**OPERATING SYSTEM LAB (COM -312)**

**Implementing First, Best and Worst Fit Contiguous memory allocation techniques by keeping a free/busy list of jobs organized by memory location **

**CSE, MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY**

## **BACHELOR OF ENGINEERING**

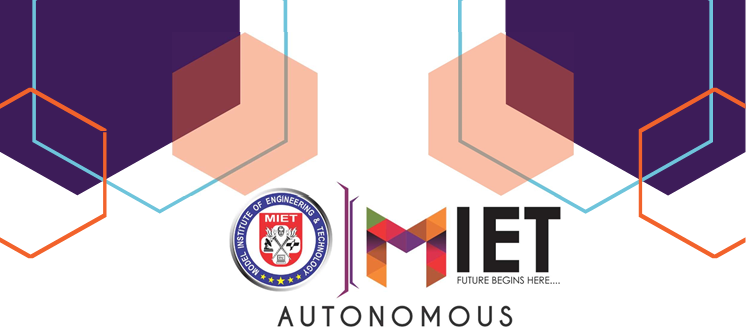
**In**

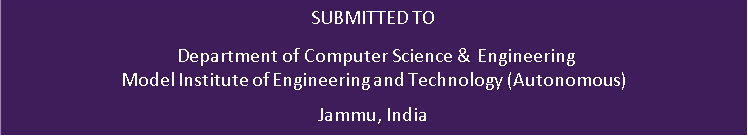
## **Computer Science & Engineering**

**SUBMITTED BY**

**ROHIN SABHARWAL(2021A1R032), HARI PRIYA DUTTA(2021A1R046),**

**VIVEK SINGH WAZIR(2021A1R036), UJJWAL SHAWN(2021A1R038)**

****

****

**ACKNOWLEDGEMENT**

Under CRIE (Center for Research, Innovation and Entrepreneurship) we both worked under the guidance of mentors of CRIE, who guided us through a way leading to professionalism and practical hands on work experience.It is indeed with a great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our Director Ankur Gupta, “MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY”, for the facilities provided to accomplish this main project. We would like to thank our Prof.Ashok Kumar , Head of the Department of Computer Science and Engineering , MIET. For this constructive criticism throughout our project .We feel elated in manifesting our sense of gratitude for our internal project guide. Asst. Saurabh Sharma , Department of Computer Science and Engineering ,MIET.He has been a constant source of inspiration for us and we are very deeply thankful to him.

**FACULTY AND MEMBER**

## **Dr. Ankur Gupta** - He is the Director at the ModeI Institute of Engineering and Technology, Jammu, India, besides being a Professor in the Department of Computer Science and Engineering. Prior to joining academia, he worked as Technical Team Lead at Hewlett Packard, developing software in the network management and e-Commerce domains. He has 2 patents to his name and 20 patents pending. He obtained his B.E, Hons. He is a senior member of the ACM, senior member IEEE and a life-member of the Computer Society of India. He has received competitive grants over Rs 2 crores from various funding agencies and faculty awards from IBM and EMC.

## **Prof. Ashok Kumar -** He has over 21 years of experience in industry, academics, research and academic administration. He did his Doctorate from IKG Punjab Technical University, Jalandhar; Master of Engineering from Punjab Engineering College, Chandigarh and Bachelor’s degree from NIT, Calicut, Kerala. He is Fellow of Institution of Electronics and Telecommunication Engineers (IETE). His research interest areas are Optical Networks, Electronics Product Design, Wireless Sensor Networks and Digital Signal Processing.. He is reviewer of many reputed national and international journals. He has also coordinated many national and international conferences.

## **Mr. Saurabh Sharma -** He is working as Assistant Professor in the Department of Computer Science at Model Institute of Engineering and Technology, Jammu and has over 10 years of experience in academics and research. He has done his M. Tech CSE in Web Mining from Maharishi Markandeshwar University, Mullana, Ambala (HR) in the year 2011.His research interest areas are Web Mining, Data Mining, Machine Learning, ANN, Pattern Classification and Object Identification. He has, to his credit, 25 research papers published in journals of national and international repute and he has guided 11 M. Tech Dissertations.

**ABSTRACT**

The technique to control and coordinate the computer memory, to assign blocks to different running programs for the optimization of an entire system's performance, is called memory management. It resides in hardware of the Operating System (OS), and in applications and programs.

