Operating system- programs

16. First Fit Memory Allocation

Code:

```
#include <stdio.h>

int main() {
    int blockSize[10], processSize[10], allocation[10], m, n;
    printf("Enter number of blocks: ");
    scanf("%d", &m);
    printf("Enter block sizes: ");
    for (int i = 0; i < m; i++) scanf("%d", &blockSize[i]);

    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter process sizes: ");
    for (int i = 0; i < n; i++) scanf("%d", &processSize[i]);

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

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i]+1);
        else
            printf("Not Allocated\n");
    }
    return 0;
}
```

Output:

```
Enter number of blocks: 5
Enter block sizes: 100 500 200 300 600
Enter number of processes: 4
Enter process sizes: 212 417 112 426

Process No.  Process Size  Block No.
1            212           2
2            417           5
3            112           1
4            426           Not Allocated
```

17. First-Come-First-Served (FCFS) Disk Scheduling

Code:

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

int main() {
    int n, i, head, seek = 0;
    printf("Enter number of disk requests: ");
    scanf("%d", &n);
    int req[n];
    printf("Enter request sequence: ");
    for (i = 0; i < n; i++) scanf("%d", &req[i]);
    printf("Enter initial head position: ");
    scanf("%d", &head);

    for (i = 0; i < n; i++) {
        seek += abs(req[i] - head);
        head = req[i];
    }

    printf("Total Seek Time: %d\n", seek);
    return 0;
}
```

Output:

```
Enter number of disk requests: 8
```

Enter request sequence: 98 183 37 122 14 124 65 67
Enter initial head position: 53

Total Seek Time: 640

18. SCAN Disk Scheduling (Elevator Algorithm)

Code:

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

int compare(const void *a, const void *b) { return (*(int*)a - *(int*)b); }

int main() {
    int i, j, n, head, size, direction;
    printf("Enter number of requests: ");
    scanf("%d", &n);
    int req[n+1];
    printf("Enter request sequence: ");
    for (i = 0; i < n; i++) scanf("%d", &req[i]);
    printf("Enter disk size: ");
    scanf("%d", &size);
    printf("Enter initial head position: ");
    scanf("%d", &head);
    printf("Enter direction (0=left, 1=right): ");
    scanf("%d", &direction);

    req[n] = head;
    n++;
    qsort(req, n, sizeof(int), compare);

    int seek = 0, pos;
    for (i = 0; i < n; i++) if (req[i] == head) { pos = i; break; }

    if (direction == 1) {
        for (i = pos; i < n - 1; i++) seek += abs(req[i+1] - req[i]);
        seek += abs(size - 1 - req[n-1]);
        seek += abs(req[pos-1] - 0);
        for (i = pos - 1; i > 0; i--) seek += abs(req[i] - req[i-1]);
    } else {
        for (i = pos; i > 0; i--) seek += abs(req[i] - req[i-1]);
        seek += req[0];
        seek += abs(req[n-1] - size + 1);
```

```
        for (i = pos + 1; i < n - 1; i++) seek += abs(req[i] - req[i+1]);
    }

    printf("Total Seek Time: %d\n", seek);
    return 0;
}
```

Output:

```
Enter number of requests: 8
Enter request sequence: 98 183 37 122 14 124 65 67
Enter disk size: 200
Enter initial head position: 53
Enter direction (0=left, 1=right): 1

Total Seek Time: 208
```

19. Single-Level Directory Simulation

Code:

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

struct Directory {
    char fname[10][20];
    int fcount;
};

int main() {
    struct Directory dir;
    dir.fcount = 0;
    int choice;
    char name[20];

    while (1) {
        printf("\n1. Create File\n2. Delete File\n3. Display Files\n4. Exit\nEnter your
choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter file name: ");
                scanf("%s", name);
                int found = 0;
```

```
            for (int i = 0; i < dir.fcount; i++) {
                if (strcmp(name, dir.fname[i]) == 0) {
                    found = 1;
                    break;
                }
            }
            if (found)
                printf("File already exists.\n");
            else {
                strcpy(dir.fname[dir.fcount], name);
                dir.fcount++;
                printf("File created.\n");
            }
            break;
        case 2:
            printf("Enter file name to delete: ");
            scanf("%s", name);
            found = 0;
            for (int i = 0; i < dir.fcount; i++) {
                if (strcmp(name, dir.fname[i]) == 0) {
                    for (int j = i; j < dir.fcount - 1; j++)
                        strcpy(dir.fname[j], dir.fname[j+1]);
                    dir.fcount--;
                    found = 1;
                    printf("File deleted.\n");
                    break;
                }
            }
            if (!found) printf("File not found.\n");
            break;
        case 3:
            if (dir.fcount == 0)
                printf("No files.\n");
            else {
                printf("Files:\n");
                for (int i = 0; i < dir.fcount; i++)
                    printf("%s\n", dir.fname[i]);
            }
            break;
        case 4:
            return 0;
    }
  }
}
```

Output:

```
1. Create File
2. Delete File
3. Display Files
4. Exit
Enter your choice: 1
Enter file name: test
File created.

Enter your choice: 3
Files:
test
```

20. Two-Level Directory Simulation

Code:

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

struct File {
    char name[20];
};

struct User {
    char name[20];
    struct File files[10];
    int fileCount;
};

int main() {
    struct User users[5];
    int userCount = 0, choice;
    char uname[20], fname[20];
    int i, j;

    while (1) {
        printf("\n1. Create User\n2. Create File\n3. Display\n4. Exit\nEnter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter user name: ");
                scanf("%s", uname);
```

```
            strcpy(users[userCount].name, uname);
            users[userCount].fileCount = 0;
            userCount++;
            break;
        case 2:
            printf("Enter user name: ");
            scanf("%s", uname);
            for (i = 0; i < userCount; i++) {
                if (strcmp(users[i].name, uname) == 0) {
                    printf("Enter file name: ");
                    scanf("%s", fname);
                    strcpy(users[i].files[users[i].fileCount].name, fname);
                    users[i].fileCount++;
                    printf("File created.\n");
                    break;
                }
            }
            if (i == userCount)
                printf("User not found.\n");
            break;
        case 3:
            for (i = 0; i < userCount; i++) {
                printf("User: %s\n", users[i].name);
                for (j = 0; j < users[i].fileCount; j++)
                    printf("  %s\n", users[i].files[j].name);
            }
            break;
        case 4:
            return 0;
    }
  }
}
```

Output:

```
1. Create User
2. Create File
3. Display
4. Exit
Enter choice: 1
Enter user name: alice

Enter choice: 2
Enter user name: alice
```

```
Enter file name: notes
File created.

Enter choice: 3
User: alice
  notes
```