Operating Systems and Systems Programming (15B11CI412)

Tutorial - 7 Memory Management

1. **Name two differences between logical and physical addresses**.

   Answer: A logical address does not refer to an actual existing address; rather, it refers to an abstract address in an abstract address space. Contrast this with a physical address that refers to an actual physical address in memory. A logical address is generated by the CPU and is translated into a physical address by the memory management unit(MMU). Therefore, physical addresses are generated by the MMU.

2. **Consider a system in which a program can be separated into two parts: code and data. The CPU knows whether it wants an instruction (instruction fetch) or data (data fetch or store). Therefore, two base-limit register pairs are provided: one for instructions and one for data. The instruction base-limit register pair is automatically read-only, so programs can be shared among different users. Discuss the advantages and disadvantages of this scheme.**

   Answer: The major advantage of this scheme is that it is an effective mechanism for code and data sharing. For example, only one copy of an editor or a compiler needs to be kept in memory, and this code can be shared by all processes needing access to the editor or compiler code. Another advantage is protection of code against erroneous modification. The only disadvantage is that the code and data must be separated, which is usually adhered to in a compiler-generated code.

3. **Why are page sizes always powers of 2?**

   Answer: Paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

4. **What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What effect would updating some byte on the one page have on the other page?**

   Answer: By allowing two entries in a page table to point to the same page frame in memory, users can share code and data. If the code is reentrant, much memory space can be saved through the shared use of large programs such as text editors, compilers, and database systems. “Copying” large amounts of memory could be effected by having different page tables point to the same memory location. However, sharing of non-reentrant code or data means that any user having access to the code can modify it and these modifications would be reflected in the other user’s “copy”.

5. **Describe a mechanism by which one segment could belong to the address space of two different processes.**

   Answer: Since segment tables are a collection of base-limit registers, segments can be shared when entries in the segment table of two different jobs point to the same physical location. The two-segment tables must have identical base pointers and the shared segment number must be the same in the two processes.

6. **Sharing segments among processes without requiring that they have the same segment number is possible in a dynamically linked segmentation system.**

   (a) Define a system that allows static linking and sharing of segments without requiring that the segment numbers be the same.

   (b) Describe a paging scheme that allows pages to be shared without requiring that the page numbers be the same.

   Answer: Both of these problems reduce to a program being able to reference both its own code and its data without knowing the segment or page number associated with the address. MULTICS solved this problem by associating four registers with each process. One register had the address of the current program segment, another had a base address for the stack, another had a base address for the global data, and so on. The idea is that all references have to be indirect through a register that maps to the current segment or page number. By changing these registers, the same code can execute for different processes without the same page or segment numbers.

7. **Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB. These partitions need to be allocated to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. Perform the allocation of processes using-**

   (a) First Fit Algorithm

   (b) Best Fit Algorithm

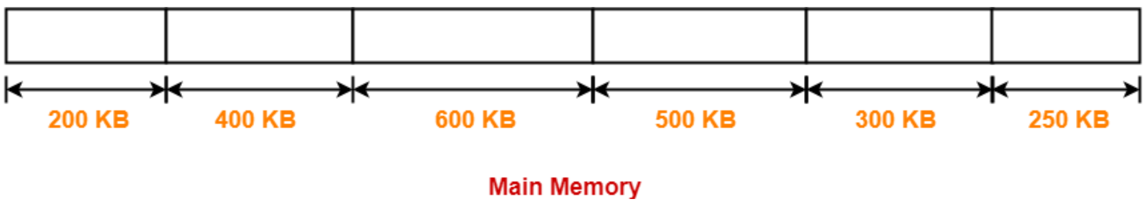
   (c) Worst Fit Algorithm

   Answer:

   According to the question,

   The main memory has been divided into fixed size partitions as-

   Let us say the given processes are-



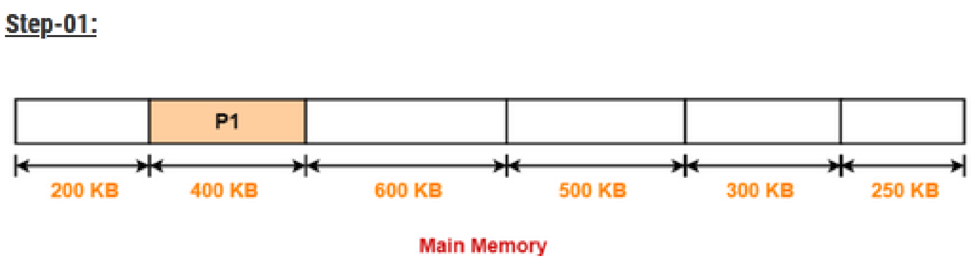
- Process P1 = 357 KB
- Process P2 = 210 KB
- Process P3 = 468 KB
- Process P4 = 491 KB

