Operating Systems!

Memory: Approaches to Memory Management (Part 3)

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

https://www.cs.montana.edu/cs460

Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

Today

- Announcements
  - ~~Yalnix~~ How are projects/proposals going?! ☺
  - *Exam 1: Nice work! **(Still)** Working on grading…*

- Upcoming Deadlines
  - **Sunday [10/18/2020] @ 11:59 PM (MST)** - PA3 (Group Assignment)

    Yalnix teams: *While PA3 == yalnix checkpoint 1, you should plan to be near completing checkpoint 2!*

  - **Sunday [10/25/2020] @ 11:59 PM (MST)** - Project Proposal

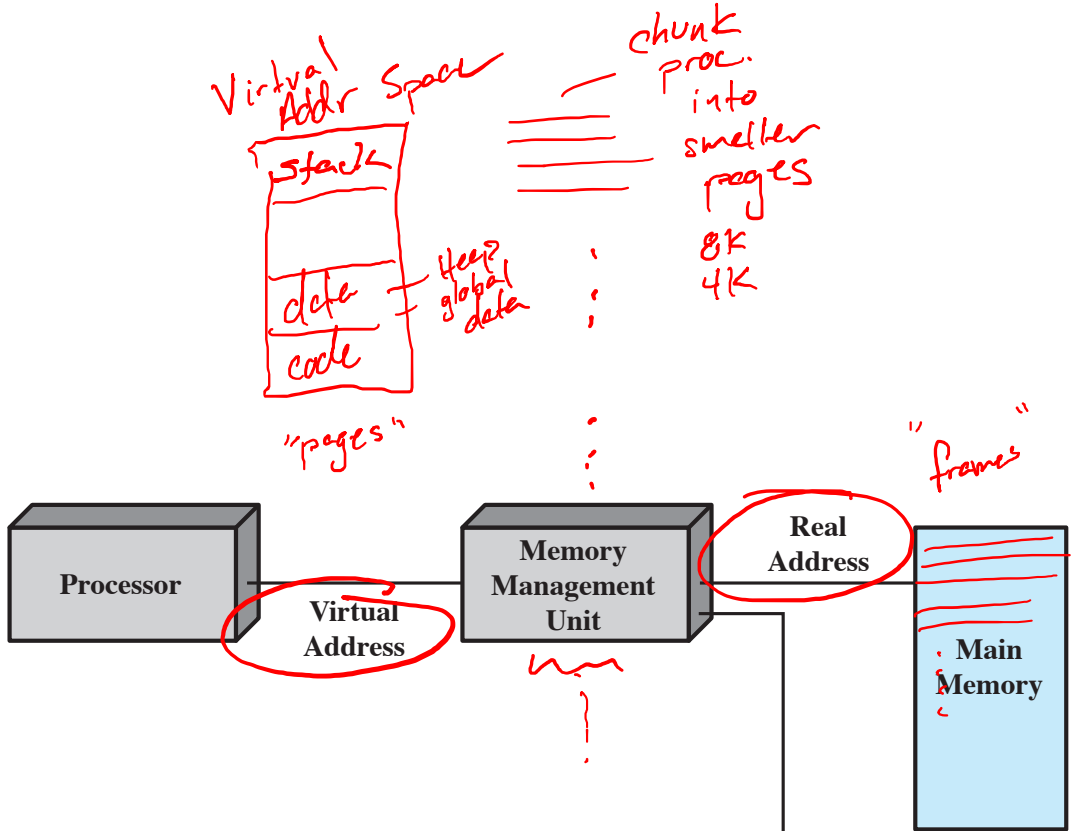
    Yalnix teams: *you should plan to be near completing checkpoint 3!*

Today *(cont.)*

Last Week: Logical vs. Physical Addresses, Paging, Address Translation, ...

