THE UNIVERSITY *of* EDINBURGH
informatics

Operating Systems (INFR10079)

2023/2024 Semester 2

Memory Management

(Allocation)

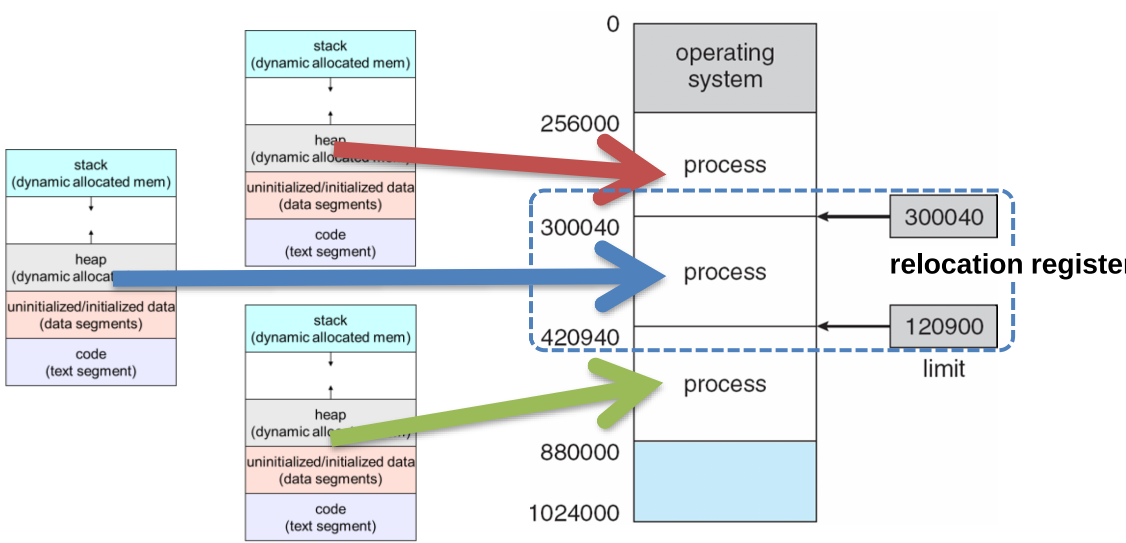
abarbala@inf.ed.ac.uk

Chapter 9.1, 9.2, 9.5

Contiguous Allocation #1

- Main memory **must** support both OS and user processes
- Memory limited resource **must** allocate efficiently

- Contiguous allocation is an *early method*
- Main memory usually into two **partitions**
  - OS (usually) held in low memory with interrupt vector
  - Processes held in high memory
    - Each contained in single contiguous section of memory



