

Memory\_Management

BCSE303P Operating Systems



***Name: Chillara V L N S Pavana Vamsi***

***Register Number: 21BCE5095***

1. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition.

Code:

#include <stdio.h>

void firstFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tFirst fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                space[j] -= process\_size[i];

                allocation[i] = j;

                break;

            }

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

void bestFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tBest fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (space[x] > space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            allocation[i] = x;

            space[x] -= process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

void worstFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tWorst fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (space[x] < space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            allocation[i] = x;

            space[x] -= process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

int main()

{

    int n;

    printf("Enter the number of sections memory of 10GB divided:");

    scanf("%d", &n);

    int space1[n];

    int space2[n];

    int space3[n];

    for (int i = 0; i < n; i++)

    {

        printf("\nEnter the divison number %d (in GB):", i + 1);

        scanf("%d", &space1[i]);

        space2[i] = space1[i];

        space3[i] = space1[i];

    }

    int m;

    printf("Enter the number process:");

    scanf("%d", &m);

    int process\_size[m];

    for (int i = 0; i < m; i++)

    {

        printf("\nEnter the allocation required for process number %d (in GB):", i + 1);

        scanf("%d", &process\_size[i]);

    }

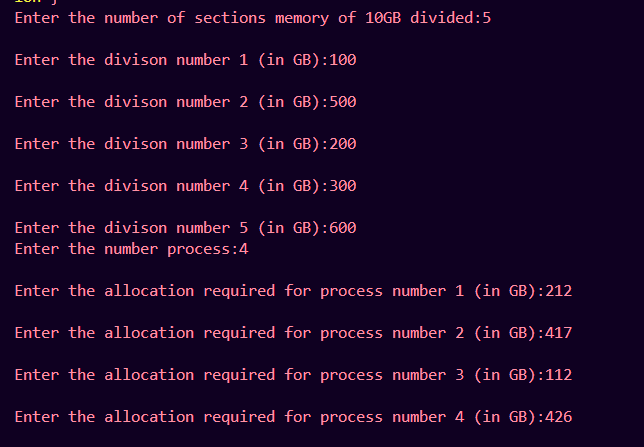
    firstFit(process\_size, m, space1, n);

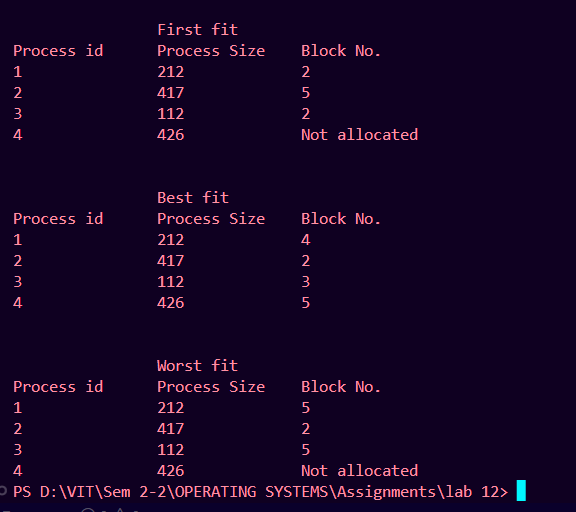
    bestFit(process\_size, m, space2, n);

    worstFit(process\_size, m, space3, n);

    return 0;

}

Output:

1. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition using 3 threads and and threads may return the name of the processes which are unallocated if any.

Code:

#include <stdio.h>

#include <pthread.h>

#define n 5

#define m 4

struct processData

{

    int space[n];

    int process\_size[m];

    int allocation[m];

    int unallocated[m];

};

void \*firstFit(void \*p)

{

    struct processData \*pd = (struct processData \*)p;

    printf("\n\n\t\tFirst fit");

    for (int i = 0; i < m; i++)

        pd->allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        for (int j = 0; j < n; j++)

        {

            if (pd->space[j] >= pd->process\_size[i])

            {

                pd->space[j] -= pd->process\_size[i];

                pd->allocation[i] = j;

                break;

            }

        }

        printf("%d\t\t%d\t\t", i + 1, pd->process\_size[i]);

        if (pd->allocation[i] == -1)

        {

            printf("Not allocated\n");

        }

        else

            printf("%i\n", pd->allocation[i] + 1);

    }

}

void \*bestFit(void \*p)

{

    struct processData \*pd = (struct processData \*)p;

    printf("\n\n\t\tBest fit");

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (pd->space[j] >= pd->process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (pd->space[x] > pd->space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            pd->allocation[i] = x;

            pd->space[x] -= pd->process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, pd->process\_size[i]);

        if (pd->allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", pd->allocation[i] + 1);

    }

}

void \*worstFit(void \*p)

{

    struct processData \*pd = (struct processData \*)p;

    printf("\n\n\t\tWorst fit");

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (pd->space[j] >= pd->process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (pd->space[x] < pd->space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            pd->allocation[i] = x;

            pd->space[x] -= pd->process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, pd->process\_size[i]);

        if (pd->allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", pd->allocation[i] + 1);

    }

}

int main()

{

    struct processData pd1, pd2, pd3;

    for (int i = 0; i < n; i++)

    {

        printf("\nEnter the divison number %d (in GB):", i + 1);

        scanf("%d", &pd1.space[i]);

    }

    for (int i = 0; i < m; i++)

    {

        printf("\nEnter the allocation required for process number %d (in GB):", i + 1);

        scanf("%d", &pd1.process\_size[i]);

