**COA Lab**

**Experiment No. 5**

**Name:** Ebrahim Hirani

**UID:** 2018130015

**Class:** SE Comps (Batch B)

**Aim:** To implement Memory Allocation Algorithms

**PARTITION ALLOCATION METHODS IN MEMORY MANAGEMENT**

* In the operating system, the following are four common memory management techniques.
  + **Single contiguous allocation:** Simplest allocation method used by MS-DOS. All memory (except some reserved for OS) is available to a process.
  + **Partitioned allocation:** Memory is divided in different blocks or partitions.Each process is allocated accroding to the requirment.
  + **Paged memory management:** Memory is divided into fixed sized units called page frames, used in a virtual memory environment.
  + **Segmented memory management:** Memory is divided in different segments (a segment is a logical grouping of the process’ data or code).In this management, allocated memory doesn’t have to be contiguous.
* Most of the operating systems (for example Windows and Linux) use Segmentation with Paging. A process is divided into segments and individual segments have pages.
* In Partition Allocation, when there is more than one partition freely available to accommodate a process’s request, a partition must be selected. To choose a particular partition, a partition allocation method is needed. A partition allocation method is considered better if it avoids internal fragmentation.
* Below are the various partition allocation schemes:
  + **First Fit:** In the first fit, the partition is allocated which is first sufficient block from the top of Main Memory.
  + **Best Fit** Allocate the process to the partition which is the first smallest sufficient partition among the free available partition.
  + **Worst Fit** Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory.
* **Is Best-Fit really best?**
  + Although best fit minimizes the wastage space, it consumes a lot of processor time for searching the block which is close to the required size.
  + Also, Best-fit may perform poorer than other algorithms in some cases.

**IMPLEMENTATION OF ALGORITHMS**

**Code:**

*def* firstFit(*memory*, *processes*):

    for process in processes:

        for partition in memory:

            if partition[0] >= process[0] and partition[1]:

                partition[1] = False

                partition.append(process[0])

                process[1] = True

                break

    print("Memory Occupancy:\n")

    for partition in memory:

        print("Capacity: " + *str*(partition[0]) + (("\tOccupied: " + *str*(partition[2])) if not partition [1] else "\tUnoccupied"))

    print("\nWaitlisted processes:")

    for process in processes:

        if not process[1]:

            print(process[0])

*def* bestFit(*memory*, *processes*):

    for process in processes:

        best = None

        bestDiff = *float*('inf')

        for i,partition in enumerate(memory):

            diff = partition[0] - process[0]

            if diff < bestDiff and partition[1] and diff >= 0:

                best = i

                bestDiff = diff

        if best:

            memory[best][1] = False

            memory[best].append(process[0])

            process[1] = True

    print("Memory Occupancy:\n")

    for partition in memory:

        print("Capacity: " + *str*(partition[0]) + (("\tOccupied: " + *str*(partition[2])) if not partition [1] else "\tUnoccupied"))

    print("\nWaitlisted processes:")

    for process in processes:

        if not process[1]:

            print(process[0])

*def* worstFit(*memory*, *processes*):

    for process in processes:

        best = None

        bestDiff = -*float*('inf')

        for i,partition in enumerate(memory):

            diff = partition[0] - process[0]

            if diff > bestDiff and partition[1] and diff >= 0:

                best = i

                bestDiff = diff

        if best:

            memory[best][1] = False

            memory[best].append(process[0])

            process[1] = True

    print("Memory Occupancy:\n")

    for partition in memory:

        print("Capacity: " + *str*(partition[0]) + (("\tOccupied: " + *str*(partition[2])) if not partition [1] else "\tUnoccupied"))

    print("\nWaitlisted processes:")

    for process in processes:

        if not process[1]:

            print(process[0])

if \_\_name\_\_ == "\_\_main\_\_":

    import copy

    memory = [[100, True],[300, True],[150, True],[650, True],[450, True],]

    processes = [[212,False],[315,False],[127,False],[470,False],]

    print("\n\nMemory:\n")

    print([m[0] for m in memory])

    print("\n\nProcesses:\n")

    print([p[0] for p in processes])

    print("\n\nFirst Fit: \n")

    firstFit(copy.deepcopy(memory),copy.deepcopy(processes))