- Agenda
  - Page Tables
  - Heaps → review fault.c 3,4
  - Stacks → review fault.c 5,6,7 + sf.c
  - The TLB
  - Switching Processes & the TLB

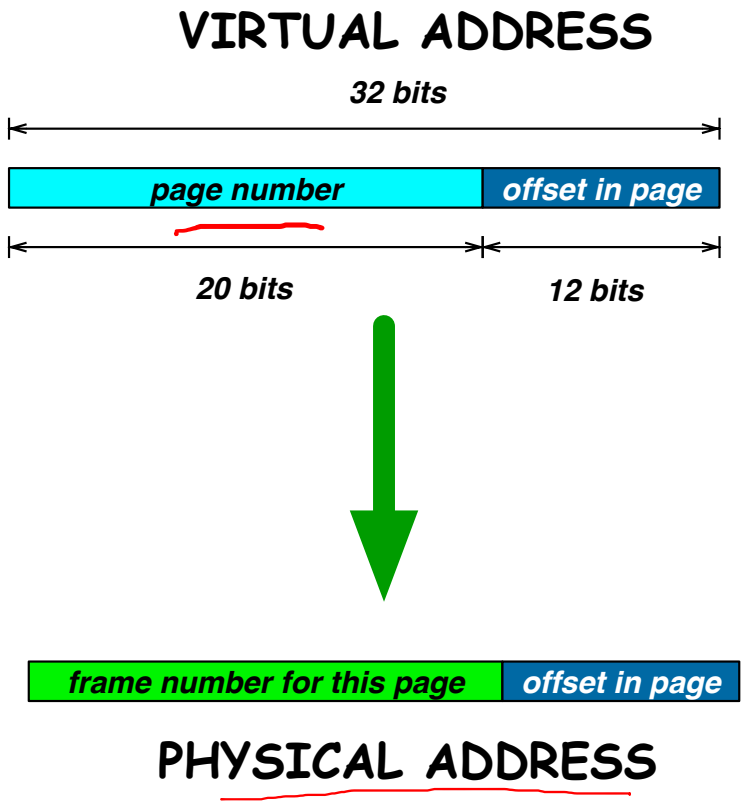
➔ Virtual Memory



Address Translation

Q: How do we find the frame number that matches a page number?

