

Team 14

Team Dynamic

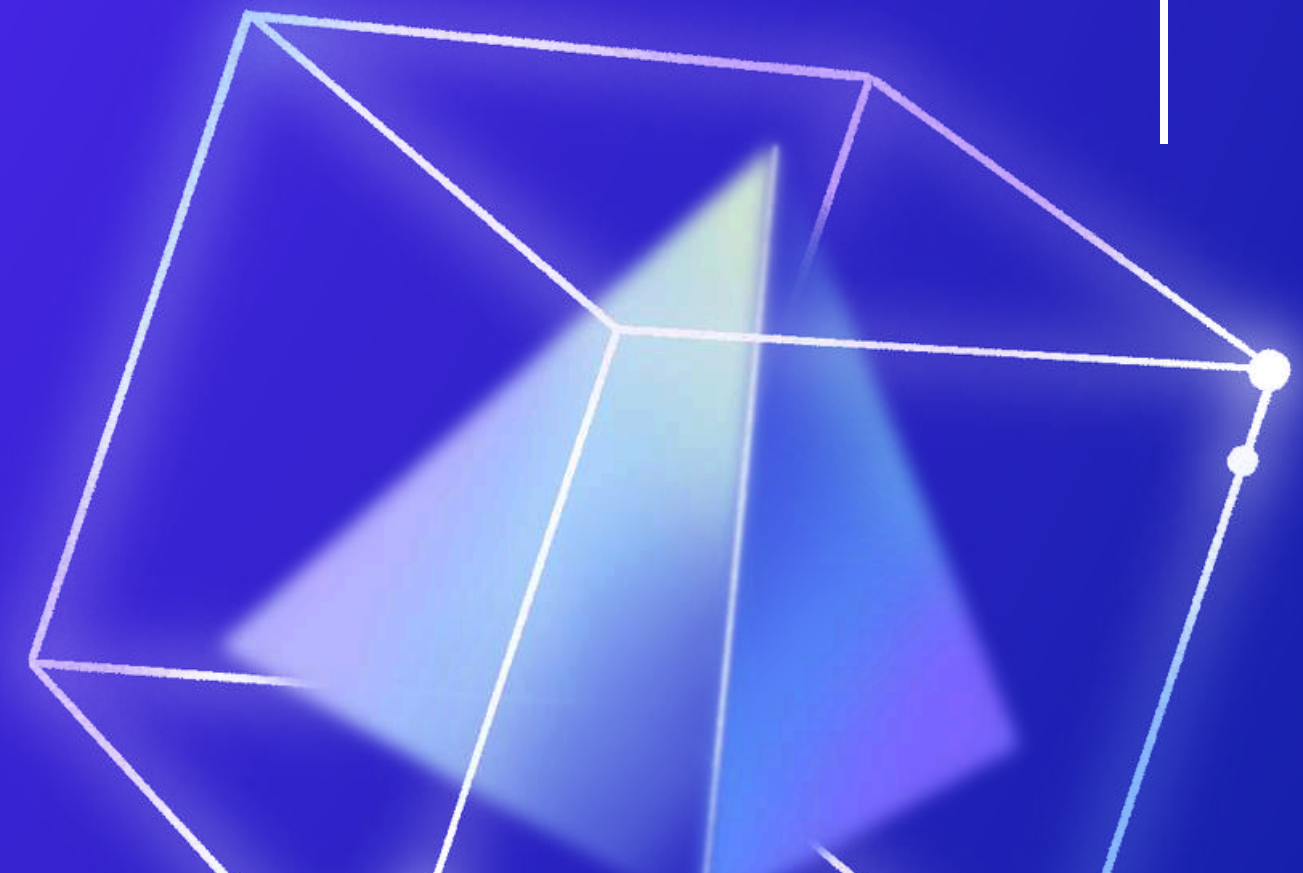
Ubuntu

OPEN-SOURCE OS



TOPICS TO DISCUSS

- | | | |
|----|----------------------------------|------|
| 01 | • IPC – Shared memory | D |
| 02 | • Scheduling Algorithm – CFS | D |
| 03 | • File allocation and protection | D-SS |
| 04 | • Memory Management | D |
| 05 | • Disk Scheduling – CFQ | T |
| 06 | • Page Replacement | T |
| 07 | • Virtual memory & TLB | T |



IPC-SHARED MEMORY

Inter-Process Communication (IPC) using shared memory is a method that allows multiple processes to access common memory space. This can be one of the most efficient ways to exchange data between processes because it avoids the need for data to be copied between processes, allowing for direct access.

DEMO

Let us look into a small example to understand how shared memory is used between processes in ubuntu.

