// Contiguous Memory Allocation Techniques – Best Fit, First Fit and Worst Fit

**Procedure:**

1. Initialization**:**

* + The program initializes an array memory representing the memory block, where each element can be either -1 (indicating an empty space) or a positive integer (indicating that a memory block is allocated).
  + It also maintains an array partitions to store the starting index of each allocated memory block and num\_partitions to keep track of the number of partitions.

1. Memory Allocation Functions:
   * firstFit(int size): Attempts to allocate a memory block of the specified size using the First Fit technique. It searches for the first available memory region that can accommodate the process. If a suitable region is found, it marks the allocated memory region in the memory array and records the starting index in the partitions array.
   * bestFit(int size): Similar to First Fit but searches for the smallest available region that can accommodate the process, minimizing memory wastage.
   * worstFit(int size): Similar to First Fit but searches for the largest available region that can accommodate the process, potentially causing more memory fragmentation.
2. Deallocate Function:
   * deallocateMemory(int index): Takes an index and deallocates the memory block starting at that index. It marks the memory block as empty (-1) and updates the partitions array to indicate that the partition is available.
3. Display Memory Function:
   * displayMemory(): Displays the current state of memory, representing allocated memory blocks with 'X' and empty memory with '.'.
4. Menu-Driven User Interface:
   * The program offers a menu-driven interface for the user to choose from the following options:
     + Allocate memory using First Fit.
     + Allocate memory using Best Fit.
     + Allocate memory using Worst Fit.
     + Deallocate memory by specifying the starting index.
     + Display the current state of memory.
     + Exit the program.
5. Main Loop:
   * The main loop continues to execute until the user chooses to exit.
   * Depending on the user's choice, the program will call one of the allocation methods, deallocate memory, or display the memory state.

hp@hp-HP-Laptop-15s-fr2xxx:~/OS\_C\_Programs$ gedit conti\_mem.c

// Develop a C Program to simulate the following contiguous memory allocation techniques

// (a) Worst fit (b) Best fit (c) First fit

//Author: Dr.Guruvammal S, Date:17/10/2023

#include <stdio.h>

#define MAX\_MEMORY\_SIZE 100

#define MAX\_PARTITIONS 10

int memory[MAX\_MEMORY\_SIZE]; // Memory block

int partitions[MAX\_PARTITIONS]; // Partition information

int num\_partitions = 0; // Number of partitions

void initializeMemory() {

for (int i = 0; i < MAX\_MEMORY\_SIZE; i++) {

memory[i] = -1; // Initialize memory with -1 (empty)

}

}

int firstFit(int size) {

for (int i = 0; i < MAX\_MEMORY\_SIZE; i++) {

if (memory[i] == -1) { // Found an empty space

int j;

for (j = i; j < MAX\_MEMORY\_SIZE && memory[j] == -1; j++) {

// Check if there is enough consecutive space for the allocation

if (j - i + 1 >= size) {

partitions[num\_partitions] = i;

num\_partitions++;

return i;

}

}

i = j; // Move the pointer to the last checked position

}

}

return -1; // Not enough space

}

int bestFit(int size) {

int bestFitIndex = -1;

int bestFitSize = MAX\_MEMORY\_SIZE;

for (int i = 0; i < MAX\_MEMORY\_SIZE; i++) {

if (memory[i] == -1) { // Found an empty space

int j;

for (j = i; j < MAX\_MEMORY\_SIZE && memory[j] == -1; j++) {

if (j - i + 1 >= size && j - i + 1 < bestFitSize) {

bestFitSize = j - i + 1;

bestFitIndex = i;

}

}

i = j;

}

}

if (bestFitIndex != -1) {

partitions[num\_partitions] = bestFitIndex;

num\_partitions++;

return bestFitIndex;

} else {

return -1; // Not enough space

}

}

int worstFit(int size) {

int worstFitIndex = -1;

int worstFitSize = 0;

for (int i = 0; i < MAX\_MEMORY\_SIZE; i++) {

if (memory[i] == -1) { // Found an empty space

int j;

for (j = i; j < MAX\_MEMORY\_SIZE && memory[j] == -1; j++) {

if (j - i + 1 >= size && j - i + 1 > worstFitSize) {

worstFitSize = j - i + 1;

worstFitIndex = i;

}

}

i = j;

}

}

if (worstFitIndex != -1) {

partitions[num\_partitions] = worstFitIndex;

num\_partitions++;

return worstFitIndex;

} else {

return -1; // Not enough space

}

}

void deallocateMemory(int index) {

if (index >= 0 && index < MAX\_MEMORY\_SIZE) {

for (int i = 0; i < num\_partitions; i++) {

if (partitions[i] == index) {

for (int j = partitions[i]; j < partitions[i] + memory[index]; j++) {

memory[j] = -1;

}

partitions[i] = -1; // Mark the partition as empty

}

}

}

}

void displayMemory() {

printf("Memory Status:\n");

for (int i = 0; i < MAX\_MEMORY\_SIZE; i++) {

if (memory[i] == -1) {

printf(".");

} else {

printf("X");

}

}

printf("\n");

}

int main() {

initializeMemory();

int choice;

int size;

while (1) {

printf("\nMemory Allocation Techniques\n");

printf("1. First Fit\n");

printf("2. Best Fit\n");

printf("3. Worst Fit\n");

printf("4. Deallocate\n");

printf("5. Display Memory\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter process size: ");

scanf("%d", &size);

if (firstFit(size) == -1) {

printf("Allocation failed: Not enough memory\n");

} else {

printf("Allocation successful\n");

}

break;

case 2:

printf("Enter process size: ");

scanf("%d", &size);

if (bestFit(size) == -1) {

printf("Allocation failed: Not enough memory\n");

} else {

printf("Allocation successful\n");

}

break;

case 3:

printf("Enter process size: ");

scanf("%d", &size);

if (worstFit(size) == -1) {

printf("Allocation failed: Not enough memory\n");

} else {

printf("Allocation successful\n");

}

break;

case 4:

printf("Enter the index to deallocate: ");

scanf("%d", &size);

deallocateMemory(size);

break;

case 5:

displayMemory();

break;

case 6:

return 0;

default:

printf("Invalid choice\n");

}

}

return 0;

}

**Execution:**

**hp@hp-HP-Laptop-15s-fr2xxx:~/OS\_C\_Programs$ cc conti\_mem.c**

**hp@hp-HP-Laptop-15s-fr2xxx:~/OS\_C\_Programs$ ./a.out**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 1**

**Enter process size: 100**

**Allocation successful**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 2**

**Enter process size: 400**

**Allocation failed: Not enough memory**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 5**

**Memory Status:**

**....................................................................................................**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 4**

**Enter the index to deallocate: 200**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 5**

**Memory Status:**

**....................................................................................................**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 2**

**Enter process size: 50**

**Allocation successful**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 3**

**Enter process size: 50**

**Allocation successful**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice: 2**

**Enter process size: 100**

**Allocation failed: Not enough memory**

**Memory Allocation Techniques**

**1. First Fit**

**2. Best Fit**

**3. Worst Fit**

**4. Deallocate**

**5. Display Memory**

**6. Exit**

**Enter your choice:**