertical position to maintain a constant current. The image is then generated in the same way as for MFM. Other techniques which have been used in the past to analyze magnetic media are the use of ferrofluid in combination with optical microscopes (which, with gigabit/square inch recording density is no longer feasible as the magnetic features are smaller than the wavelength of visible light) and a number of exotic techniques which require significant sample preparation and expensive equipment. In comparison, MFM can be performed through the protective overcoat applied to magnetic media, requires little or no sample preparation, and can produce results in a very short time.

Even for a relatively inexperienced user, the time to start getting images of the data on a drive platter is about 5 minutes. To start getting useful images of a particular track requires more than a passing knowledge of disk formats, but these are well-documented, and once the correct location on the platter is found a single image would take approximately 2-10 minutes depending on the skill of the operator and the resolution required. With one of the more expensive MFM, it is possible to automate a collection sequence and theoretically possible to collect an image of the entire disk by changing the MFM controller software.”

“With the ever-increasing data density on disk platters and a corresponding reduction in feature size and use of exotic techniques to record data on the medium, it's unlikely that anything can be recovered from any recent drive except perhaps a single level via basic error-canceling techniques. In particular, the drives in use at the time that this paper was originally written are long since extinct, so the methods that applied specifically to the older, lower-density technology don't apply anymore. Conversely, with modern high-density drives, even if you've got 10KB of sensitive data on a drive and can't erase it with 100% certainty, the chances of an adversary being able to find the erased traces of that 10KB in 200GB of other erased traces are close to zero.

Any modern drive will most likely be a hopeless task, with ultra-high densities and the use of perpendicular recording MFM would not get a usable image, and then the use of EPRML will mean that even if we could magically transfer some sort of image into a file, the ability to decode that to recover the original data would be quite challenging.”

Reference:

[1] <https://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html>

**Question 13. (10 points) How does segmentation differ from variable-partition programming?**

**ANSWER:**

“**In** **Segmentation:**

1. A process is divided into Segments.
2. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments.
3. Segmentation gives the user’s view of the process which paging does not give. Here the user’s view is mapped to physical memory.
4. There are types of segmentation:
   1. **Virtual memory segmentation –**  
      Each process is divided into a number of segments, not all of which are residents at any one point in time.
   2. **Simple segmentation –**  
      Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.
5. There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

**Segment Table –** It maps two-dimensional Logical addresses into one-dimensional Physical addresses. It’s each table entry has:

* **Base Address:**Itcontains the starting physical address where the segments reside in memory.
* **Limit:** It specifies the length of the segment.

1. Address generated by the CPU is divided into:
   1. Segment number (s): Number of bits required to represent the segment.
   2. Segment offset (d): Number of bits required to represent the size of the segment.”

“**In variable-partition programming:**

1. It is one of the simplest methods of allocating memory where processes are assigned to variably sized partitions in memory, where each partition may contain exactly one process.
2. In this variable partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
3. Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
4. Eventually, the memory contains a set of holes of various sizes.
5. As processes enter the system, the operating system takes into account the memory requirements of each process and the amount of available memory space in determining which processes are allocated memory.
6. When a process is allocated space, it is loaded into memory, where it can then compete for CPU time.
7. When a process terminates, it releases its memory, which the operating system may then provide to another process.”

**Differences between Segmentation and variable-partition programming are as follows:**

1. Variable-partitioning provides a single continuous block of memory to the whole process, whereas segmentation associates many segments to that process.
2. Segmentation divides a process into various segments whereas in the variable partition scheme the memory is divided into different sized partitions and not the process.
3. There are 2 types of segmentation: Virtual memory segmentation and Simple segmentation, whereas there is only one type of variable-partitioning scheme.
4. In segmentation, a segment table “maps two-dimensional Logical addresses into one-dimensional Physical addresses” whereas the same methodology is not used in a variable-partitioning scheme.

**Reference:**

[1] Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin, pg 358, 9.2.2 Memory Allocation

[2] <https://www.geeksforgeeks.org/segmentation-in-operating-system/>

**Question 14. (10 points) In what ways do segmentation/paging systems incur fragmentation?**

**ANSWER:**“ Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. As programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use. The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called fragmentation.

There are mainly two types of fragmentation in the operating system. These are as follows:

1. Internal Fragmentation
2. External Fragmentation

The causes of the two types of fragmentation are as follows:

**Paging** allows us to store processes in memory in a discontinuous space. To implement this technique, we divide the processes into pages. Pages are blocks of fixed sizes. Similarly, we also divide physical memory into frames. When executing a process, the CPU converts each page’s logical address to a physical address. Then, using the segmented paging technique, we partition the address provided by the CPU into a page number and a page offset. We know that paging divides programs into fixed-size pages. However, at some point, it would occur that a program wouldn’t need precisely the whole page. Therefore, it can leave a free partition of the memory, leading to memory with allocated unused spaces**. This is precisely how internal fragmentation occurs in memory.** When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes internal fragmentation.

Let’s take an example. We’re taking a program that is divided into two blocks of equal size 2KB each. Hence, we allocate 4KB of non-contiguous space in the memory. But if a program needs only 3KB, then an allocated and unused 1KB space is left on memory, causing internal fragmentation:”

Diagram

Description automatically generated

“**External fragmentation** happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as external fragmentation.”

Chart

Description automatically generated

**“**The basic premise of **segmentation** says that segment sizes are variable. Every process is divided into a number of chunks where each chunk is called a **segment**. Each process is loaded by bringing all of its segments into the main memory. Every segment of the process is loaded into the main memory by creating partitions dynamically matching the size of each segment. This creates an exact fit for every segment. Segmentation is free of internal fragmentation but suffers from **external fragmentation.** Every program/process may occupy more than one non-contiguous segment, similar to dynamic partitioning. Consider a case where a larger segment is evicted and a segment that is smaller is put in its place. As the new segment is smaller it leaves an area in the segment which remains unused. **This is external fragmentation.** The 'holes' created by external fragmentation are dealt with by implementing a process called 'compaction'. This is a costly process with large overhead and hence must not be called very often.”

Reference:

[1] <https://www.javatpoint.com/fragmentation-in-operating-system>

[2] <https://www.baeldung.com/cs/internal-vs-external-fragmentation-paging>

[3] <https://gcallah.github.io/OperatingSystems/Segmentation.html>

**Question 15. (10 points) Compare the overhead incurred by shortest seek time first (SSTF) and First Come, First Serve (FCFS).**

**ANSWER:**

**FCFS (First-Come-First-Serve) Disk Scheduling Algorithm:**

“FCFS (First-Come-First-Serve) is the easiest disk scheduling algorithm among all the scheduling algorithms. In the FCFS disk scheduling algorithm, each input/output request is served in the order in which the requests arrive. In this algorithm, starvation does not occur because FCFS addresses each request.

**Advantages of FCFS Disk scheduling Algorithm**

The advantages of the FCFS disk scheduling algorithm are:

1. In FCFS disk scheduling, there is no indefinite delay.
2. There is no starvation in FCFS disk scheduling because each request gets a fair chance.

**Disadvantages of FCFS Disk Scheduling Algorithm**

The disadvantages of the FCFS disk scheduling algorithm are:

1. FCFS scheduling is not offered as the best service.
2. In FCFS, scheduling disk time is not optimized.”

**We will consider the same example for both FCFS and SSTF Disk Scheduling Algorithms:**

“Input:

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

**Output:**

Total number of seek operations = 510

Seek Sequence is

176

79

34

60

92

11

41

114

The following chart shows the sequence in which requested tracks are serviced using FCFS.

**A picture containing text, skiing, slope, line

Description automatically generated**

**Therefore, the total seek count is calculated as:**

= (176-50)+(176-79)+(79-34)+(60-34)+(92-60)+(92-11)+(41-11)+(114-41)

= 510 ”

**SSTF (Shortest Seek Time First) Disk Scheduling Algorithm:**

“SSTF is another type of scheduling algorithm. In this type of disk scheduling, the job which has less seek time will be executed first. So, in SSTF (shortest seek time first) scheduling, we have to calculate the seek time first. and after calculating the seek time, each request will be served on the basis of seek time. The request which is close to the disk arm will be first executed. There are some drawbacks in FCFS. To overcome the limitations that arise in the FCFS. SSTF scheduling is implemented.

**Advantages of SSTF Disk Scheduling**

The advantages of SSTF disk scheduling are:

1. In SSTF disk scheduling, the average response time is decreased.
2. Increased throughput.

**Disadvantages of SSTF Disk Scheduling**

The disadvantages of SSTF disk scheduling are:

1. In SSTF, there may be a chance of starvation.
2. SSTF is not an optimal algorithm.
3. There are chances of overhead in SSTF disk scheduling because, in this algorithm, we have to calculate the seek time in advanced.
4. The speed of this algorithm can be decreased because direction could be switched frequently.”

