Operating Systems Lab

Fall 2024-25(L59+60)

Student Name:- Parth Suri

Registration No:- 22BDS0116

Class No. :- VL2024250102445

Faculty Name:- Rahul Srivastava

Page Replacement Algorithms

Experiment 1: First-In-First-Out (FIFO) Page Replacement

Pseudocode:

```
1. Initialize an empty queue `frames` with a fixed size equal to the number of frames.
2. Initialize `page_faults` to 0.
3. For each `page` in `page_reference_string`:
   a. If `page` is NOT in `frames`:
      i. Increment `page_faults`.
      ii. If `frames` is full:
          - Remove the oldest page (front of the queue).
      iii. Add `page` to the end of `frames`.
4. Output the total `page_faults`.
```

Experiment 2: Least Recently Used (LRU) Page Replacement

Pseudocode:

```
1. Initialize an empty list `frames` with a fixed size equal to the number of frames.
2. Initialize `page_faults` to 0.
3. For each `page` in `page_reference_string`:
   a. If `page` is in `frames`:
      i. Remove `page` from `frames`.
      ii. Append `page` to the end of `frames` (marking it as recently used).
   b. Else:
      i. Increment `page_faults`.
      ii. If `frames` is full:
          - Remove the first page (least recently used).
      iii. Add `page` to the end of `frames`.
4. Output the total `page_faults`.
```

Experiment 3: Optimal Page Replacement

Pseudocode:

```
1. Initialize an empty list `frames` with a fixed size equal to the number of frames.
2. Initialize `page_faults` to 0.
3. For each `page` in `page_reference_string` at index `i`:
    a. If `page` is in `frames`, continue to next page.
    b. Else:
        i. Increment `page_faults`.
        ii. If `frames` is full:
            - Initialize `furthest_index` to -1 and `page_to_replace` as None.
            - For each `p` in `frames`:
                a. Find the next occurrence of `p` after index `i` in `page_reference_string`.
                b. If `p` does not appear again, select it as `page_to_replace`.
                c. If the next occurrence is further than `furthest_index`, update `furthest_index` and set `page_to_replace` to `p`.
            - Remove `page_to_replace` from `frames`.
        iii. Add `page` to `frames`.
4. Output the total `page_faults`.
```

Experiment 4: Least Frequently Used (LFU) Page Replacement

Pseudocode:

