Inversion of Control and Event Loops

July 17, 2017

Homework 5

Homework 5

- Deadline extended to Wednesday, July 19
- Chance to discuss
  - Regular expressions
  - Parsing strategies
  - Useful techniques

Example Approach

1. Preprocess input to handle known problems

    1.1. Abbreviations

    1.2. Nested sentences

2. Iteratively identify longest possible strings

3. Post-process result to undo transformations

Event Loops and Inversion of Control

Responsive Programming

- We want programs to respond quickly
  - Web Services
  - GUIs
- Sometimes, responsiveness > speed

servers/BlockingServer.java –>

I/O Latency

- L1: 3 cycles
- L2: 14 cycles
- RAM: 250 cycles
- DISK: 41,000,000 cycles
- NETWORK: 240,000,000 cycles

Slide by Ryan Dahl,
https://s3.amazonaws.com/four.livejournal/20091117/jsconf.pdf

Responsiveness Issues

- Waiting (blocking) kills responsiveness
- Identify where the program waits
- Allow it to do other things

What "Blocks" / "Waits"?

Blocking Operations

- sleep()
- read()
- write()
- connect() / accept() / etc

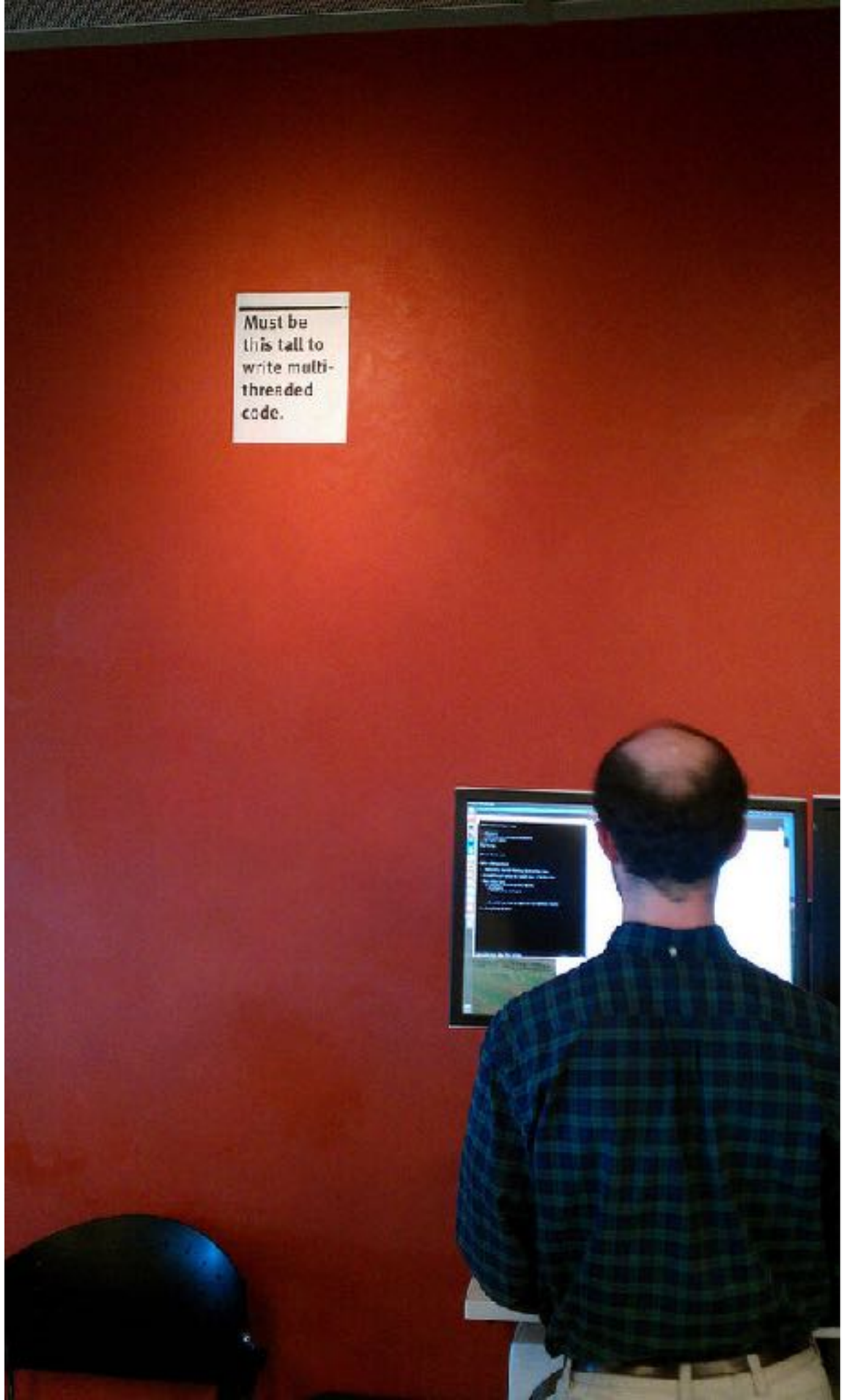
Responsiveness, Step 1

- Threading
- 1 thread for each request
- Kernel switches execution for us

servers/ThreadedServer.java –>

Threading Issues?

- Threads scale, but not infinitely
- More overhead in leaning on the kernel
- around 1-10k threads, and you'll have problems
  - Stack memory
  - Switching overhead


Must be
this tall to
write multi-
threaded
code.

10k Threads, Really?

- Popular web services
  - 100k-10m concurrent users
- GUIs
  - User events
  - Timers
  - Application events

Insight

- Threads create a lot of "work" just to wait
  - Stack memory
  - Context information
- More intelligent way to wait?

Event Loop

Event Loop

- Invert Control
- Register Handlers
- Have system switch during anything that blocks / waits

Inversion of Control

- Also know as "async" programming
- Register code that happens on events
- Have system call that code when that event happens
- Until it happens, keep doing other things

inversion-of-control/Read{Blocking, Async}.java

->

Pattern in Async / Event Loops

- Register for events
- Wait for the system to call our code
- Rely on the system to do all the scheduling for us

servers/AsyncServer.java –>

GUI Applications

- Generally written using Async
- Register for events (onClick, onAppear, onScroll)
- Wait for the system to call us

android-BasicNetworking/* –>