Memory management 2

CS503: Operating systems, Spring 2019

Pedro Fonseca
Department of Computer Science
Purdue University

Copyright © 2018 by Douglas Comer And CRC Press. All rights reserved.

Edited by Byoungyoung Lee and Pedro Fonseca

Previous lecture

- Divide memory manager into two pieces:
  - Low-level used in kernel to allocate address spaces
  - High-level used to handle abstraction of virtual memory and paging within an address space
- Two conceptual memory types in low-level pieces
  - Heap
  - Stack
- (Globals are shared)

Recall: Low-level memory manager in Xinu

- Stack allocation/free in Xinu

```
addr = getstk(numbytes);
freestk(addr, numbytes);
```

- Heap allocation/free in Xinu

```
addr = getmem(numbytes);
freemem(addr, numbytes);
```

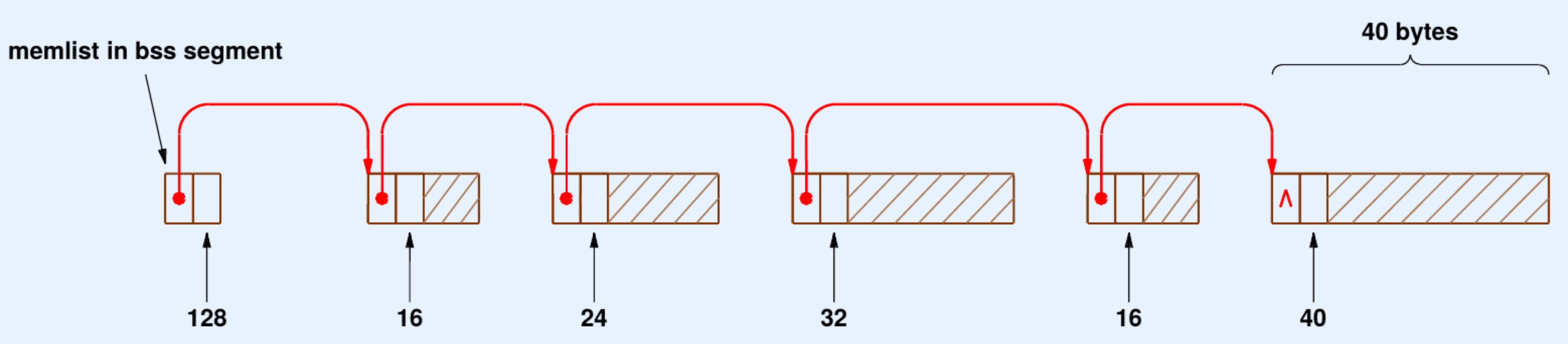
Recall: Xinu low-level allocation

- One free area
- Single free list used for both heap and stack allocation:
  - Ordered by increasing address
  - Singly-linked
  - Initialized at system startup to contain all free memory
- Allocation policy
  - Heap: **first-fit**
  - Stack: **last-fit**
  - Results in two conceptual memory pools

Recall: Allocation technique in Xinu

- **getmem():**
  - Round up the requested allocation
  - Walk free memory list
  - **First-fit**: choose first free block that is large enough
- **freemem():**
  - Find prev/next such that the address of the block being freed lies between *prev* and *next*
  - Either:
    - Coalesce with the previous block if new block is contiguous
    - Add the new block to the free list

How does the free list layout look like?



Stack allocation in Xinu

- **getstk()** and **freestk()**
  - Last fit version of **getmem()**

Process creation

- Requirements
  - Allocate stack for the new process
  - Fill in process table entry
  - Place pseudo call on stack as if new process was called

Creating process

- **create()**:
  - Allocate a process table entry
  - Allocate a stack
  - Place values on the stack as if the top-level function was called (pseudo-call)
    - Will eventually return to INITRET (**userret()**)
  - Arrange saved state so context switch can switch to the process
    - Recall **ctxsw()**

Killing process

- **kill()**:
  - Implements process termination
  - Action depends on state of process:
    - If process is on the ready list, must remove it
    - If process is waiting on a semaphore, must adjust the semaphore count

Killing process (cont.)

- **kill()**:
  - Look carefully at the code:
    - Step 1: free the process’s stack
    - Step 2: perform other actions
  - Consider a current process killing itself
    - The call to resched() occurs after the process’s stack has been freed
    - Any potential issues?