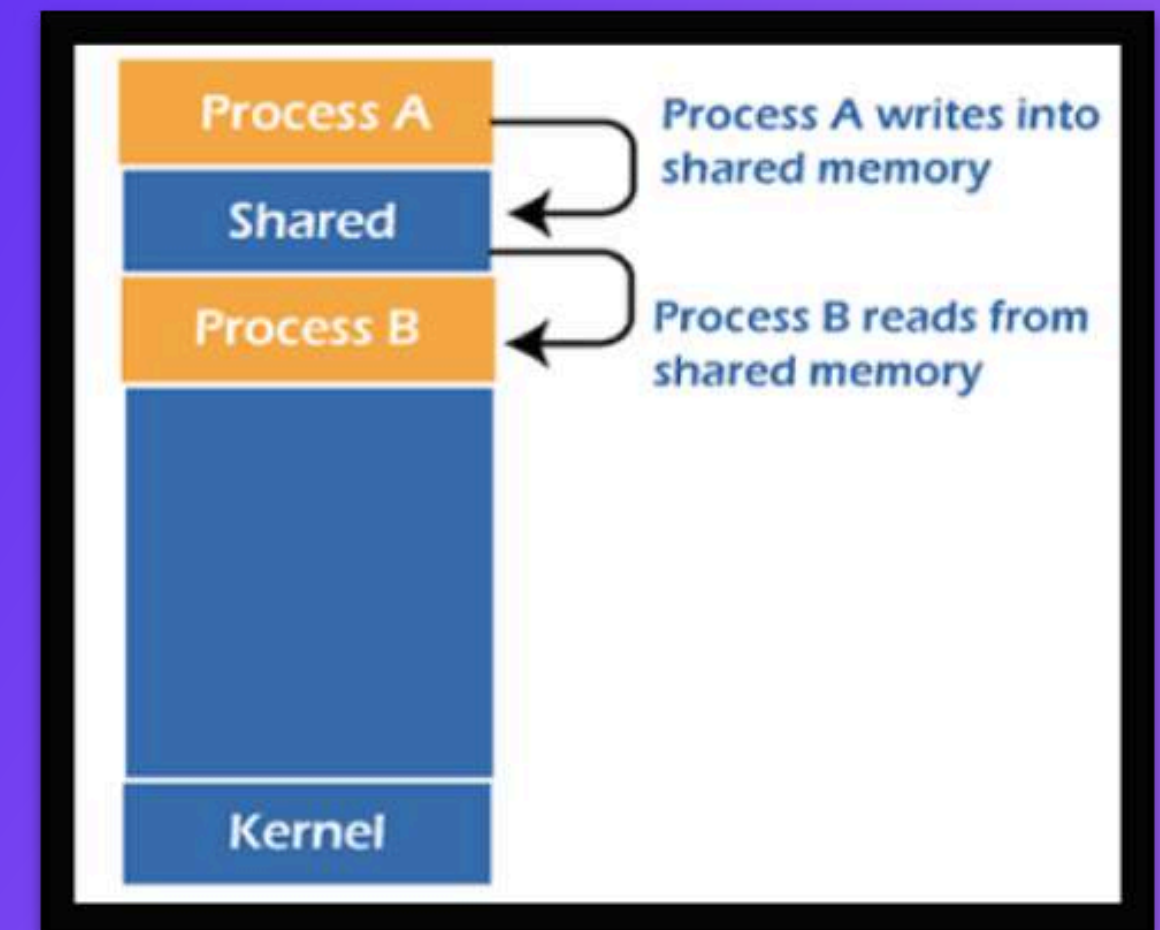
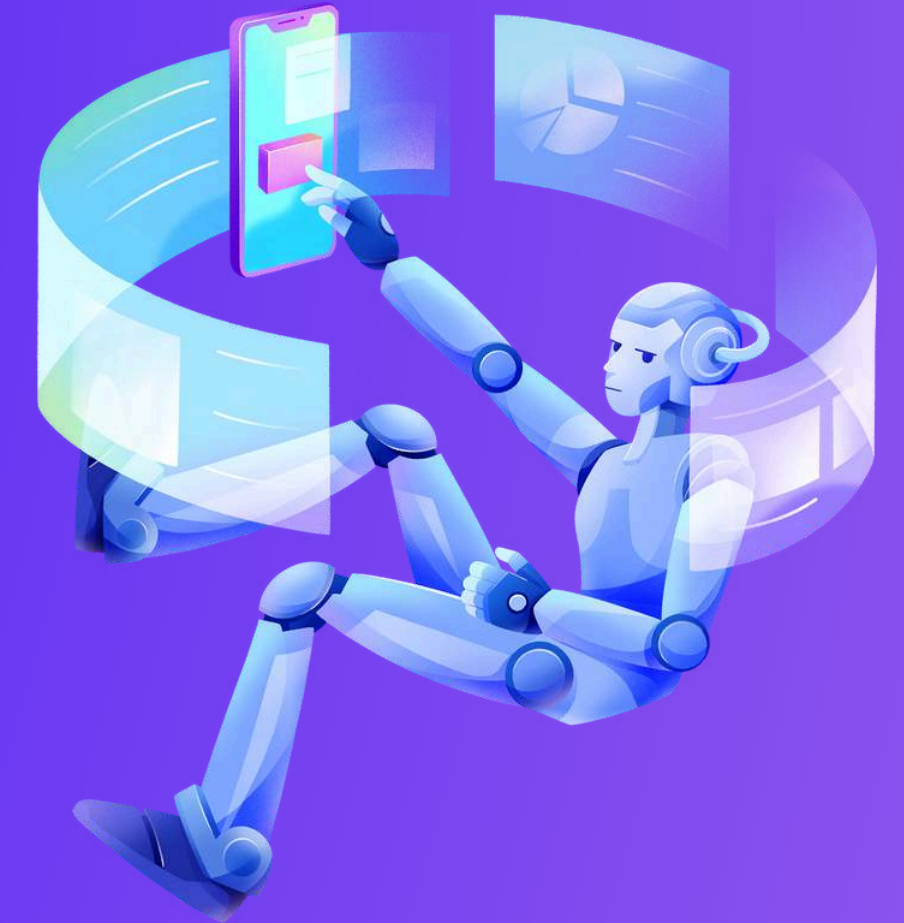
create -> `ftok()`, `shmget()`

