



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
GENERAL SEMESTER 2023-24

B.Tech - CSE

BCSE303P: Operating Systems Lab

5. Implement the below concepts:

- a) Dynamic memory allocation algorithms - First-fit, Best-fit, Worst-fit algorithms.
- b) Page Replacement Algorithms - FIFO, LRU and Optimal.

a)

```
main.c
1 #include <stdio.h>
2
3 #define MAX_MEMORY_SIZE 100
4
5 // Memory block structure
6 typedef struct {
7     int start;
8     int size;
9     int allocated;
10 } MemoryBlock;
11
12 // Function prototypes
13 void initializeMemory();
14 void displayMemory();
15 int firstFit(int processSize);
16 int bestFit(int processSize);
17 int worstFit(int processSize);
18
19 // Global variables
20 MemoryBlock memory[MAX_MEMORY_SIZE];
21
22 int main() {
23     int choice, processSize, allocatedIndex;
24
25     // Initialize memory
26     initializeMemory();
27
28     do {
29         // Display menu
30         printf("\n--- Memory Allocation Menu ---\n");
31         printf("1. First Fit\n");
32         printf("2. Best Fit\n");
33         printf("3. Worst Fit\n");
34         printf("4. Exit\n");
35         printf("Enter your choice: ");
36         scanf("%d", &choice);
37
38         switch (choice) {
39             case 1:
40                 printf("Enter process size: ");
41                 scanf("%d", &processSize);
42                 allocatedIndex = firstFit(processSize);
43                 if (allocatedIndex != -1)
44                     printf("Process allocated at index %d\n", allocatedIndex);
45                 else
46                     printf("No suitable memory block found\n");
47                 displayMemory();
48                 break;
49
50             case 2:
51                 printf("Enter process size: ");
52                 scanf("%d", &processSize);
53                 allocatedIndex = bestFit(processSize);
54                 if (allocatedIndex != -1)
55                     printf("Process allocated at index %d\n", allocatedIndex);
56                 else
57                     printf("No suitable memory block found\n");
58                 displayMemory();
59                 break;
60
61             case 3:
62                 printf("Enter process size: ");
63                 scanf("%d", &processSize);
64                 allocatedIndex = worstFit(processSize);
65                 if (allocatedIndex != -1)
66                     printf("Process allocated at index %d\n", allocatedIndex);
67                 else
68                     printf("No suitable memory block found\n");
69                 displayMemory();
70                 break;
71         }
72     } while (choice != 4);
73 }
```

```

72     case 4:
73         printf("Exiting...\n");
74         break;
75
76     default:
77         printf("Invalid choice. Please try again.\n");
78     }
79 } while (choice != 4);
80
81 return 0;
82 }
83
84 // Initialize memory blocks
85 void initializeMemory() {
86     int i;
87
88     for (i = 0; i < MAX_MEMORY_SIZE; i++) {
89         memory[i].start = i * 10; // Assuming each block size is 10
90         memory[i].size = 10;
91         memory[i].allocated = 0;
92     }
93 }
94
95 // Display memory blocks
96 void displayMemory() {
97     int i;
98
99     printf("\n--- Memory Blocks ---\n");
100    printf("Start\tSize\tAllocated\n");
101    for (i = 0; i < MAX_MEMORY_SIZE; i++)
102        printf("%d\t%d\t%d\n", memory[i].start, memory[i].size, memory[i].allocated);
103 }

```

```

105 // First Fit allocation algorithm
106 int firstFit(int processSize) {
107     int i;
108
109     for (i = 0; i < MAX_MEMORY_SIZE; i++) {
110         if (memory[i].allocated == 0 && memory[i].size >= processSize) {
111             memory[i].allocated = 1;
112             return i;
113         }
114     }
115
116     return -1;
117 }

```

```

119 // Best Fit allocation algorithm
120 int bestFit(int processSize) {
121     int i, bestFitIndex = -1, bestFitDiff = MAX_MEMORY_SIZE;
122
123     for (i = 0; i < MAX_MEMORY_SIZE; i++) {
124         if (memory[i].allocated == 0 && memory[i].size >= processSize) {
125             int diff = memory[i].size - processSize;
126             if (diff < bestFitDiff) {
127                 bestFitDiff = diff;
128                 bestFitIndex = i;
129             }
130         }
131     }
132
133     if (bestFitIndex != -1)
134         memory[bestFitIndex].allocated = 1;
135
136     return bestFitIndex;
137 }