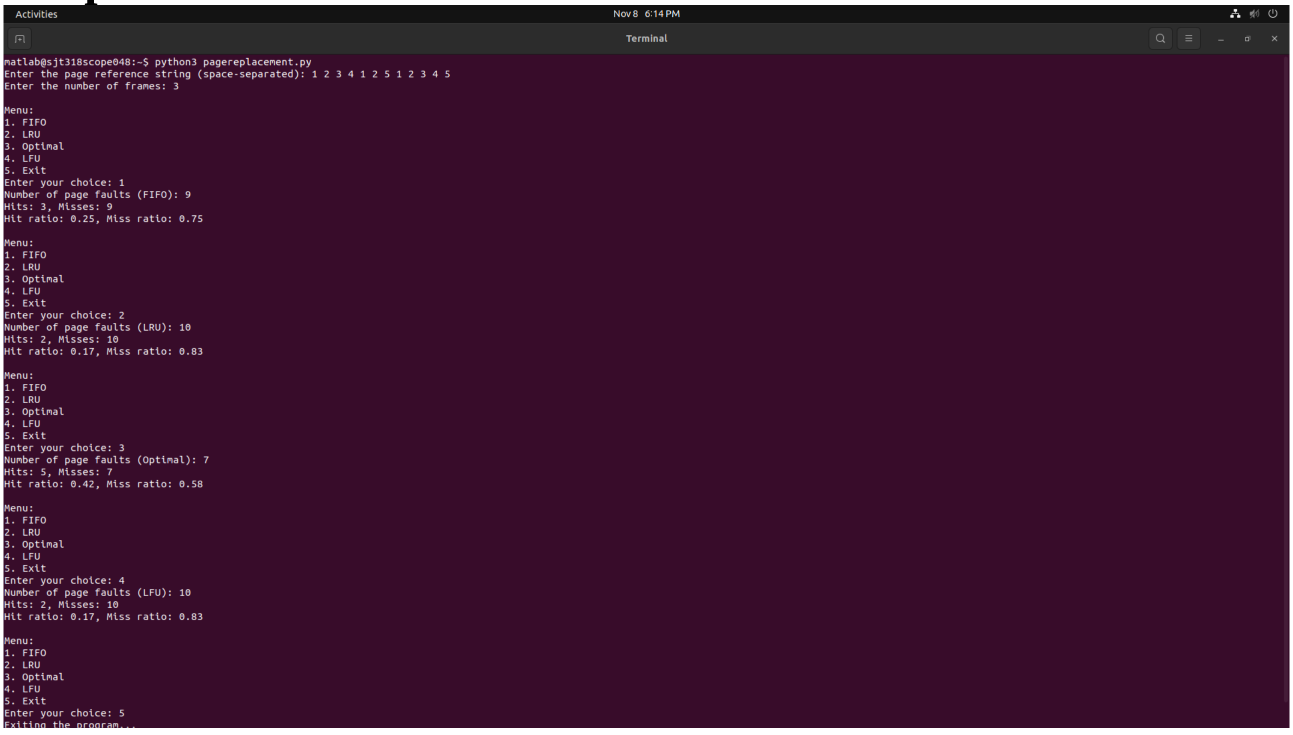
```
1. Initialize an empty list `frames` with a fixed size equal to the number of frames.
2. Initialize `frequency_count` as a dictionary to store the frequency of each page.
3. Initialize `page_faults` to 0.
4. For each `page` in `page_reference_string`:
    a. If `page` is in `frames`:
        i. Increment `frequency_count[page]`.
    b. Else:
        i. Increment `page_faults`.
        ii. If `frames` is full:
            - Find the page in `frames` with the lowest frequency in `frequency_count`.
            - If there is a tie, choose the least recently used page among the least frequent.
            - Remove that page from `frames` and `frequency_count`.
        iii. Add `page` to `frames` and set `frequency_count[page]` to 1.

5. Output the total `page_faults`.
```