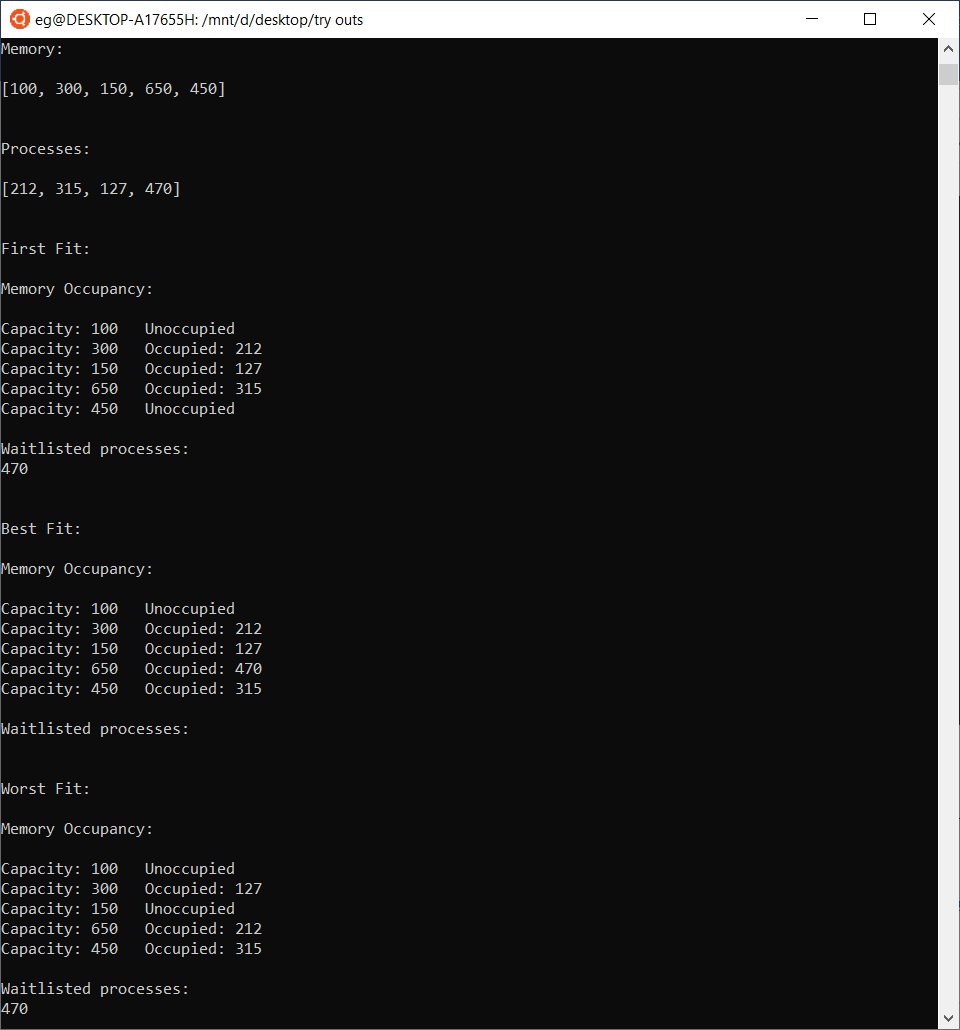
    print("\n\nBest Fit: \n")

    bestFit(copy.deepcopy(memory),copy.deepcopy(processes))

    print("\n\nWorst Fit: \n")

    worstFit(copy.deepcopy(memory),copy.deepcopy(processes))

**Output:**



**CONCLUSION**

* Learnt the Memory Allocation Algorithms.
* Implemented the algorithms using Python and displayed Memory occupancy along with the waitlisted processes.