Memory management is the main part of the Operating System which is basically used to control or handle the primary memory. Processes move between the disk and the main memory, during the time of execution. It keeps track of every memory location. It also checks the memory requirement for the processes and allocates memory as per that requirement. It also makes decisions about when the memory is allocated to the process. It updates the status of the memory whenever memory gets freed.

**First Fit Memory Allocation :-** This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, the first job claims the first available memory with space more than or equal to its size.

**Best Fit Memory Allocation :-** This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently.

**Worst Fit Memory Allocation:-**  In this allocation technique, the process traverses the whole memory and always searches for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

**CONTENTS**

Acknowledgement i

Faculty and member ii

Abstract iii

Contents iv

Introduction v

Objective vi

Terminology(flowchart) vii

Algorithm viii

Implementation ix

a) Coding

b) Output

References x

**1. INTRODUCTION**

* **First Fit Memory Allocation :-**

This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, first job claims the first available memory with space more than or equal to it’s size. The operating system doesn’t search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

* **Advantages of First-Fit Memory Allocation:**  
  It is fast in processing. As the processor allocates the nearest available memory partition to the job, it is very fast in execution.
* **Disadvantages of First-Fit Memory Allocation :**  
  It wastes a lot of memory. The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not. It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.
* **Best Fit Memory Allocation :-**

This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

* **Advantages of Best-Fit Allocation :**   
  Memory Efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient. To save memory from getting wasted, it is the best method.
* **Disadvantages of Best-Fit Allocation :**   
  It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work

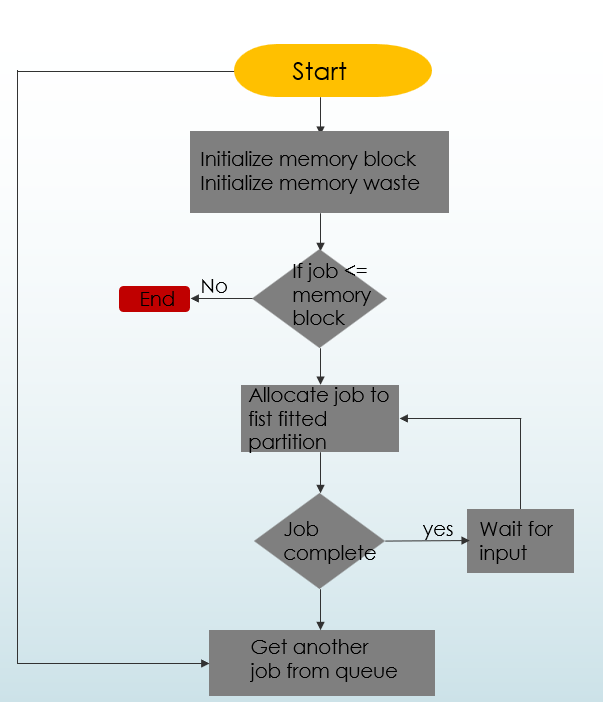
**Worst Fit Memory Allocation:-**

In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

* **Advantages of Worst-Fit Allocation :**   
  Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.
* **Disadvantages of Worst-Fit Allocation :**   
  It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

**2. OBJECTIVE**

* For both fixed and dynamic memory allocation schemes, the operating system must keep a list of each memory location noting which are free and which are busy.
* Then as new jobs come into the system, the free partitions must be allocated.
* These partitions may be allocated by **First-Fit Memory Allocation**, **Best-Fit Memory Allocation** and **Worst-Fit Memory Allocation**.
* Objective is to implement these allocation schemes on the Linux operating system .
* Users should be given a choice to select the particular allocation strategy.