0x5660e853



0x?????853

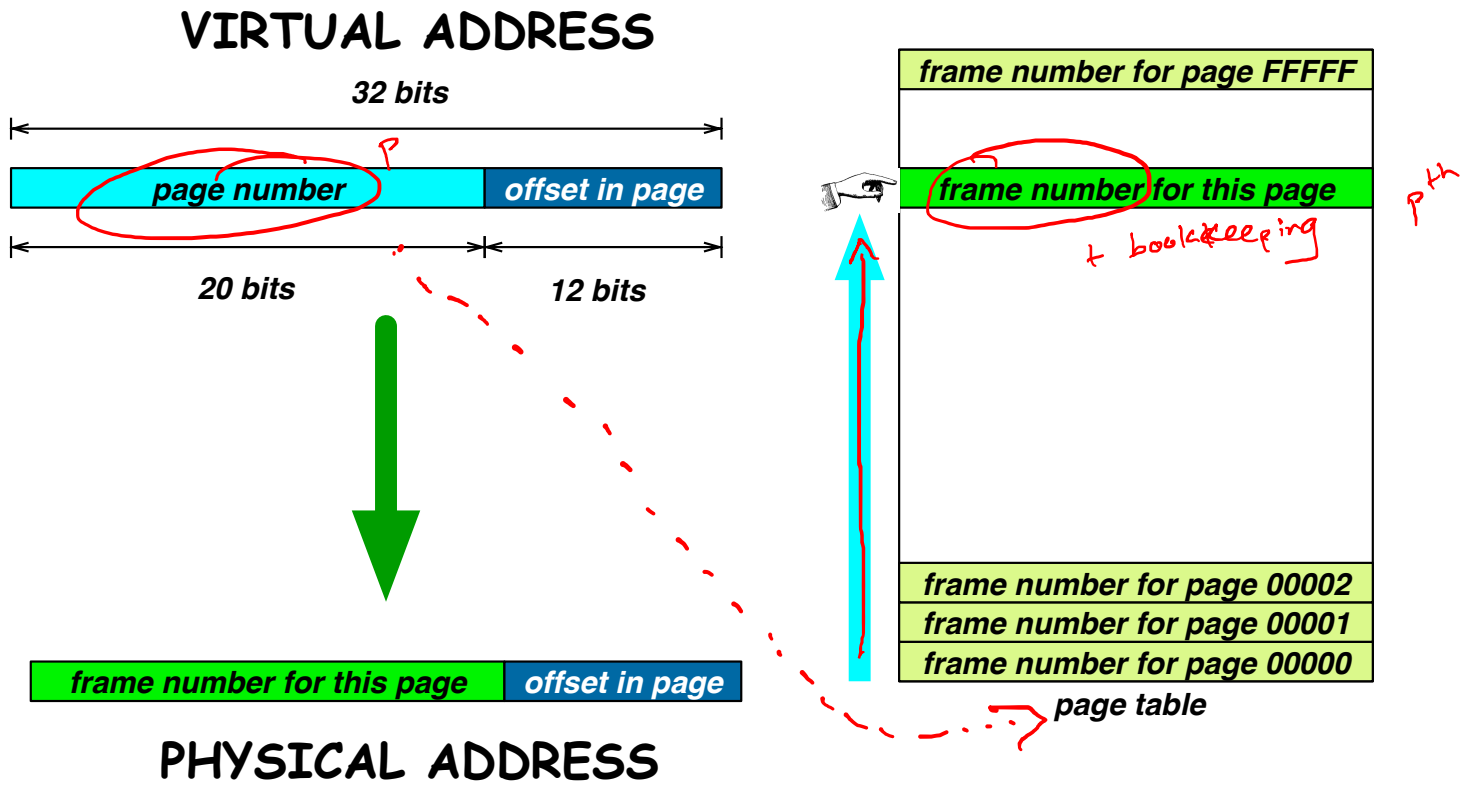
Array... page is an index ⇒ frame

see: psize.c
➔ (-m32) psize32 and (-m64) psize64

Address Translation

A: We use a page table!

