CS 350 Notes

Matthew Visser

Oct 18, 2011

Mistake from last class: the volatile should go by the pointer.

1 Shared Virtual Memory

Shared object files are files like `*.so` or `*.dll`. These are shared libraries that are loaded into memory.

1.1 Kernel Address Space

Having the kernel use virtual addresses is confusing, because the kernel needs the virtual address mapping to work before it can use it. Because of this, we just have the kernel use physical addresses.

The kernel can have its own page table, but it's tricky to do so. If the kernel gets an address, it's physical, so it must reverse the mapping first.

Linux and most other OS's have a shared segment for every process that holds an address space in the kernel. MIPS helps the OS do this by reserving the upper half of address space for the kernel. The upper half is divided into three kernel segments: `kseg0`, `kseg1`, `kseg2`.

The MIPS processor helps by

1. Addresses $\geq 0x80000000$ are *privileged*.
2. `kseg0`: virtual address `0x80000000` is mapped to physical address `0x00000000` without a TLB.
    - This makes booting the machine a lot easier. We now know exactly what address the machine starts executing at, so we can put our boot loader there.
3. `kseg1` starts at `0xa0000000` and is mapped to `0x00000000` without a TLB and bypassing CPU cache. We have this segment for IO devices.

4. `kseg2` is there for any addresses in `kseg0` that are over the address range. This is not used in OS/161.

Midterm up to slide 28 of `vm.pdf`.

2 Notes About Midterm

Date:	Oct 25
Time:	4:30 - 6:30
Notes:	There is no lecture on this date. Instead, office hours are held from 1000h to 1100h in DC 3349, not in the lab.

How to study for the midterm?

There are some sample midterms on the course web page.

3 Assignment 2 Discussion

User programs to test with are in `testbin`. Different programs use different system calls.

3.1 Write and Exit

Get write to console and `_exit` working and run palin.

1. Define a handler function and call it from `mips_syscall`
2. Copy data from user space.
3. Do Logic of `write` or `_exit`.

How do we print data to the console? We use `kprintf`.

1. `kprintf` uses a lock to write strings atomically.
2. Calls `putch()` which puts one character to the console. The name of the console is called `/dev/generic/console` or "`con:`". This device is *asynchronous*.

How do we copy data from user space into the kernel?

A kernel can dereference a pointer from user space because the kernel is mapped into the address space of every process. We need to copy data because `kprintf` could cause our thread to sleep. We might also want data to be aligned in the kernel.

We can copy using `copyin/copyout` and `copyinstr/copyoutstr`. These copy functions call `memcpy`. The string copy functions call `copystr` which is like `strncpy`.

UIO Move → moves data from/ to kernel buffer and user level buffer. Specified in an uio struct.