Concertant LLP

Them Threads, Them Threads, Them Useless Threads

Dr Russel Winder

Partner, Concertant LLP

russel.winder@concertant.com

Copyright © 2008 Russel Winder

Aims and Objectives of the Session

- Raise awareness that threads are the problem as much as the solution.
- Show that threads are a low-level language implementation tool not an application programming tool.
- Show that higher-level abstractions are needed to manage parallelism in the brave, new, multicore world.

Copyright © 2008 Russel Winder

Structure of the Session

- Look at the threads support in Java, C++, and C.
- Look at some of the most obvious problems.
- Look at how Erlang and Occam deal with this.
- Tentatively, propose a new next layer model for Java, C++, and C.

Copyright © 2008 Russel Winder

On Expertise In Programming

Dr Russel Winder

Partner, Concertant LLP

russel.winder@concertant.com

Copyright © 2008 Russel Winder

Aims and Objectives

- To convince people that learning and using multiple programming paradigms is the road to improved competence.

Copyright © 2008 Russel Winder

On Expertise

- Expertise is *not* the number of years of experience in a given language – though that is a factor.
- Expertise is strongly related to the number of different types of language a person is fluent in.

Learning and being able to use C, C++, Java, Python, Ruby, Groovy, Fortran, Haskell, Erlang, etc. ***properly*** *makes you a better programmer.*

Copyright © 2008 Russel Winder

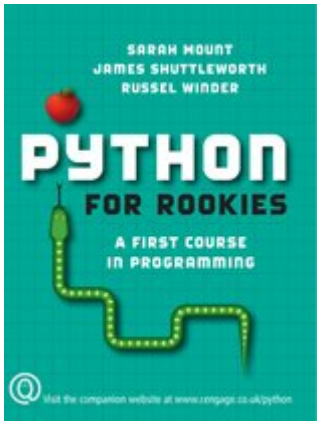
Subliminal Advertising

Python for Rookies

Sarah Mount, James Shuttleworth and

Russel Winder

Thomson Learning



Now called Cengage Learning.

Learners of Python need this book.

Copyright © 2008 Russel Winder

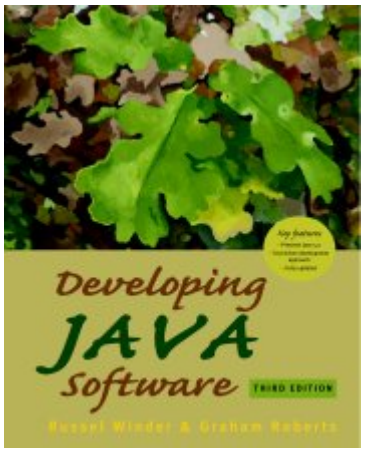
More Subliminal Advertising

Developing Java Software

Third Edition

Russel Winder and Graham Roberts

Wiley



Learners of Java need this book.

Copyright © 2008 Russel Winder

Anti Advertising

Developing C++ Software

Second Edition

Russel Winder

Wiley

Only buy this book is you are studying the history of C++ and how it was taught.

Learners of C++ used to need this book.

Copyright © 2008 Russel Winder

My Camera and Me

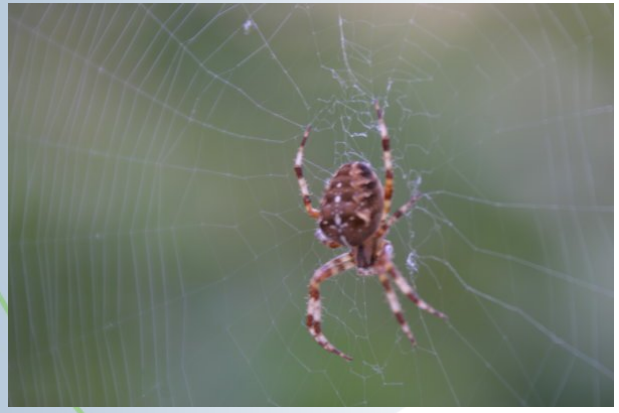
Dr Russel Winder

Partner, Concertant LLP

russel.winder@concertant.com

Copyright © 2008 Russel Winder

Some Photos I Have Taken



Copyright © 2008 Russel Winder

The Keywords

- Multiprogramming
- Multitasking
- Multithreading
- Concurrency
- Multiprocessing
- Parallelism

- SISD
- SIMD
- MISD
- MIMD

- Tightly coupled
- Loosely coupled

- Data parallel

- Multicore
- Cluster

- Livelock
- Deadlock

Copyright © 2008 Russel Winder

A Bit of History
In the Beginning

- Computer hardware was expensive and hard to find; maximizing utilization was critical.
- Multitasking (multiple concurrent processes) was crucial for maximizing availability and utilization.
- Virtualization was flirted with, but it did not become mainstream – though recently it has become de rigeuer.

Copyright © 2008 Russel Winder

A Bit of History
The Early Middle Period

- Tasks/processes seen as too heavyweight.
- Lightweight processes, aka threads, introduced.
- Issues:
  - Tasks/processes had hardware and operating system support, threads did not.
  - Tasks/processes have separate memory; threads are a shared memory approach.

Copyright © 2008 Russel Winder

A Bit of History
The Late Middle Period

- Sun LWP
- PThreads
- ???

Copyright © 2008 Russel Winder

A Bit of History
Modern Times

- Java integrates threads in the core model.
  - One model makes programming more consistent.
  - Monitors as well as locks are an integral part of the language.
- C++ has no standard so the plethora remains.
  - C++0x shows that the C++ committee consider threads need standardizing.
  - Basically PThreads.
  - What happened to Boost.Thread?
  - No integration of higher-level mechanisms.

Copyright © 2008 Russel Winder

Quick Quiz

- Who said:

  "Multithreaded programming is fraught with many challenges, and can rightly be considered something that the majority of programmers should steer clear of."

Copyright © 2008 Russel Winder

Consequences

- All computers are parallel processors.
- Multicore processors and multiprocessor systems offer threads as the mechanism of control.

- The majority of programmers should steer clear of computers.

- Only write single-threaded programs.

Copyright © 2008 Russel Winder

Thread Safety

- Java really brought thread-based programming to the masses.
- C and C++ had threads but they were an add on.
- Java brought "thread safety" front and centre:
  - Library classes made thread safe.
  - Programmers taught to think about concurrency and thread safety.
  - Thread safety kills performance of single-threaded applications.

Copyright © 2008 Russel Winder

The StringBuffer Problem

- StringBuffer is a thread-safe class (well as much as possible).
- It is heavyweight and slow.
- String manipulation is at the core of everything.
- StringBuilder is a necessity.

Is there "too much thread safety"?

No just inappropriate thread safety.

Copyright © 2008 Russel Winder

Collections as an Example

- The basic data structures are not thread safe, they are as fast as possible in a single-threaded context.
- Thread safety is achieved through added adapters.

Collection.synchronizedSet

Copyright © 2008 Russel Winder

Collections as Anti-Pattern?

public static <T> Set<T> synchronizedSet(Set<T> s)
Returns a synchronized (thread-safe) set backed by the specified set. In order to guarantee serial access, it is critical that all access to the backing set is accomplished through the returned set.
It is imperative that the user manually synchronize on the returned set when iterating over it:

```
 Set s = Collections.synchronizedSet(new HashSet());
     ...
 synchronized(s) {
     Iterator i = s.iterator(); // Must be in the synchronized block
     while (i.hasNext())
         foo(i.next());
 }
```

Failure to follow this advice may result in non-deterministic behavior.
The returned set will be serializable if the specified set is serializable.
Parameters:
s - the set to be "wrapped" in a synchronized set.
Returns:
a synchronized view of the specified set.

From Sun's API manual

Copyright © 2008 Russel Winder

Java Util Concurrent

- Does ConcurrentSkipListSet solve the problem?

Possibly, but probably not.

Copyright © 2008 Russel Winder

Dissention

- Concurrency in Java is hard:
  - Threads is the biggest problem in all Java training.
  - java.util.concurrency still doesn't make it easy enough.
- Occam and Erlang never bought into the threads model for programming:
  - Message passing is the only model.
  - No synchronization because there is no shared memory.

Copyright © 2008 Russel Winder

The Problems of Threads

- Shared memory.
- Flow control.
- Shared memory.
- Flow control.
- Shared memory.
- Flow control.
- Shared memory.

- Flow control.
- Shared memory.
- Flow control.
- Shared memory.
- Flow control.
- Share memory.
- Flow control.

Copyright © 2008 Russel Winder

Event-driven Programming

- Event loop
- Callbacks/even handlers to deal with things.
- Standard architecture for GUI and networking.
  - Swing/AWT
  - Gnome
  - Twisted Internet

Does this solve the problem – no.

Copyright © 2008 Russel Winder

Taxonomy Of Parallelism

- Shared memory.
- Distributed Memory:
  - Slow communications:
    - errors likely.
  - Fast communications:
    - errors unlikely.

Copyright © 2008 Russel Winder

Cluster Parallelism

- Processes on separate computers.
- Communication by message passing.

- PVM
- MPI
- Erlang

Copyright © 2008 Russel Winder

Multiprocessor Parallelism

- Single system.
  - Shared memory.
  - Bus-based message passing.
- Processes/IPC
- Threads
- Message passing:
  - Occam
  - Erlang

Copyright © 2008 Russel Winder

What To Do

- Refuse to use multiple threads.
- Never use shared mutable state.
- Always use message passing.

Put a layer on Java for doing message passing.

Something pairing with java.util.concurrent.

Copyright © 2008 Russel Winder

What have the HPC People Done?

- OpenMP
  - Data parallelism.
  - Fine-grain parallelism.
  - Thread implemented.
- MPI:
  - Message passing.
  - Cluster-level parallelism.
  - Process implemented.

Copyright © 2008 Russel Winder

Summary

- Threads are needed but they are not a programmer tool.
- Why bother with an explicit threads API when OpenMP is available?
- Why not use a language that supports parallelism directly:
  - Erlang
  - Haskell
  - Occam

Copyright © 2008 Russel Winder