detach -> `shmdt()`

attach -> `shmat()`

delete -> `shmctl()`

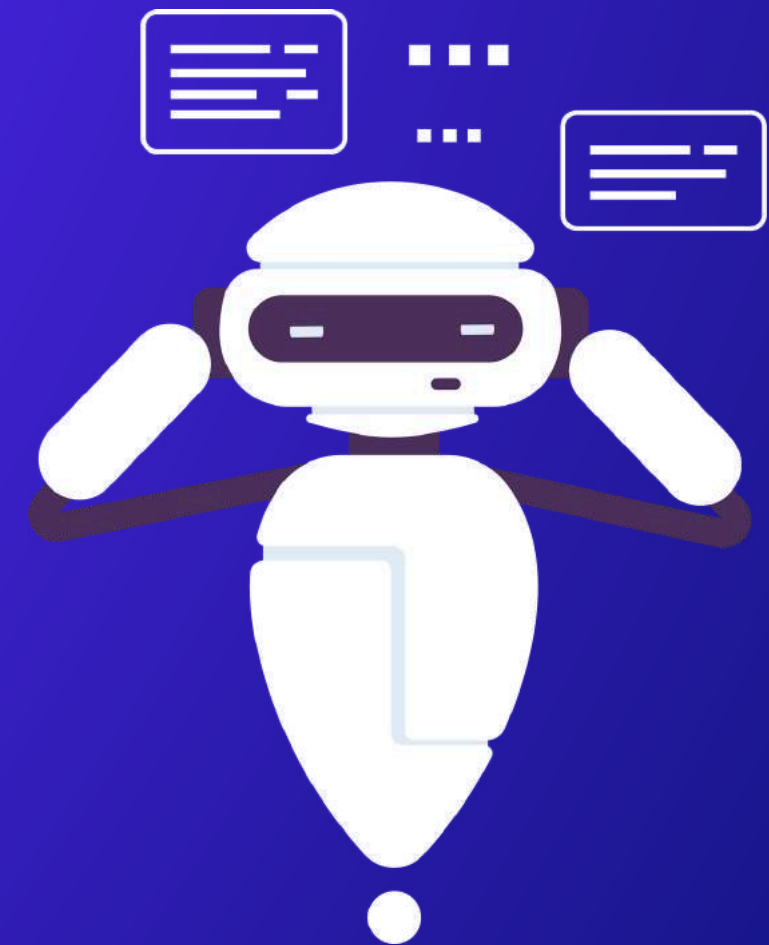
```
aadhira@AadhiLaps:~$ gcc shared.c
aadhira@AadhiLaps:~$ ./a.out
Enter the message to be shared: Hi this side
Process A has written the message to shared memory.
Process B is reading the message from shared memory: Hi this side
Process B has finished reading the message from shared memory.
Shared memory segment has been deleted.
```



SCHEDULING ALGORITHM - CFS

Ubuntu, like most Linux distributions, uses the Completely Fair Scheduler (CFS) as its default scheduling algorithm for the Linux kernel.

DEMO



FILE ALLOCATION AND PROTECTION

01

Creating, Deleting, and Manipulating Files

To create, delete, and manipulate files on an ext4 file system in Ubuntu, you can use standard Unix commands.

```
aadhira@AadhiLaps:~$ touch myfile.txt  
aadhira@AadhiLaps:~$ echo "Hello, World!" > myfile.txt
```

Deleting Files

You can delete files using the **rm** command.

```
aadhira@AadhiLaps:~$ rm myfile.txt  
rm: cannot remove 'myfile.txt': No such file or directory
```

Manipulating Files

Common commands to manipulate files include **cp** for copying, **mv** for moving or renaming, and **cat** or **less** for viewing contents.

Listing Files and Directories

Command: **ls**

Explanation: This command lists the files and directories in the current directory. Adding options such as **-l** (long format) or **-a** (including hidden files) can provide more detailed or comprehensive listings.

