OPERATING SYSTEMS LAB
DIGITAL ASSIGNMENT-4

AADITYA ROSHAN
22BIT0250

Memory Allocation Algorithms

1. First Fit

Code:

```
#include <stdio.h>

void firstFit(int blockSize[], int blocks, int processSize[], int processes) {
    int allocation[processes];
    for (int i = 0; i < processes; i++)
        allocation[i] = -1;
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < blocks; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;}
        }
    }
    for (int i = 0; i < processes; i++) {
        printf("Process %d -> ", i + 1);
        if (allocation[i] != -1)
            printf("Block %d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
int main() {
       int blocks, processes;
       printf("Enter the number of blocks: ");
       scanf("%d", &blocks);
       int blockSize[blocks];

       printf("Enter the size of each block: ");
       for (int i = 0; i < blocks; i++) {
           scanf("%d", &blockSize[i]);
       }
       printf("Enter the number of processes: ");
       scanf("%d", &processes);
       int processSize[processes];

       printf("Enter the size of each process: ");
       for (int i = 0; i < processes; i++) {
           scanf("%d", &processSize[i]);
       }
       firstFit(blockSize, blocks, processSize, processes);
       return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi first_fit.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc first_fit.c -o first_fit
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./first_fit
Enter the number of blocks: 5
Enter the size of each block: 100 300 50 200 250
Enter the number of processes: 5
Enter the size of each process: 200 150 100 25 50
Process 1 -> Block 2
Process 2 -> Block 4
Process 3 -> Block 1
Process 4 -> Block 2
Process 5 -> Block 2
```

2. Best Fit

Code:

```
#include <stdio.h>

void bestFit(int blockSize[], int blocks, int processSize[], int processes) {
    int allocation[processes];
    for (int i = 0; i < processes; i++)
        allocation[i] = -1;

    for (int i = 0; i < processes; i++) {
        int bestIdx = -1;

       for (int j = 0; j < blocks; j++) {
            if (blockSize[j] >= processSize[i]) {

                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx])
                    bestIdx = j;
            }
        }

`       if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i];
        }
    }

    for (int i = 0; i < processes; i++) {
        printf("Process %d -> ", i + 1);
        if (allocation[i] != -1)
            printf("Block %d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main() {
    int blocks, processes;

    printf("Enter the number of blocks: ");
    scanf("%d", &blocks);
    int blockSize[blocks];

    printf("Enter the size of each block: ");
    for (int i = 0; i < blocks; i++) {
        scanf("%d", &blockSize[i]);
    }
    printf("Enter the number of processes: ");
    scanf("%d", &processes);
    int processSize[processes];
    printf("Enter the size of each process: ");
    for (int i = 0; i < processes; i++) {
        scanf("%d", &processSize[i]);
}
    bestFit(blockSize, blocks, processSize, processes);
    return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi best_fit.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc best_fit.c -o best_fit
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./best_fit
Enter the number of blocks: 5
Enter the size of each block: 100 300 50 200 250
Enter the number of processes: 5
Enter the size of each process: 200 150 100 25 50
Process 1 -> Block 4
Process 2 -> Block 5
Process 3 -> Block 1
Process 4 -> Block 3
Process 5 -> Block 5
```

3. **Worst Fit**

Code:

```
#include <stdio.h>

void worstFit(int blockSize[], int blocks, int processSize[], int
processes) {
    int allocation[processes];


    for (int i = 0; i < processes; i++)
        allocation[i] = -1;


    for (int i = 0; i < processes; i++) {
        int worstIdx = -1;
        for (int j = 0; j < blocks; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx])
                    worstIdx = j;
            }
        }


        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            blockSize[worstIdx] -= processSize[i];
        }
    }

    for (int i = 0; i < processes; i++) {
        printf("Process %d -> ", i + 1);
        if (allocation[i] != -1)
            printf("Block %d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main() {
    int blocks, processes;

    printf("Enter the number of blocks: ");
    scanf("%d", &blocks);
    int blockSize[blocks];

    printf("Enter the size of each block: ");
    for (int i = 0; i < blocks; i++) {
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the number of processes: ");
    scanf("%d", &processes);
    int processSize[processes];
```

```
    printf("Enter the size of each process: ");
    for (int i = 0; i < processes; i++) {
        scanf("%d", &processSize[i]);
    }

    worstFit(blockSize, blocks, processSize, processes);
    return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi worst_fit.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc worst_fit.c -o worst_fit
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./worst_fit
Enter the number of blocks: 5
Enter the size of each block: 100 300 50 200 250
Enter the number of processes: 5
Enter the size of each process: 200 150 100 25 50
Process 1 -> Block 2
Process 2 -> Block 5
Process 3 -> Block 4
Process 4 -> Block 1
Process 5 -> Block 2
```