0x5660e853



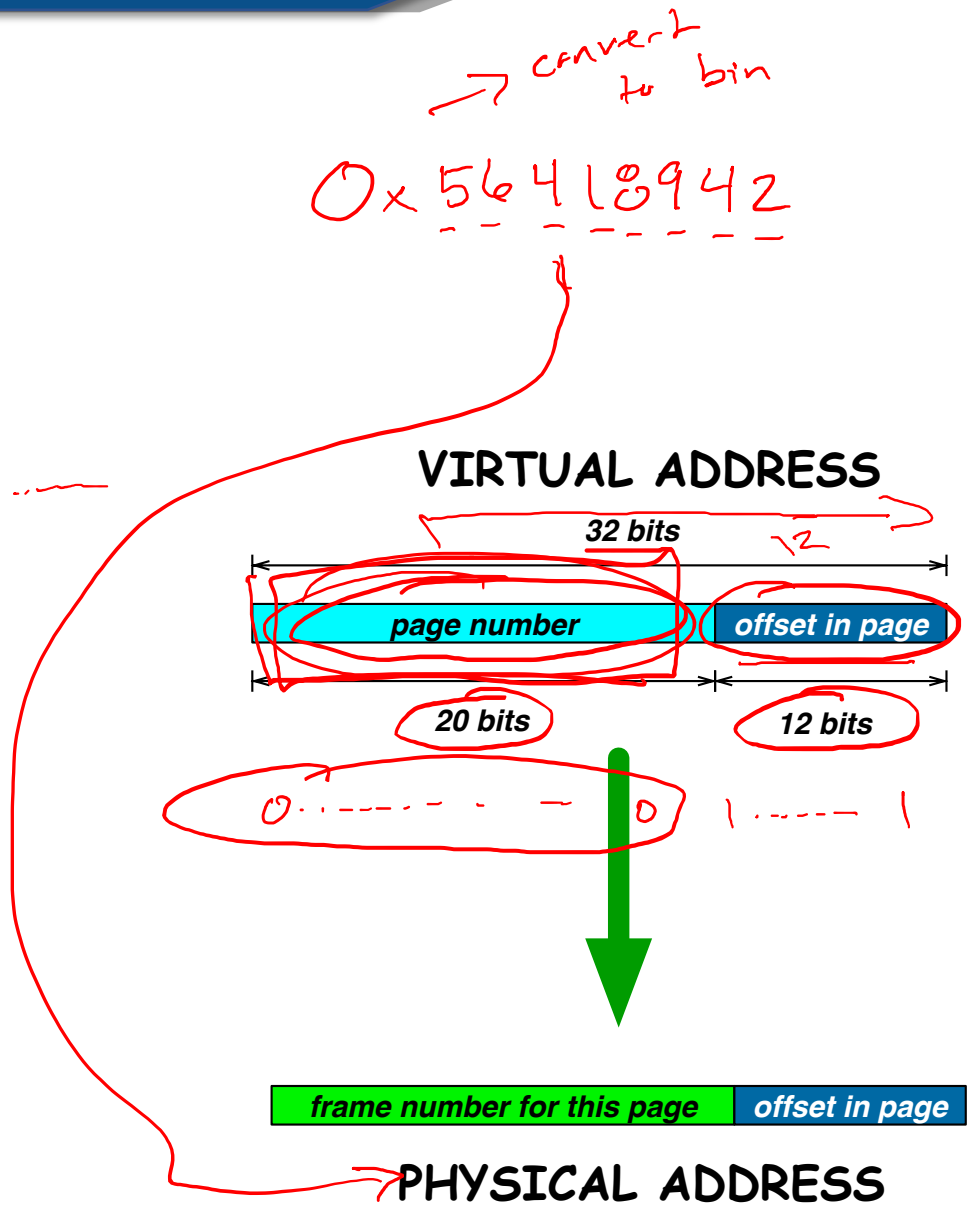
0x?????853

see: psize.c
➔ (-m32) psize32 and (-m64) psize64

Address Translation

Activity! (Think-Pair-Share)

- How to quickly **get the page number** to index into the page table?
- How to quickly find the physical address of **the zeroth byte in frame f**?
  - hint: think binary operations!
- How do we quickly **add the offset** to it?
- **Who** does this translation?



The Heap

Heaps of Fun!

Top of the Heap

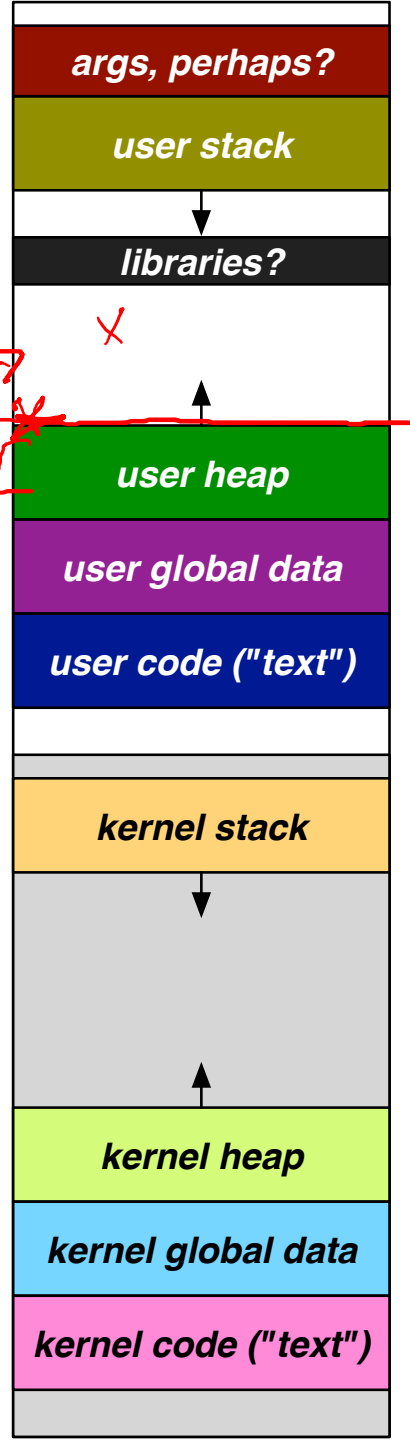
A look at heap memory management

- The first address *right after the heap* is often called the **"break"**

- We can manage the heap region **explicitly** via syscalls!

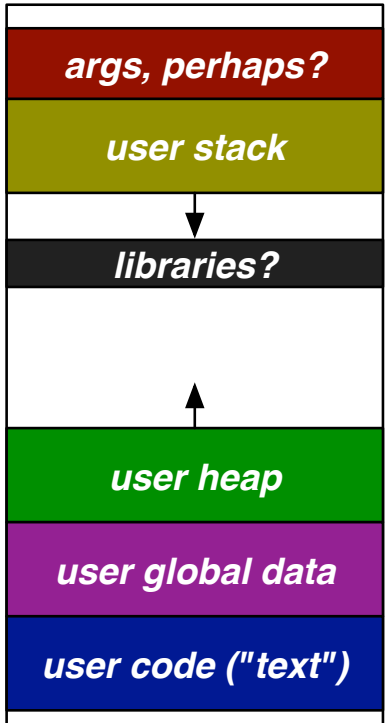
  - **brk**(address)
    ➔ try to set "break" to **address**

  - **sbrk**(delta)
    ➔ try to increase heap by **delta** bytes



Reflection: You might want to think about what the kernel might do when it implements these syscalls.