**“Considering the same example for SSTF as in the FCFS case:**

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}   
Initial head position = 50

The following chart shows the sequence in which requested tracks are serviced using SSTF.

A picture containing text, skiing, slope

Description automatically generated

Therefore, total seek count is calculated as:

= (50-41)+(41-34)+(34-11)+(60-11)+(79-60)+(92-79)+(114-92)+(176-114)

= 204”

As we can see, the total seek count is lesser for SSTF (=240) than for FCFS (=510) algorithm for the above example. This is true in general. But, when compared to FCFS, SSTF has a considerable overhead in finding out the closest request. In SSTF, the seek requests have to be sorted based on the seek times before the controller takes any action.

Reference:

[1] <https://www.tutorialandexample.com/fcfs-disk-scheduling-algorithm>

[2] <https://www.geeksforgeeks.org/program-for-sstf-disk-scheduling-algorithm/?msclkid=36a38168bd1611ec88dbf9262f125f55>

[3] <https://www.geeksforgeeks.org/fcfs-disk-scheduling-algorithms/>

[4] <https://www.geeksforgeeks.org/difference-between-fcfs-and-sstf-disk-scheduling-algorithm/>

**Question 16. (10 points) For each of the following workloads, identify whether a cluster-based or a client–server DFS model would handle the workload best. Explain your answers.**

1. **Hosting student files in a university lab.**
2. **Processing data sent by the Hubble telescope.**
3. **Sharing data with multiple devices from a home server. (textbook)**

**ANSWER**

1. For hosting student files in a university lab, a client-server DFS model would be able to handle the workload best. Following are the reasons for that:
2. The number of students is moderate and so all their student records data can be very easily and efficiently stored locally on a single server. In the client-server DFS model “the server stores both files and metadata on attached storage”.
3. This single server can be handled by the faculty of the lab. Necessary security and authentication policies can be set by the server to protect the data. “The server is responsible for carrying out authentication, checking the requested file permissions, and if warranted, delivering the file to the requesting client.”
4. “Clients are connected to the server through a network and can request access to files in the DFS by contacting the server through a well-known protocol such as NFS Version 3.” The clients, in this case, are the computers of the students and they can easily get connected to the common server through the network and can then request access to their files.
5. Multiple students can get connected to the server at the same time as this functionality is supported by the client-server DFS model.
6. For processing data sent by the Hubble telescope, a cluster-based DFS model would be able to handle the workload best. Following are the reasons for that:
   1. “As the amount of data, I/O workload, and processing expand, so does the need for a DFS to be fault-tolerant and scalable. Large bottlenecks cannot be tolerated, and system component failures must be expected. The cluster-based architecture was developed in part to meet these needs.” Since the data sent by the Hubble telescope would be enormous, a single server would not be enough for storing and processing the data and hence a cluster of machines/servers would be ideal for this task.
   2. In the cluster-based DFS model, “each file chunk is stored on a data server and is replicated a certain number of times (for example, three times) to protect against component/hardware failure (which is very common) and for faster access to the data (servers containing the replicated chunks have fast access to those chunks).” Since the Hubble telescope data is large and important, there is a need for faster access to the data and its safety. With data replication for a certain number of times, both these requirements can be met in the cluster-based DFS model.
   3. “One or more clients are connected via a network to a master metadata server and several data servers that house “chunks” (or portions) of files. The metadata server keeps a mapping of which data servers hold chunks of which files, as well as a traditional hierarchical mapping of directories and files.” Several clients can parallelly and swiftly access parts of the otherwise huge data as per their requirements and do their own processing of the data in an efficient and faster manner.
7. For sharing data with multiple devices from a home server, a client-server DFS model would be able to handle the workload best. Following are the reasons for that:
8. The data to store and share in this case is not that large, so the data can be very easily and efficiently stored locally on a single server in the client-server DFS model.
9. The number of devices accessing the data is also not that large in amount, so the client-server model can easily handle the moderate number of clients in the network.
10. The associated task is not that huge, and there are not many issues regarding scalability, faster processing, and fault tolerance associated with it which are the main target points of setting up a cluster-based DFS model. The associated requirements for the task can be easily managed by a client-server DFS model. Besides, a cluster-based DFS model is too expensive to set up for such a small task.

**Reference:**

[1] Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin, pgs 758, 759, 19.6.1 The Client–Server DFS Model, 19.6.2 The Cluster-Based DFS Model

**Question 17. (10 points)**

1. **List three characteristics that make Beowulf clusters high performance**
2. **What problem does a high availability cluster with shared storage have?**

**ANSWER:**

**(a)** “A Beowulf cluster is a computer cluster of what are normally identical, commodity-grade computers networked into a small local area network with libraries and programs installed which allow processing to be shared among them. The result is a high-performance parallel computing cluster from the inexpensive personal computer hardware.”

The three characteristics that make Beowulf clusters high performance is as follows:

1. “A Beowulf cluster uses a multicomputer architecture. It features a parallel computing system that usually consists of one or more master nodes and one or more compute nodes, or cluster nodes, interconnected via widely available network interconnects. All of the nodes in a typical Beowulf cluster are commodity systems–PCs, workstations, or servers–running commodity software such as Linux. Free operating systems, such as Linux, are available, reliable, and well-supported, and are distributed with complete source code, encouraging the development of additional tools including low-level drivers, parallel file systems, and communication libraries.” Since it is made of commodity-grade computers which are essentially connected via a local area network, it is **highly scalable** as it would be very easy to add more compute resources to it.
2. “Commodity networking, especially Fast Ethernet, has made it possible to design distributed-memory systems with **relatively high bandwidths** and tolerably low latencies at low cost. With the power and low prices of today’s off-the-shelf PCs and the availability of 100/1.000 Mb/s Ethernet interconnect, it makes sense to combine them to build HighPerformance-Computing and Parallel Computing environment.”
3. “With free versions of Linux and public domain software packages, no commercially available parallel computing system can compete with the price of the Beowulf system.Another key component contributing to **forward compatibility is the system software** used on Beowulf. With the maturity and robustness of Linux, GNU software and the ‘standardization’ of message passing via PVM and MPI, programmers now have a guarantee that the programs they write will run on future Beowulf Clusters, regardless of who makes the processors or the networks.”
4. “One of the main differences between a Beowulf Cluster and a COW (Cluster of Workstations) is the fact that Beowulf behaves more like a single machine rather than many workstations. The nodes in the cluster don’t sit on people’s desks; they are dedicated to running cluster jobs. It is usually connected to the outside world through only a single node”

“That said the main characteristics of a Beowulf Cluster can be summarized in the following points:

– Very high performance-price ratio.

– Easy scalability.

– Recycling possibilities of the hardware components.

– Guarantee of usability/upgradeability in the future.”

Reference:

[1] <https://en.wikipedia.org/wiki/Beowulf_cluster>

[2] <https://subs.emis.de/LNI/Proceedings/Proceedings13/44_BeowulfClusterforHigh-PerfComp.pdf>

[3] <https://www.pcquest.com/high-performance-computing-beowulf-clusters/?msclkid=b674c3d1bd1f11ecbdb2ce1a4b4f3827>

**(b)** “High availability (HA) is a component of a technology system that eliminates single points of failure to ensure continuous operations or uptime for an extended period. High Availability ensures your systems, databases, and applications operate when and as needed. Imagine running dozens of nodes that depend on a single database server for their functionality. Regardless of the number of nodes, the failure of one node does not affect the persistent state of the others. But if the database fails, the entire cluster is unusable. Therefore, the database is a single point of failure. This is known as a shared disk cluster.”

“High availability (HA) aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.”

“Shared storage is a medium accessible by all of the subscribers in a network, intended for file storage and allowing simultaneous access by multiple subscribers without the need to duplicate files to their computers. Because only one file server exists containing all the data, backup and archiving processes are streamlined, and the risk of conflicting backups or archives is eliminated.”

