**Lab no 9**

**Memory Management**

***Objectives:***

* What is memory management?
* What is pagination?
* Implementation of First, Next, Best and Worst fit algorithm.

**Memory Allocation:**

To gain proper memory utilization, memory allocation must be allocated efficient manner. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process. Thus, the degree of multiprogramming is obtained by the number of partitions.

Memory allocation is a process by which computer programs are assigned memory or space. It is of three types:

1. **First Fit Allocation**

The first hole that is big enough is allocated to the program.

1. **Best Fit Allocation**

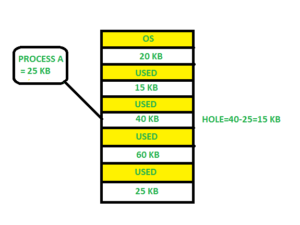
The smallest hole that is big enough is allocated to the program.

1. **Worst Fit Allocation**

The largest hole that is big enough is allocated to the program.

**First fit:-**

In the first fit, the first available free hole fulfills the requirement of the process allocated.



**Methods discussed**

We will look at two different methods –

* **Method 1 –** Blocks allowed to keep just one single process
* **Method 2 –** Blocks allowed to keep multiple processes, if partitioned fragmentation big enough for new processes

## Method 1

Run

def FirstFit(block\_Size, blocks, process\_Size, proccesses):

# code to store the block id of the block that needs to be allocated to a process

allocate = [-1] \* proccesses

occupied = [False] \* blocks

# Any process is assigned with the memory at the initial stage

# find a suitable block for each process

# the blocks are allocated as per their size

for i in range(proccesses):

for j in range(blocks):

if not occupied[j] and (block\_Size[j] >= process\_Size[i]):

# assign the block j to p[i] process

allocate[i] = j

occupied[j] = True

break

print("Process No. Process Size Block No.")

for i in range(proccesses):

print(str(i + 1) + "\t\t\t" + str(process\_Size[i]) + "\t\t\t", end=" ")

if allocate[i] != -1:

print(allocate[i] + 1)

else:

print("Not Allocated")

# Driver code

block\_Size = [100, 50, 30, 120, 35]

process\_Size = [20, 60, 70, 40]

m = len(block\_Size)

n = len(process\_Size)

FirstFit(block\_Size, m, process\_Size, n)

## Method 2

This allows multiple processes to share the same block space if the size is enough to keep another process.

Run

def FirstFit(block\_Size, m, process\_Size, n):

# code to store the block id of the block that needs to be allocated to a process

allocate = [-1] \* n

# Any process is assigned with the memory at the initial stage

# find a suitable block for each process

# the blocks are allocated as per their size

for i in range(n):

for j in range(m):

if block\_Size[j] >= process\_Size[i]:

# assign the block j to p[i] process

allocate[i] = j

# available block memory is reduced

block\_Size[j] -= process\_Size[i]

break

print("Process No. Process Size Block No.")

for i in range(n):

print(str(i + 1) + "\t\t\t" + str(process\_Size[i]) + "\t\t\t", end=" ")

if allocate[i] != -1:

print(allocate[i] + 1)

else:

print("Not Allocated")

# Driver code

block\_Size = [100, 50, 30, 120, 35]

process\_Size = [20, 60, 70, 40]

m = len(block\_Size)

n = len(process\_Size)

FirstFit(block\_Size, m, process\_Size, n)

**Task:**

* Create code for Next-fit, Best-fit and Worst-fit.

