1. Given a dynamic memory partition setup with current holes of (A) 100K, (B) 500K, (C) 200K, (D) 300K, and (E) 600K (in order), show how each of the First-fit, Best-fit and Worst-fit algorithms would place processes of size 212K, 417K, 112K, and 426K (in order)? Indicate for each algorithm the total space left, the smallest available hole, the largest available hole, and the processes (if any) that are unable to be placed. Which algorithm makes the most efficient use of memory?

Answer:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Partition | First-fit  Algorithm | | Best-fit  Algorithm | | Worst-Fit  Algorithm | |
| Process  Placed | Space  Left | Process  Placed | Space  Left | Process  Placed | Space  Left |
| A. 100K |  | 100 |  | 100 |  | 100 |
| B. 500K | P1 212K  P3 112K | 176 | P2 417K | 83 | P2 417K | 83 |
| C. 200K |  | 200 | P3 112K | 88 |  | 200 |
| D. 300K |  | 300 | P1 212K | 88 |  | 300 |
| E. 600K | P2 417K | 183 | P4 426K | 174 | P1 212K  P3 112K | 276 |
| Unable to place | P4 426K |  |  |  | P4 426K |  |
| Total space left |  | 959 |  | 533 |  | 959 |
| Smallest available |  | 100 |  | 83 |  | 83 |
| Largest available |  | 300 |  | 174 |  | 300 |

Most efficient algorithm?: For the process list supplied, the Best-fit algorithm was the only one able to place all processes in memory and therefore had the smallest amount of total space left.

1. Consider a logical address space of 8 pages of 2048 words each, mapped onto a physical memory of 32 frames (assume system is word addressable).

* 1. How many words does the logical address space contain? How many bits are there in the logical address?

**Virtual address space: max # pages \* maximum page size**

**= 8 pages of 2048 words = 16384 words**

**(3 bit page number + 11 bit offset in page = 14 bit address)**

* 1. How many words does the physical address space contain? How many bits are there in the physical address?

**Physical address space: max # frames \* max page size**

**= 32 frames of 2048 words = 65536 words**

**(5 bit frame reference + 11 bit offset in page = 16 bit address)**

1. The following is a core map of a virtual memory system that has a page size of 100.

|  |  |  |
| --- | --- | --- |
| Frame # | Process ID | Page # |
| 0 | 1 | 2 |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 0 |
| 4 | 1 | 3 |

1. To which physical address does logical address 175 of process 1 map? If this logical address does not map to any physical address, write "does not map".

**logical address of 175 of process one is located on page 1, offset 75**

**page 1 of process 1 is currently located in frame 1**

**the base address of frame 1 is 100 + offset 75 = physical address of 175**

b. To which physical address does logical address 38 of process 2 map? If this logical address does not map to any physical adddress, write "does not map".

**logical address 38 of process 2 is located on page 0, offset 38**

**page 0 of process 2 is not currently allocated to a frame**

therefore, the logical address 38 of process 2, does not map to a physical address

c.       Which logical address of which process maps to physical address 27.

**physical address 27 is located at frame 0, offset 27**

**frame 0 currently holds page 2 of process 1**

**so physical address 27 maps to process 1, logical address 227 (2 \* 100 + 27)**