So we can say that if the shared storage faces an issue then all the other systems in the cluster which rely on the aforementioned shared storage will be affected. So, in order to build a genuinely Highly Available (HA) system, we need to remove this shared storage which is a "single point of failure". This problem can be solved if the shared storage can be duplicated and all the duplications of the shared storage can be made accessible to all the machines in the cluster. However, the issues associated with this approach are:

i. Increased complexity and cost

ii. More space would be required

Reference:

[1] <https://us.sios.com/what-we-do/high-availability/>

[2] <https://en.wikipedia.org/wiki/High_availability>

[3] <https://cloud.netapp.com/blog/cvo-blg-high-availability-cluster-concepts-and-architecture>

[4] <https://www.techtarget.com/whatis/definition/shared-storage>

**Question 18. (10 points)**

**Stat: Write your own version of the command line program stat, which simply calls the stat() system call on a given file or directory. Print out file size, number of blocks allocated, reference (link) count, and so forth. What is the link count of a directory, as**

**the number of entries in the directory changes?**

**ANSWER:**

The code is provided as a .c file in Canvas. The executable file of the code is also provided.

Answer to the second part of this question:

“A Link count shows how many directory entries link to the file/directory. A file's link count is displayed in the second column of output from the ls -l command. This number represents the total number of links that have been created to point to the data. The link count of a directory increases whenever a sub-directory is created. When the hard link file is moved or deleted, the link count of the original file gets reduced.”

Reference:

[1] <https://frameboxxindore.com/windows/what-is-link-count-in-linux.html>

[2] <https://www.theunixschool.com/2012/10/link-count-file-vs-directory.html>

**Sample output:**

Please enter the file/directory path relative to the present working directory: a.out

Provided path is a file, the file properties are as follows:

1. The file/directory size is: 17096

2. The number of blocks allocated is: 40

3. The reference (link) count is: 1

4. The file/directory user ID is: 1000

5. The preferred block size for I/O file system of the file/directory is: 4096

6. The file/directory group ID is: 1000

7. The total number of blocks in multiples of 512 bytes is: 40

Note: 'r' denotes read permission, 'w' denotes write permission, 'x' denotes execute permission, '-' denotes that the particular permission is not present

8. The File/directory Permissions for User: rwx

9. The File/directory Permissions for Group: rwx

10. The File/directory Permissions for Others: rwx

11. The file/directory was created on: 19/3/2022 (DD/MM/YYYY) at 8 hours 47 minutes 30 seconds

12. The file/directory was modified on: 19/3/2022 (DD/MM/YYYY) at 8 hours 47 minutes 30 seconds

**Question 20 (20 points) (No question 19 on exam)**

**Using the Texas Cyber Range to write a program that implements the following disk-scheduling algorithms:**

**a. FCFS**

**b. SCAN**

**c. C-SCAN**

**Your program will service a disk with 5,000 cylinders numbered 0 to 4,999. The program will generate a random series of 1,000-cylinder requests and service them according to each of the algorithms listed above. The program will be passed the initial position of the disk head (as a parameter on the command line) and report the total amount of head movement required by each algorithm.**

**ANSWER:**

The code is provided as a .c file in Canvas. The executable file of the code is also provided.

**Sample output:**

Generating a random series of 1,000-cylinder requests (1000 random integers) between 0 and 5000

Printing the 1,000-cylinder requests

4460, 3330, 2626, 4361, 3520, 436, 3682, 1464, 4452, 3293, 3707, 670, 2606, 521, 2609, 4263, 1264, 3552, 2808, 810, 4278, 3310, 2936, 780, 1462, 1295, 538, 4747, 3541, 849, 456, 3001, 4179, 4434, 2362, 2699, 4870, 2397, 515, 674, 690, 4223, 1344, 3297, 4744, 3954, 3912, 1008, 3858, 3073, 3170, 3137, 1383, 1106, 3917, 2845, 2402, 807, 2593, 943, 3008, 4401, 296, 3540, 3835, 4011, 2591, 57, 1408, 3107, 731, 3450, 2330, 3428, 1747, 3426, 2382, 2012, 4435, 2592, 85, 3957, 729, 2820, 1416, 999, 665, 3818, 3158, 4610, 1113, 1167, 4011, 2761, 4707, 4198, 1772, 2298, 4255, 3180, 1757, 1339, 1631, 4087, 4767, 4730, 2514, 3501, 3094, 3301, 1093, 3179, 2258, 3175, 999, 3674, 4174, 3017, 3844, 3684, 2627, 1309, 4851, 1639, 4071, 910, 2189, 843, 4561, 1445, 376, 1318, 2784, 3359, 1758, 3903, 4441, 4272, 2404, 2536, 3925, 3497, 715, 2535, 1672, 3067, 1210, 2198, 1084, 1406, 883, 3711, 2716, 2086, 1702, 3139, 4349, 3892, 334, 3910, 1689, 710, 1580, 825, 