# Comparison of Memory Organization Schemes

The basic concept of this assignment is to compare the main memory organization schemes. The objective is to analyze three groups of memory allocation schemes: contiguous memory allocation, pure segmentation, and pure paging. We will need to identify if External / Internal Fragmentation occurs in each of the three groups. We will also need to identify the potential for sharing segments between user processes of each group.

The first of the schemes is contiguous memory allocation. Memory is divided up into 2 parts: kernel & user. Kernel memory could be placed in low or high memory, based on the location of the interrupt vector (default is low). Each process is placed in a single section of memory. That section is contiguous to the section containing the next process. External fragmentation exists due to the segment length being dependent on the size of the process’s memory. It occurs when there is no segment available to meet the memory requirements of a user process. Something to consider is that using a compaction algorithm to reorganize memory is not efficient. What makes contiguous memory allocation unique is its necessity of searching for blocks of memory of a specific length. The top three algorithms are Best Fit, Worst Fit, and First Fit. Do we search all blocks and find the segment that has a length as close as possible to the memory of a process? Do we take the segment that has the biggest length? Do we take the first block that has a length strictly greater than the memory of a process? Each of these methods results in External Fragmentation. Internal fragmentation exists due to the segment length being dependent on the size of the process’s memory. It occurs when the segment size exceeds the memory requirements of the process. If we use the third method, first fit, we might end up with a larger segment than needed. Since the allocation process requires that each process gets its own memory block, no segments exist that can be shared. An application of contiguous memory allocation is found in Kernel Memory.

The second of the three schemes is pure segmentation. Separate memory into several segments that have a name and length. The programmer specifies <name, offset> (i.e. arr[i]). The name is actually a reference to the segment-number, so the programmer is actually specifying <segment-number, offset> (ie. P[arr][i]). Logically addressing segments of physical memory into main memory allows the memory of a process to be stored noncontiguously in physical memory. External fragmentation can still occur if the physical memory is not a multiple of the segmentation length. Internal Fragment doesn’t occur because segments of equal length are loaded into physical memory a single time and reused. Shared Segments are possible due to reentrant blocks of physical memory. An application of segmentation can be found in Libraries and DLLs.

The third of the three schemes is pure paging. Logically addressing segments of physical memory into main memory with fixed-sized blocks called frames. Logically addressing main memory into blocks of the same size called pages. Each address is separated into two parts: page-number & page-offset. The page-number is the index of the page table. The page table contains the base address in physical memory. The physical memory addresses are calculated by combining the page-number with the page-offset. No external fragmentation due to the equivalent segment size of the logical addressing. Internal Fragment doesn’t occur because segments of equal length are loaded into physical memory a single time and reused. Shared Segments are possible due to reentrant blocks of physical memory. This allows the memory of a process to be stored noncontiguously. Applications of paging can be found in User Program Memory.

Pure Paging has the advantage over the other two memory allocation schemes due to its double use of the logical addressing scheme, allowing for the elimination of both external and internal fragmentation. According to our book (Silberschatz, Galvin and Gagne, 2014), “…we have no external fragmentation: any free frame can be allocated to a process that needs it.” Paging allows for reentrant code and reusable non-contiguous memory which lends itself to segment sharing. It is also the current method employed in most modern operating systems. Here is a chart to display which of the three schemes has external fragmentation, internal fragmentation, or segment sharing.

|  |  |  |  |
| --- | --- | --- | --- |
|  | External Fragmentation | Internal Fragmentation | Shared segments |
| Contiguous | Yes | Yes | No |
| Pure Segmentation | Yes | No | Yes |
| Pure Paging | No | No | Yes |

References:

[1] Silberschatz, A., Galvin, P. and Gagne, G. (2014). *Operating system concepts*. 9th ed. Hoboken, N.J: Wiley.