Page Replacement Algorithms

1. First-in-First-out (FIFO)

Code:

```
#include <stdio.h>

void FIFO(int pages[], int n, int capacity) {
    int frame[capacity], pageFaults = 0, index = 0, hits = 0;
    for (int i = 0; i < capacity; i++)
        frame[i] = -1;
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                hits++;
                break;
            }
        }
     page (miss)
        if (!found) {
            frame[index] = pages[i];
            index = (index + 1) % capacity;
            pageFaults++;
        }
    }
    printf("Page Faults: %d\n", pageFaults);
    printf("Page Hits: %d\n", hits);
    float hitRatio = (float)hits / n;
    float missRatio = (float)pageFaults / n;
    printf("Hit Ratio: %.2f\n", hitRatio);
    printf("Miss Ratio: %.2f\n", missRatio);
}

int main() {
    int n, capacity;

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

    printf("Enter the page reference string: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &capacity);
    FIFO(pages, n, capacity);

    return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi fifo.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc fifo.c -o fifo
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./fifo
Enter the number of pages: 12
Enter the page reference string: 1 3 0 3 5 6 3 2 4 3 0 3
Enter the number of frames: 3
Page Faults: 10
Page Hits: 2
Hit Ratio: 0.17
Miss Ratio: 0.83
```

2. Optimal Page Replacement (OPT)

Code:

```
#include <stdio.h>

void Optimal(int pages[], int n, int capacity) {
    int frame[capacity], pageFaults = 0, hits = 0;
    for (int i = 0; i < capacity; i++)
        frame[i] = -1;
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                hits++;
                break;}
        }
        if (!found) {
            int replaceIndex = -1, farthest = i + 1;
            for (int j = 0; j < capacity; j++) {
                int nextUse = -1;
                for (int k = i + 1; k < n; k++) {
                    if (frame[j] == pages[k]) {
                        nextUse = k;
                        break;
                    }
                }
                if (nextUse == -1) {
                    replaceIndex = j;
                    break;
                } else if (nextUse > farthest) {
                    farthest = nextUse;
                    replaceIndex = j;
                }
            }
            if (replaceIndex == -1) {
                for (int j = 0; j < capacity; j++) {
                    if (frame[j] == -1) {
                        replaceIndex = j;
                        break;
                    }
                }
            }
            frame[replaceIndex] = pages[i];
            pageFaults++;
        }
    }
    printf("Page Faults: %d\n", pageFaults);
    printf("Page Hits: %d\n", hits);
    float hitRatio = (float)hits / n;
    float missRatio = (float)pageFaults / n;
    printf("Hit Ratio: %.2f\n", hitRatio);
    printf("Miss Ratio: %.2f\n", missRatio);
}
```

```
int main() {
    int n, capacity;

    printf("Enter the number of pages: ");
    scanf("%d", &n);
    int pages[n];
    printf("Enter the page reference string: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &capacity);
    Optimal(pages, n, capacity);
    return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi opt.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc opt.c -o opt
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./opt
Enter the number of pages: 12
Enter the page reference string: 1 3 0 3 5 6 3 2 4 3 0 3
Enter the number of frames: 3
Page Faults: 7
Page Hits: 5
Hit Ratio: 0.42
Miss Ratio: 0.58
```

3. Least Recently Used (LRU)

Code:

```
#include <stdio.h>

void LRU(int pages[], int n, int capacity) {
    int frame[capacity], pageFaults = 0, hits = 0;
    for (int i = 0; i < capacity; i++)
        frame[i] = -1;

    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                hits++;
                break;
            }
        }
        if (!found) {
            int replaceIndex = -1, leastRecent = -1;
            for (int j = 0; j < capacity; j++) {
                int isRecent = 0;
                for (int k = i - 1; k >= 0; k--) {
                    if (frame[j] == pages[k]) {
                        isRecent = 1;
                        if (leastRecent == -1 || k < leastRecent) {
                            leastRecent = k;
                            replaceIndex = j;
                        }
                        break;
                    }
                }
              if (frame[j] == -1) {
                    replaceIndex = j;
                    break;
                }
            }
            frame[replaceIndex] = pages[i];
            pageFaults++;
        }
    }

    printf("Page Faults: %d\n", pageFaults);
    printf("Page Hits: %d\n", hits);


    float hitRatio = (float)hits / n;
    float missRatio = (float)pageFaults / n;

    printf("Hit Ratio: %.2f\n", hitRatio);
    printf("Miss Ratio: %.2f\n", missRatio);
}
```

```c
int main() {
    int n, capacity;

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

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

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

    LRU(pages, n, capacity);

    return 0;
}
```

Output:

```
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ vi lru.c
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ gcc lru.c -o lru
aaditya@AadityaPC:~/OS_Lab/LAB DA-4$ ./lru
Enter the number of pages: 12
Enter the page reference string: 1 3 0 3 5 6 3 2 4 3 0 3
Enter the number of frames: 3
Page Faults: 8
Page Hits: 4
Hit Ratio: 0.33
Miss Ratio: 0.67
```