FILE ALLOCATION AND PROTECTION

02

1. Checking Disk Usage:

- **Command:** `df -h`
- **Explanation:** This command displays the disk space usage for all mounted filesystems in a human-readable format (-h). It shows information such as total size, used space, available space, and the filesystem type.

```
aadhira@AadhiLaps:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          230G  175G   55G   77% /
none            230G  175G   55G   77% /dev
none            230G  175G   55G   77% /run
none            230G  175G   55G   77% /run/lock
none            230G  175G   55G   77% /run/shm
none            230G  175G   55G   77% /run/user
tmpfs           230G  175G   55G   77% /sys/fs/cgroup
A:\              7.9G  242M   7.6G    4% /mnt/a
C:\             230G  175G   55G   77% /mnt/c
aadhira@AadhiLaps:~$
```

2. Checking File and Directory Sizes:

- **Command:** `du -h filename/directory`
- **Explanation:** This command displays the disk space usage of the specified file or directory in a human-readable format. It provides information on the size of individual files or directories.

```
aadhira@AadhiLaps:~$ du -h samp2.c
4.0K    samp2.c
aadhira@AadhiLaps:~$
```

3. Monitoring Disk Usage:

- **Command:** `du -h --max-depth=1 /path | sort -hr`
- **Explanation:** This command shows disk usage for each directory within a specified path, sorted by size in a human-readable format. It helps identify which directories are consuming the most space.

```
aadhira@AadhiLaps:~$ du -h --max-depth=1 /home/aadhira | sort -hr
155M    /home/aadhira
154M    /home/aadhira/.vscode-server
364K    /home/aadhira/obs8
172K    /home/aadhira/obs6
132K    /home/aadhira/obs7
122K    /home/aadhira/ossamp
84K     /home/aadhira/obs9
76K     /home/aadhira/obs07
61K     /home/aadhira/ADS00PS
48K     /home/aadhira/obs08
40K     /home/aadhira/obs10
0       /home/aadhira/.local
aadhira@AadhiLaps:~$
```

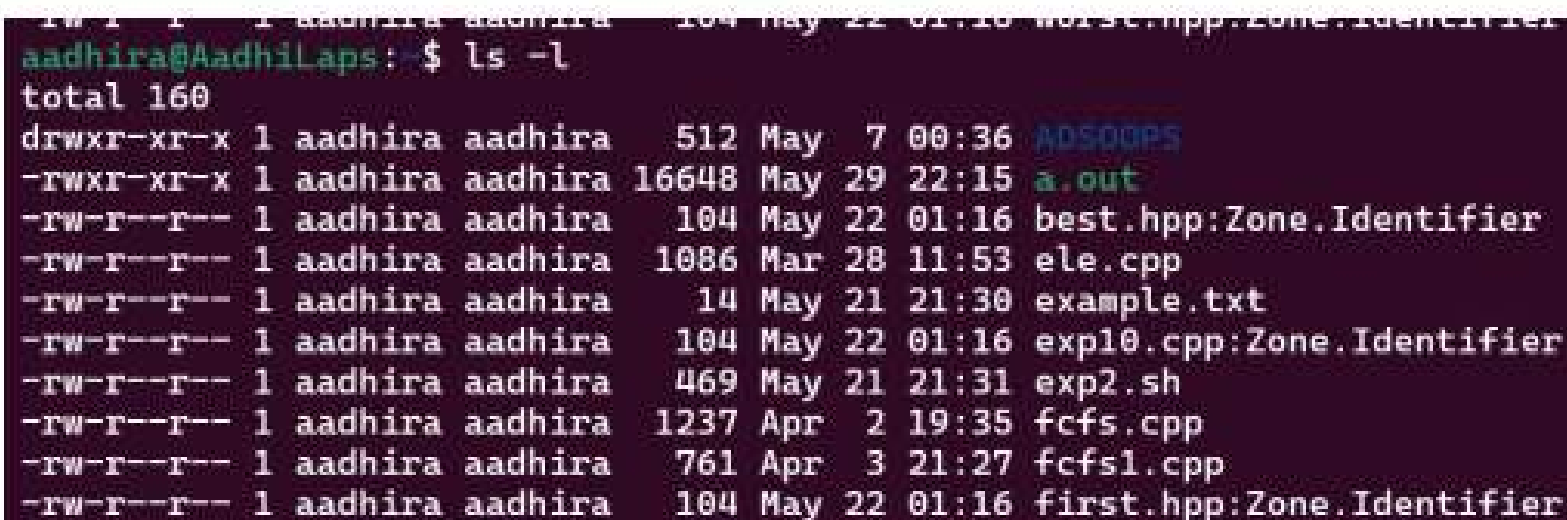
FILE ALLOCATION AND PROTECTION

LINUX FILE PERMISSIONS

Permissions are displayed as a 10-character string (e.g., **-rwxr-xr--**):

- The first character indicates the file type (**-** for a regular file, **d** for a directory).
- The next nine characters represent the permissions for the owner, group, and others, respectively.

