|  |  |  |  |
| --- | --- | --- | --- |
| **Fr. Conceicao Rodrigues College of Engineering**  **Department of Computer Engineering** | | | |
| **Student’s Roll No** |  | **Students Name** |  |
| **Date of Performance** |  | **SE Computer – Div** | **A / B** |

**Aim:** Study Memory Management

**Lab Outcome:**

**CSL403.4:** Implement various memory management techniques and evaluate their performances.

**Problem Statements:**

Implement Dynamic Partitioning Placement Algorithms

(a)Best Fit (b) First-Fit (c )Worst-Fit

1. Given the number of holes and their sizes, number of blocks to be placed in memory and their sizes, find which algorithm would be resulting in effective utilization of memory.

2. Give the allotment of blocks to holes in each algorithm

**References:**

[**https://www.youtube.com/watch?v=oYfzZU2Z6Tk&t=626s**](https://www.youtube.com/watch?v=oYfzZU2Z6Tk&t=626s)

|  |  |  |  |
| --- | --- | --- | --- |
| **On time Submission(2)** | **Knowledge of Topic(4)** | **Implementation and Demonstraion(4)** | **Total (10)** |
|  |  |  |  |
| **Signature of Faculty** |  | **Date of Submission** |  |

1.First Fit

blocks = [100, 500, 200, 300, 600]

process = [212, 417, 112, 426]

m = len(blocks)

n = len(process)

allocation = [-1]\*n

for i in range(n):

    for j in range(m):

        if blocks[j]>=process[i]:

            choosen = blocks[i]

            allocation[i] = choosen

            blocks[j] -= process[i]

            break

    print('memory blocks after ',i+1,' process -->',blocks)

print()

print("Process No.    Process Size            Block")

for i in range(n):

    print('    ',i + 1, "          ", process[i],end = "\t\t\t")

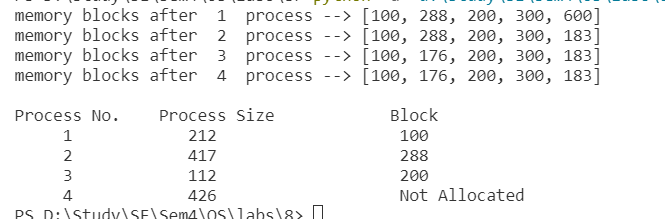
    if allocation[i] != -1:

        print(allocation[i])

    else:

        print("Not Allocated")

Output:



2.best fit

blocks = [100, 500, 200, 300, 600]

process = [212, 417, 112, 426, 80, 2, 900]

m = len(blocks)

n = len(process)

allocation = [-1]\*n

for i in range(n):

    index = -1

    best = max(blocks)

    for j in range(m):

        if blocks[j]>=process[i]:

            if blocks[j]<=best:

                best = blocks[j]

                index = j

    if index is not -1:

        allocation[i]=best

        blocks[index] -= process[i]

    print('memory blocks after ',i+1,' process -->',blocks)

print("Process No.    Process Size            Block")

for i in range(n):

    print('    ',i + 1, "          ", process[i],end = "\t\t\t")

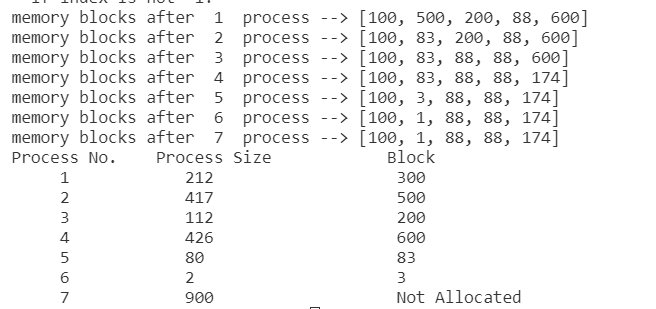
    if allocation[i] != -1:

        print(allocation[i])

    else:

        print("Not Allocated")

Output:



3.Worst fit

blocks = [100, 500, 200, 300, 600]

process = [212, 417, 112, 426, 80, 2, 900]

m = len(blocks)

n = len(process)

allocation = [-1]\*n

print('memory allocation')

for i in range(n):

    index = -1

    worst = min(blocks)

    for j in range(m):

        if blocks[j]>=process[i]:

            if blocks[j]>=worst:

                worst = blocks[j]

                index = j

    if index is not -1:

        allocation[i]=worst

        blocks[index] -= process[i]

    print('memory blocks after ',i+1,' process -->',blocks)

print()

print("Process No.    Process Size            Block")

for i in range(n):

    print('    ',i + 1, "          ", process[i],end = "\t\t\t")

    if allocation[i] != -1:

        print(allocation[i])

    else:

        print("Not Allocated")

Output:

