

```
main.c  address.txt  ⋮
1  #include <stdio.h>
2
3  // Logical defines.
4  #define TRUE 1
5  #define FALSE 0
6
7  // Page defines.
8  #define PAGE_SIZE 100
9  #define MAX_FRAMES 5
10
11 // Paging algorithm modes.
12 #define FIFO 1
13 #define LRU 2
14 #define OPT 3
15
16 typedef struct {
17     int pageNum;
18     int usage;
19     int lastUsed;
20     int timeStamp; // Add timestamp for LRU
21 } pageData;
22
23 int adds[100];
24 pageData frames[MAX_FRAMES];
25
26 int readAddressStream(char *filename);
27 void showAdds(int numAdds);
28 int pageReplace(int numAdds, char mode);
29 int searchFrameTable(int pageNum, int nFrames, char mode);
30 void showFrameTable(int nFrames);
31 int getIndexOfOldestPage(int nFrames);
32 int getIndexOfLRUPage(int nFrames);
33 int getIndexOfBeladyPage(int nFrames, int numAdds);
34
35 int main(void) {
36     int numAdds;
37     int pageFaults;
38
39     // Send message to the user.
40     printf("Hello TV Land! \n");
41
42     // Read the incoming address stream from the input file.
43     numAdds = readAddressStream("address.txt");
44     printf("numAdds = %d \n", numAdds);
45
46     // Show the addresses to the user.
```

```
46 // Show the addresses to the user.
47 showAdds(numAdds);
48
49 // Implement the FIFO page replacement algorithm.
50 printf("Page replacement (FIFO) \n");
51 pageFaults = pageReplace(numAdds, FIFO);
52 printf("pageFaults = %d \n", pageFaults);
53
54 // Implement the LRU page replacement algorithm.
55 printf("\n");
56 printf("Page replacement (LRU) \n");
57 pageFaults = pageReplace(numAdds, LRU);
58 printf("pageFaults = %d \n", pageFaults);
59
60 // Implement Belady's page replacement algorithm.
61 printf("\n");
62 printf("Page replacement (Belady's OPT) \n");
63 pageFaults = pageReplace(numAdds, OPT);
64 printf("pageFaults = %d \n", pageFaults);
65 }
66
67 int readAddressStream(char *filename) {
68     FILE *in;
69     int address;
70     int j;
71     in = fopen(filename, "r");
72     j = 0;
73     while (fscanf(in, "%d", &address) != EOF) {
74         adds[j] = address;
75         j = j + 1;
76     }
77     fclose(in);
78     return j;
79 }
80
81 void showAdds(int numAdds) {
82     int j;
83     printf("Address Stream. \n");
84     for (j = 0; j < numAdds; j++) {
85         printf("%d \n", adds[j]);
86     }
87 }
88
89 int searchFrameTable(int pageNum, int nFrames, char mode) {
90     int j;
91     int frameIndex = -1;
92     char searching = TRUE;
```

```
main.c address.txt
92 char searching = TRUE;
93
94 for (j = 0; j < nFrames && searching; j++) {
95     if (frames[j].pageNum == pageNum) {
96         frameIndex = j;
97         searching = FALSE;
98     }
99 }
100
101 return frameIndex;
102 }
103
104 int getIndexOfOldestPage(int nFrames) {
105     int j;
106     int old = frames[0].lastUsed;
107     int oldIndex = 0;
108
109     for (j = 1; j < nFrames; j++) {
110         if (frames[j].lastUsed < old) {
111             old = frames[j].lastUsed;
112             oldIndex = j;
113         }
114     }
115
116     return oldIndex;
117 }
118
119 int getIndexOfLRUPage(int nFrames) {
120     int j;
121     int leastRU = frames[0].timeStamp;
122     int lIndex = 0;
123
124     for (j = 1; j < nFrames; j++) {
125         if (frames[j].timeStamp < leastRU) {
126             leastRU = frames[j].timeStamp;
127             lIndex = j;
128         }
129     }
130
131     return lIndex;
132 }
133
134 int getIndexOfBeladyPage(int nFrames, int numAdds) {
135     int j;
136     int opt = -1;
137     int oIndex = -1;
```

```
136 int opt = -1;
137 int oIndex = -1;
138
139 // Simple approach: Assume future references follow the same order as the initial sequence.
140 for (j = 0; j < nFrames; j++) {
141     int futureIndex = searchFrameTable(frames[j].pageNum, numAdds, OPT);
142     if (futureIndex > opt) {
143         opt = futureIndex;
144         oIndex = j;
145     }
146 }
147
148 return oIndex;
149 }
150
151 int pageReplace(int numAdds, char mode) {
152     int j;
153     int pageNum;
154     int frameNum, nFrames = 0; // Initialize nFrames
155     int repFrame;
156     int pageFaults = 0;
157
158     for (j = 0; j < numAdds; j++) {
159         // Use the current address as the page number
160         pageNum = adds[j] / PAGE_SIZE;
161
162         // Calculate page and offset. (Assume simple logic, replace if needed)
163         // int offset = pageNum % PAGE_SIZE;
164
165         // Search frame table to see if the page is present in memory.
166         frameNum = searchFrameTable(pageNum, nFrames, mode);
167
168         if (frameNum == -1) {
169             // If the page is not found in the page table, add it.
170             if (nFrames < MAX_FRAMES) {
171                 // If there is room in the table, add the frame.
172                 frames[nFrames].pageNum = pageNum;
173                 frames[nFrames].usage = 1;
174                 frames[nFrames].lastUsed = j; // Update timestamp for LRU
175                 nFrames++;
176             } else {
177                 // Page fault.
178                 switch (mode) {
179                     case FIFO:
180                         // Find the oldest frame.
181                         repFrame = getIndexOfOldestPage(nFrames);
```

```

main.c address.txt
180 // Find the oldest frame.
181 repFrame = getIndexOfOldestPage(nFrames);
182 break;
183 case LRU:
184 // Find the least recently used frame.
185 repFrame = getIndexOfLRUPage(nFrames);
186 break;
187 case OPT:
188 // Find the frame used furthest in the future.
189 repFrame = getIndexOfBeladyPage(nFrames, numAdds);
190 break;
191 }
192
193 // Replace the frame.
194 frames[repFrame].pageNum = pageNum;
195 frames[repFrame].usage = 1;
196 frames[repFrame].lastUsed = j; // Update timestamp for LRU
197 pageFaults++;
198 }
199 } else {
200 // Frame was found in the table.
201 // Update the usage count and last time used.
202 frames[frameNum].usage++;
203 frames[frameNum].lastUsed = j; // Update timestamp for LRU
204 }
205
206 // Show the frame table to the user.
207 showFrameTable(nFrames);
208 }
209
210 return pageFaults;
211 }
212
213 void showFrameTable(int nFrames) {
214 int j;
215 printf("Frame Table - ");
216 for (j = 0; j < MAX_FRAMES; j++) {
217 if (j < nFrames) {
218 printf("%d ", frames[j].pageNum);
219 } else {
220 printf("# ");
221 }
222 }
223 printf("\n");
224 }
225
226

```

```
Hello TV Land!
numAdds = 20
Address Stream.
721
43
121
222
44
327
45
428
223
328
45
329
224
122
225
46
123
722
47
124
```

Page replacement (FIFO)

```
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2
```

Page replacement (LRU)

```
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
```

```
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2
```

Page replacement (LRU)

```

Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2

```

Page replacement (Belady's OPT)

[illegible]