Operating system

11. Multithreading

Code:

```
#include <stdio.h>
#include <pthread.h>

void* print_message(void* ptr) {
    char* message = (char*) ptr;
    printf("%s\n", message);
    return NULL;
}

int main() {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, print_message, "Thread 1: Hello");
    pthread_create(&thread2, NULL, print_message, "Thread 2: World");

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    return 0;
}
```

Output:

```
Thread 1: Hello
Thread 2: World
```

12. FIFO Paging

Code:

```
#include <stdio.h>

int main() {
    int pages[50], n, frames[10], f, i, j = 0, k, pageFaults = 0, found;
    printf("Enter number of pages: ");
    scanf("%d", &n);
    printf("Enter the pages: ");
    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;

    for(i = 0; i < n; i++) {
        found = 0;
        for(k = 0; k < f; k++) {
            if(frames[k] == pages[i]) {
                found = 1;
                break;
            }
        }
        if(!found) {
            frames[j] = pages[i];
            j = (j + 1) % f;
            pageFaults++;
        }
    }
    printf("Total Page Faults: %d\n", pageFaults);
    return 0;
}
```

Output:

```
Enter number of pages: 12
Enter the pages: 1 2 3 4 1 2 5 1 2 3 4 5
Enter number of frames: 3

Total Page Faults: 9
```

13. LRU Paging

Code:

```
#include <stdio.h>

int findLRU(int time[], int n) {
    int i, min = time[0], pos = 0;
    for(i = 1; i < n; ++i) {
        if(time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
```

```
    return pos;
}

int main() {
    int pages[50], frames[10], time[10], n, f, i, j, pos, pageFaults = 0, counter = 0,
found;
    printf("Enter number of pages: ");
    scanf("%d", &n);
    printf("Enter the pages: ");
    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;
        time[i] = 0;
    }

    for(i = 0; i < n; ++i) {
        found = 0;
        for(j = 0; j < f; ++j) {
            if(frames[j] == pages[i]) {
                counter++;
                time[j] = counter;
                found = 1;
                break;
            }
        }
        if(!found) {
            pos = findLRU(time, f);
            counter++;
            frames[pos] = pages[i];
            time[pos] = counter;
            pageFaults++;
        }
    }

    printf("Total Page Faults: %d\n", pageFaults);
    return 0;
}
```

Output:

```
Enter number of pages: 12
```

Enter the pages: 1 2 3 4 1 2 5 1 2 3 4 5
Enter number of frames: 3

Total Page Faults: 10

14. Optimal Paging

Code:

```
#include <stdio.h>

int predict(int pages[], int frames[], int pn, int index, int fn) {
    int res = -1, farthest = index;
    for(int i = 0; i < fn; i++) {
        int j;
        for(j = index; j < pn; j++) {
            if(frames[i] == pages[j]) {
                if(j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if(j == pn)
            return i;
    }
    return (res == -1) ? 0 : res;
}

int main() {
    int pages[50], frames[10], n, f, i, j, hit = 0, found;
    printf("Enter number of pages: ");
    scanf("%d", &n);
    printf("Enter the pages: ");
    for(i = 0; i < n; i++) scanf("%d", &pages[i]);
    printf("Enter number of frames: ");
    scanf("%d", &f);

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

    for(i = 0; i < n; i++) {
```

```
        found = 0;
        for(j = 0; j < f; j++) {
            if(frames[j] == pages[i]) {
                hit++;
                found = 1;
                break;
            }
        }
        if(!found) {
            if(index < f)
                frames[index++] = pages[i];
            else {
                int pos = predict(pages, frames, n, i + 1, f);
                frames[pos] = pages[i];
            }
        }
    }
    printf("Total Page Faults: %d\n", n - hit);
    return 0;
}
```

Output:

```
Enter number of pages: 12
Enter the pages: 1 2 3 4 1 2 5 1 2 3 4 5
Enter number of frames: 3
Total Page Faults: 7
```

15. Sequential File Allocation

Code:

```
#include <stdio.h>

struct File {
    int startBlock;
    int length;
};

int main() {
    int memory[100] = {0}, i, j, n;
    struct File files[10];
```

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

    for(i = 0; i < n; i++) {
        int start, len;
        printf("Enter start block and length for file %d: ", i+1);
        scanf("%d%d", &start, &len);
        int allocated = 1;

        for(j = start; j < start + len; j++) {
            if(memory[j] == 1) {
                allocated = 0;
                break;
            }
        }

        if(allocated) {
            for(j = start; j < start + len; j++)
                memory[j] = 1;
            files[i].startBlock = start;
            files[i].length = len;
            printf("File %d allocated.\n", i+1);
        } else {
            printf("File %d cannot be allocated.\n", i+1);
        }
    }

    return 0;
}
```

Output:

```
Enter number of files: 3
Enter start block and length for file 1: 2 3
Enter start block and length for file 2: 5 2
Enter start block and length for file 3: 3 2

File 1 allocated.
File 2 allocated.
File 3 cannot be allocated.
```