Lecture 11 — Multithreading

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

December 29, 2014

Objective

Goal: a brief introduction to multithreading

About Multithreading

First definition: a Process.

Process has three main components:

- An executable program,
- Data created/needed by the program, and
- Execution context of the program

A process has at least one Thread.

About Multithreading

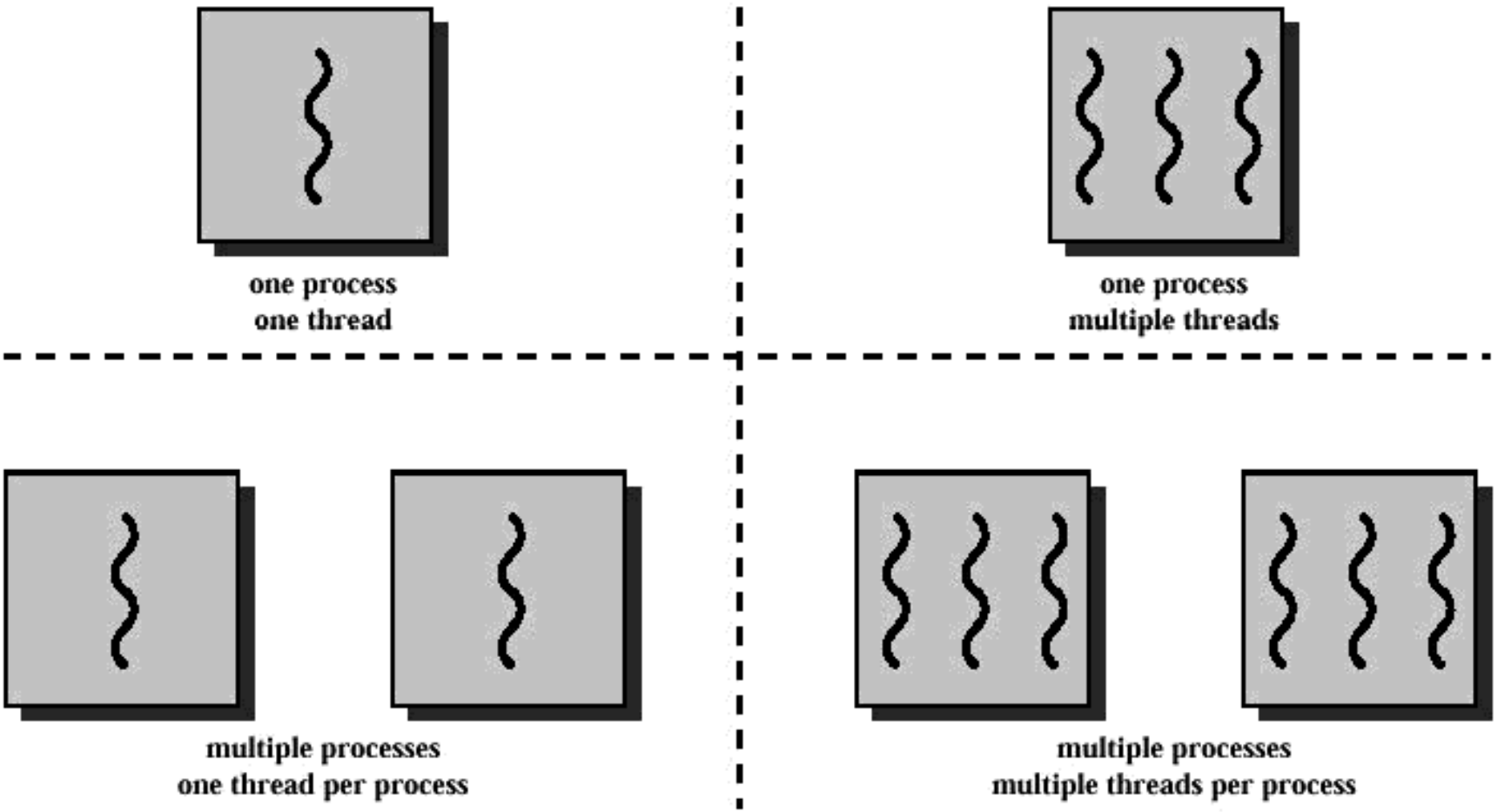
Thread: A short form of Thread of Execution.

A sequence of executable commands that can be scheduled to run on the CPU.

A multithreaded program uses more than one thread at least some of the time.