421, 3338, 4728, 4863, 3962, 3484, 2399, 4239, 3333, 4466, 1775, 1358, 2533, 4337, 3556, 3617, 743, 791, 3681, 4811, 4230, 1735, 2950, 3579, 1979, 3285, 3841, 20, 347, 1773, 845, 769, 1464, 1925, 632, 426, 409, 3031, 1018, 3743, 2497, 2793, 101, 1383, 2130, 9, 1352, 4225, 2153, 1385, 4037, 1383, 3121, 3339, 4962, 1452, 2976, 3803, 1473, 3324, 576, 3670, 445, 3392, 1948, 1077, 3819, 2357, 460, 1189, 2452, 4309, 3982, 3905, 2044, 2464, 267, 3397, 1689, 2420, 4782, 2078, 3803, 4255, 418, 117, 708, 4746, 272, 3533, 4422, 2200, 2203, 4867, 593, 4151, 2296, 764, 2861, 4108, 1953, 1665, 4770, 935, 1923, 1814, 4751, 2190, 211, 1440, 962, 1346, 3519, 4765, 601, 289, 4882, 1309, 1387, 154, 4842, 810, 2354, 3398, 2029, 2947, 3901, 678, 3711, 1762, 4786, 2016, 4780, 4556, 2951, 1703, 2723, 4054, 3893, 4286, 495, 4855, 632, 366, 972, 1234, 2007, 2206, 3895, 3394, 3712, 90, 556, 1066, 3488, 3938, 366, 2389, 4616, 429, 504, 754, 2446, 284, 1663, 1749, 1987, 4386, 804, 2232, 3672, 2651, 3439, 4305, 4369, 4411, 1891, 1376, 1617, 786, 1122, 329, 876, 3031, 2747, 716, 1969, 4465, 3106, 2937, 1247, 3610, 3691, 45, 3894, 354, 3146, 2233, 4740, 3950, 4465, 4765, 2953, 2904, 422, 2322, 2315, 2313, 50, 284, 4451, 2525, 1965, 328, 1908, 1064, 1044, 3877, 530, 502, 3166, 1777, 464, 1857, 3174, 4358, 2212, 1320, 1591, 3304, 1623, 1056, 4421, 4576, 312, 4843, 3251, 2627, 3508, 4653, 4263, 2960, 2178, 1228, 4640, 4086, 2293, 2036, 4315, 4175, 2539, 2481, 2304, 3003, 691, 478, 3714, 4255, 1798, 1657, 3911, 3421, 4066, 3333, 4350, 4378, 4528, 3953, 2006, 3037, 3606, 2621, 2349, 2137, 202, 1989, 2575, 2495, 377, 3243, 3022, 4268, 2076, 326, 3624, 2767, 2156, 2338, 2022, 306, 3995, 934, 80, 4413, 619, 4430, 3792, 147, 3383, 2150, 4536, 3341, 4771, 1885, 478, 4973, 226, 3054, 3820, 604, 2649, 1842, 4872, 4725, 3520, 3496, 2493, 676, 2186, 867, 2335, 2534, 3153, 2415, 1947, 3772, 1845, 739, 272, 1580, 2889, 4808, 1273, 4013, 1694, 1752, 3986, 3272, 1158, 2807, 3876, 3807, 1001, 101, 4884, 874, 4949, 2377, 2902, 2136, 4597, 237, 1022, 2750, 2652, 2969, 1523, 849, 61, 1795, 3781, 4302, 2955, 55, 3315, 1001, 3159, 2302, 4274, 4317, 1461, 3150, 4476, 3814, 4603, 4360, 4688, 4553, 3090, 2591, 3041, 2687, 2828, 415, 437, 481, 3384, 3312, 2682, 4797, 1459, 1464, 4100, 4415, 1519, 2415, 416, 4678, 1069, 4690, 347, 3882, 4193, 1175, 2697, 148, 1887, 2385, 1053, 1329, 1328, 446, 4016, 4157, 861, 806, 990, 598, 470, 3672, 395, 3282, 1488, 847, 2697, 3007, 3263, 4465, 4037, 684, 508, 4384, 4567, 1053, 1911, 3616, 1201, 151, 2353, 2255, 1480, 3682, 2701, 1849, 4191, 4915, 4007, 181, 513, 4477, 205, 2260, 2759, 1694, 3108, 1808, 1053, 2723, 1274, 91, 4759, 3134, 827, 4326, 4187, 4091, 2942, 388, 4242, 296, 3995, 722, 330, 3049, 2571, 4521, 4316, 2930, 1054, 4829, 2408, 2611, 3441, 1519, 657, 1549, 4680, 1711, 624, 2306, 3154, 384, 440, 333, 1062, 4627, 776, 357, 1367, 18, 2005, 1715, 2093, 2335, 1116, 1016, 1856, 432, 3947, 4262, 1613, 2707, 1873, 54, 4226, 2531, 2956, 3906, 594, 4932, 1212, 100, 1668, 3004, 1785, 2731, 3983, 2562, 3088, 1703, 3932, 1445, 3418, 1025, 3780, 886, 2042, 1988, 1318, 2341, 1250, 4283, 48, 4475, 689, 4274, 3358, 4997, 4533, 3952, 4930, 2097, 404, 2950, 102, 2190, 681, 4085, 4752, 121, 788, 3684, 1566, 558, 1062, 1698, 1444, 3104, 3686, 4114, 1797, 1288, 4749, 1845, 2116, 1791, 2471, 474, 1788, 3356, 779, 3070, 454, 1183, 1021, 1908, 4725, 3054, 2345, 4477, 3176, 4486, 4514, 1094, 1396, 576, 2793, 4193, 32, 2831, 4659, 1829, 472, 4409, 3674, 2588, 1200, 2497, 4414, 4340, 854, 193, 2411, 2660, 2729, 3432, 4568, 2454, 1486, 3265, 3284, 1014, 2751, 2798, 2109, 4148, 4726, 1254, 4693, 4758, 4085, 4352, 1587, 909, 3761, 1613, 4849, 1313, 462, 4264, 654, 1316, 809, 3065, 3976, 4890, 2849, 4896, 2345, 687, 3162, 629, 1702, 913, 4779, 163, 1413, 4505, 1417, 1106, 4263, 1854, 1811, 2202, 2764, 572, 167, 3965, 3238, 629, 3229, 3892, 1946, 391, 3309, 2274, 281, 2510, 2171, 2626, 3197, 333, 4607, 1251, 2598, 738, 1414, 364, 243, 2831, 2822, 858, 1038, 4633, 3060, 154, 1558, 4579, 4119, 4796, 209, 3701, 40, 3507, 4092, 4701, 2133, 725, 2211, 4304, 4704, 408, 989, 663, 3012, 3588, 1402, 4426, 304, 1645, 3610, 3126, 3856, 4648, 4112, 1916, 1154, 670, 2848, 273, 1818, 4409, 326, 3210, 4268, 770, 2911, 1401, 1496, 1474, 706, 1200, 3234, 3047, 3215, 1246, 2987, 4617, 673, 3291, 1263, 4283, 2770, 119, 283, 1882, 3387, 1437, 3904, 2587, 3062, 2074, 1996, 3389, 284, 1264, 4159, 4547, 4018, 2007, 1021, 1076, 4559, 4255, 4123, 2775, 1854, 2111, 2392, 2527, 1754, 7, 3162, 4524, 1478, 3445, 2758, 1218, 1234, 1662, 3805, 4296, 88, 2154, 2685, 372, 4770, 3197, 1271, 3788, 204, 2292, 4864, 4764, 2900, 340, 2539, 4754, 2451, 1283, 3633, 4205, 1291, 1795, 82, 4121, 1592, 2840, 339, 2826, 855, 497, 2122, 943, 4003, 1160, 2668, 3773, 4357, 3939, 3914, 913, 2584, 130, 677, 1836,

