Virtualization, System Calls, and Privileged Execution

Administrivia

- Announce grading guide for project #1
    - Subsequent projects will be graded similarly
- Coding (and time spent doing it) is not the most important part of this class, *concepts* are! (This is reflected in the class syllabus.)

Student Questions

- Answer questions on the project and the set of tools

Virtualization

Why virtualize (in general)? To create the illusion that a desirable aspect of the system is present when in fact it is missing. (Alternative definition: to create the illusion that a process has machines and resources at its disposal, when in fact it be sharing them or otherwise not have them instantly available.)

Virtualization raises the level of abstraction of (hardware) resources to a point where they are meaningful to the applications.

- Why virtualize the CPU?
    - To create the illusion of parallel execution
        - via timer interrupts and multiprogramming
- Why virtualize memory?
    - To create the illusion of "infinite" memory
        - regardless of the actual size of the physical memory on the machine
            - You wouldn't want to write your code that knows how much memory there is, since you then have to rewrite it every time more memory is added.
    - To create the illusion of each program having the memory at its disposal
        - … by providing a virtual address space with addresses between 0 and 2^{32}-1 whereas in fact those virtual addresses are mapped to physical ones that only the OS needs to know
- Why virtualize disks?
    - To abstract the details of how data is stored on disks (file systems, etc.)
    - To create the illusion of transparency of data access
        - … even if another process preempts using the file system in the course of processing the current request for data
- Why virtualize the network?
    - To create the illusion of proximity of resources
        - You request a resource and it (magically) shows up whether local or remote
    - To create the illusion of a reliable transfer (of bits)
        - … even when the underlying medium is error-prone
- Can we virtualize any other resource? Do we? What is the purpose? Examples? What is virtual in the real world around us? Car ignition systems, online purchases, non-face to face communication, even plant seed growing. Is there value in virtualizing?

- What does VMWare stand for and what does it do? What resource is it virtualizing? (Emulators virtualize entire architectures.)

System Calls

- Why are they necessary? Why are useful services protected (run by the kernel)?
- Some system calls you need to know about:
    - fork() "clones" the current address space; returns the child process IDs in the parent process and 0 in the child process (or –1 in case of error);
    - execv() replaces the current process image with a new one (including a new stack, code text, environment, etc.); executed in the child process;
    - wait() is executed in the parent process to wait for child termination (determined by its process ID);

Privilege bit, privileged mode

- Why are they necessary?
    - To isolate / distinguish the user programs from those of the kernel in order to protect sensitive common resources from accidental or malicious corruption
        - A parallel to how the government protects commons (air, water, transportation, etc.)
    - Certain systems may want to be able to control/intercept all inputs to a process so they can reproducibly re-run it later.
- Protection requires hardware support
    - Some early systems did not have such support (e.g., Intel 808x processors) and operating systems running on top of them (e.g., MS-DOS up to Windows 95) were vulnerable.
- Downsides of having protected mode
    - Crossing the user / system boundary has performance implications if done too often
        - Many modern systems try to move functionality into user space to avoid it, and use other mechanisms to protect one process from inflicting damage on other processes.

Quiz Question

- In Linux the first 64KB of every address space are protected, i.e., a user process cannot access them without that resulting in a segmentation fault. What is the rationale behind protecting those bytes? Why is that not a "silver bullet" still?