Homework: OS Bugs

Read: Bugs as Deviant Behaviour paper. (See readings.)

Hand-In Procedure

You are to turn in this homework during lecture. Please write up your answers to the exercises below and hand them in to a 6.828 staff member at the beginning of lecture.

OS Invariants

In case, after all that OS hacking, you're wondering why 6.170 was a pre-requisite: here's a homework on invariants and specifications.

Operating system code must obey many rules for correctness and performance (e.g. check user pointers before using them in kernel mode). These rules aren't normally checked by compilers like `gcc` and Engler et al make an argument for using a meta-compiler to check the source code for system-specific rule violations.

Think about your JOS implementation. What are some unstated invariants in the JOS kernel?

Here are some simple invariants that should generally be true across all our implementations:

- Interrupts are disabled in kernel mode.
- All registers are saved and restored on a context switch.
- Application code is never executed with priority level (CPL) 0.
- Never write to a page on the free page list.
- If a page is on the free list, it is not mapped in any environment's address space below `KERNBASE`.
- The addresses for `UENVS`, `UPAGES` and `UVPT` are never mapped writable in any address space.
- There is only one `GDT` and only one `TSS`.

There are a large number of other invariants and rules that you could probably state about your specific JOS implementation.

Question Have you encountered any nasty bugs that would have been avoided if a metacompiler had alerted you that you were violating some rule or invariant you assumed your code was obeying?

[feel free to recount mind-numbing debugging marathons here]

This completes the homework.

In fact, this completes all homeworks!