Please enter the initial position of the disk head:

50

Please choose from the following options:

| 1 -> Run all 3 algorithms one-by-one: FCFS, SCAN (for both left and right directions), C-SCAN |

| 2 -> Run FCFS |

| 3 -> Run SCAN (will show ouput for both left and right directions) |

| 4 -> Run C-SCAN |

Enter an integer from 1, 2, 3, or 4 which corresponds to running all disk-scheduling algorithms, running FCFS only, running SCAN only (will show ouput for both left and right directions) , or running C-SCAN only

1

############## Running FCFS disk-scheduling algorithm: ##############

The seek sequence for this algorithm is:

4460, 3330, 2626, 4361, 3520, 436, 3682, 1464, 4452, 3293, 3707, 670, 2606, 521, 2609, 4263, 1264, 3552, 2808, 810, 4278, 3310, 2936, 780, 1462, 1295, 538, 4747, 3541, 849, 456, 3001, 4179, 4434, 2362, 2699, 4870, 2397, 515, 674, 690, 4223, 1344, 3297, 4744, 3954, 3912, 1008, 3858, 3073, 3170, 3137, 1383, 1106, 3917, 2845, 2402, 807, 2593, 943, 3008, 4401, 296, 3540, 3835, 4011, 2591, 57, 1408, 3107, 731, 3450, 2330, 3428, 1747, 3426, 2382, 2012, 4435, 2592, 85, 3957, 729, 2820, 1416, 999, 665, 3818, 3158, 4610, 1113, 1167, 4011, 2761, 4707, 4198, 1772, 2298, 4255, 3180, 1757, 1339, 1631, 4087, 4767, 4730, 2514, 3501, 3094, 3301, 1093, 3179, 2258, 3175, 999, 3674, 4174, 3017, 3844, 3684, 2627, 1309, 4851, 1639, 4071, 910, 2189, 843, 4561, 1445, 376, 1318, 2784, 3359, 1758, 3903, 4441, 4272, 2404, 2536, 3925, 3497, 715, 2535, 1672, 3067, 1210, 2198, 1084, 1406, 883, 3711, 2716, 2086, 1702, 3139, 4349, 3892, 334, 3910, 1689, 710, 1580, 825, 421, 3338, 4728, 4863, 3962, 3484, 2399, 4239, 3333, 4466, 1775, 1358, 2533, 4337, 3556, 3617, 743, 791, 3681, 4811, 4230, 1735, 2950, 3579, 1979, 3285, 3841, 20, 347, 1773, 845, 769, 1464, 1925, 632, 426, 409, 3031, 1018, 3743, 2497, 2793, 101, 1383, 2130, 9, 1352, 4225, 2153, 1385, 4037, 1383, 3121, 3339, 4962, 1452, 2976, 3803, 1473, 3324, 576, 3670, 445, 3392, 1948, 1077, 3819, 2357, 460, 1189, 2452, 4309, 3982, 3905, 2044, 2464, 267, 3397, 1689, 2420, 4782, 2078, 3803, 4255, 418, 117, 708, 4746, 272, 3533, 4422, 2200, 2203, 4867, 593, 4151, 2296, 764, 2861, 4108, 1953, 1665, 4770, 935, 1923, 1814, 4751, 2190, 211, 1440, 962, 1346, 3519, 4765, 601, 289, 4882, 1309, 1387, 154, 4842, 810, 2354, 3398, 2029, 2947, 3901, 678, 3711, 1762, 4786, 2016, 4780, 4556, 2951, 1703, 2723, 4054, 3893, 4286, 495, 4855, 632, 366, 972, 1234, 2007, 2206, 3895, 3394, 3712, 90, 556, 1066, 3488, 3938, 366, 2389, 4616, 429, 504, 754, 2446, 284, 1663, 1749, 1987, 4386, 804, 2232, 3672, 2651, 3439, 4305, 4369, 4411, 1891, 1376, 1617, 786, 1122, 329, 876, 3031, 2747, 716, 1969, 4465, 3106, 2937, 1247, 3610, 3691, 45, 3894, 354, 3146, 2233, 4740, 3950, 4465, 4765, 2953, 2904, 422, 2322, 2315, 2313, 50, 284, 4451, 2525, 1965, 328, 1908, 1064, 1044, 3877, 530, 502, 3166, 1777, 464, 1857, 3174, 4358, 2212, 1320, 1591, 3304, 1623, 1056, 4421, 4576, 312, 4843, 3251, 2627, 3508, 4653, 4263, 2960, 2178, 1228, 4640, 4086, 2293, 2036, 4315, 4175, 2539, 2481, 2304, 3003, 691, 478, 3714, 4255, 1798, 1657, 3911, 3421, 4066, 3333, 4350, 4378, 4528, 3953, 2006, 3037, 3606, 2621, 2349, 2137, 202, 1989, 2575, 2495, 377, 3243, 3022, 4268, 2076, 326, 3624, 2767, 2156, 2338, 2022, 306, 3995, 934, 80, 4413, 619, 4430, 3792, 147, 3383, 2150, 4536, 3341, 4771, 1885, 478, 4973, 226, 3054, 3820, 604, 2649, 1842, 4872, 4725, 3520, 3496, 2493, 676, 2186, 867, 2335, 2534, 3153, 2415, 1947, 3772, 1845, 739, 272, 1580, 2889, 4808, 1273, 4013, 1694, 1752, 3986, 3272, 1158, 2807, 3876, 3807, 1001, 101, 4884, 874, 4949, 2377, 2902, 2136, 4597, 237, 1022, 2750, 2652, 2969, 1523, 849, 61, 1795, 3781, 4302, 2955, 55, 3315, 1001, 3159, 2302, 4274, 4317, 1461, 3150, 4476, 3814, 4603, 4360, 4688, 4553, 3090, 2591, 3041, 2687, 2828, 415, 437, 481, 3384, 3312, 2682, 4797, 1459, 1464, 4100, 4415, 1519, 2415, 416, 4678, 1069, 4690, 347, 3882, 4193, 1175, 2697, 148, 1887, 2385, 1053, 1329, 1328, 446, 4016, 4157, 861, 806, 990, 598, 470, 3672, 395, 3282, 1488, 847, 2697, 3007, 3263, 4465, 4037, 684, 508, 4384, 4567, 1053, 1911, 3616, 1201, 151, 2353, 2255, 1480, 3682, 2701, 1849, 4191, 4915, 4007, 181, 513, 4477, 205, 2260, 2759, 1694, 3108, 1808, 1053, 2723, 1274, 91, 4759, 3134, 827, 4326, 4187, 4091, 2942, 388, 4242, 296, 3995, 722, 330, 3049, 2571, 4521, 4316, 2930, 1054, 4829, 2408, 2611, 3441, 1519, 657, 1549, 4680, 1711, 624, 2306, 3154, 384, 440, 333, 1062, 4627, 776, 357, 1367, 18, 2005, 1715, 2093, 2335, 1116, 1016, 1856, 432, 3947, 4262, 1613, 2707, 1873, 54, 4226, 2531, 2956, 3906, 594, 4932, 1212, 100, 1668, 3004, 1785, 2731, 3983, 2562, 3088, 1703, 3932, 1445, 3418, 1025, 3780, 886, 2042, 1988, 1318, 2341, 1250, 4283, 48, 4475, 689, 4274, 3358, 4997, 4533, 3952, 4930, 2097, 404, 2950, 102, 2190, 681, 4085, 4752, 121, 788, 3684, 1566, 558, 1062, 1698, 1444, 3104, 3686, 4114, 1797, 1288, 4749, 1845, 2116, 1791, 2471, 474, 1788, 3356, 779, 3070, 454, 1183, 1021, 1908, 4725, 3054, 2345, 4477, 3176, 4486, 4514, 1094, 1396, 576, 2793, 4193, 32, 2831, 4659, 1829, 472, 4409, 3674, 2588, 1200, 2497, 4414, 4340, 854, 193, 2411, 2660, 2729, 3432, 4568, 2454, 1486, 3265, 3284, 1014, 2751, 2798, 2109, 4148, 4726, 1254, 4693, 4758, 4085, 4352, 1587, 909, 3761, 1613, 4849, 1313, 462, 4264, 654, 1316, 809, 3065, 3976, 4890, 2849, 4896, 2345, 687, 3162, 629, 1702, 913, 4779, 163, 1413, 4505, 1417, 1106, 4263, 1854, 1811, 2202, 2764, 572, 167, 3965, 3238, 629, 3229, 3892, 1946, 391, 3309, 2274, 281, 2510, 2171, 2626, 3197, 333, 4607, 1251, 2598, 738, 1414, 364, 243, 2831, 2822, 858, 1038, 4633, 3060, 154, 1558, 4579, 4119, 4796, 209, 3701, 40, 3507, 4092, 4701, 2133, 725, 2211, 4304, 4704, 408, 989, 663, 3012, 3588, 1402, 4426, 304, 1645, 3610, 3126, 3856, 4648, 4112, 1916, 1154, 670, 2848, 273, 1818, 4409, 326, 3210, 4268, 770, 2911, 1401, 1496, 1474, 706, 1200, 3234, 3047, 3215, 1246, 2987, 4617, 673, 3291, 1263, 4283, 2770, 119, 283, 1882, 3387, 1437, 3904, 2587, 3062, 2074, 1996, 3389, 284, 1264, 4159, 4547, 4018, 2007, 1021, 1076, 4559, 4255, 4123, 2775, 1854, 2111, 2392, 2527, 1754, 7, 3162, 4524, 1478, 3445, 2758, 1218, 1234, 1662, 3805, 4296, 88, 2154, 2685, 372, 4770, 3197, 1271, 3788, 204, 2292, 4864, 4764, 2900, 340, 2539, 4754, 2451, 1283, 3633, 4205, 1291, 1795, 82, 4121, 1592, 2840, 339, 2826, 855, 497, 2122, 943, 4003, 1160, 2668, 3773, 4357, 3939, 3914, 913, 2584, 130, 677, 1836,

The total number of head movements for the FCFS algorithm is= 1641300

The average seek duration for the algorithm is= 1641.300049

############## Will now run SCAN disk-scheduling algorithm from both left and right directions: ##############

############## 1. Running SCAN disk-scheduling algorithm from right direction: ##############

The seek sequence for this algorithm is:

50, 54, 55, 57, 61, 80, 82, 85, 88, 90, 91, 100, 101, 101, 102, 117, 119, 121, 130, 147, 148, 151, 154, 154, 163, 167, 181, 193, 202, 204, 205, 209, 211, 226, 237, 243, 267, 272, 272, 273, 281, 283, 284, 284, 284, 289, 296, 296, 304, 306, 312, 326, 326, 328, 329, 330, 333, 333, 334, 339, 340, 347, 347, 354, 357, 364, 366, 366, 372, 376, 377, 384, 388, 391, 395, 404, 408, 409, 415, 416, 418, 421, 422, 426, 429, 432, 436, 437, 440, 445, 446, 454, 456, 460, 462, 464, 470, 472, 474, 478, 478, 481, 495, 497, 502, 504, 508, 513, 515, 521, 530, 538, 556, 558, 572, 576, 576, 593, 594, 598, 601, 604, 619, 624, 629, 629, 632, 632, 654, 657, 663, 665, 670, 670, 673, 674, 676, 677, 678, 681, 684, 687, 689, 690, 691, 706, 708, 710, 715, 716, 722, 725, 729, 731, 738, 739, 743, 754, 764, 769, 770, 776, 779, 780, 786, 788, 791, 804, 806, 807, 809, 810, 810, 825, 827, 843, 845, 847, 849, 849, 854, 855, 858, 861, 867, 874, 876, 883, 886, 909, 910, 913, 913, 934, 935, 943, 943, 962, 972, 989, 990, 999, 999, 1001, 1001, 1008, 1014, 1016, 1018, 1021, 1021, 1022, 1025, 1038, 1044, 1053, 1053, 1053, 1054, 1056, 1062, 1062, 1064, 1066, 1069, 1076, 1077, 1084, 1093, 1094, 1106, 1106, 1113, 1116, 1122, 1154, 1158, 1160, 1167, 1175, 1183, 1189, 1200, 1200, 1201, 1210, 1212, 1218, 1228, 1234, 1234, 1246, 1247, 1250, 1251, 1254, 1263, 1264, 1264, 1271, 1273, 1274, 1283, 1288, 1291, 1295, 1309, 1309, 1313, 1316, 1318, 1318, 1320, 1328, 1329, 1339, 1344, 1346, 1352, 1358, 1367, 1376, 1383, 1383, 1383, 1385, 1387, 1396, 1401, 1402, 1406, 1408, 1413, 1414, 1416, 1417, 1437, 1440, 1444, 1445, 1445, 1452, 1459, 1461, 1462, 1464, 1464, 1464, 1473, 1474, 1478, 1480, 1486, 1488, 1496, 1519, 1519, 1523, 1549, 1558, 1566, 1580, 1580, 1587, 1591, 1592, 1613, 1613, 1617, 1623, 1631, 1639, 1645, 1657, 1662, 1663, 1665, 1668, 1672, 1689, 1689, 1694, 1694, 1698, 1702, 1702, 1703, 1703, 1711, 1715, 1735, 1747, 1749, 1752, 1754, 1757, 1758, 1762, 1772, 1773, 1775, 1777, 1785, 1788, 1791, 1795, 1795, 1797, 1798, 1808, 1811, 1814, 1818, 1829, 1836, 1842, 1845, 1845, 1849, 1854, 1854, 1856, 1857, 1873, 1882, 1885, 1887, 1891, 1908, 1908, 1911, 1916, 1923, 1925, 1946, 1947, 1948, 1953, 1965, 1969, 1979, 1987, 1988, 1989, 1996, 2005, 2006, 2007, 2007, 2012, 2016, 2022, 2029, 2036, 2042, 2044, 2074, 2076, 2078, 2086, 2093, 2097, 2109, 2111, 2116, 2122, 2130, 2133, 2136, 2137, 2150, 2153, 2154, 2156, 2171, 2178, 2186, 2189, 2190, 2190, 2198, 2200, 2202, 2203, 2206, 2211, 2212, 2232, 2233, 2255, 2258, 2260, 2274, 2292, 2293, 2296, 2298, 2302, 2304, 2306, 2313, 2315, 2322, 2330, 2335, 2335, 2338, 2341, 2345, 2345, 2349, 2353, 2354, 2357, 2362, 2377, 2382, 2385, 2389, 2392, 2397, 2399, 2402, 2404, 2408, 2411, 2415, 2415, 2420, 2446, 2451, 2452, 2454, 2464, 2471, 2481, 2493, 2495, 2497, 2497, 2510, 2514, 2525, 2527, 2531, 2533, 2534, 2535, 2536, 2539, 2539, 2562, 2571, 2575, 2584, 2587, 2588, 2591, 2591, 2592, 2593, 2598, 2606, 2609, 2611, 2621, 2626, 2626, 2627, 2627, 2649, 2651, 2652, 2660, 2668, 2682, 2685, 2687, 2697, 2697, 2699, 2701, 2707, 2716, 2723, 2723, 2729, 2731, 2747, 2750, 2751, 2758, 2759, 2761, 2764, 2767, 2770, 2775, 2784, 2793, 2793, 2798, 2807, 2808, 2820, 2822, 2826, 2828, 2831, 2831, 2840, 2845, 2848, 2849, 2861, 2889, 2900, 2902, 2904, 2911, 2930, 2936, 2937, 2942, 2947, 2950, 2950, 2951, 2953, 2955, 2956, 2960, 2969, 2976, 2987, 3001, 3003, 3004, 3007, 3008, 3012, 3017, 3022, 3031, 3031, 3037, 3041, 3047, 3049, 3054, 3054, 3060, 3062, 3065, 3067, 3070, 3073, 3088, 3090, 3094, 3104, 3106, 3107, 3108, 3121, 3126, 3134, 3137, 3139, 3146, 3150, 3153, 3154, 3158, 3159, 3162, 3162, 3166, 3170, 3174, 3175, 3176, 3179, 3180, 3197, 3197, 3210, 3215, 3229, 3234, 3238, 3243, 3251, 3263, 3265, 3272, 3282, 3284, 3285, 3291, 3293, 3297, 3301, 3304, 3309, 3310, 3312, 3315, 3324, 3330, 3333, 3333, 3338, 3339, 3341, 3356, 3358, 3359, 3383, 3384, 3387, 3389, 3392, 3394, 3397, 3398, 3418, 3421, 3426, 3428, 3432, 3439, 3441, 3445, 3450, 3484, 3488, 3496, 3497, 3501, 3507, 3508, 3519, 3520, 3520, 3533, 3540, 3541, 3552, 3556, 3579, 3588, 3606, 3610, 3610, 3616, 3617, 3624, 3633, 3670, 3672, 3672, 3674, 3674, 3681, 3682, 3682, 3684, 3684, 3686, 3691, 3701, 3707, 3711, 3711, 3712, 3714, 3743, 3761, 3772, 3773, 3780, 3781, 3788, 3792, 3803, 3803, 3805, 3807, 3814, 3818, 3819, 3820, 3835, 3841, 3844, 3856, 3858, 3876, 3877, 3882, 3892, 3892, 3893, 3894, 3895, 3901, 3903, 3904, 3905, 3906, 3910, 3911, 3912, 3914, 3917, 3925, 3932, 3938, 3939, 3947, 3950, 3952, 3953, 3954, 3957, 3962, 3965, 3976, 3982, 3983, 3986, 3995, 3995, 4003, 4007, 4011, 4011, 4013, 4016, 4018, 4037, 4037, 4054, 4066, 4071, 4085, 4085, 4086, 4087, 4091, 4092, 4100, 4108, 4112, 4114, 4119, 4121, 4123, 4148, 4151, 4157, 4159, 4174, 4175, 4179, 4187, 4191, 4193, 4193, 4198, 4205, 4223, 4225, 4226, 4230, 4239, 4242, 4255, 4255, 4255, 4255, 4262, 4263, 4263, 4263, 4264, 4268, 4268, 4272, 4274, 4274, 4278, 4283, 4283, 4286, 4296, 4302, 4304, 4305, 4309, 4315, 4316, 4317, 4326, 4337, 4340, 4349, 4350, 4352, 4357, 4358, 4360, 4361, 4369, 4378, 4384, 4386, 4401, 4409, 4409, 4411, 4413, 4414, 4415, 4421, 4422, 4426, 4430, 4434, 4435, 4441, 4451, 4452, 4460, 4465, 4465, 4465, 4466, 4475, 4476, 4477, 4477, 4486, 4505, 4514, 4521, 4524, 4528, 4533, 4536, 4547, 4553, 4556, 4559, 4561, 4567, 4568, 4576, 4579, 4597, 4603, 4607, 4610, 4616, 4617, 4627, 4633, 4640, 4648, 4653, 4659, 4678, 4680, 4688, 4690, 4693, 4701, 4704, 4707, 4725, 4725, 4726, 4728, 4730, 4740, 4744, 4746, 4747, 4749, 4751, 4752, 4754, 4758, 4759, 4764, 4765, 4765, 4767, 4770, 4770, 4771, 4779, 4780, 4782, 4786, 4796, 4797, 4808, 4811, 4829, 4842, 4843, 4849, 4851, 4855, 4863, 4864, 4867, 4870, 4872, 4882, 4884, 4890, 4896, 4915, 4930, 4932, 4949, 4962, 4973, 4997, 48, 45, 40, 32, 20, 18, 9, 7,

The total number of head movements for the SCAN algorithm

when you initially traverse from the right direction is= 9941

The average seek duration for the algorithm is= 9.941000

############## 2. Running SCAN disk-scheduling algorithm from left direction: ##############

The seek sequence for this algorithm is:

