Operating Systems W7L2 - Memory Management I and II

Class	💿 Operating Systems
Created	@Oct 14, 2020 5:07 PM
Materials	06 - Memory Management II.pdf
Reviewed	□
Type	Lecture

> ❗ Next lecture will be a revision. Z will be answering questions and giving problems to work on solving

Memory Management I

- Effective = correct, efficient = correct *and* done well

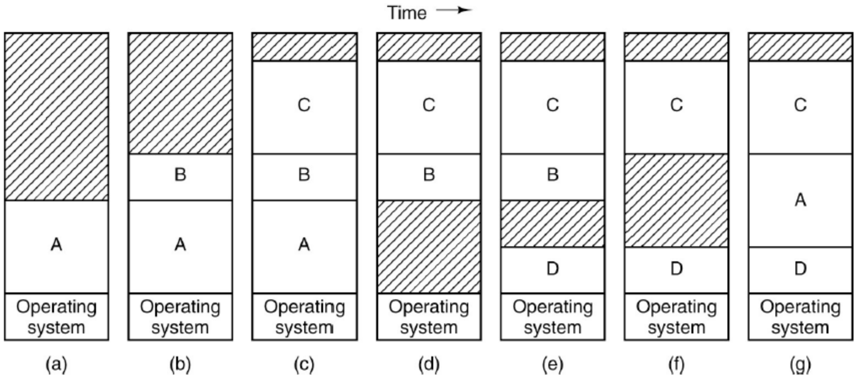
Address Space: Base and Limit

- Base and limit are two registers
    - Base is the start address of a program in physical memory
    - Limit (aka bound) is the length of the program
- The MMU (memory management unit) is the piece of hardware, within a processor, that does all this
- Only the OS can modify base and limit
- It's a logical system, but not quite the best

- Main drawback is that if memory psace is not enough, then we may need to *swap* programs out of memory

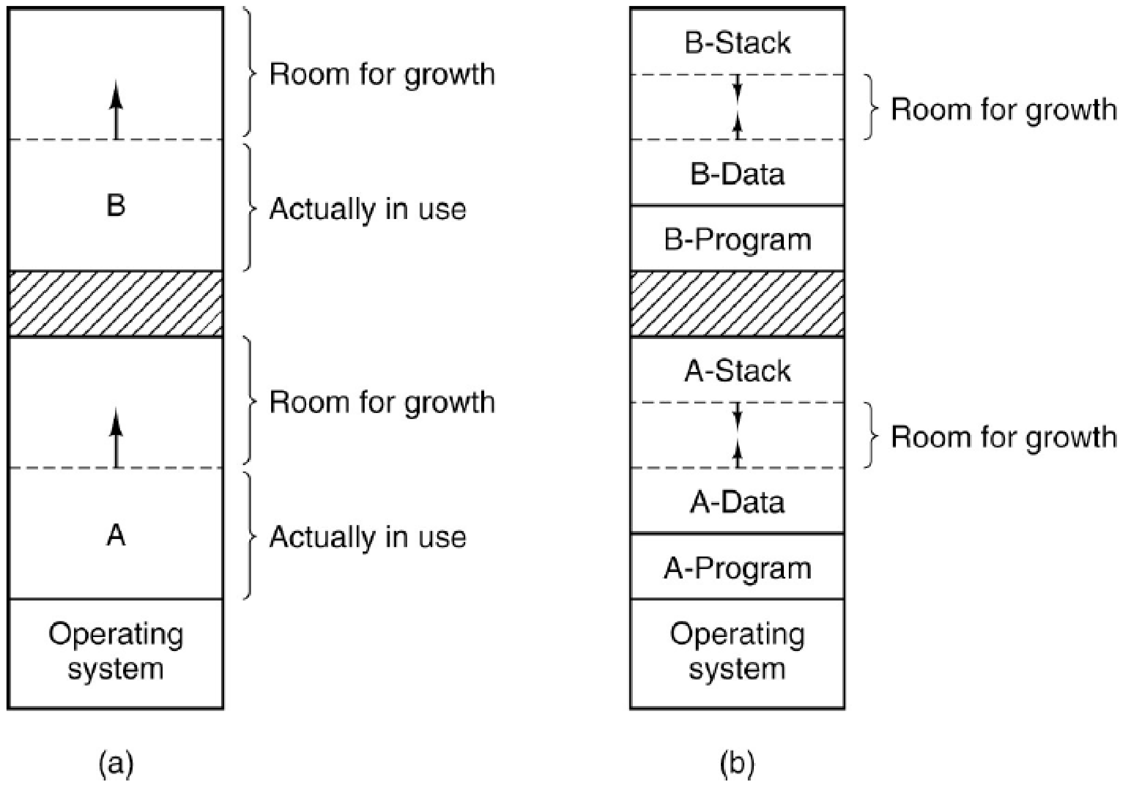
Swapping

▼ Diagram of swapping over time



- There isn't any "pushing" of processes, the OS will just swap things around to get enough space
- **Holes** are created as porcesses go in and out of memory
- **Memory Compaction:** Holes are combined

