# Heap Memory Management and Garbage Collection

CS439: Principles of Computer Systems

# Bringing It Together

- Processes are the OS's main abstraction for protection
- A process defines an address space---but this is a virtual/logical address space
- Virtual address spaces are mapped onto physical address spaces in physical memory
- *Paging* describes how the OS provides the illusion of infinite memory
  - Page tables aid address translation and data finding
  - Page faults identify which pages are needed and not yet in memory
  - Page replacement algorithms decide which pages to evict when a free frame is needed
  - Or we can create free frames ahead of time using the working set
- The heap is part of a process's virtual address space (and is thus paged!)
  - Heap can be managed explicitly---user directs the heap memory manager
    - Allocation policies: free list and bump pointer
    - Deallocation: only for free list!
  - Heap may be managed automatically---user still allocates memory, but the heap manager automatically manages it
    - Same allocation policies
    - Deallocation: garbage collection!
      - Identification and Collection

# Vocabulary (pt 1)

automatic memory management
explicit memory management
runtime system
programmer control
free block
heap memory manager
coalescing with neighbors
arbitrary request sequences
memory re-use
page
bump pointer
delete
binning
fast allocation
memory leak
dynamically allocated data
malloc
new
external fragmentation
free
safe pointers
internal fragmentation
free list
allocator
heap
programmer burden

# Vocabulary (pt 2)

nursery
mark-sweep
dead objects
sweep-to-free
allocation
program roots
mark-compact
compaction
old space
tracing
mutator
free list
SDS
from space

identification
locality
reclamation
stack
locality
mutator
evacuate
to space
allocation time
registers
heap
semi-space
bump pointer
copying

reference counting
contemporaneously allocated objects
address space
generational hypothesis
reachable objects
garbage collection
live objects
young space
multi-pass collection
space efficiency
objects connected by pointers
safe pointers
non-copying
incremental object reclamation