To list files with detailed permissions in the current directory:



```
drwxr-xr-x 1 aadhira aadhira  512 May  7 00:36 ADS00PS
-rwxr-xr-x 1 aadhira aadhira 16648 May 29 22:15 a.out
-rw-r--r-- 1 aadhira aadhira  104 May 22 01:16 best.hpp:Zone.Identifier
-rw-r--r-- 1 aadhira aadhira 1086 Mar 28 11:53 ele.cpp
-rw-r--r-- 1 aadhira aadhira   14 May 21 21:30 example.txt
-rw-r--r-- 1 aadhira aadhira  104 May 22 01:16 exp10.cpp:Zone.Identifier
-rw-r--r-- 1 aadhira aadhira  469 May 21 21:31 exp2.sh
-rw-r--r-- 1 aadhira aadhira 1237 Apr  2 19:35 fcfs.cpp
-rw-r--r-- 1 aadhira aadhira  761 Apr  3 21:27 fcfs1.cpp
-rw-r--r-- 1 aadhira aadhira  104 May 22 01:16 first.hpp:Zone.Identifier
```


FILE ALLOCATION AND PROTECTION

LINUX FILE PERMISSIONS

04

Changing File Permissions

The file **example.txt** initially had permissions **-rw-r--r--**, allowing read and write access for the owner (**aadhira**), and read-only access for the group and others.

```
aadhira@AadhiLaps:~$ ls -l example.txt
-rw-r--r-- 1 aadhira aadhira 12 May 29 23:24 example.txt
aadhira@AadhiLaps:~$ chmod go-r example.txt
aadhira@AadhiLaps:~$ ls -l example.txt
-rw----- 1 aadhira aadhira 12 May 29 23:24 example.txt
aadhira@AadhiLaps:~$
```

Using the command **chmod go-r example.txt**, read permissions for the group and others were removed.

The new permissions became **-rw-----**, meaning only the owner (**aadhira**) can read and write to the file, while the group and others have no permissions.

Changing File Ownership

Change the ownership of **round.cpp** to user **username** and group **groupname**

```
aadhira@AadhiLaps:~$ ls -l samp1.c
-rw-r--r-- 1 aadhira aadhira 2011 May  9 13:34 samp1.c
aadhira@AadhiLaps:~$ sudo chown aadhira:audio example.txt
aadhira@AadhiLaps:~$ ls -l example.txt
-rw----- 1 aadhira audio 12 May 29 23:24 example.txt
```

`sudo chown new_owner:new_group example.txt`

MEMORY MANAGEMENT

THE BUDDY SYSTEM IS A MEMORY ALLOCATION ALGORITHM COMMONLY USED IN OPERATING SYSTEMS TO MANAGE PHYSICAL MEMORY EFFICIENTLY. IN UBUNTU, LIKE MANY OTHER LINUX DISTRIBUTIONS, THE BUDDY SYSTEM IS EMPLOYED BY THE KERNEL FOR MEMORY ALLOCATION. HERE'S HOW THE BUDDY SYSTEM WORKS:

THE AVAILABLE PHYSICAL MEMORY IS DIVIDED INTO FIXED-SIZE BLOCKS, TYPICALLY POWERS OF 2, SUCH AS 4KB, 8KB, 16KB, ETC. EACH MEMORY BLOCK IS FURTHER SPLIT INTO SMALLER BLOCKS, KNOWN AS "BUDDIES," WHICH ARE HALVES OF THE ORIGINAL BLOCK SIZE.

BINARY TREE STRUCTURE

ALLOCATION

DEALLOCATION

DEMO

DISC SCHEDULING - CFQ

COMPLETELY FAIR QUEUEING :

Completely Fair Queuing (CFQ) is a disk scheduling algorithm used in Linux to ensure that each process gets a fair share of the disk's bandwidth. CFQ divides processes into different priority classes and allocates time slices for I/O requests accordingly. Let's break down CFQ with an example and explain how the different priority classes work.