**3. Methodology ( flowchart )**

**a**. **First fit**:

Fig. 1

**b. Best Fit:**

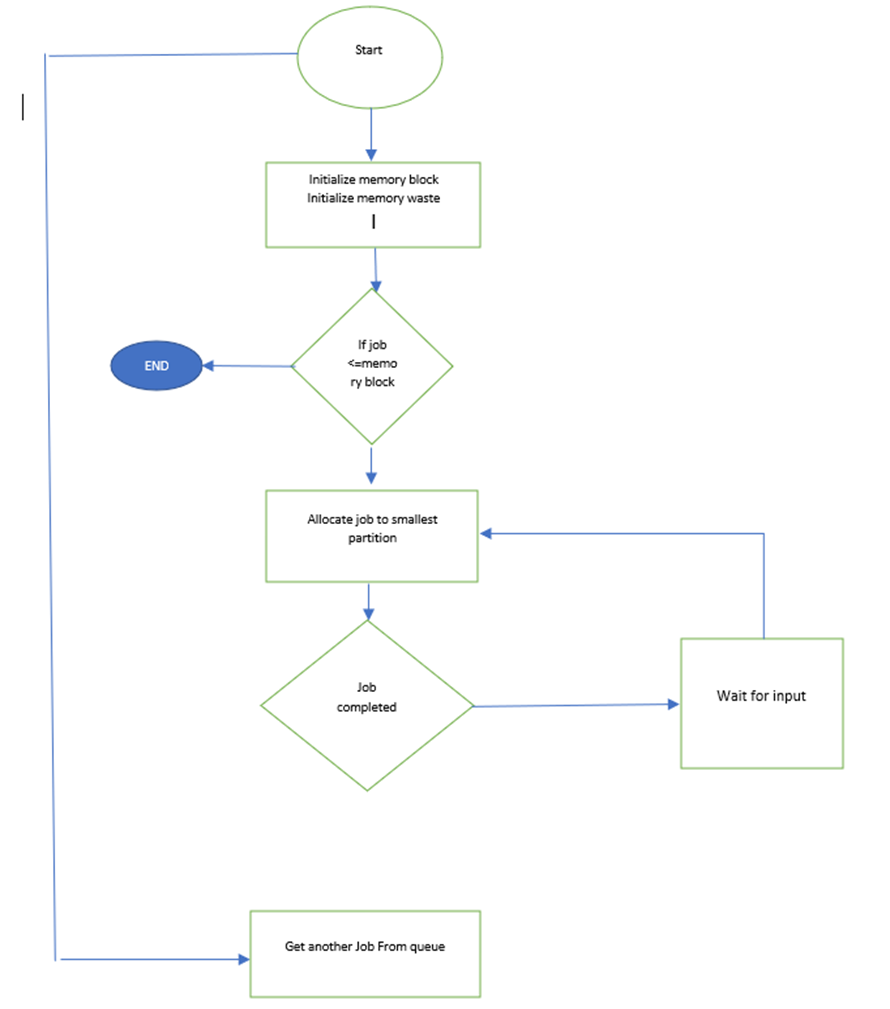
****

Fig. 2

**c. Worst Fit:**

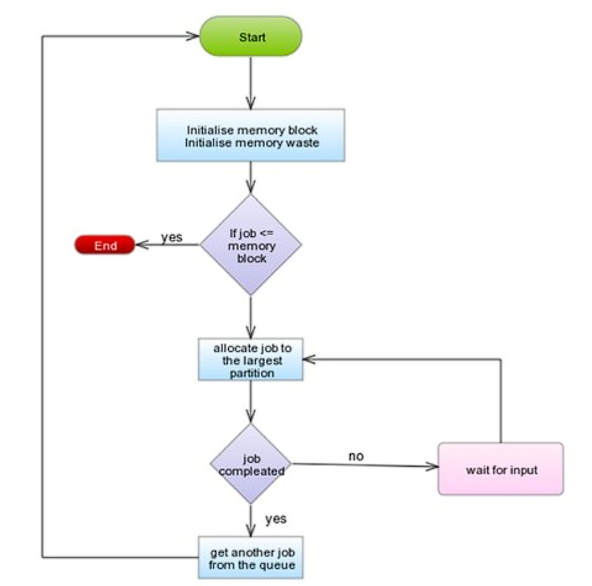


Fig. 3

**4. Algorithm**

**First Fit Memory Allocation Algorithm :**

First-Fit Allocation In the first tit algorithm the allocator keeps a list of free blocks (known as the free list) and, on receiving a request for memory, scares along the list for the first block that is large enough to satisfy the request if the chosen block is significantly larger than that requested, then it is usually split, and the remainder added to the list as another free block The first fic algorithm performs reasonably well, as it ensures that allocations are quick When recycling free blocks there is a choice as to where to add the blocks to the free list effectively in what orders the free lot is kept

**Algorithm for allocate (n)**

Size(block)= n + size(header)

Scan free list for first block with nWords >=size(block)

If block not found

Failure (time for garbage collection!)

Else if free block nWords >= Free block nWords – size(block)

In-use block n words = size(block

Else

Unlink Block from free list

Return pointer to block

"Threshold must be at least size header) +1 to leave room for header and link Threshold can be set higher to combat fragmentation

**Allocation time is O(K)** (K-number of Blocks in free list)

**Best Fit Memory Allocation Algorithm:**

1. Get no. of Processes and no. of blocks.
2. After that get the size of each block and process requests.
3. Then select the best memory block that can be allocated using the above definition.
4. Display the processes with the blocks that are allocated to a respective process.
5. Value of Fragmentation is optional to display to keep track of wasted memory.
6. Stop.