Threads may be created and destroyed dynamically.

Processes and Threads



Thread States

Conceptually, a thread can be in one of three states:

- **Executing**
- **Ready**
- **Blocked**

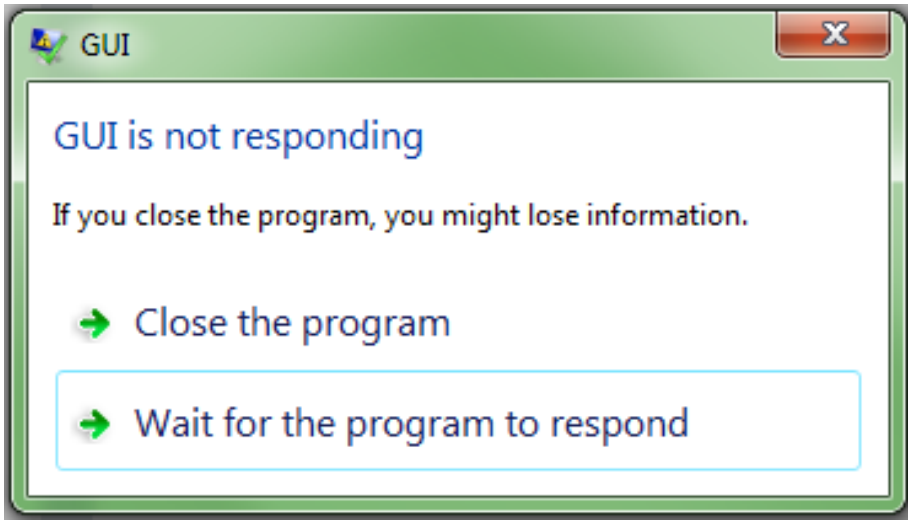
About Multithreading

Very few programs written today are single threaded.

It is typical to separate UI from processing.

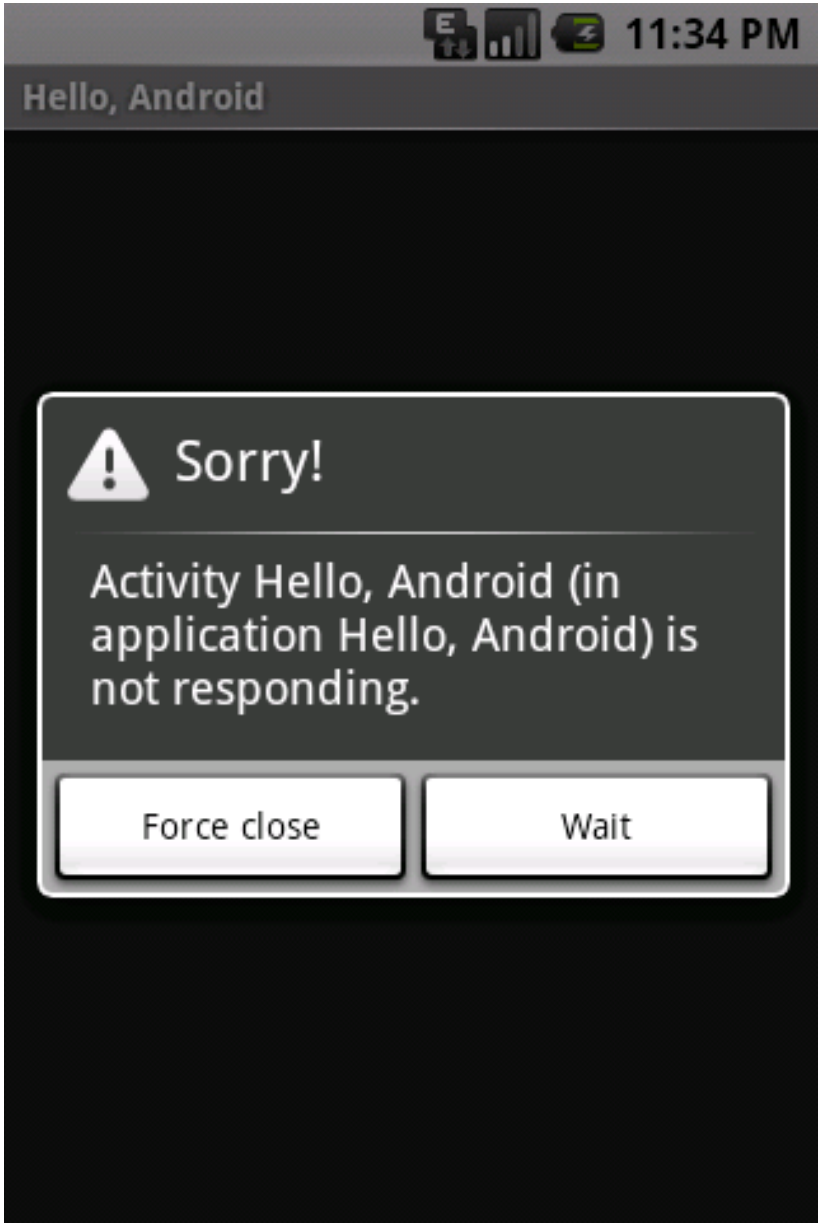
Example: File Transfer (FTP) program.