▼ Diagrams

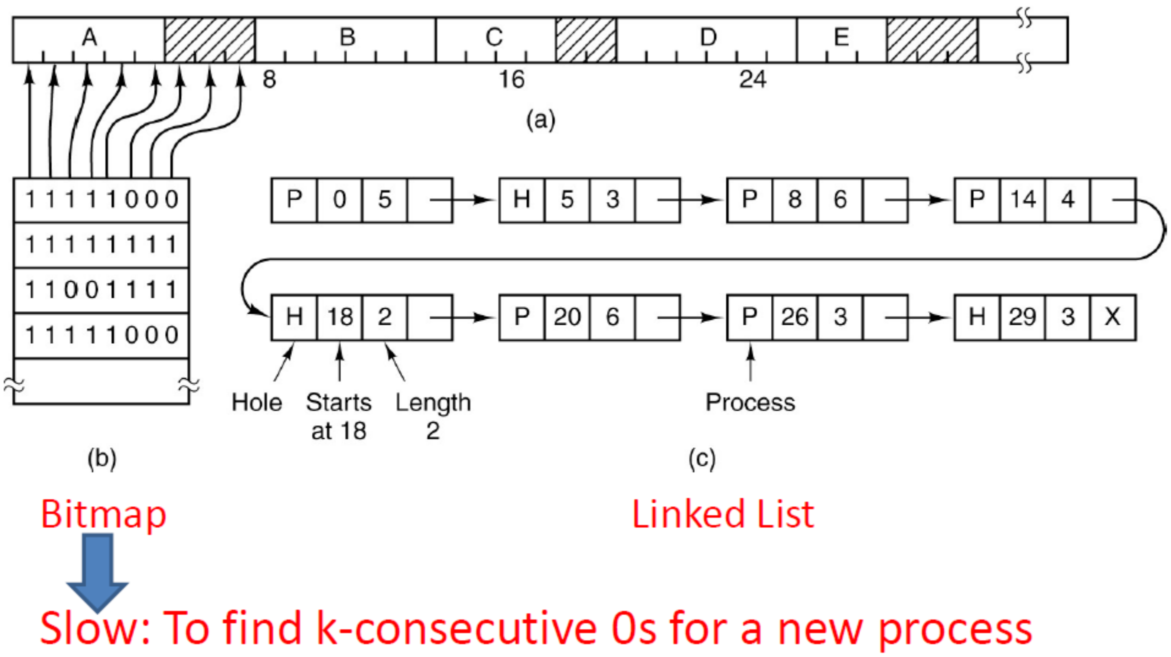


- We do *not* know the size of local variables beforehand, whereas we know the size of global and static variables

Managing Free Memory

- Bitmaps
  - Memory is divided into allocation units *of equal size*
  - Each unit is mapped with a `0` (free) or a `1` (not free) corresponding to bits in bitmap (`1` and `0` could also be in opposite order)
- Linked list of allocated and free memory segments
  - Segments are of different sizes
  - Nodes of linked list store availability
  - LinkedList easier than array so that holes can be combined more easily

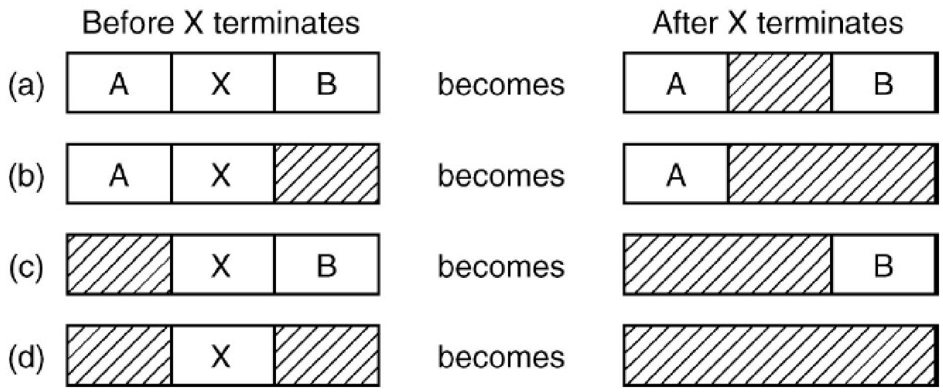
▼ Bitmap and LinkedList Diagram — Why is bitmap slower than linked list?