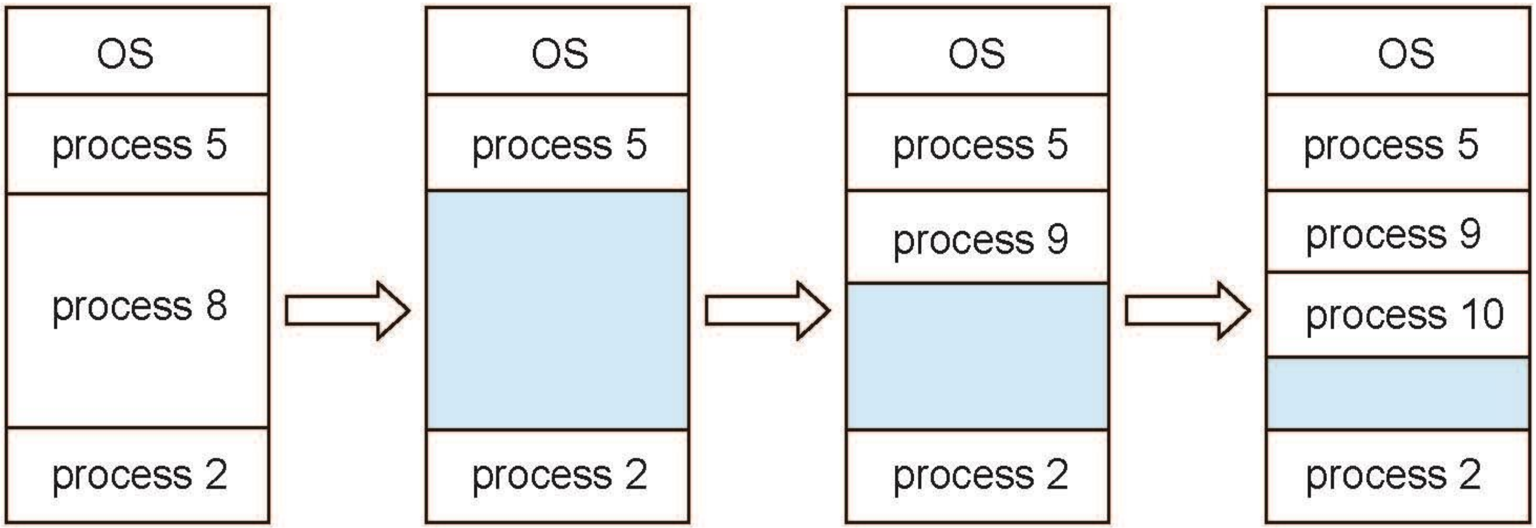
Contiguous Allocation #2

- Relocation and limit registers
  - To **protect user processes**
    - from each other
    - from changing OS code and data
  - *Relocation register* contains value of smallest physical address
  - *Limit register* contains range of logical addresses
    - Each logical address must be less than the limit register

- MMU translates logical address
  - *transparently, during execution*

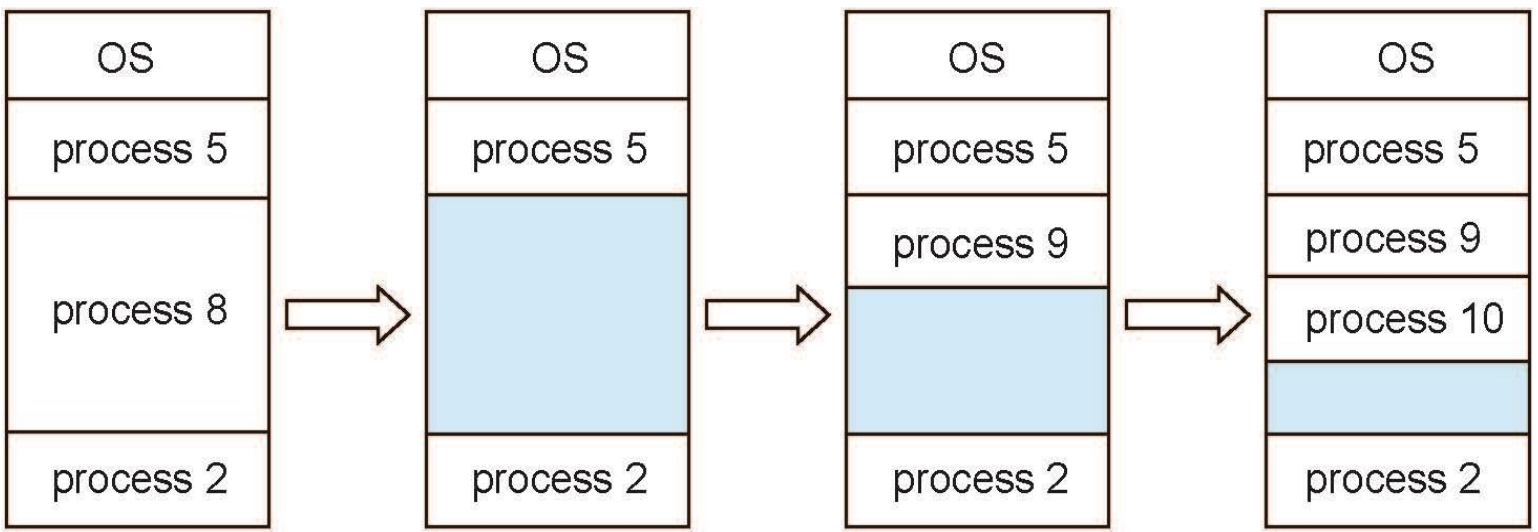
Multiple-partition Allocation #1

- Multiple-partition allocation
  - Variable partition size (size of a program) **for efficiency**
  - Degree of multiprogramming **limited** by number of partitions



