 ***DEPARTMENT OF COMPUTER ENGINEERING***

Experiment No.

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | SNA |
| Assisting Teachers | Ms. Rasika Ransing |
| Laboratory | M310B – Computer Engineering Laboratory |

|  |  |  |
| --- | --- | --- |
| Student Name | B. Niranjana | |
| Roll Number | 18104A0014 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment Number |  | |
| Experiment Title | Write a program to implement dynamic partitioning placement algorithms i.e Best Fit, First-Fit, Worst-Fit etc | |
| Resources / Apparatus Required | Hardware: PC | Software:  Compiler |
| Objectives  (Skill Set / Knowledge Tested / Imparted) | Memory Management technique | |
| Theory: | Memory management is a form of resource management applied to computer memory. The essential requirement of memory management is to provide ways to dynamically allocate portions of memory to programs at their request, and free it for reuse when no longer needed. This is critical to any advanced computer system where more than a single process might be underway at any time.  Several methods have been devised that increase the effectiveness of memory management. Virtual memory systems separate the memory addresses used by a process from actual physical addresses, allowing separation of processes and increasing the size of the virtual address space beyond the available amount of RAM using paging or swapping to secondary storage. The quality of the virtual memory manager can have an extensive effect on overall system performance. | |
| Code | //best  def bestFit(blockSize, m, processSize, n):  allocation = [-1] \* n  for i in range(n):  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 bestIdx != -1:  allocation[i] = bestIdx  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")    if \_\_name\_\_ == '\_\_main\_\_':  blockSize = [100, 500, 200, 300, 600]  processSize = [212, 417, 112, 426]  m = len(blockSize)  n = len(processSize)    bestFit(blockSize, m, processSize, n)  //worst  def worstFit(blockSize, m, processSize, n):  allocation = [-1] \* n    for i in range(n):  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 wstIdx != -1:  allocation[i] = wstIdx  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")  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)  //first fit  def worstFit(blockSize, m, processSize, n):  allocation = [-1] \* n    for i in range(n):  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 wstIdx != -1:  allocation[i] = wstIdx  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")  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: |  | |