48, 45, 40, 32, 20, 18, 9, 7, 50, 54, 55, 57, 61, 80, 82, 85, 88, 90, 91, 100, 101, 101, 102, 117, 119, 121, 130, 147, 148, 151, 154, 154, 163, 167, 181, 193, 202, 204, 205, 209, 211, 226, 237, 243, 267, 272, 272, 273, 281, 283, 284, 284, 284, 289, 296, 296, 304, 306, 312, 326, 326, 328, 329, 330, 333, 333, 334, 339, 340, 347, 347, 354, 357, 364, 366, 366, 372, 376, 377, 384, 388, 391, 395, 404, 408, 409, 415, 416, 418, 421, 422, 426, 429, 432, 436, 437, 440, 445, 446, 454, 456, 460, 462, 464, 470, 472, 474, 478, 478, 481, 495, 497, 502, 504, 508, 513, 515, 521, 530, 538, 556, 558, 572, 576, 576, 593, 594, 598, 601, 604, 619, 624, 629, 629, 632, 632, 654, 657, 663, 665, 670, 670, 673, 674, 676, 677, 678, 681, 684, 687, 689, 690, 691, 706, 708, 710, 715, 716, 722, 725, 729, 731, 738, 739, 743, 754, 764, 769, 770, 776, 779, 780, 786, 788, 791, 804, 806, 807, 809, 810, 810, 825, 827, 843, 845, 847, 849, 849, 854, 855, 858, 861, 867, 874, 876, 883, 886, 909, 910, 913, 913, 934, 935, 943, 943, 962, 972, 989, 990, 999, 999, 1001, 1001, 1008, 1014, 1016, 1018, 1021, 1021, 1022, 1025, 1038, 1044, 1053, 1053, 1053, 1054, 1056, 1062, 1062, 1064, 1066, 1069, 1076, 1077, 1084, 1093, 1094, 1106, 1106, 1113, 1116, 1122, 1154, 1158, 1160, 1167, 1175, 1183, 1189, 1200, 1200, 1201, 1210, 1212, 1218, 1228, 1234, 1234, 1246, 1247, 1250, 1251, 1254, 1263, 1264, 1264, 1271, 1273, 1274, 1283, 1288, 1291, 1295, 1309, 1309, 1313, 1316, 1318, 1318, 1320, 1328, 1329, 1339, 1344, 1346, 1352, 1358, 1367, 1376, 1383, 1383, 1383, 1385, 1387, 1396, 1401, 1402, 1406, 1408, 1413, 1414, 1416, 1417, 1437, 1440, 1444, 1445, 1445, 1452, 1459, 1461, 1462, 1464, 1464, 1464, 1473, 1474, 1478, 1480, 1486, 1488, 1496, 1519, 1519, 1523, 1549, 1558, 1566, 1580, 1580, 1587, 1591, 1592, 1613, 1613, 1617, 1623, 1631, 1639, 1645, 1657, 1662, 1663, 1665, 1668, 1672, 1689, 1689, 1694, 1694, 1698, 1702, 1702, 1703, 1703, 1711, 1715, 1735, 1747, 1749, 1752, 1754, 1757, 1758, 1762, 1772, 1773, 1775, 1777, 1785, 1788, 1791, 1795, 1795, 1797, 1798, 1808, 1811, 1814, 1818, 1829, 1836, 1842, 1845, 1845, 1849, 1854, 1854, 1856, 1857, 1873, 1882, 1885, 1887, 1891, 1908, 1908, 1911, 1916, 1923, 1925, 1946, 1947, 1948, 1953, 1965, 1969, 1979, 1987, 1988, 1989, 1996, 2005, 2006, 2007, 2007, 2012, 2016, 2022, 2029, 2036, 2042, 2044, 2074, 2076, 2078, 2086, 2093, 2097, 2109, 2111, 2116, 2122, 2130, 2133, 2136, 2137, 2150, 2153, 2154, 2156, 2171, 2178, 2186, 2189, 2190, 2190, 2198, 2200, 2202, 2203, 2206, 2211, 2212, 2232, 2233, 2255, 2258, 2260, 2274, 2292, 2293, 2296, 2298, 2302, 2304, 2306, 2313, 2315, 2322, 2330, 2335, 2335, 2338, 2341, 2345, 2345, 2349, 2353, 2354, 2357, 2362, 2377, 2382, 2385, 2389, 2392, 2397, 2399, 2402, 2404, 2408, 2411, 2415, 2415, 2420, 2446, 2451, 2452, 2454, 2464, 2471, 2481, 2493, 2495, 2497, 2497, 2510, 2514, 2525, 2527, 2531, 2533, 2534, 2535, 2536, 2539, 2539, 2562, 2571, 2575, 2584, 2587, 2588, 2591, 2591, 2592, 2593, 2598, 2606, 2609, 2611, 2621, 2626, 2626, 2627, 2627, 2649, 2651, 2652, 2660, 2668, 2682, 2685, 2687, 2697, 2697, 2699, 2701, 2707, 2716, 2723, 2723, 2729, 2731, 2747, 2750, 2751, 2758, 2759, 2761, 2764, 2767, 2770, 2775, 2784, 2793, 2793, 2798, 2807, 2808, 2820, 2822, 2826, 2828, 2831, 2831, 2840, 2845, 2848, 2849, 2861, 2889, 2900, 2902, 2904, 2911, 2930, 2936, 2937, 2942, 2947, 2950, 2950, 2951, 2953, 2955, 2956, 2960, 2969, 2976, 2987, 3001, 3003, 3004, 3007, 3008, 3012, 3017, 3022, 3031, 3031, 3037, 3041, 3047, 3049, 3054, 3054, 3060, 3062, 3065, 3067, 3070, 3073, 3088, 3090, 3094, 3104, 3106, 3107, 3108, 3121, 3126, 3134, 3137, 3139, 3146, 3150, 3153, 3154, 3158, 3159, 3162, 3162, 3166, 3170, 3174, 3175, 3176, 3179, 3180, 3197, 3197, 3210, 3215, 3229, 3234, 3238, 3243, 3251, 3263, 3265, 3272, 3282, 3284, 3285, 3291, 3293, 3297, 3301, 3304, 3309, 3310, 3312, 3315, 3324, 3330, 3333, 3333, 3338, 3339, 3341, 3356, 3358, 3359, 3383, 3384, 3387, 3389, 3392, 3394, 3397, 3398, 3418, 3421, 3426, 3428, 3432, 3439, 3441, 3445, 3450, 3484, 3488, 3496, 3497, 3501, 3507, 3508, 3519, 3520, 3520, 3533, 3540, 3541, 3552, 3556, 3579, 3588, 3606, 3610, 3610, 3616, 3617, 3624, 3633, 3670, 3672, 3672, 3674, 3674, 3681, 3682, 3682, 3684, 3684, 3686, 3691, 3701, 3707, 3711, 3711, 3712, 3714, 3743, 3761, 3772, 3773, 3780, 3781, 3788, 3792, 3803, 3803, 3805, 3807, 3814, 3818, 3819, 3820, 3835, 3841, 3844, 3856, 3858, 3876, 3877, 3882, 3892, 3892, 3893, 3894, 3895, 3901, 3903, 3904, 3905, 3906, 3910, 3911, 3912, 3914, 3917, 3925, 3932, 3938, 3939, 3947, 3950, 3952, 3953, 3954, 3957, 3962, 3965, 3976, 3982, 3983, 3986, 3995, 3995, 4003, 4007, 4011, 4011, 4013, 4016, 4018, 4037, 4037, 4054, 4066, 4071, 4085, 4085, 4086, 4087, 4091, 4092, 4100, 4108, 4112, 4114, 4119, 4121, 4123, 4148, 4151, 4157, 4159, 4174, 4175, 4179, 4187, 4191, 4193, 4193, 4198, 4205, 4223, 4225, 4226, 4230, 4239, 4242, 4255, 4255, 4255, 4255, 4262, 4263, 4263, 4263, 4264, 4268, 4268, 4272, 4274, 4274, 4278, 4283, 4283, 4286, 4296, 4302, 4304, 4305, 4309, 4315, 4316, 4317, 4326, 4337, 4340, 4349, 4350, 4352, 4357, 4358, 4360, 4361, 4369, 4378, 4384, 4386, 4401, 4409, 4409, 4411, 4413, 4414, 4415, 4421, 4422, 4426, 4430, 4434, 4435, 4441, 4451, 4452, 4460, 4465, 4465, 4465, 4466, 4475, 4476, 4477, 4477, 4486, 4505, 4514, 4521, 4524, 4528, 4533, 4536, 4547, 4553, 4556, 4559, 4561, 4567, 4568, 4576, 4579, 4597, 4603, 4607, 4610, 4616, 4617, 4627, 4633, 4640, 4648, 4653, 4659, 4678, 4680, 4688, 4690, 4693, 4701, 4704, 4707, 4725, 4725, 4726, 4728, 4730, 4740, 4744, 4746, 4747, 4749, 4751, 4752, 4754, 4758, 4759, 4764, 4765, 4765, 4767, 4770, 4770, 4771, 4779, 4780, 4782, 4786, 4796, 4797, 4808, 4811, 4829, 4842, 4843, 4849, 4851, 4855, 4863, 4864, 4867, 4870, 4872, 4882, 4884, 4890, 4896, 4915, 4930, 4932, 4949, 4962, 4973, 4997,

The total number of head movements for the SCAN algorithm

when you initially traverse from the left direction is= 5047

The average seek duration for the algorithm is= 5.047000

############## Running C-Scan disk-scheduling algorithm: ##############

The seek sequence for this algorithm is:

50, 54, 55, 57, 61, 80, 82, 85, 88, 90, 91, 100, 101, 101, 102, 117, 119, 121, 130, 147, 148, 151, 154, 154, 163, 167, 181, 193, 202, 204, 205, 209, 211, 226, 237, 243, 267, 272, 272, 273, 281, 283, 284, 284, 284, 289, 296, 296, 304, 306, 312, 326, 326, 328, 329, 330, 333, 333, 334, 339, 340, 347, 347, 354, 357, 364, 366, 366, 372, 376, 377, 384, 388, 391, 395, 404, 408, 409, 415, 416, 418, 421, 422, 426, 429, 432, 436, 437, 440, 445, 446, 454, 456, 460, 462, 464, 470, 472, 474, 478, 478, 481, 495, 497, 502, 504, 508, 513, 515, 521, 530, 538, 556, 558, 572, 576, 576, 593, 594, 598, 601, 604, 619, 624, 629, 629, 632, 632, 654, 657, 663, 665, 670, 670, 673, 674, 676, 677, 678, 681, 684, 687, 689, 690, 691, 706, 708, 710, 715, 716, 722, 725, 729, 731, 738, 739, 743, 754, 764, 769, 770, 776, 779, 780, 786, 788, 791, 804, 806, 807, 809, 810, 810, 825, 827, 843, 845, 847, 849, 849, 854, 855, 858, 861, 867, 874, 876, 883, 886, 909, 910, 913, 913, 934, 935, 943, 943, 962, 972, 989, 990, 999, 999, 1001, 1001, 1008, 1014, 1016, 1018, 1021, 1021, 1022, 1025, 1038, 1044, 1053, 1053, 1053, 1054, 1056, 1062, 1062, 1064, 1066, 1069, 1076, 1077, 1084, 1093, 1094, 1106, 1106, 1113, 1116, 1122, 1154, 1158, 1160, 1167, 1175, 1183, 1189, 1200, 1200, 1201, 1210, 1212, 1218, 1228, 1234, 1234, 1246, 1247, 1250, 1251, 1254, 1263, 1264, 1264, 1271, 1273, 1274, 1283, 1288, 1291, 1295, 1309, 1309, 1313, 1316, 1318, 1318, 1320, 1328, 1329, 1339, 1344, 1346, 1352, 1358, 1367, 1376, 1383, 1383, 1383, 1385, 1387, 1396, 1401, 1402, 1406, 1408, 1413, 1414, 1416, 1417, 1437, 1440, 1444, 1445, 1445, 1452, 1459, 1461, 1462, 1464, 1464, 1464, 1473, 1474, 1478, 1480, 1486, 1488, 1496, 1519, 1519, 1523, 1549, 1558, 1566, 1580, 1580, 1587, 1591, 1592, 1613, 1613, 1617, 1623, 1631, 1639, 1645, 1657, 1662, 1663, 1665, 1668, 1672, 1689, 1689, 1694, 1694, 1698, 1702, 1702, 1703, 1703, 1711, 1715, 1735, 1747, 1749, 1752, 1754, 1757, 1758, 1762, 1772, 1773, 1775, 1777, 1785, 1788, 1791, 1795, 1795, 1797, 1798, 1808, 1811, 1814, 1818, 1829, 1836, 1842, 1845, 1845, 1849, 1854, 1854, 1856, 1857, 1873, 1882, 1885, 1887, 1891, 1908, 1908, 1911, 1916, 1923, 1925, 1946, 1947, 1948, 1953, 1965, 1969, 1979, 1987, 1988, 1989, 1996, 2005, 2006, 2007, 2007, 2012, 2016, 2022, 2029, 2036, 2042, 2044, 2074, 2076, 2078, 2086, 2093, 2097, 2109, 2111, 2116, 2122, 2130, 2133, 2136, 2137, 2150, 2153, 2154, 2156, 2171, 2178, 2186, 2189, 2190, 2190, 2198, 2200, 2202, 2203, 2206, 2211, 2212, 2232, 2233, 2255, 2258, 2260, 2274, 2292, 2293, 2296, 2298, 2302, 2304, 2306, 2313, 2315, 2322, 2330, 2335, 2335, 2338, 2341, 2345, 2345, 2349, 2353, 2354, 2357, 2362, 2377, 2382, 2385, 2389, 2392, 2397, 2399, 2402, 2404, 2408, 2411, 2415, 2415, 2420, 2446, 2451, 2452, 2454, 2464, 2471, 2481, 2493, 2495, 2497, 2497, 2510, 2514, 2525, 2527, 2531, 2533, 2534, 2535, 2536, 2539, 2539, 2562, 2571, 2575, 2584, 2587, 2588, 2591, 2591, 2592, 2593, 2598, 2606, 2609, 2611, 2621, 2626, 2626, 2627, 2627, 2649, 2651, 2652, 2660, 2668, 2682, 2685, 2687, 2697, 2697, 2699, 2701, 2707, 2716, 2723, 2723, 2729, 2731, 2747, 2750, 2751, 2758, 2759, 2761, 2764, 2767, 2770, 2775, 2784, 2793, 2793, 2798, 2807, 2808, 2820, 2822, 2826, 2828, 2831, 2831, 2840, 2845, 2848, 2849, 2861, 2889, 2900, 2902, 2904, 2911, 2930, 2936, 2937, 2942, 2947, 2950, 2950, 2951, 2953, 2955, 2956, 2960, 2969, 2976, 2987, 3001, 3003, 3004, 3007, 3008, 3012, 3017, 3022, 3031, 3031, 3037, 3041, 3047, 3049, 3054, 3054, 3060, 3062, 3065, 3067, 3070, 3073, 3088, 3090, 3094, 3104, 3106, 3107, 3108, 3121, 3126, 3134, 3137, 3139, 3146, 3150, 3153, 3154, 3158, 3159, 3162, 3162, 3166, 3170, 3174, 3175, 3176, 3179, 3180, 3197, 3197, 3210, 3215, 3229, 3234, 3238, 3243, 3251, 3263, 3265, 3272, 3282, 3284, 3285, 3291, 3293, 3297, 3301, 3304, 3309, 3310, 3312, 3315, 3324, 3330, 3333, 3333, 3338, 3339, 3341, 3356, 3358, 3359, 3383, 3384, 3387, 3389, 3392, 3394, 3397, 3398, 3418, 3421, 3426, 3428, 3432, 3439, 3441, 3445, 3450, 3484, 3488, 3496, 3497, 3501, 3507, 3508, 3519, 3520, 3520, 3533, 3540, 3541, 3552, 3556, 3579, 3588, 3606, 3610, 3610, 3616, 3617, 3624, 3633, 3670, 3672, 3672, 3674, 3674, 3681, 3682, 3682, 3684, 3684, 3686, 3691, 3701, 3707, 3711, 3711, 3712, 3714, 3743, 3761, 3772, 3773, 3780, 3781, 3788, 3792, 3803, 3803, 3805, 3807, 3814, 3818, 3819, 3820, 3835, 3841, 3844, 3856, 3858, 3876, 3877, 3882, 3892, 3892, 3893, 3894, 3895, 3901, 3903, 3904, 3905, 3906, 3910, 3911, 3912, 3914, 3917, 3925, 3932, 3938, 3939, 3947, 3950, 3952, 3953, 3954, 3957, 3962, 3965, 3976, 3982, 3983, 3986, 3995, 3995, 4003, 4007, 4011, 4011, 4013, 4016, 4018, 4037, 4037, 4054, 4066, 4071, 4085, 4085, 4086, 4087, 4091, 4092, 4100, 4108, 4112, 4114, 4119, 4121, 4123, 4148, 4151, 4157, 4159, 4174, 4175, 4179, 4187, 4191, 4193, 4193, 4198, 4205, 4223, 4225, 4226, 4230, 4239, 4242, 4255, 4255, 4255, 4255, 4262, 4263, 4263, 4263, 4264, 4268, 4268, 4272, 4274, 4274, 4278, 4283, 4283, 4286, 4296, 4302, 4304, 4305, 4309, 4315, 4316, 4317, 4326, 4337, 4340, 4349, 4350, 4352, 4357, 4358, 4360, 4361, 4369, 4378, 4384, 4386, 4401, 4409, 4409, 4411, 4413, 4414, 4415, 4421, 4422, 4426, 4430, 4434, 4435, 4441, 4451, 4452, 4460, 4465, 4465, 4465, 4466, 4475, 4476, 4477, 4477, 4486, 4505, 4514, 4521, 4524, 4528, 4533, 4536, 4547, 4553, 4556, 4559, 4561, 4567, 4568, 4576, 4579, 4597, 4603, 4607, 4610, 4616, 4617, 4627, 4633, 4640, 4648, 4653, 4659, 4678, 4680, 4688, 4690, 4693, 4701, 4704, 4707, 4725, 4725, 4726, 4728, 4730, 4740, 4744, 4746, 4747, 4749, 4751, 4752, 4754, 4758, 4759, 4764, 4765, 4765, 4767, 4770, 4770, 4771, 4779, 4780, 4782, 4786, 4796, 4797, 4808, 4811, 4829, 4842, 4843, 4849, 4851, 4855, 4863, 4864, 4867, 4870, 4872, 4882, 4884, 4890, 4896, 4915, 4930, 4932, 4949, 4962, 4973, 4997, 4999, 0, 7, 9, 18, 20, 32, 40, 45, 48,

The total number of head movements for the C-Scan algorithm is= 9996

The average seek duration for the algorithm is= 9.996000