Chapter 4. Concurrency

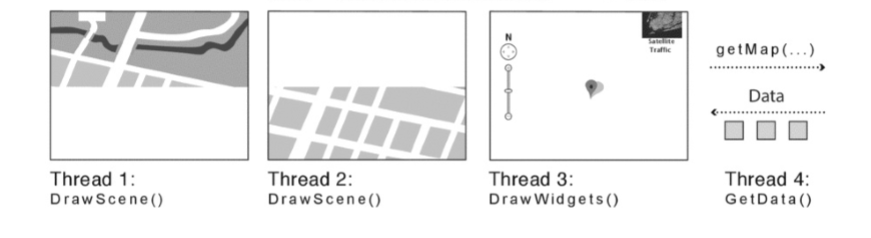
*Concurrency*: multiple activities that can happen at the same time.

Both a challenge for the OS developer and the apps developer.

*Thread abstraction*: the illusion that we have nearly infinite number of virtual processors, while the physical hardware is more limited.

# 4.1 Thread use cases

The idea with the thread abstraction is that in a program we can represent each concurrent task as a thread. Traditional programming can be considered as singe-threaded, while a program that is multi-threaded have each task and sequence in its own thread.



One use case with multi-threaded programming. Drawing the map with two different threads, and giving the UI its own thread, as with the data fetching with has its own thread, makes the program run faster and more responsive.

**4.1.1 Four Reasons to use Threads**

1. Program structure: expressing logically concurrent tasks.

* Programs often simulate real-world apps that have concurrent activities.
* Each concurrent task is expressed as a separate thread.
* Server example, it needs to handle many users at the same time, cannot wait for each task to finish. Therefore servers often give each user/task its own thread.

2. Responsiveness: shifting work to run in the background.

* By giving the UI its own thread, it can still function while other threads are performing work in the background.
* E.g. the cancel button should always work even if the downloaded page is gigantic.
* The OS kernel also uses threads. E.g. when writing to a file, it uses a separated thread to flush out the data to the disk. This way it preserves its responsiveness.

3.Performance: exploiting multiple processor

* Programs can use threads on a multiprocessor to do work in parallel.
* Number of threads do not need to match number of processors since we the abstraction.

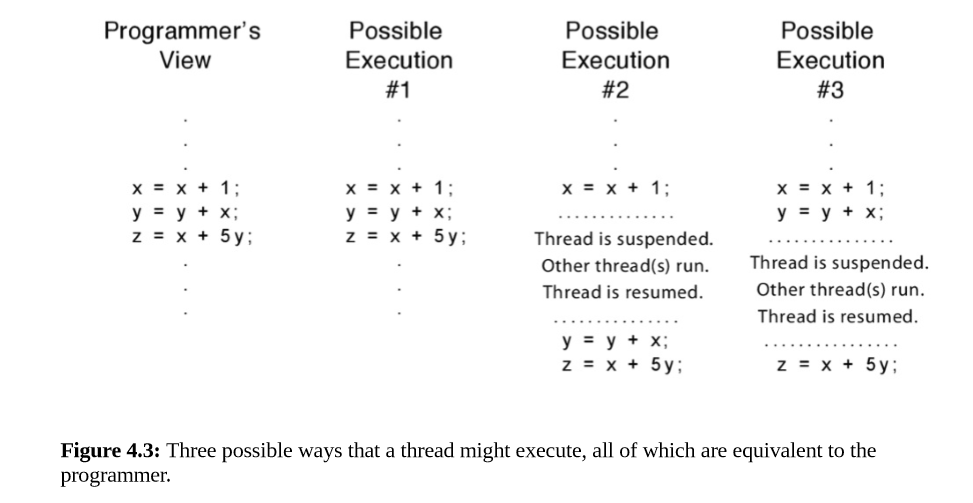
4. Performance: managing I/O devices

* Computers must often interact with the outside world via I/O to do useful work.
* Processors are must faster than I/O therefore it is much more efficient to use a thread for I/O.
* The processor can perform other instructions while the I/O thread is waiting for data.

# 4.2 Thread Abstraction

**4.2.1 Running, Suspending and Resuming Threads**

To implement the illusion of thread abstraction, the OS include a thread scheduler, that can switch between threads that are running and those that are ready but not running.



And not reading lessius abstract away you know this formality so we can assume that there are certain number of professors just the same as well see with believe that it's kind of unlimited and very fast memory when in fact it's gonna fix to provide that we just I'm bit so we have this program at traction we have another task want to do in in send it and we can just assume that those are getting run we don't actually know which process is there getting run on but we assume like sort of that each thread has its own process we don't really need to think about the details below that but the physical reality is that um we have you know maybe 2 processes physically in the system and so they operating system can schedule to read that they can run at the same time almost to process just waiting to run and I need to finish their time is expired wilc will contact switch name a process