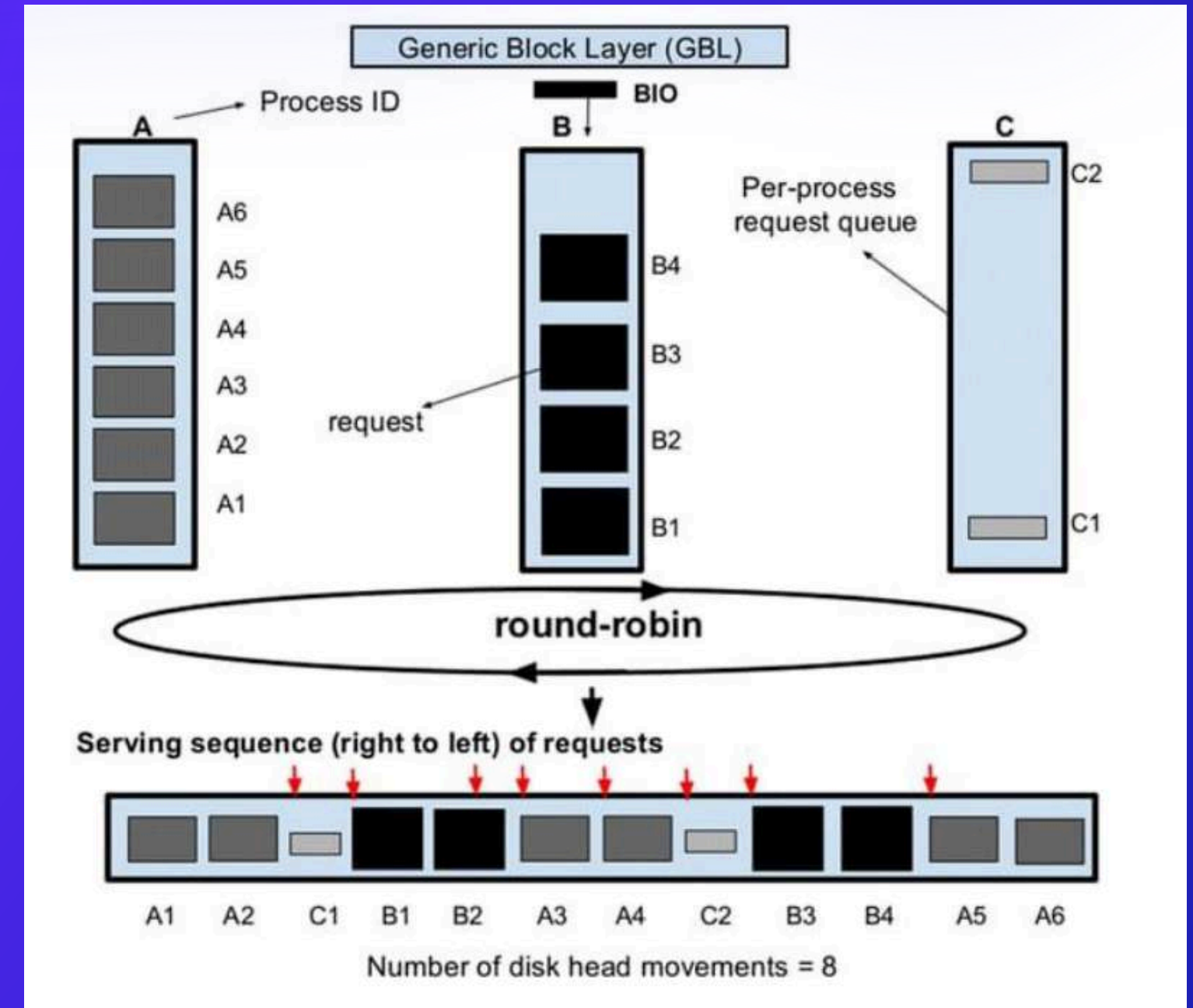
Basic Principles of CFQ

1. **Queue Management:** CFQ maintains separate I/O request queues for each process.
2. **Time Slices:** Each queue is given a time slice during which it can issue I/O requests to the disk.
3. **Priority Classes:** CFQ categorizes requests into three priority classes: real-time, best-effort, and idle.
4. **Fairness and Throughput:** CFQ aims to balance fairness (equal access for all processes) and throughput (efficient use of disk bandwidth).

DISC SCHEDULING - CFQ

Example Breakdown

1. **A1 to A2**: No disk head movement required as they are consecutive requests from Process A.
2. **A2 to C1**: The disk head must move to service C1, resulting in a disk head movement.
3. **C1 to B1**: Another disk head movement to service B1.
4. **B1 to B2**: No disk head movement required as they are consecutive requests from Process B.
5. **B2 to A3**: Disk head movement required to service A3.
6. **A3 to A4**: No disk head movement required as they are consecutive requests from Process A.
7. **A4 to C2**: Disk head movement required to service C2.
8. **C2 to B3**: Disk head movement required to service B3.
9. **B3 to B4**: No disk head movement required as they are consecutive requests from Process B.
10. **B4 to A5**: Disk head movement required to service A5.
11. **A5 to A6**: No disk head movement required as they are consecutive requests from Process A.



PAGE REPLACEMENT ALGO.

SECOND CHANCE (OR CLOCK) PAGE REPLACEMENT ALGORITHM :

Overview: The Second Chance page replacement algorithm, also known as the Clock algorithm, is designed to optimize memory management by offering a balance between efficiency and simplicity. It is a more practical version of the Least Recently Used (LRU) algorithm, aiming to reduce overhead while maintaining effective page replacement.

Algorithm Steps:

1. Page Reference:

- If the referenced page is already in memory (in the frames array), set its corresponding second chance bit to 1, indicating recent use.

2. Page Fault (Page not in memory):

- If there is an empty frame available, insert the new page into that frame and set the second chance bit to 0.
- If the memory is full, proceed as follows:
 - Use the pointer to check each page cyclically.
 - If the second chance bit of the current page (pointed by the pointer) is 0, replace it with the new page and reset the second chance bit.
 - If the second chance bit is 1, set it to 0 and move the pointer to the next page, repeating the search process until a replaceable page is found.

3. Update:

After handling the page fault or reference, update the pointer to the next frame as necessary and increment the page fault count if a page replacement occurred.

PAGE REPLACEMENT ALGO.