**Worst Fit Memory Allocation Algorithm:**

1. Input memory blocks and processes with sizes.
2. Initialize all memory blocks as free.

1. Start by picking each process and find the maximum block size that can be assigned to current process i.e., find max(bockSize[1], blockSize[2],.....blockSize[n]) > processSize[current],
2. If found then assign it to the current process.
3. If not then leave that process and keep checking the further processes.

**5. Implementation**

**a) Coding**

//C Program for Implementing First, Best and Worst Fir Contiguous memory allocation techniques by keeping free/busy list of jobs organized by memory location.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

 struct node {

int start;

int end;

int size;

struct node \*next;

};

struct node \*head = NULL;

struct node \*tail = NULL;

 void insert(int start, int end, int size)

{ struct node\*temp=structnode\*)malloc(sizeof(struct node));

temp->start = start;

temp->end = end;

temp->size = size;

temp->next = NULL;

if(head == NULL) {

head = temp;

tail = temp; }

else {

tail->next = temp;

tail = temp; }}

void display() {

struct node \*temp = head;

while(temp != NULL) {

printf("%d\t%d\t%d\n", temp->start, temp->end, temp->size);

temp = temp->next; }}

void first\_fit(int start, int end, int size)

{

struct node \*temp = head;

while(temp != NULL)

{ if(temp->size >= size)

{

printf("\nJob Allocated at %d\n", temp->start);

temp->start = temp->start + size;

temp->size = temp->size - size;

return; }

temp = temp->next; }

printf("\nJob Not Allocated\n"); }

void best\_fit(int start, int end, int size)

{ struct node \*temp = head;

struct node \*best = NULL;

while(temp != NULL)

{ if(temp->size >= size)

{ if(best == NULL)

{ best = temp;

}

else if(best->size > temp->size)

{ best = temp;

} }

temp = temp->next; }

if(best != NULL) {

printf("\nJob Allocated at %d\n", best->start);

best->start = best->start + size;

best->size = best->size - size; }

else {

printf("\nJob Not Allocated\n"); }

if(worst != NULL) void worst\_fit(int start, int end, int size) {

struct node \*temp = head;

struct node \*worst = NULL;

while(temp != NULL)

{

if(temp->size >= size)

{

if(worst == NULL)

{

worst = temp; }

}

else if(worst->size < temp->size)

{

worst = temp;

}

temp = temp->next;

}

{

printf("\nJob Allocated at %d\n", worst->start);

worst->start = worst->start + size;

worst->size = worst->size - size;

}

Else

{

printf("\nJob Not Allocated\n");

}

}

int main()

{

int start, end, size, choice;

char ch;

do {

printf("\nEnter the start, end and size of the memory block: ");

scanf("%d%d%d", &start, &end, &size);

insert(start, end, size);

printf("\nDo you want to enter more memory blocks? (y/n): ");

scanf(" %c", &ch);

}while(ch == 'y' || ch == 'Y');

printf("\nThe free/busy list of memory blocks is: \n");

display();

do {

printf("\nEnter the size of the job: ");

scanf("%d", &size);

printf("\n1. First Fit\n2. Best Fit\n3. Worst Fit\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: first\_fit(start, end, size);

break;

case 2: best\_fit(start, end, size);

break;

case 3: worst\_fit(start, end, size);

break;

default: printf("\nInvalid Choice\n"); }

printf("\nDo you want to enter more jobs? (y/n): ");

scanf(" %c", &ch); }

while(ch == 'y' || ch == 'Y');