Not Responding



(From StackOverflow.com)

Not Responding



(From linuxtopia.org)

Multithreading: FTP

Solution? Multithreading

When an upload is ready to start, a new thread is created.

Thread runs in the background.

UI Thread does not wait for the upload.

Android official guide says to use this.

Inter-Process Communication

In most modern OSes, each process runs as if it's in its own world.

If this was not the case, processes could read memory of other processes!

Can still happen in embedded systems.

Inter-Process Communication

Because of the “walls” between processes, communication is hard.

Alternative: communicate between multiple threads!

No enforcement of rules by the OS between threads.

Inter-Process Communication

Alternative: Inter-Process Communication (IPC)

Used for data sharing, message passing, function calls.

Can be set up with files, shared memory, etc.

Android IPC

Intents: used to ask another process to do something.

Explicit: Specify what process should handle the Intent.

Implicit: Ask OS to find programs that can handle the Intent.

If there is more than one, user gets a popup asking to choose.

Example: YouTube link can open in Browser or YouTube App.

Intent is one way, but there is an alternative.

Binder: client-server model for comunication.

Details of the Binder are beyond the scope of this course.

Processes vs. Threads

Threads ware faster to create and destroy than processes.

Less time to switch between threds in the same process.

Threads within the same process share memory and files.
Can communicate easily.

Processes vs. Threads

No protection between threads in the same process.

If any thread encounters an error, whole process will be halted;
Independent processes can keep going if a related one stops.

Multithreading: Execution

Assume a computer with 1 processor; 1 thread at a time

Still support multiple threads.

Time division: task switching.

Picture 3 threads.

User perception: threads executing in parallel

Multithreading: Execution

Now: desktops, laptops, cell phones: multicore processors.

Multicore = multiple threads executing at once.

Time slicing still occurs if needed.

Multithreading

Two kinds of multithreading: Co-operative and Pre-emptive

Co-operative Multithreading

Used to be fairly common - even in Mac OS 9.

Threads yield the CPU.

Problem: greedy threads (never yield).

Pre-Emptive Multithreading

Solution: let the OS decide when thread switch.

Every thread acts as if it's the only one.

No need to decide when is a good time to yield.

OS forces a thread switch when it is time.

Pre-Emptive vs Co-operative

Usually you will see Pre-Emptive.

Embedded systems might not manage threads; might have to use co-operative.

Co-operative an be more efficient if everyone plays nicely.

However, one rogue thread can wreck it for everyone.

Parallelism

Multiple threads at the same time = tasks completed faster?

Depends on the nature of the task!

Fully paralellized: $2 \times$ Threads = $2 \times$ Speed

Partially paralellized: $2 \times$ Threads = $(1 < n < 2) \times$ Speed

Cannot be paralellized: $2 \times$ Threads = $1 \times$ Speed

Multithreading Bugs

Challenging to convert single threaded code to multithreaded.

Multithreaded programs are prone to new types of bugs.

Multithreaded bugs will come up later in the course, but we'll take time for a short preview.

Debugging Parallel Programming

Let's imagine we have an instance of an object `Location` that has two co-ordinates, `x` and `y`.

```
location.setX(5);    location.setX(10);
location.setY(7);    location.setY(0);
```

Location Example

Even if each method is atomic, we cannot guarantee an order.

Consider this order:

1. Set x to 5
2. Set x to 10
3. Set y to 0
4. Set y to 7

Result: $(10, 7)$ - Inconsistent!

We'll see how to solve this when we get to debugging.

Example: RaceCondition

Now we'll examine in Eclipse some code examples where we have a race condition.

(See the code examples folder in Learn for the source.)