    }

    for (int i = 0; i < m; i++)

        pd1.allocation[i] = -1;

    pd2 = pd1;

    pd3 = pd1;

    pthread\_t first\_t, best\_t, worst\_t;

    pthread\_create(&first\_t, NULL, (void \*)firstFit, (void \*)&pd1);

    pthread\_join(first\_t, NULL);

    int flag = m;

    for (int i = 0; i < m; i++)

    {

        if (pd1.allocation[i] == -1)

        {

            printf("\nProcess Id %d is unallocated\n", i);

            flag--;

        }

    }

    if (m == flag)

        printf("\nAll process are successfully allocated\n");

    pthread\_create(&best\_t, NULL, (void \*)bestFit, (void \*)&pd2);

    pthread\_join(best\_t, NULL);

    flag = m;

    for (int i = 0; i < m; i++)

    {

        if (pd2.allocation[i] == -1)

        {

            printf("\nProcess Id %d is unallocated\n", i);

            flag--;

        }

    }

    if (m == flag)

        printf("\nAll process are successfully allocated\n");

    pthread\_create(&worst\_t, NULL, (void \*)worstFit, (void \*)&pd3);

    pthread\_join(worst\_t, NULL);

    flag = m;

    for (int i = 0; i < m; i++)

    {

        if (pd3.allocation[i] == -1)

        {

            printf("\nProcess Id %d is unallocated\n", i);

            flag--;

        }

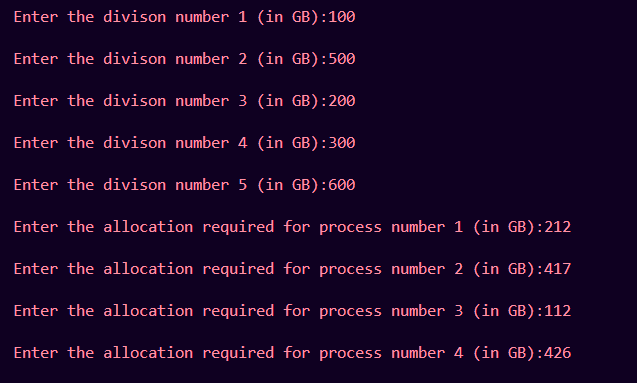
    }

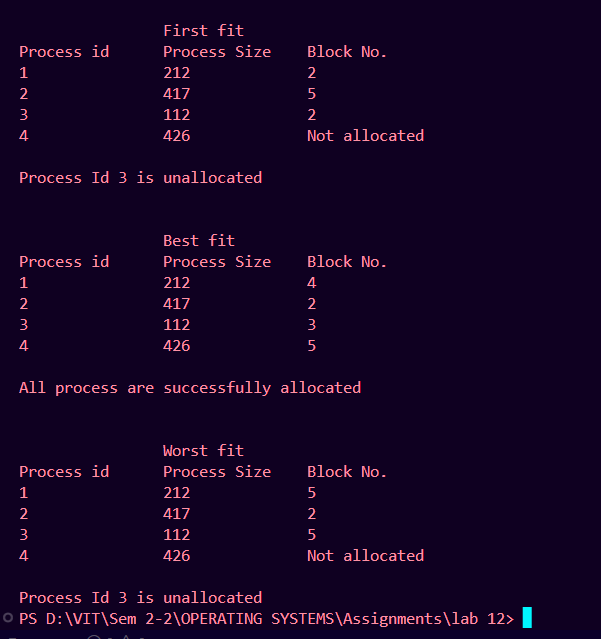
    if (m == flag)

        printf("\nAll process are successfully allocated\n");

    return 0;

}

Output:



1. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition with assumption of block sizes and processes memory request sizes are in process.txt file. So your program should read the data from the file and perform the memory allocations.

Code:

#include <stdio.h>

void firstFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tFirst fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                space[j] -= process\_size[i];

                allocation[i] = j;

                break;

            }

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

void bestFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tBest fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (space[x] > space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            allocation[i] = x;

            space[x] -= process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

void worstFit(int process\_size[], int m, int space[], int n)

{

    printf("\n\n\t\tWorst fit");

    int allocation[m];

    for (int i = 0; i < m; i++)

        allocation[i] = -1;

    printf("\nProcess id\tProcess Size\tBlock No.\n");

    for (int i = 0; i < m; i++)

    {

        int x = -1;

        for (int j = 0; j < n; j++)

        {

            if (space[j] >= process\_size[i])

            {

                if (x == -1)

                    x = j;

                else if (space[x] < space[j])

                    x = j;

            }

        }

        if (x != -1)

        {

            allocation[i] = x;

            space[x] -= process\_size[i];

        }

        printf("%d\t\t%d\t\t", i + 1, process\_size[i]);

        if (allocation[i] == -1)

            printf("Not allocated\n");

        else

            printf("%i\n", allocation[i] + 1);

    }

}

int main()

{

    int n, m;

    FILE \*fp = fopen("process.txt", "r");

    fscanf(fp, "%d", &n);

    fscanf(fp, "%d", &m);

    int space1[n];

    int space2[n];

    int space3[n];

    for (int i = 0; i < n; i++)

    {

        fscanf(fp, "%d", &space1[i]);

        space2[i] = space1[i];

        space3[i] = space1[i];

    }

    int process\_size[m];

    for (int i = 0; i < m; i++)

    {

        fscanf(fp, "%d", &process\_size[i]);

    }

    firstFit(process\_size, m, space1, n);

    bestFit(process\_size, m, space2, n);

    worstFit(process\_size, m, space3, n);

    fclose(fp);

    return 0;

}