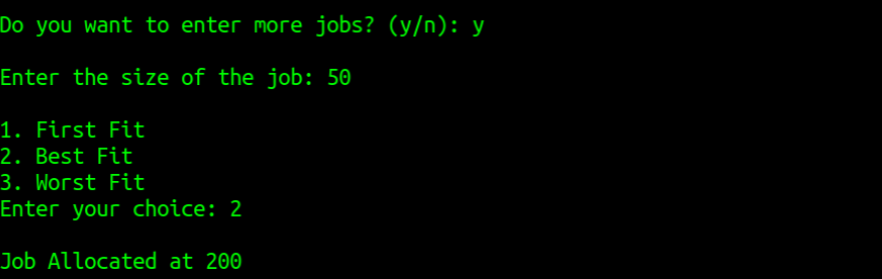
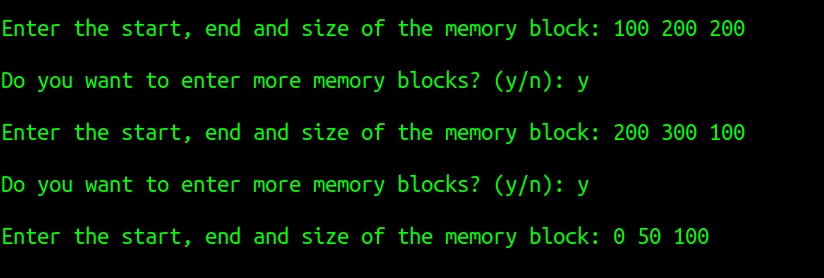
printf("\nThe free/busy list of memory blocks is: \n");

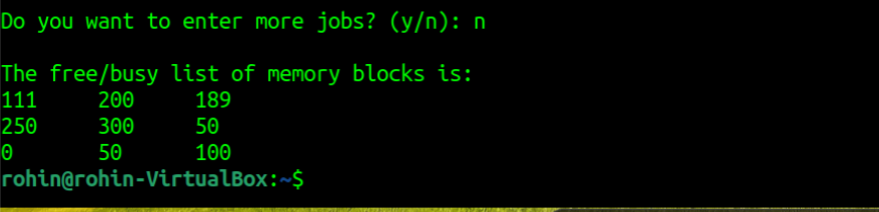
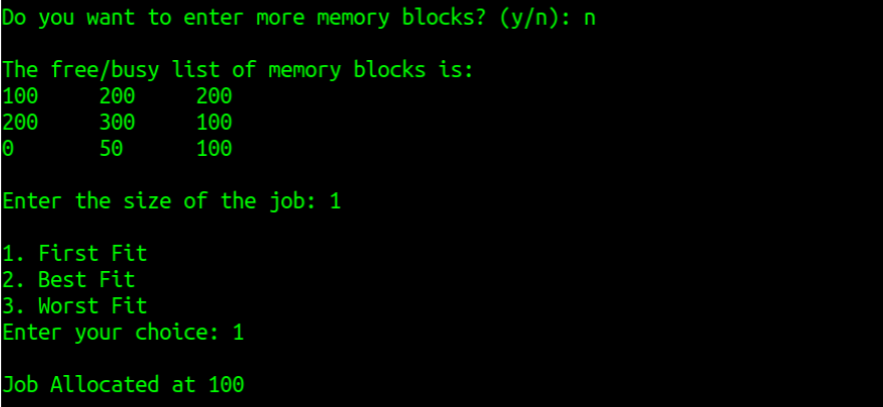
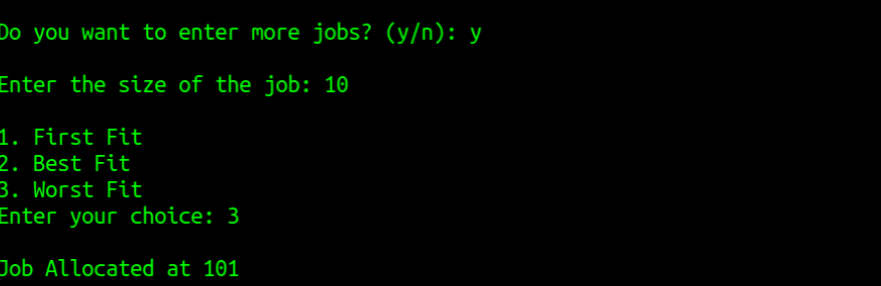
display();

return 0;

}

**b) Test output**

****The output of the C program is given below :

****

**References**

1. <https://www.geeksforgeek.org/first-fit-allocation-in-operating-systems/>
2. <https://www.geeksforgeeks.org/best-fit-allocation-in-operating-system/>
3. <https://prepinsta.com/operating-systems/first-fit-best-fit-worst-fit-in-os-example/>
4. <https://prepinsta.com/operating-systems/page-replacement-algorithms/worst-fit/worst-fit-in-c/>
5. <https://iq.opengenus.org/first-best-and-worst-fit-strategies/>
6. <https://www.thecrazyprogrammer.com/2017/01/first-fit-algorithm-in-c.html>