Multiple-partition Allocation #2

- Multiple-partition allocation
  - **Hole:** block of available memory; holes of various sizes
  - When a process arrives, **OS allocates** memory from a hole large enough to accommodate it
  - Process exiting, **returns partition to OS** , adjacent free partitions combined
  - Operating system maintains information about
    - allocated partitions
    - free partitions (hole)



Dynamic Storage-Allocation Problem

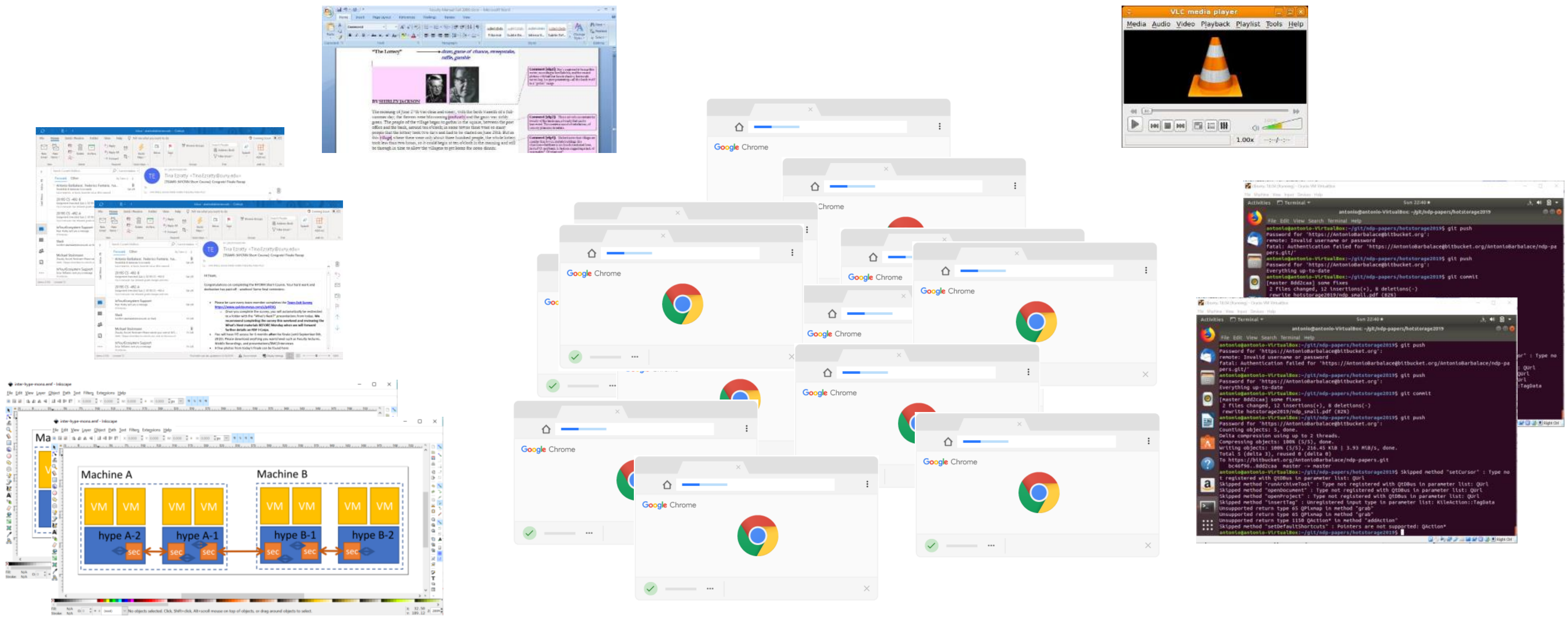
How to satisfy a request of size ***n*** from a list of free holes?

