Processes and Threads – Distinctions and Motivations Behind Each

Questions

- Answering student questions

Reflections on the project

- What was the most difficult aspect of the project?
  - Finding where things are in the kernel? Making the changes? Getting it to finally work? Others?
- How did you find the Linux code structure? Easy to understand? Not so easy? Was it well componentized?
- Was the midnight turn-in time better than the 8:30am time just before lectures?

Processes vs. threads

- What are the main differences? (I was once asked this in a job interview.)
  - A thread (inside a given process) is uniquely defined by a stack, a program counter (PC), and a set of registers.
  - All threads in a process share the same code, heap, and static variable segments.
- What is lightweight about threads?
  - Switching between them doesn't require cache and TLB flushing, both of which are expensive operations taking the majority of the switching time.
    - A (process) context switching takes roughly ~1µs on modern machines.
    - It is an overhead that must be minimized / amortized across useful computations
    - Context shouldn't be switched too often, but the need for sharing and better resource utilization requires it (> 10,000 switches/sec)
  - Then why not have only threads and abandon process isolation?
    - There are legitimate reasons to isolate processes from one another.
      - E.g.: They don't need to (or, should not) share any data.
- Which of these applications are multithreaded and which are single-threaded? What motivates this?
  - Web servers
  - Web clients / browsers
  - Microsoft Word
  - Java Virtual Machine
  - your command interpreter
  - other applications

Communication models

- Shared memory – fast on the same machine (avoids kernel boundary crossings), but applications need to ensure protection and synchronization themselves (can't rely on the kernel)
- Message passing – for infrequent exchanges, communication across machines
- Where do threads fit?
  - They share memory and need to be protected from other threads.

The Java protection model

- Motivation behind it
  - All code lives in a single address space (with the virtual machine) in order to enable efficient sharing
    - ... between downloaded (mobile) code and local code
    - What is virtual about the Java Virtual Machine (JVM)?

    - Language-based (fine-grained) protection of resources at a level above that which the OS provides
        - There's more to it than type safety - it's a complex mechanism, similar to but less sophisticated than that in .NET.

Quiz Question

- How can one programmatically determine which way the stack grows – up or down? (Yet another job interview question.)

Other (stack-related) questions

- Does it make sense to more efficiently allocate the memory for thread stacks when there are multiple threads? What would the cost be?
    - Principle: “Make common case fast, and the uncommon case correct.”
        - Also applies for the issue of threads vs. processes (if sharing is essential and common)