```

```

139 // Worst Fit allocation algorithm
140 int worstFit(int processSize) {
141     int i, worstFitIndex = -1, worstFitDiff = -1;
142
143     for (i = 0; i < MAX_MEMORY_SIZE; i++) {
144         if (memory[i].allocated == 0 && memory[i].size >= processSize) {
145             int diff = memory[i].size - processSize;
146             if (diff > worstFitDiff) {
147                 worstFitDiff = diff;
148                 worstFitIndex = i;
149             }
150         }
151     }
152
153     if (worstFitIndex != -1)
154         memory[worstFitIndex].allocated = 1;
155
156     return worstFitIndex;
157 }
158 }

```

OUTPUT:

FIRST FIT-

PURVA SHARMA 21BCE0169

Console: connected (Running: 30 seg)

```

--- Memory Allocation Menu ---
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter your choice: 1
Enter process size: 2
Process allocated at index 0

```

```

--- Memory Blocks ---
Start  Size  Allocated
0      10     1
10     10     0
20     10     0
30     10     0
40     10     0
50     10     0
60     10     0
70     10     0
80     10     0
90     10     0
100    10     0
110    10     0

```

BEST FIT-

```
PURVA SHARMA 21BCE0169

> Console: connected (Running: 12 seg)

--- Memory Allocation Menu ---
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter your choice: 2
Enter process size: 2
Process allocated at index 0

--- Memory Blocks ---
Start  Size  Allocated
0      10    1
10     10    0
20     10    0
30     10    0
40     10    0
50     10    0
60     10    0
70     10    0
80     10    0
90     10    0
100    10    0
110    10    0
```

WORST FIT-

```
PURVA SHARMA 21BCE0169

> Console: connected (Running: 16 seg)

--- Memory Allocation Menu ---
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter your choice: 3
Enter process size: 2
Process allocated at index 0

--- Memory Blocks ---
Start  Size  Allocated
0      10    1
10     10    0
20     10    0
30     10    0
40     10    0
50     10    0
60     10    0
70     10    0
80     10    0
90     10    0
100    10    0
110    10    0
```

b)

```
PURVA SHARMA 21BCE0169

main.c
1 #include <stdio.h>
2
3 #define MAX_PAGES 100
4 #define MAX_FRAMES 10
5
6 // Function prototypes
7 void initializeFrames();
8 int isPageInFrames(int page, int frameCount);
9 int getPageIndex(int page, int frameCount);
10 int getFifoPageIndex(int frameCount);
11 int getLruPageIndex(int frameCount, int usedCount[MAX_FRAMES]);
12 int getOptimalPageIndex(int frameCount, int pages[], int pageIndices[], int currentIndex);
13
14 // Global variables
15 int frames[MAX_FRAMES];
16 int pageFaults = 0;
17
18 int main() {
19     int pageReferences[MAX_PAGES];
20     int pageIndices[MAX_PAGES];
21     int frameCount, referenceCount;
22     int i, j;
23
24     // Read input
25     printf("Enter the number of frames: ");
26     scanf("%d", &frameCount);
27
28     printf("Enter the number of page references: ");
29     scanf("%d", &referenceCount);
30
31     printf("Enter the page reference string: ");
32     for (i = 0; i < referenceCount; i++) {
33         scanf("%d", &pageReferences[i]);
34         pageIndices[i] = -1;
35     }
36 }
```

```

37 // FIFO algorithm
38 initializeFrames();
39
40 for (i = 0; i < referenceCount; i++) {
41     if (!isPageInFrames(pageReferences[i], frameCount)) {
42         int index = getFifoPageIndex(frameCount);
43         frames[index] = pageReferences[i];
44         pageFaults++;
45     }
46 }
47
48 printf("\nFIFO Page Replacement:\n");
49 printf("Page Faults: %d\n", pageFaults);
50
51 // LRU algorithm
52 initializeFrames();
53 pageFaults = 0;
54 int usedCount[MAX_FRAMES] = {0};
55
56 for (i = 0; i < referenceCount; i++) {
57     if (!isPageInFrames(pageReferences[i], frameCount)) {
58         int index = getLruPageIndex(frameCount, usedCount);
59         frames[index] = pageReferences[i];
60         pageFaults++;
61     }
62     usedCount[getPageIndex(pageReferences[i], frameCount)] = i + 1;
63 }
64
65 printf("\nLRU Page Replacement:\n");
66 printf("Page Faults: %d\n", pageFaults);
67
68 // LRU algorithm
69 initializeFrames();
70 pageFaults = 0;
71 int usedCount[MAX_FRAMES] = {0};
72
73 for (i = 0; i < referenceCount; i++) {
74     if (!isPageInFrames(pageReferences[i], frameCount)) {
75         int index = getLruPageIndex(frameCount, usedCount);
76         frames[index] = pageReferences[i];
77         pageFaults++;
78     }
79     usedCount[getPageIndex(pageReferences[i], frameCount)] = i + 1;
80 }
81
82 printf("\nLRU Page Replacement:\n");
83 printf("Page Faults: %d\n", pageFaults);
84
85 // Optimal algorithm
86 initializeFrames();
87 pageFaults = 0;
88
89 for (i = 0; i < referenceCount; i++) {
90     if (!isPageInFrames(pageReferences[i], frameCount)) {
91         int index = getOptimalPageIndex(frameCount, &pageReferences[i], &pageIndices[i], i);
92         frames[index] = pageReferences[i];
93         pageFaults++;
94     }
95 }
96
97 printf("\nOptimal Page Replacement:\n");
98 printf("Page Faults: %d\n", pageFaults);
99
100 return 0;
101 }
102