- **First-fit**: Allocate the ***first*** hole that is big enough
- **Best-fit**: Allocate the ***smallest*** hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- **Worst-fit**: Allocate the ***largest*** hole; must also search entire list
  - Produces the largest leftover hole

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

Multiple Programs: Swapping (1)

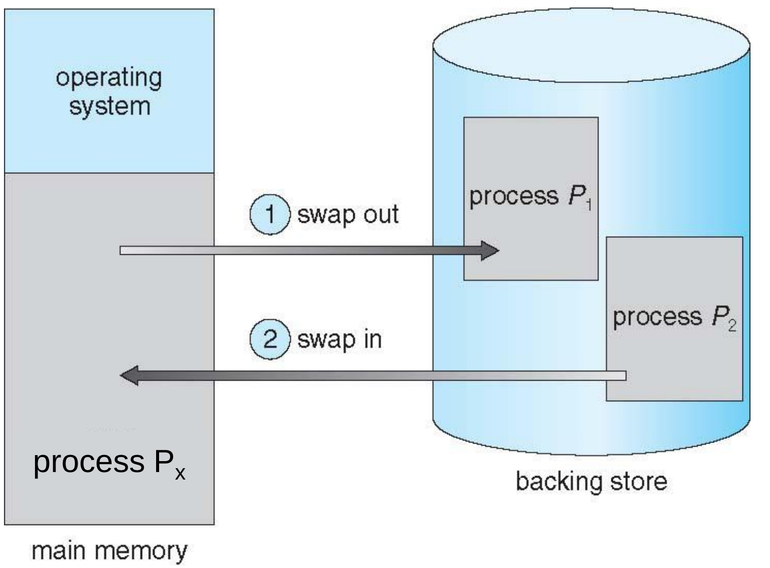
- How many programs are currently executed?
- Do they all fit in memory?



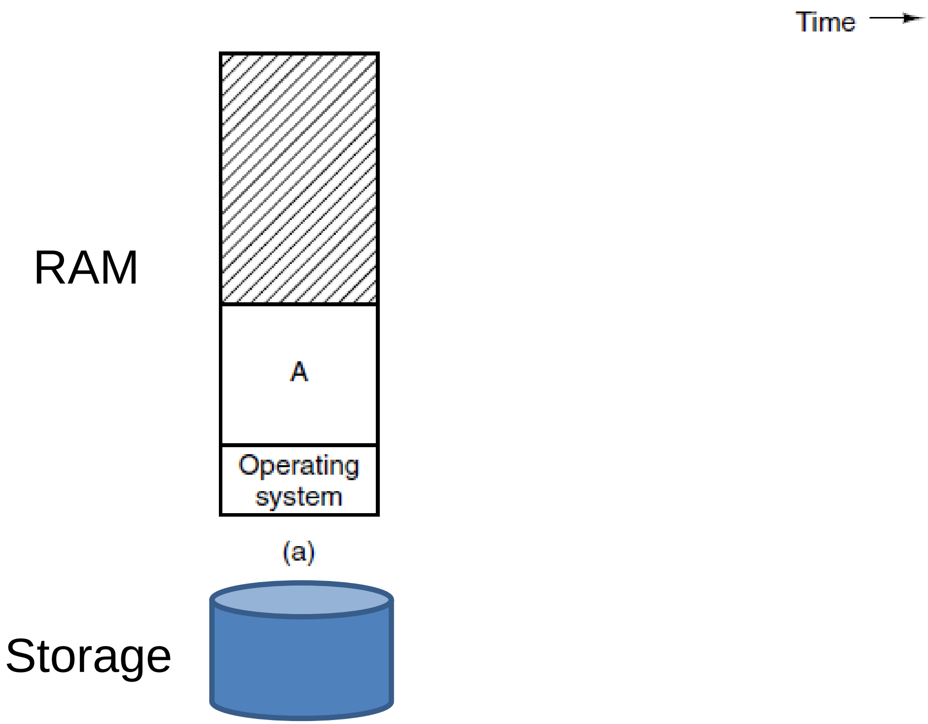
Multiple Programs: Swapping (2)

- How many programs are currently executed?
- Do they all fit in memory?

