


Memory Management 


The functions of any OS include:

- o Keeping track of free and used memory areas.
- o Allocating enough memory to each starting process.
- o Protecting the memory area of each process from unauthorized access.
- o De-allocating memory from terminating processes.

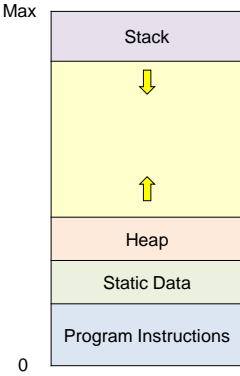
ELC 467– Spring 2020 Lecture 7 Page 6

Slide 1

We now start the study of some operating system functions related to memory management. The operating system is responsible for assigning each process a portion in memory in which it can run, and protecting this portion from unauthorized access by other processes. System uses appropriate algorithms and data structures to perform these functions.

Memory Management 

A typical layout of process memory contains different sections as shown.



The diagram illustrates the memory layout of a process. It is a vertical stack of four sections. From top to bottom: 'Stack' (purple), 'Heap' (orange), 'Static Data' (green), and 'Program Instructions' (blue). The top of the Stack is labeled 'Max' and the bottom of the Program Instructions is labeled '0'. A yellow arrow points downwards from the top of the Stack, and another yellow arrow points upwards from the bottom of the Heap, indicating their growth directions.

ELC 467– Spring 2020 Lecture 7 Page 7

Slide 2

Typically, memory used by a process stores program instructions, static variables and data structures, and dynamic heap (growing into larger addresses) and stack (growing into smaller addresses). On contrast to the process execution time which is generally unknown to the system, we can assume that process memory requirements are declared and known to the system. Until otherwise stated, we assume that process must work from undivided (or contiguous) area in memory.

Memory Management

In a single- programmed system, OS occupies an area in memory, and the rest of memory is available for the single running process.

In a **multitasking system**, memory is shared between a number of concurrent user processes.

Source: [Silberscahtz 18]

ELC 467– Spring 2020 Lecture 7 Page 8

Slide 4

The idea of relocatable processes and base address was mentioned in first week lecture. Here, the limit is the threshold of memory addresses that process should not go across.

Memory Management

Hardware **Memory Management Unit (MMU)** allows relocation of processes, translates logical addresses to physical addresses, and prevents a process in user mode from accessing addresses outside its assigned area.

Source: [Silberscahtz 18]

ELC 467– Spring 2020 Lecture 7 Page 9

Slide 5

The logical address is the offset we mentioned before. The operation of checking if limit is passed, and adding base to offset will be needed for each memory access. For efficiency, these operations must be done in hardware MMU and not by software.

Memory Management by Fixed Partitions

- Divides memory into n partitions (not necessarily equal).
- Assigns each new process a partition large enough to run it.
- Process owns this partition until it is terminated (or swapped out).

Operating system 8M
2M
4M
6M
8M
8M
12M
16M

ELC 467- Spring 2020 Lecture 7 Page 10

Slide 6

The first memory management technique we consider is the method of fixed partitions. Here, memory is divided into a fixed number of partitions with fixed sizes and limits. Each of these partitions can be assigned to a process. Giving different sizes to partitions allow running processes with different memory requirements.

Memory Management by Fixed Partitions

- Divides memory into n partitions (not necessarily equal).
- Assigns each new process a partition large enough to run it.
- Process owns this partition until it is terminated (or swapped out).


If several partitions can be assigned to some new process, which one to select?

Operating system 8M
2M
4M
6M
8M
8M
12M
16M

ELC 467- Spring 2020 Lecture 7 Page 10

Slide 8

The obvious answer here is the smallest one. Size of partition assigned to a process must be equal to or larger than process requirements. If larger, the additional area cannot be used by any other process, and is hence wasted. This can be minimized if we select the smallest partition that can be used by the process.

Memory Management by Fixed Partitions 


Disadvantages of fixed partitions:

- Limits the **degree of multi-programming** : i.e number of concurrent processes that the system can handle.
- Typically results in large wasted memory area as a result of **internal fragmentation**: unused areas inside partitions.

ELC 467– Spring 2020 Lecture 7 Page 11

Slide 9

If we have n partitions, system cannot run more than n processes concurrently, even if memory still have free space. In addition, internal fragmentation refers to the phenomenon that parts of partitions will be assigned to processes but not actually used. Fixed partitions are not used in current systems, except possibly for some simple embedded systems.

Memory Management by Variable Partitions 

Variable partitions are similar to fixed partitions, but *number and sizes* of partitions change dynamically.

When a process is brought into main memory, it is allocated **exactly as much memory as it requires**. When this memory is deallocated, **splitting and merging** of partitions are possible.