```

```

103 // Initialize frames
104 void initializeFrames() {
105     int i;
106     for (i = 0; i < MAX_FRAMES; i++)
107         frames[i] = -1;
108 }
109
110 // Check if page is present in frames
111 int isPageInFrames(int page, int frameCount) {
112     int i;
113     for (i = 0; i < frameCount; i++) {
114         if (frames[i] == page)
115             return 1;
116     }
117     return 0;
118 }
119
120 // Get the index of a page in frames
121 int getPageIndex(int page, int frameCount) {
122     int i;
123     for (i = 0; i < frameCount; i++) {
124         if (frames[i] == page)
125             return i;
126     }
127     return -1;
128 }
129
130 // Get the index of the page to replace using FIFO algorithm
131 int getFifoPageIndex(int frameCount) {
132     return (pageFaults % frameCount);
133 }
134

```





```

123 // Get the index of the page to replace using LRU algorithm
124 int getLruPageIndex(int frameCount, int usedCount[MAX_FRAMES]) {
125     int i, minIndex = 0, minValue = usedCount[0];
126
127     for (i = 1; i < frameCount; i++) {
128         if (usedCount[i] < minValue) {
129             minValue = usedCount[i];
130             minIndex = i;
131         }
132     }
133
134     return minIndex;
135 }
136
137 // Get the index of the page to replace using Optimal algorithm
138 int getOptimalPageIndex(int frameCount, int pages[], int pageIndices[], int currentIndex) {
139     int i, j, maxIndex = 0, maxValue = -1;
140
141     for (i = 0; i < frameCount; i++) {
142         if (pageIndices[i] == -1)
143             return i;
144
145         int found = 0;
146
147         for (j = currentIndex + 1; j < MAX_PAGES; j++) {
148             if (pages[j] == frames[i]) {
149                 if (j > maxIndex) {
150                     maxIndex = j;
151                     maxValue = i;
152                 }
153             }
154             found = 1;
155             break;
156         }
157
158         if (!found)
159             return i;
160     }
161     return maxIndex;
162 }
163

```

OUTPUT:

 PURVA SHARMA 21BCE0169

>_    Console: connection closed (Running: 15 seg) 

```

Enter the number of frames: 6
Enter the number of page references: 4
Enter the page reference string: 2 3 7 6

```

```

FIFO Page Replacement:
Page Faults: 4

```

```

LRU Page Replacement:
Page Faults: 4

```

```

Optimal Page Replacement:
Page Faults: 4

```

PAJAMA PADMAI