Code for next fit:

# Python3 program for next fit

# memory management algorithm

# Function to allocate memory to

# blocks as per Next fit algorithm

def NextFit(blockSize, m, processSize, n):

    # Stores block id of the block

    # allocated to a process

    # Initially no block is assigned

    # to any process

    allocation = [-1] \* n

    j = 0

    t = m-1

    # pick each process and find suitable blocks

    # according to its size ad assign to it

    for i in range(n):

        # Do not start from beginning

        while j < m:

            if blockSize[j] >= processSize[i]:

                # allocate block j to p[i] process

                allocation[i] = j

                # Reduce available memory in this block.

                blockSize[j] -= processSize[i]

                # sets a new end point

                t = (j - 1) % m

                break

            if t == j:

                # sets a new end point

                t = (j - 1) % m

                # breaks the loop after going through all memory block

                break

            # mod m will help in traversing the

            # blocks from starting block after

            # we reach the end.

            j = (j + 1) % m

    print("Process No. Process Size Block no.")

    for i in range(n):

        print("\t", i + 1, "\t\t\t", processSize[i],end = "\t\t\t")

        if allocation[i] != -1:

            print(allocation[i] + 1)

        else:

            print("Not Allocated")

# Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

    blockSize = [5, 10, 20]

    processSize = [10, 20, 5]

    m = len(blockSize)

    n = len(processSize)

    NextFit(blockSize, m, processSize, n)

Ouput:

Graphical user interface, text

Description automatically generated

Code for best fit:

# Python3 implementation of Best - Fit algorithm

# Function to allocate memory to blocks

# as per Best fit algorithm

def bestFit(blockSize, m, processSize, n):

    # Stores block id of the block

    # allocated to a process

    allocation = [-1] \* n

    # pick each process and find suitable

    # blocks according to its size ad

    # assign to it

    for i in range(n):

        # Find the best fit block for

        # current process

        bestIdx = -1

        for j in range(m):

            if blockSize[j] >= processSize[i]:

                if bestIdx == -1:

                    bestIdx = j

                elif blockSize[bestIdx] > blockSize[j]:

                    bestIdx = j

        # If we could find a block for

        # current process

        if bestIdx != -1:

            # allocate block j to p[i] process

            allocation[i] = bestIdx

            # Reduce available memory in this block.

            blockSize[bestIdx] -= processSize[i]

    print("Process No. Process Size  Block no.")

    for i in range(n):

        print(i + 1, "       ", processSize[i],

                                end = "      ")

        if allocation[i] != -1:

            print(allocation[i] + 1)

        else:

            print("Not Allocated")

# Driver code

if \_\_name\_\_ == '\_\_main\_\_':

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

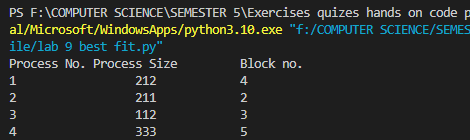
    processSize = [212, 211, 112, 333]

    m = len(blockSize)

    n = len(processSize)

    bestFit(blockSize, m, processSize, n)

Output:



Code for wrost fit:

# Python3 implementation of worst - Fit algorithm

# Function to allocate memory to blocks as

# per worst fit algorithm

def worstFit(blockSize, m, processSize, n):

    # Stores block id of the block

    # allocated to a process

    # Initially no block is assigned

    # to any process

    allocation = [-1] \* n

    # pick each process and find suitable blocks

    # according to its size ad assign to it

    for i in range(n):

        # Find the best fit block for

        # current process

        wstIdx = -1

        for j in range(m):

            if blockSize[j] >= processSize[i]:

                if wstIdx == -1:

                    wstIdx = j

                elif blockSize[wstIdx] < blockSize[j]:

                    wstIdx = j

        # If we could find a block for

        # current process

        if wstIdx != -1:

            # allocate block j to p[i] process

            allocation[i] = wstIdx

            # Reduce available memory in this block.

            blockSize[wstIdx] -= processSize[i]

    print("Process No. Process Size Block no.")

    for i in range(n):

        print(i + 1, "       ",

            processSize[i], end = "  ")

        if allocation[i] != -1:

            print(allocation[i] + 1)

        else:

            print("Not Allocated")

# Driver code

if \_\_name\_\_ == '\_\_main\_\_':

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

    processSize = [212, 417, 112, 426]

    m = len(blockSize)

    n = len(processSize)

    worstFit(blockSize, m, processSize, n)

Output:

Text

Description automatically generated