Midterm Review

CS 241

March 5, 2012

Announcements

Midterm exam: Tomorrow (Tue), 7 – 9 pm

Location according to last name

- A-Liang: 1404 SC
- Lim-Z: 1320 DCL

Discussion this week? YES

Lecture on Friday? NO

Today

- All yours
- Special requests: Page tables, page faults, MMU, page tables, TLB, page tables, multilevel page tables, ...
- Malloc algorithms
- Synchronization

Virtual memory: key concept review

Virtual memory

- Memory addresses used by an application
- Unrelated to physical address
- May not even be stored in physical memory

Physical memory

- The RAM in your computer

Memory Management Unit (MMU)

- Hardware which translates virtual to physical addresses every time any program accesses any memory

Virtual memory: key concept review

Page

- Unit in which OS allocates memory to applications
- MMU also works in units of pages

Page table

- Data structure used by MMU to remember virtual-to-physical mapping
- One per process (why?)
- Created by OS, stored in memory (top level, at least)
- (What events modify the page table?)

Translation Lookaside Buffer (TLB)

- Cache of virtual-to-physical mappings
- Faster than extra memory references needed to look up in page table
- Must be flushed when switching between apps

Virtual memory: key concept review

Multilevel page table

- Top level page table points to other page tables rather than individual pages
- What is the point of this?

Segmentation fault

- Program accesses memory outside the segments that it is allowed to access (e.g., deref NULL, write past end of heap, etc.)

Page faults

- Happen when the virtual page (which the application is trying to access) is not currently mapped to a valid physical page
- Seg fault is one kind of page fault
- When is a page fault "normal"?

From practice exam

Consider the following code segment:

```
void *ptr = malloc(1024 * sizeof(char));
printf("%p", ptr);
```

When this program is run as two separate processes, you notice the following output:

Process #1: 0x49301240
Process #2: 0xac382ac0

Based on the output above, what can be determined about the address contained in ptr?

A. The address of Process #1 is located before the address of Process #2 in physical memory.
B. The address of Process #2 is located before the address of Process #1 in physical memory.
C. The address of Process #1 and Process #2 is located in the same physical memory.
D. None of the above.

Address translation: single PT

x = 5;

Virtual page number Offset

store 5 in: 01010110 01010011 0101**1010 10100010**

How is this translated?

Need:

- Size of a page = 4 KB (our starting assumption; varies across different hardware architectures)
- Size of page table: # entries = 2^{32} / 4 KB = 2^{20}
- Size of a page table: # bytes = 2^{20} * 4 bytes = 2^{22} bytes
- Offset size (# bits of offset in virtual address) = 12 bits
- Virtual page number size = 32 – 12 = 20 bits
- Page table data structure

Address translation: 2-level PT

x = 5;

store 5 in 01010111 01010011 01011010 10100010

How is this translated?