Code:-

```
class PageReplacement:
    def __init__(self, pages, capacity):
        self.pages = pages
        self.capacity = capacity
    # FIFO Page Replacement
    def fifo(self):
        page_queue = []
        page_faults = 0
        hits = 0
        for page in self.pages:
            if page not in page_queue:
                if len(page_queue) >= self.capacity:
                    page_queue.pop(0) # Remove the first page (FIFO)
                page_queue.append(page)
                page_faults += 1
            else:
                hits += 1 # Page was found (hit)
        misses = page_faults
        hit_ratio = hits / len(self.pages)
        miss_ratio = misses / len(self.pages)
        return page_faults, hits, misses, hit_ratio, miss_ratio
    # LRU Page Replacement
    def lru(self):
        page_stack = []
        page_faults = 0
        hits = 0
        for page in self.pages:
            if page not in page_stack:
                if len(page_stack) >= self.capacity:
                    page_stack.pop(0) # Remove the least recently used page
                page_stack.append(page)
                page_faults += 1
            else:
                hits += 1 # Page was found (hit)
            page_stack.remove(page)
            page_stack.append(page) # Move the used page to the end
(recently used)
        misses = page_faults
        hit_ratio = hits / len(self.pages)
        miss_ratio = misses / len(self.pages)
        return page_faults, hits, misses, hit_ratio, miss_ratio
    # Optimal Page Replacement
    def optimal(self):
        page_queue = []
        page_faults = 0
        hits = 0
        for i, page in enumerate(self.pages):
            if page not in page_queue:
                if len(page_queue) >= self.capacity:
                    farthest = -1
                    index_to_remove = -1
                    for j in range(len(page_queue)):
                        try:
                            next_use = self.pages.index(page_queue[j], i + 1)
                        except ValueError:
                            next_use = float('inf')
                        if next_use > farthest:
                            farthest = next_use
                            index_to_remove = j
                    page_queue.pop(index_to_remove)
                page_queue.append(page)
                page_faults += 1
            else:
                hits += 1 # Page was found (hit)
        misses = page_faults
        hit_ratio = hits / len(self.pages)
        miss_ratio = misses / len(self.pages)
        return page_faults, hits, misses, hit_ratio, miss_ratio
    # LFU Page Replacement
    def lfu(self):
        page_freq = {}
        page_queue = []
        page_faults = 0
        hits = 0
        for page in self.pages:
            if page not in page_queue:
                if len(page_queue) >= self.capacity:
                    lfu_page = min(page_freq, key=page_freq.get) # Least
    frequently used page
                    page_queue.remove(lfu_page)
                    del page_freq[lfu_page]
                page_queue.append(page)
                page_freq[page] = page_freq.get(page, 0) + 1
                page_faults += 1
            else:
                page_freq[page] += 1
                hits += 1 # Page was found (hit)
        misses = page_faults
        hit_ratio = hits / len(self.pages)
        miss_ratio = misses / len(self.pages)
        return page_faults, hits, misses, hit_ratio, miss_ratio
    def main():
    pages = list(map(int, input("Enter the page reference string (space-
    separated): ").split()))
    capacity = int(input("Enter the number of frames: "))
    page_replacement = PageReplacement(pages, capacity)
    while True:
        print("\nMenu:")
        print("1. FIFO")
        print("2. LRU")
        print("3. Optimal")
        print("4. LFU")
        print("5. Exit")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            page_faults, hits, misses, hit_ratio, miss_ratio =
    page_replacement.fifo()
            print(f"Number of page faults (FIFO): {page_faults}")
            print(f"Hits: {hits}, Misses: {misses}")
            print(f"Hit ratio: {hit_ratio:.2f}, Miss ratio: {miss_ratio:.2f}")
        elif choice == 2:
            page_faults, hits, misses, hit_ratio, miss_ratio =
    page_replacement.lru()
            print(f"Number of page faults (LRU): {page_faults}")
            print(f"Hits: {hits}, Misses: {misses}")
            print(f"Hit ratio: {hit_ratio:.2f}, Miss ratio: {miss_ratio:.2f}")
```

```
        elif choice == 3:
            page_faults, hits, misses, hit_ratio, miss_ratio =
page_replacement.optimal()
            print(f"Number of page faults (Optimal): {page_faults}")
            print(f"Hits: {hits}, Misses: {misses}")
            print(f"Hit ratio: {hit_ratio:.2f}, Miss ratio: {miss_ratio:.2f}")
        elif choice == 4:
            page_faults, hits, misses, hit_ratio, miss_ratio =
page_replacement.lfu()
            print(f"Number of page faults (LFU): {page_faults}")
            print(f"Hits: {hits}, Misses: {misses}")
            print(f"Hit ratio: {hit_ratio:.2f}, Miss ratio: {miss_ratio:.2f}")
        elif choice == 5:
            print("Exiting the program...")
            break
        else:
            print("Invalid choice! Please try again.")
if __name__ == "__main__":
    main()
```

Output:-



```
matlab@sjt318scope048:~$ python3 pagereplacement.py
Enter the page reference string (space-separated): 1 2 3 4 1 2 5 1 2 3 4 5
Enter the number of frames: 3

Menu:
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Exit
Enter your choice: 1
Number of page faults (FIFO): 9
Hits: 3, Misses: 9
Hit ratio: 0.25, Miss ratio: 0.75

Menu:
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Exit
Enter your choice: 2
Number of page faults (LRU): 10
Hits: 2, Misses: 10
Hit ratio: 0.17, Miss ratio: 0.83

Menu:
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Exit
Enter your choice: 3
Number of page faults (Optimal): 7
Hits: 5, Misses: 7
Hit ratio: 0.42, Miss ratio: 0.58

Menu:
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Exit
Enter your choice: 4
Number of page faults (LFU): 10
Hits: 2, Misses: 10
Hit ratio: 0.17, Miss ratio: 0.83

Menu:
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Exit
Enter your choice: 5
Exiting the program...
```