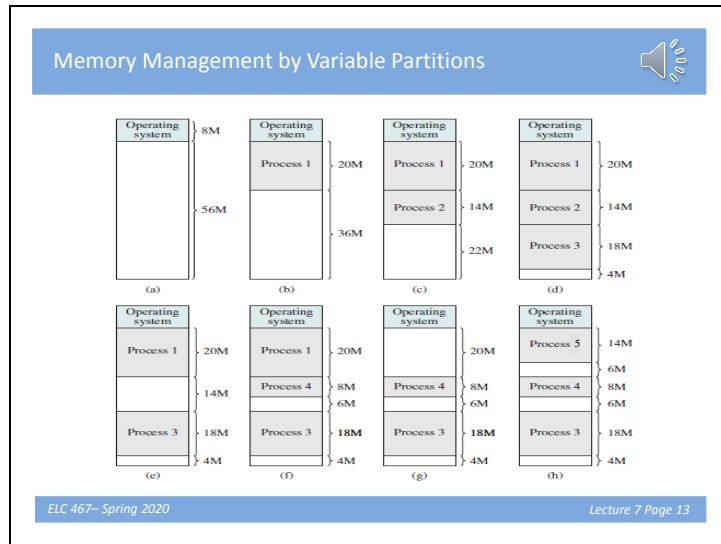
ELC 467– Spring 2020 Lecture 7 Page 12

Slide 10

In this technique also, a memory partition of enough size is reserved for each process from its start to its termination. However, the partitions into which the memory is divided change dynamically.

Slide 11

Process is assigned the exact memory size it needs. If this is taken from a previously larger partition, the remainder of this partition can be used by some other process. Adjacent partitions that become free can be merged into one larger partition. These variable partitions enable avoiding the disadvantages of fixed partitions.



Slide 12

In figure (a), the operating system only is running, reserving a memory partition for itself. In figures (b),(c), and (d) processes 1,2, and 3 are assigned their memory requirements. In figure (e) process 2 terminates, thus releasing the partition it held. When process 4 starts in figure (f), it needs only 8M, thus the partition of size 14 M is split into 8M assigned to new process, and 6M that remains free. Same occurs when process 5 starts.

Memory Management by Variable Partitions 🔊

Variable partitions are similar to fixed partitions, but *number and sizes* of partitions change dynamically.


When a process is brought into main memory, it is allocated exactly as much memory as it requires. When this memory is deallocated, splitting and merging of partitions are possible.

If several areas can be assigned to some new process, **which one to select**? The answer is not obvious as in the case of fixed partitions.

ELC 467– Spring 2020 Lecture 7 Page 12

Slide 13

In fixed partitions, the answer to this question was to select the smallest, to minimize the waste in memory. Here, no waste occurs as a result of internal fragmentation. So what is the best selection?

Memory Management by Variable Partitions 

- Best-fit algorithm
Assigns new process memory from the smallest area with enough free space.
Not always a good policy, as it usually results in many small leftover holes that are not enough for any process, and are thus effectively wasted (**External fragmentation**).


ELC 467– Spring 2020 Lecture 7 Page 14

Slide 14

If we also take the memory requirements from the smallest partition, we call this the best-fit selection.

Slide 15

This tends to minimize the remaining free areas. These remaining areas may be not enough to run any process. Note that we still assume that a process needs to operate in one undivided partition. After using best fit for some time, the free area in memory will be scattered into small holes that are not enough to run processes. This phenomenon is called external fragmentation (i.e. fragments occur outside of partitions).

Memory Management by Variable Partitions 

- Best-fit algorithm
Assigns new process memory from the smallest area with enough free space.
Not always a good policy, as it usually results in many small leftover holes that are not enough for any process, and are thus effectively wasted (External fragmentation).
- Worst-fit algorithm
Assigns new process memory from the largest area with enough free space.
No large holes are left for large processes to run.


ELC 467– Spring 2020 Lecture 7 Page 14

Slide 16

If we select instead the largest partition, we call this the worst-fit algorithm. It should avoid the disadvantage of best-fit.

Slide 17

However, a new problem arises. Since in worst-fit we systematically split large partitions, we expect that no large partitions will be left after a while. Processes with large memory requirements will not be able to run.

Memory Management by Variable Partitions

- First-fit algorithm
Assigns new process memory from area with enough free space and least address.

- Next-fit algorithm
Similar to first-fit, but starts search after last assigned area (circular first-fit).

These result in faster operation and less memory waste than best-fit and worst-fit.

ELC 467- Spring 2020Lecture 7 Page 15

Slide 18

Since selection of smallest or largest partitions will both lead to problems, it is better not to base decision on size. The first-fit algorithm begins looking for a suitable partition from the start of memory, and assigns the first partition it finds. Sometimes this will be small, and sometimes it will be large, so disadvantages of best-fit and worst-fit will not persist.

Slide 19

A slightly different method is the next-fit algorithm. It also assigns the first partition it finds irrespective of size. However, it does not begin the search from the start of memory each time. Instead, it begins search after the last assigned area. This usually will speed up the search, as in first-fit the start of memory will typically be fragmented, and suitable partitions will be found at higher addresses. The name circular first-fit may be used since when search reaches the end of memory, it returns back to the start.



Example: Starting from the shown state, the following occurred in order:

- Process A terminated.
- Process C terminated.
- Process E starts requiring 100K.
- Process F starts requiring 160K.

Where will each algorithm place E and F?

free	304K
D	200K
C	150K
B	100K
A	200K
OS	70K

Slide 20

Note here that the order of events is important as it will change the results.

Slide 21

Now we have free areas of sizes 200 K, 150 K, and 304 K.