Example Walkthrough:

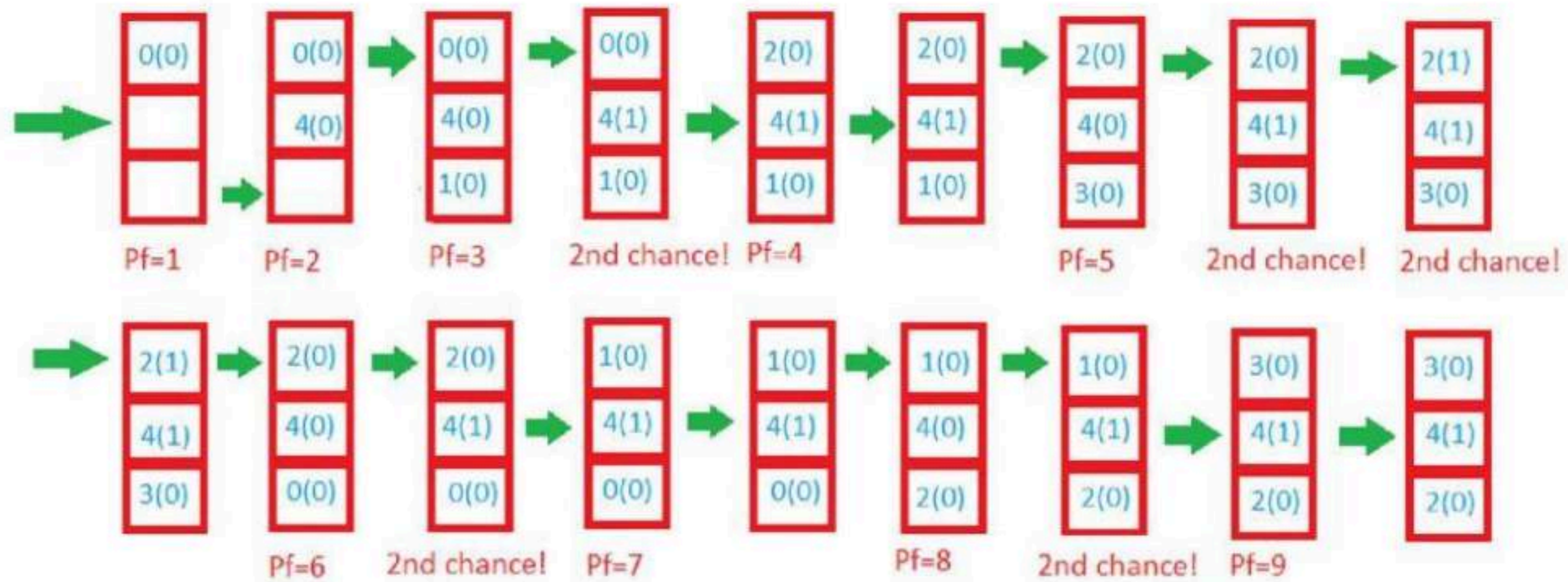
- **Pass-6 to Pass-18:** This sequence illustrates how the algorithm processes page references, updates second chance bits, and performs replacements. Pages that have been accessed recently are given another chance before being replaced, reflecting the algorithm's strategy.

Performance:

- The Second Chance algorithm performs similarly to LRU in terms of page fault count but with significantly reduced complexity and overhead.
- This makes it a practical choice for systems requiring efficient memory management without the computational burden of maintaining full LRU tracking.

Conclusion: The Second Chance page replacement algorithm offers a well-balanced solution for memory management, providing efficiency comparable to LRU but with simpler implementation. By leveraging recent page access history to give pages a "second chance," it optimizes the trade-off between performance and complexity, making it suitable for various practical applications in operating systems.

Page sequence: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4



VIRTUAL MEMORY & TLB

The Translation Lookaside Buffer (TLB) is a specialized cache used to improve the speed of virtual address translation. It is part of the memory management unit (MMU) in a computer's CPU.

In Ubuntu, the Translation Lookaside Buffer (TLB) operates in conjunction with the Linux kernel and hardware MMU (Memory Management Unit) to optimize virtual memory management.

TLB in Virtual Memory Management on Ubuntu The Translation Lookaside Buffer (TLB) is an essential component in Ubuntu's virtual memory management system. It helps optimize the translation of virtual addresses to physical addresses, significantly enhancing performance.



ROLE OF TLB IN UBUNTU

01

- **1. Address Translation :** Ubuntu, like other Linux distributions, uses virtual memory to allow processes to use a larger address space than the physical memory. The TLB helps speed up this address translation process.
-

02

- **2. Performance Monitoring with perf :** Ubuntu provides tools to monitor TLB performance, such as perf. This tool can measure the number of TLB loads and misses, helping diagnose performance issues.
-

03

- **3. Reducing TLB Misses with HugePages :** HugePages can help reduce TLB misses by using larger memory pages, thereby reducing the number of entries needed in the TLB.

ROLE OF TLB IN UBUNTU

Address Translation

1. **Without TLB:** Every memory access requires a page table lookup:
 - A process accesses virtual address 0xBEEF1234.
 - The CPU must traverse the page tables to find the physical address 0xDEAD5678.
 - This lookup involves multiple memory accesses, causing delays.
2. **With TLB:** The TLB caches recent translations:
 - A process accesses virtual address 0xBEEF1234.
 - The CPU checks the TLB and finds the cached translation to 0xDEAD5678.
 - The physical address is used immediately, skipping the slow page table lookup.