Slow: To find k-consecutive 0s for a new process

Segments of LinkedLists are generally faster than bitmaps, which have to find consecutive free spaces

▼ Double LinkedList Diagram



- Segments *can* be broken, but only when a process needs space
  - For example, if a process is given 75% of available memory, then a new node will be created with that other 25% of memory

▼ Different allocation methods

1. First fit
2. Best fit
3. Next fit
4. Worst fit
5. ... [There are many more]

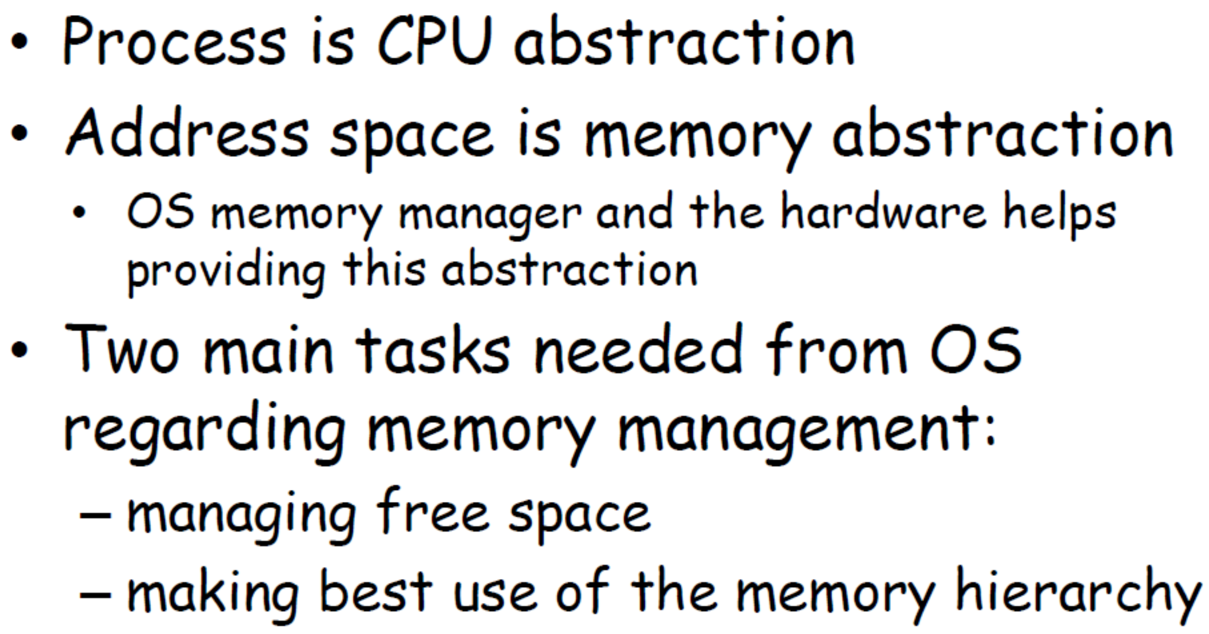
Memory Management Tasks

- ▼ What is memory management's main role?

  To bring processes into main memory for execution by the processor

- Involves *virtual memory* and is based on *segmentation* and *paging*

Conclusion Slide



- Process is CPU abstraction
- Address space is memory abstraction
  - OS memory manager and the hardware helps providing this abstraction
- Two main tasks needed from OS regarding memory management:
  - – managing free space
  - – making best use of the memory hierarchy

Memory Management II

- You can never really have "enough" memory → Determining "enough" isn't possible since programs keep growing in size

- If each program exceeds memory we have, why don't we run out of memory?
- Logical, or virtual, addresses that are dyanimcally translated into physical address at run time are *memory references*
- A process can be broken up into several pieces (pages) — To be covered more when we resume talking about memory
- **Virtual Memory:** Mapping from logical (virtual) address space to physical address space

---

> ❗ If you have questions for the review lecture, you can put them on NYU Classes. Be specific!