- Swapping
  a) **SWAP IN:** Bringing in memory a process in its entirety (from disk)
  b) **RUN:** Running it for a while
  c) **SWAP OUT:** Putting it back from memory (to disk)
  - About **running** processes, no programs!
  - Idle processes are stored on disk,
    - Do not take up any memory when they are not running

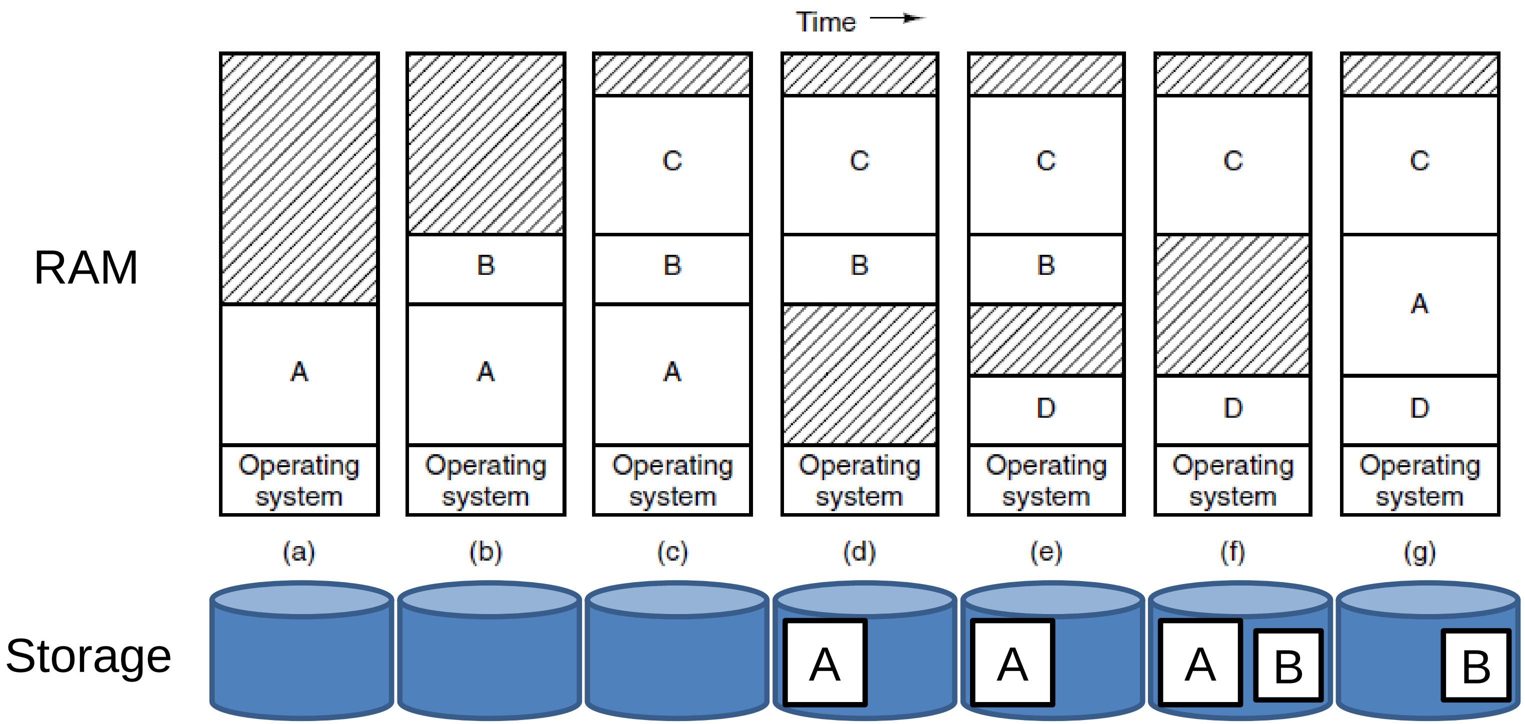


Multiple Programs: Swapping (3)



