# Memory Management


Session: 2021 – 2025

**Submitted by:**

Muhammad Yaqoob 2021-CS-118

**Supervised by:**

Ms. Abqa Javed

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# Contents

# 1 Memory Management

## 1.1 Best Fit

1. Implement the **BEST FIT** memory management by taking a number of processes,and block and their respective size and implement it. You need to print a table showing the process size, block size and block number which is assigned to current process.

```
~$ ./bestFit
Enter the number of memory blocks: 4
Enter the size of each memory block:
3 6 3 4
Enter the number of processes: 4
Enter the size of each process:
2 4 5 3

Process No.     Process Size    Block No.
   1               2            1
   2               4            4
   3               5            2
   4               3            3
~$
```

FIGURE 1: Best Fit

### 1.1.1 Source Code:

```
#include <stdio.h>

#define MAX_BLOCKS 100
#define MAX_PROCESSES 100

int main() {
    int blocks[MAX_BLOCKS], processes[MAX_PROCESSES];
    int n_blocks, n_processes;
    int i, j;

    printf("Enter the number of memory blocks: ");
```

```
    scanf("%d", &n_blocks);

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

    printf("Enter the number of processes: ");
    scanf("%d", &n_processes);

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

    int allocation[n_processes];

    for (i = 0; i < n_processes; i++) {
        allocation[i] = -1;
        int best_block = -1;

        for (j = 0; j < n_blocks; j++) {
            if (blocks[j] >= processes[i]) {
                if (best_block == -1) {
                    best_block = j;
                } else if (blocks[j] < blocks[best_block]) {
                    best_block = j;
                }
            }
        }

        if (best_block != -1) {
            allocation[i] = best_block;
            blocks[best_block] -= processes[i];
        }
    }

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

    return 0;
}
```

## 1.2 First Fit

2. Implement the **FIRST FIT** memory management by taking a number of processes,and block and their respective size and implement it. You need to print a table showing the process size, block size and block number which is assigned to current process.

```
~$ ./firstFit
Enter the number of memory blocks: 4
Enter the size of each memory block:
4 2 4 2
Enter the number of processes: 4
Enter the size of each process:
3 4 3 2

Process No.     Process Size    Block No.
 1               3              1
 2               4              3
 3               3              Not Allocated
 4               2              2
~$
```

FIGURE 2: First Fit

### 1.2.1 Source Code:

```
#include <stdio.h>

#define MAX_BLOCKS 100
#define MAX_PROCESSES 100

int main() {
    int blocks[MAX_BLOCKS], processes[MAX_PROCESSES];
    int n_blocks, n_processes;
    int i, j;

    printf("Enter the number of memory blocks: ");
    scanf("%d", &n_blocks);

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

```
    printf("Enter the number of processes: ");
    scanf("%d", &n_processes);

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

    int allocation[n_processes];

    for (i = 0; i < n_processes; i++) {
        allocation[i] = -1;

        for (j = 0; j < n_blocks; j++) {
            if (blocks[j] >= processes[i]) {
                allocation[i] = j;
                blocks[j] -= processes[i];
                break;
            }
        }
    }

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

    return 0;
}
```

## 1.3 Worst Fit

### 1.3.1 Source Code:

```
#include <stdio.h>

#define MAX_BLOCKS 100
#define MAX_PROCESSES 100

int main() {
    int blocks[MAX_BLOCKS], processes[MAX_PROCESSES];
    int n_blocks, n_processes;
    int i, j;

    printf("Enter the number of memory blocks: ");
    scanf("%d", &n_blocks);

    printf("Enter the size of each memory block:\n");
```

```
~$ ./worstFit
Enter the number of memory blocks: 4
Enter the size of each memory block:
3 5 2 4
Enter the number of processes: 5
Enter the size of each process:
5 7 4 3 8

Process No.     Process Size    Block No.
   1               5           2
   2               7           Not Allocated
   3               4           4
   4               3           1
   5               8           Not Allocated
~$
```

FIGURE 3: Worst Fit

```
for (i = 0; i < n_blocks; i++) {
    scanf("%d", &blocks[i]);
}

printf("Enter the number of processes: ");
scanf("%d", &n_processes);

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

int allocation[n_processes];

for (i = 0; i < n_processes; i++) {
    allocation[i] = -1;
    int worst_block = -1;

    for (j = 0; j < n_blocks; j++) {
        if (blocks[j] >= processes[i]) {
            if (worst_block == -1) {
                worst_block = j;
            } else if (blocks[j] > blocks[worst_block]) {
                worst_block = j;
            }
        }
```

```
        }

        if (worst_block != -1) {
            allocation[i] = worst_block;
            blocks[worst_block] -= processes[i];
        }
    }

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

    return 0;
}
```