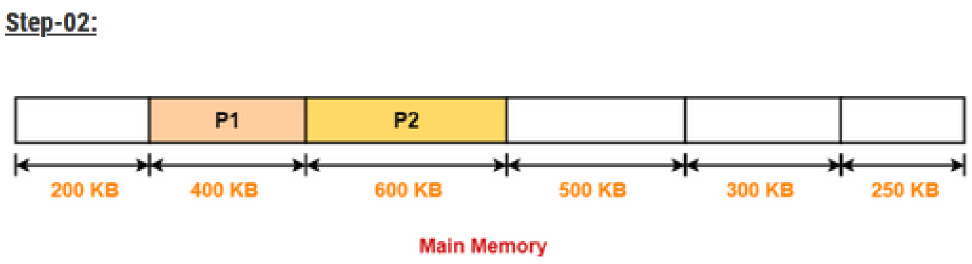
Allocation Using First Fit Algorithm

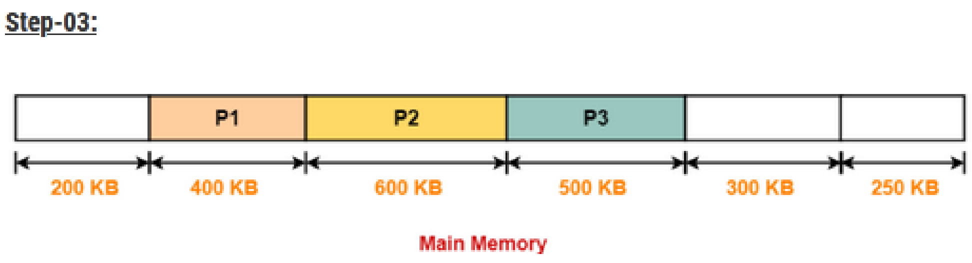
In First Fit Algorithm,

- Algorithm starts scanning the partitions serially.
- When a partition big enough to store the process is found, it allocates that partition to the process.

The allocation of partitions to the given processes is shown below-







Step-04:

- Process P4 can not be allocated the memory.
- This is because no partition of size greater than or equal to the size of process P4 is available.

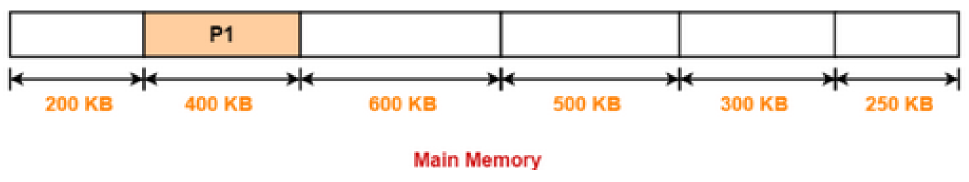
Allocation Using Best Fit Algorithm

In Best Fit Algorithm,

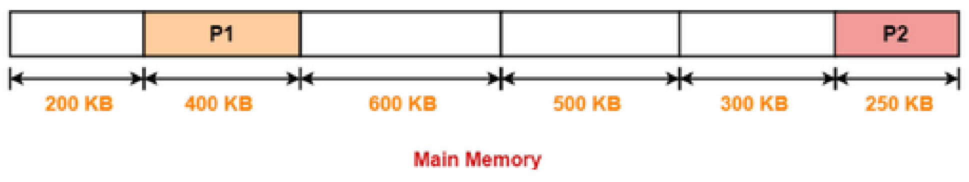
- Algorithm first scans all the partitions.
- It then allocates the partition of smallest size that can store the process.

The allocation of partitions to the given processes is shown below-

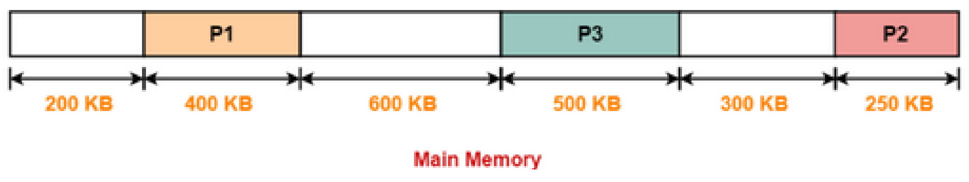
Step-01:



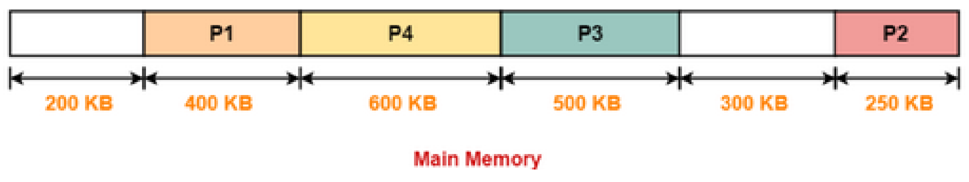
Step-02:



Step-03:



Step-04:

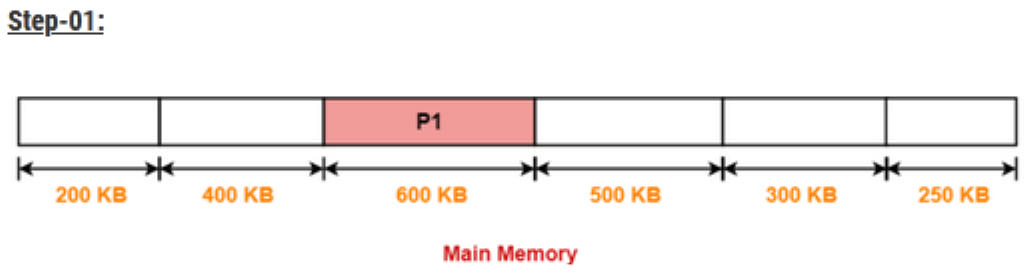


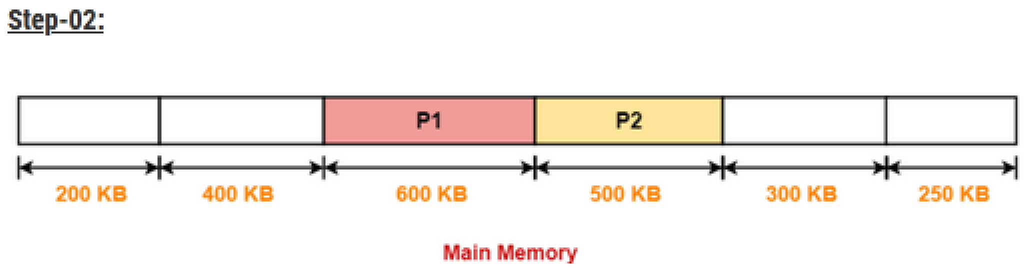
Allocation Using Worst Fit Algorithm

In Worst Fit Algorithm,

- Algorithm first scans all the partitions.
- It then allocates the partition of largest size to the process.

The allocation of partitions to the given processes is shown below-





Step-03:

- Process P3 and Process P4 can not be allocated the memory.
- This is because no partition of size greater than or equal to the size of process P3 and process P4 is available.

8. **A certain computer system has the segmented paging architecture for virtual memory. The memory is byte-addressable. Both virtual and physical address spaces contain 2^{16} bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consist of 2-byte page table entries. What is the minimum page size in bytes so that the page table for a segment requires at most one page to store it?**

   Answer:

   Given-

   - Virtual Address Space = Process size = 2^{16} bytes

- Physical Address Space = Main Memory size = 2^{16} bytes
- Process is divided into 8 equal size segments
- Page table entry size = 2 bytes

Let page size = n bytes.
Now, since page table has to be stored into a single page, so we must have:
Size of page table <= Page size

Size of Each Segment
Size of each segment
= Process size / Number of segments
= 2^{16} bytes / 8
= 2^{16} bytes / 2^3
= 2^{13} bytes
= 8 KB

Number of Pages Of Each Segment
Number of pages each segment is divided
= Size of segment / Page size
= 8 KB / n bytes
= (8K / n) pages

Size of Each Page Table
Size of each page table
= Number of entries in page table × Page table entry size
= Number of pages the segment is divided × 2 bytes
= (8K / n) × 2 bytes
= (16K / n) bytes

Page Size
Substituting values in the above condition, we get-
$(16K/n) bytes <= n bytes$
$(16K/n) <= n$

$n^2 >= 16K$

$n^2 >= 2^{14}$

$n >= 2^7$

Thus, minimum page size possible = 2^7 bytes = 128 bytes.

9. **A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain 2^{16} bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consist of 2-byte page table entries. Assume that each page table entry contains (besides other information) 1 valid bit, 3 bits for page protection and 1 dirty bit. How many bits are available in the page table entry for storing the aging information for the page? Assume that page size is 512 bytes.**

   Answer:

   Given-

   - Virtual Address Space = Process size = 2^{16} bytes
   - Physical Address Space = Main Memory size = 2^{16} bytes
   - Process is divided into 8 equal size segments
   - Page table entry size = 2 bytes = 16 bits
   - Page table entry besides other information contains 1 valid bit, 3 protection bits, 1 dirty bit
   - Page size = 512 bytes

   Number of Frames in Main Memory

   Number of frames in main memory

   = Size of main memory / Page size

   = 2^{16} bytes / 512 bytes

   = 2^{16} bytes / 2^9 bytes

   = 2^7 frames

   Thus, Number of bits required for frame identification in page table entry = 7 bits

Number Of Bits Available For Storing Aging Information

Number of bits available for storing aging information = Number of bits in page table entry – (Number of bits required for frame identification + 1 valid bit + 3 protection bits + 1 dirty bit)

= 16 bits – (7 + 1 + 3 + 1) bits

= 16 bits – 12 bits

= 4 bits