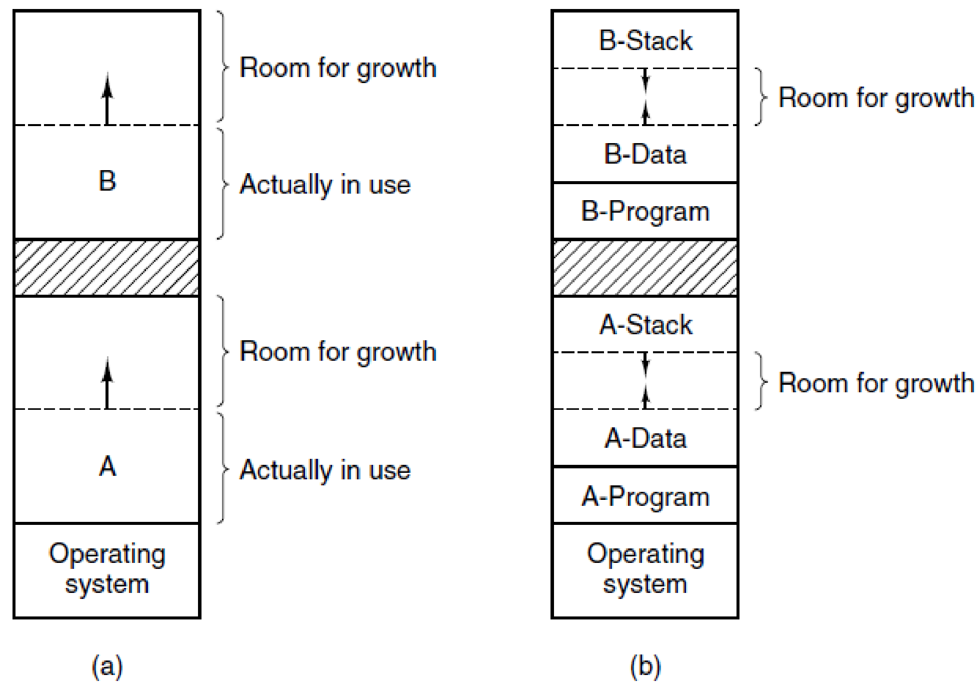
Memory allocation changes as processes come into memory and leave it. The shaded regions are unused memory (MOS Figure 3-4)

Multiple Programs: Swapping (3)



Memory allocation changes as processes come into memory and leave it. The shaded regions are unused memory (MOS Figure 3-4)

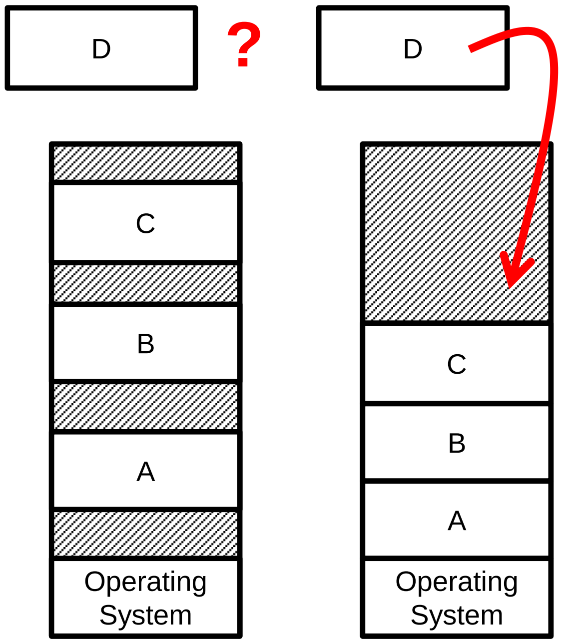
Multiple Programs: Growing Programs



(a) Allocating space for a growing program at the end – if more space is needed, relocate
(b) Allocating space for a growing program in its address space – max amount of growth
(c) Both solutions are not ideal

Multiple Programs: Memory Fragmentation #1

- Process creation and exit, swapping, process growing and shrinking
- Create **memory holes**
  - New processes, memory is available, cannot be placed
- **Compaction**
  - Move all processes tightly together
    - By memory copy or
    - By swapping out and in
    - Computationally expensive
  - What if a process needs to grow its heap or stack?



Multiple Programs: Memory Fragmentation #2

- **External Fragmentation**
  - You allocate the exact amount of memory requested
  - Total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation**
  - You allocate more than what required
  - Allocated memory may be slightly larger than requested memory
- First fit analysis reveals given *N* blocks allocated, 0.5 *N* blocks lost to fragmentation
  - 1/3 may be unusable

Concepts

- Relocatable binaries
- MMU
- Logical Address Space vs Physical Address Space
- Contiguous Memory Allocation
  - First fit, best fit, worst fit
- Swapping
- Growing programs
- Fragmentation
  - Compaction