control and concurrency

unix weapons school

Local concurrency

Of two "schools" of concurrency, we will focus on the first:

Shared memory	Operate on a common store, establishing mutual exclusion via control objects.
Message passing	Operate on disjoint stores, exchanging messages containing common data.

NB: Message passing is easily implemented in terms of shared memory, requiring a simple class. The converse is untrue[1], since memory operations are already built into the language.

[1] In C, anyway. In C++, we can do some operator overloading tricks—see Andrei Alexandrescu's "Modern C++ Design" (2001).

Necessities of fine-grained locking

What hardware primitives are required for scalable locking[2]?

- *Infinite consensus number*—For a concurrent object X and r/w registers W, the largest n for which there exists a consensus protocol $\{P_1 \dots P_n\}$ on W, X. The number is infinite if no such n exists. Herlihy taxonimized these:

Consensus number	**Object**
1	Atomic r/w registers
2	Test+set, Fetch+add
$2n-2$	n-register assignment
∞	Compare+swap/LLSC, FIFO+peek

- *Linearizability*—**FIXME**

[2]Material for this slide is due Lorenzo Alvisi's Fall 2004 CS380D "Distributed Computing" slides at UT-Austin.

FIXME about 6 more pages...

Recommended reading

- Edsger Dijkstra. "Communicating Sequential Processes" (1968).
- Maurice Herlihy. "Impossibility and universality results for wait-free synchronization" (1988).
- "Volatile Considered Harmful." Linux kernel documentation.
- Fraser and Harris. "Concurrent Programming Without Locks" (2007) and/or Fraser, "Practical Lock-Freedom" (2003).
- Ulrich Drepper. "Futexes are Tricky" (2011).
- Desnoyers et al. "User-Level Implementations of RCU" (2012).
- Hans Boehm. "Miscompiling Programs with "Benign" Data Races" (2011).
- Michael Chynowe. "Implementing Scalable Atomic Locks for Multicore Intel EM64T and IA32 Architectures" (2012).
- Kogan and Petrank, "Wait-free queues with multiple enqueuers and dequeuers" (2011).
- Culler and Singh. *Parallel Computer Architecture* (1999).
- *Intel 64 and IA-32 Architectures Software Developer's Manuals*.

Hack on!

- "A river made up of the events which happen, a violent stream. As soon as a thing has been seen, it is carried away. Another comes in its place; this too will be carried away."

  —Marcus Aurelius

- "Time keeps everything from happening all at once."

  —Henri Bergson

- "We shall come to see that time does not exist."

  —Julian Barbour