OK, so how does **malloc** work?



- What do **malloc()** and friends actually do?

```
type * ptr = malloc (sizeof(type));
...
free(ptr)
```

- Suppose you had to write such a library?
  - What do you need to keep track of?
  - What algorithmic/design issues do you need to worry about?
  - How do you evaluate if you're doing a good job?

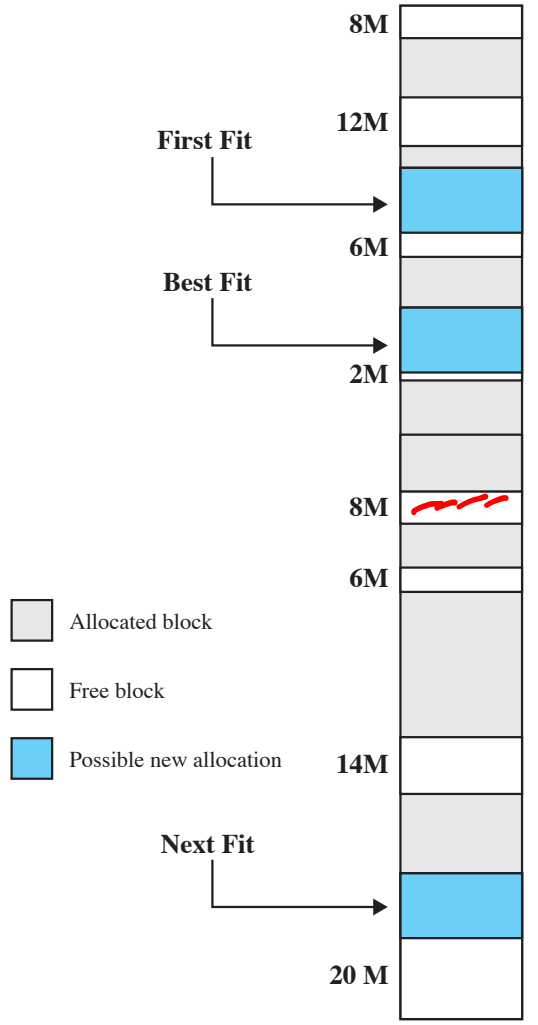
see: **new_heap.c & fault.c (3,4)**
*(Also, try running fault with **strace**!)*

OK, so how does **malloc** work?

- What do **malloc()** and friends actually do?

```
type * ptr = malloc (sizeof(type));
...
free(ptr)
```

- Suppose you had to write such a library?
  - What do you need to keep track of?
  - What algorithmic/design issues do you need to worry about?
  - How do you evaluate if you're doing a good job?



It's dynamic storage allocation again!

see: **new_heap.c & fault.c (3,4)**
*(Also, try running fault with **strace**!)*

The Stack

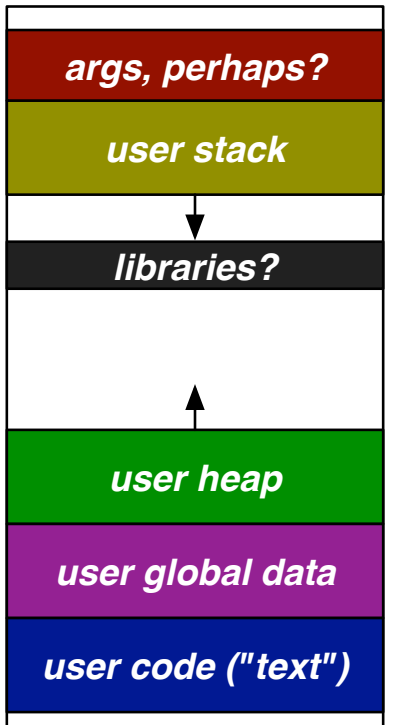
Stacks of Mystery

Bottom of the Stack

- Stacks "grow" in a different way...
  -> implicit growth!

- Basic idea:
  - if access is within max limit, allocate page(s)
  - else send SIGSEGV (which typically kills it)

- *How?* Page Faults!
  - **major fault**
    vs.
  - **minor fault**



See: **How does stack allocation work in Linux?**
https://unix.stackexchange.com/a/239323

see: **fault.c (5,6,7)**
*(Why don't we see syscalls when running **strace**...)*