Tribhuvan University
Faculty of Humanities and Social Sciences

Lab report on:
Operating System Lab 7: Memory allocation

Submitted to:
Mr. Roshan Maharjan,
Er. Himal Chand Thapa ,
Department of Computer Application,
Himalaya College of Engineering,
Chyasal,Lalitpur

Submitted by:
Sujal Gurung
Roll no: 34
BCA II/IV
September 21, 2023

1 Objectives

- implement 1st fit & best fit memory allocation algorithms

2 Introduction

Previously, we talked about fixed partitioning method for loading processes from disk to memory. There are a few algorithms for doing os. The ones we cover in this lab are:

- **First fit algorithm:** When a process requests to be loaded, it is placed into the 1st partition that is large enough to store it.
- **Best fit algorithm:** CPU goes through all available partitions & plaitces process in the smallest one that can still accomodate the process. By doing so, it aims to find the "best fit" and minimize internal fragmentation.

3 Lab Work

3.1 Write a program to show first fit allocation in memory management

```c
#include <stdio.h>
#define NUM_BLOCKS 5
#define NUM_REQUESTS 7

int blocks[NUM_BLOCKS];
int requests[NUM_REQUESTS];

void print_state() {
    int i;
    printf("Blocks: ");
    for (i = 0; i < NUM_BLOCKS; i++) {
        printf("%d  \t", blocks[i]);
    }
    printf("\nRequests: ");
    for (i = 0; i < NUM_REQUESTS; i++) {
        printf("%d \t ", requests[i]);
    }
    printf("\n");
}

void first_fit() {
    int i, j;
    for (i = 0; i < NUM_REQUESTS; i++) {
        for (j = 0; j < NUM_BLOCKS; j++) {
            if (requests[i] <= blocks[j]) {
                printf("Allocated %d to block %d\n", requests[i], j);
                blocks[j] -= requests[i];
                break;
            }
        }
        if (j == NUM_BLOCKS) {
            printf("Unable to allocate %d\n", requests[i]);
        }
```

```
    }
}
int main() {
    int i;

    // Initialize the blocks and requests arrays
    printf("Enter sizes of %d blocks ", NUM_BLOCKS);
    for(i=0; i<NUM_BLOCKS; i++) {
        scanf("%d", &blocks[i]);
    }
    printf("Enter sizes of %d requests ", NUM_REQUESTS);
    for(i=0; i<NUM_REQUESTS; i++) {
        scanf("%d", &requests[i]);
    }

    // Print the initial state of the system
    printf("Initial state:\n");
    print_state();

    // Allocate memory using the first fit algorithm
    first_fit();

    // Print the final state of the system
    printf("Final state:\n");
    print_state();
    return 0;
}
```

Output:

```
Enter sizes of 5 blocks 100 500 200 300 600
Enter sizes of 7 requests 150 200 50 400 100 250 300

Initial state:
Blocks: 100     500     200     300     600
Requests: 150    200     50      400     100     250     300

Allocated 150 to block 1
Allocated 200 to block 1
Allocated 50 to block 0
Allocated 400 to block 4
Allocated 100 to block 1
Allocated 250 to block 3
Unable to allocate 300

Final state:
Blocks: 50      50      200     50      200
Requests: 150    200     50      400     100     250     300
```

3.2 Write a program to show best fit allocation in memory management

```
#include <stdio.h>
#include <limits.h>  // Include this header for INT_MAX
#define NUM_BLOCKS 5
#define NUM_REQUESTS 7
```

```c
int blocks[NUM_BLOCKS];
int requests[NUM_REQUESTS];

void print_state() {
    int i;
    printf("Blocks: ");
    for (i = 0; i < NUM_BLOCKS; i++) {
        printf("%d  \t", blocks[i]);
    }
    printf("\nRequests: ");
    for (i = 0; i < NUM_REQUESTS; i++) {
        printf("%d \t ", requests[i]);
    }
    printf("\n");
}

void best_fit() {
    int i, j;
    for (i = 0; i < NUM_REQUESTS; i++) {
        int best_block = -1; // Initialize to an invalid block index
        int min_fragmentation = INT_MAX; // Initialize to a large value

        for (j = 0; j < NUM_BLOCKS; j++) {
            if (requests[i] <= blocks[j] && blocks[j] - requests[i] <
            ↪  min_fragmentation) {
                best_block = j;
                min_fragmentation = blocks[j] - requests[i];
            }
        }

        if (best_block != -1) {
            printf("Allocated %d to block %d\n", requests[i], best_block);
            blocks[best_block] -= requests[i];
        } else {
            printf("Unable to allocate %d\n", requests[i]);
        }
    }
}

int main() {
    int i;

    // Initialize the blocks and requests arrays
    printf("Enter sizes of %d blocks ", NUM_BLOCKS);
    for (i = 0; i < NUM_BLOCKS; i++) {
        scanf("%d", &blocks[i]);
    }
    printf("\nEnter sizes of %d requests ", NUM_REQUESTS);
    for (i = 0; i < NUM_REQUESTS; i++) {
        scanf("%d", &requests[i]);
    }

    // Print the initial state of the system
    printf("Initial state:\n");
    print_state();

    // Allocate memory using the best fit algorithm
    best_fit();
```

```
    // Print the final state of the system
    printf("Final state:\n");
    print_state();
    return 0;
}
```

Output:

```
Enter sizes of 5 blocks 100 500 200 300 600
Enter sizes of 7 requests 150 200 50 400 100 250 300

Initial state:
Blocks: 100     500     200     300     600
Requests: 150    200     50      400     100     250     300

Allocated 150 to block 2
Allocated 200 to block 3
Allocated 50 to block 2
Allocated 400 to block 1
Allocated 100 to block 0
Allocated 250 to block 4
Allocated 300 to block 4

Final state:
Blocks: 0       100     0       100     50
Requests: 150    200     50      400     100     250     300
```

4 Conclusion

Thus, we learned about memory allocation algorithms for fixed partitioning by implementing & testing them. We found that Best fit was the better alogrithm ss it allocated memory for all requests, unlike first fit.