Impact of TLB in performance in UBUNTU

Improved Memory Access Time

- **TLB Hits:** When the TLB contains the required translation, memory access is almost instantaneous.
- **TLB Misses:** If the translation is not in the TLB, the CPU must perform a slower page table lookup.

CPU Efficiency

- **Reduced Overhead:** TLB hits reduce the overhead of address translation, allowing the CPU to perform other tasks more efficiently.
- **Context Switching:** During context switches, TLB entries may become invalid. Techniques like TLB tagging with process identifiers help mitigate this issue.

Program Behavior

- **Locality of Reference:** Programs with good locality of reference (frequently accessing a small set of memory locations) benefit more from TLB, resulting in higher TLB hit rates.
- **Using Large Pages:** Configuring applications to use HugePages can reduce TLB misses and improve performance.

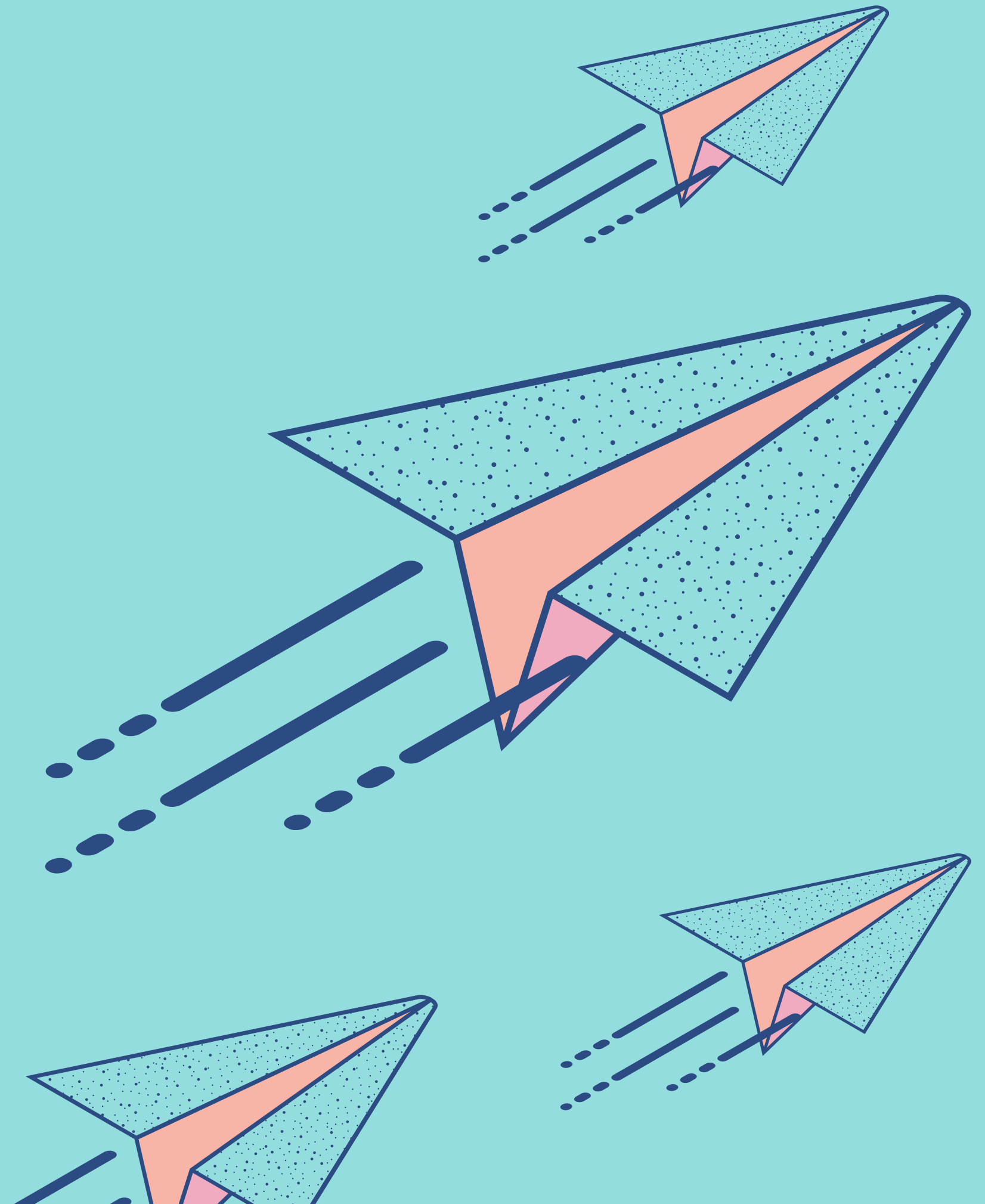
Member contribution

TEAM 14
UBUNTU

Aadhira	Gowtham	Ragul	Aravinth	Vicky
(Demo) IPC - Shared Memory File - Management	(Demo) Scheduling Algorithm Memory Management	Disk scheduling	Page Replacement	Virtual Memory & TLB

Do you have any questions?

ASK US! We hope you
learned something new.



THANK YOU!

