Name: Ayush.D.Chougule
Roll no: 08 TYCSE - D
PRN: 1032231720

OSL

13/10/25

Assignment - 08

* Aim: Write a program to simulate the first in first out (FIFO) page replacement algorithm.

* Objective:
- To understand Various concept:
  - Memory management
  - Paging
  - ~~Paging~~ Virtual memory.
  - Page replacement stratergise.

* Theory:

i) Memory management requirements:
Efficient memory management ensure optimal system performance. It must allocate memory dynamically, protect process from each other and support multitasking while minimizing fragmentation.

ii) Paging:
Is a memory management technique that divides physical memory into fixed size blocks called frames and logical memory into pages. It eliminates external fragmentation and allows non-contigeous memory Allocation, improving flexibility and efficiency.

iii) FIFO Algorithm:
It is a page replacement algorithm replaces the oldest page in memory when a new page needs to be loaded. It maintains a queue of pages and removes the one that arrived first, regardless of usage frequency.

iv) Belady's Anomaly:

Occurs when increasing the number of page frames result in mo
page faults in certain algorithm like FIFO. This count e
behaviour shows that more memory doesn't always mea
performance, highlighting the need for better replacement s

* Input:
* Output:

} AT LAST

* Platform: Linux.
* Programming language: C/C++.
* Conclusion: Efficient memory management ensure performance p
FIFO and anomalies highlighting the need for sm
replacement strategies.

* FAQ:

1]. Describe page table, frame table and explain the hardware sup
required to implement paging.

→ Page table → Maps virtual pages to physical memory fra
Each process having it's own page table.

Frame table ⇒ Tracks the status of physical memory frames-
they are free/occupied.

Hardware support for paging ⇒
Paging requires a memory management unit (mmu), to translate Virtual address to physical address using the page table. Translation lookaside buffer (TLB) speeds up access by caching recent translations.

2]. Explain first fit, Best fit and worst fit terms.
→ i) First fit ⇒ Allocate the first Available block large enough

ii) Best fit ⇒ choose the smallest block that fits, minimizing wasted space.

iii) Worst fit ⇒ Picks the largest block, leaving big left over space.

3]. Illustrate example of Dynamic and fixed partitioning.
→ - Dynamic partitioning ⇒ memory is allocated as per needed, reducing waste but causing fragmentation.
- Fixed partitioning ⇒ memory is divided into 2 fixed sizes, simple but may waste space if process size varies.

Eg: Dynamic partitioning ⇒ Process A needs 200kb.
Process B needs 300 Kb
Process C needs 250 Kb.

Available memory = 1000 Kb.
So, Process A, B, C are allocated memory.

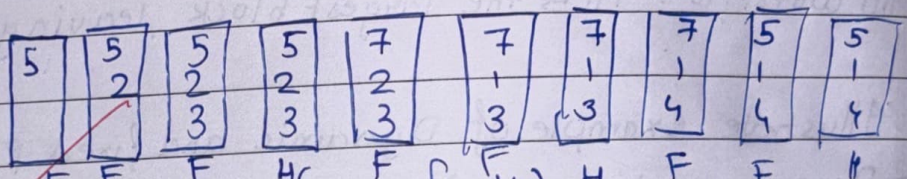
Remaining = 250 Kb
If B finish = 300kb more is free
If C finish = 250 kb more is free which can be allocated.

Eg: Fixed partitioning : Block 1, 2, 3, 4 = 250 Kb each.

Process A needs 200Kb = Assigned to block 1.
Process B needs 300 Kb = Cannot fit in any block even if memory is available.

* Input:
Enter number of pages : 10
Enter the page reference string : 5 2 3 2 7 1 3 4 5 1
Enter number of frames : 3

* OUTPUT.



page 5 → 5 – – (page fault)
page 2 → 5 2 – (page fault)
page 3 → 5 2 3 (page fault)
Page 2 → 5 2 3 (Hit)
page 7 → 7 2 3 (page fault)
page 1 → 7 1 3 (Page fault)
Page 3 → 7 1 3 (Hit ~~page fault~~)
page 4 → 7 1 4 (Page fault)
page 5 → 5 1 4 (Page fault)
page 1 → 5 1 4 (HIT)

total pages : 10
total page faults : 7
Total Hits. : 3
Fault ratio : 0.7
Hit ratio : 0.3

CODE:

```
#include <stdio.h>

int main() {
    int pages[50], frames[10];
    int n, f, i, j, next = 0, faults = 0, hits = 0;
    float hit_ratio, fault_ratio;

    printf("Enter number of pages: ");
    scanf("%d", &n);

    printf("Enter the page reference string: ");
    for (i = 0; i < n; i++)
        scanf("%d", &pages[i]);

    printf("Enter number of frames: ");
    scanf("%d", &f);


    for (i = 0; i < f; i++)
        frames[i] = -1;

    printf("\n--- FIFO Page Replacement ---\n");


    for (i = 0; i < n; i++) {
        int found = 0;


        for (j = 0; j < f; j++) {
```

```
        if (frames[j] == pages[i]) {
            found = 1;
            hits++;
            break;
        }
    }

    if (!found) {
        frames[next] = pages[i];
        next = (next + 1) % f;
        faults++;
    }

    printf("\nPage %2d --> ", pages[i]);
    for (j = 0; j < f; j++) {
        if (frames[j] == -1)
            printf(" - ");
        else
            printf("%2d ", frames[j]);
    }

    if (found)
        printf(" (Hit)");
    else
        printf(" (Page Fault)");
}

hit_ratio = (float)hits / n;
fault_ratio = (float)faults / n;

printf("\n\nTotal Pages: %d", n);
```

```
    printf("\nTotal Page Faults: %d", faults);
    printf("\nTotal Hits: %d", hits);
    printf("\nFault Ratio: %.2f", fault_ratio);
    printf("\nHit Ratio: %.2f\n", hit_ratio);

    return 0;
}
```

OUTPUT:

```
computer@computerVY:~$ gedit fifo.c
computer@computerVY:~$ gcc fifo.c
computer@computerVY:~$ ./a.out
Enter number of pages: 10
Enter the page reference string: 5 2 3 2 7 1 3 4 5 1
Enter number of frames: 3

--- FIFO Page Replacement ---

Page  5 -->  5  -  -  (Page Fault)
Page  2 -->  5  2  -  (Page Fault)
Page  3 -->  5  2  3  (Page Fault)
Page  2 -->  5  2  3  (Hit)
Page  7 -->  7  2  3  (Page Fault)
Page  1 -->  7  1  3  (Page Fault)
Page  3 -->  7  1  3  (Hit)
Page  4 -->  7  1  4  (Page Fault)
Page  5 -->  5  1  4  (Page Fault)
Page  1 -->  5  1  4  (Hit)

Total Pages: 10
Total Page Faults: 7
Total Hits: 3
Fault Ratio: 0.70
Hit Ratio: 0.30
```