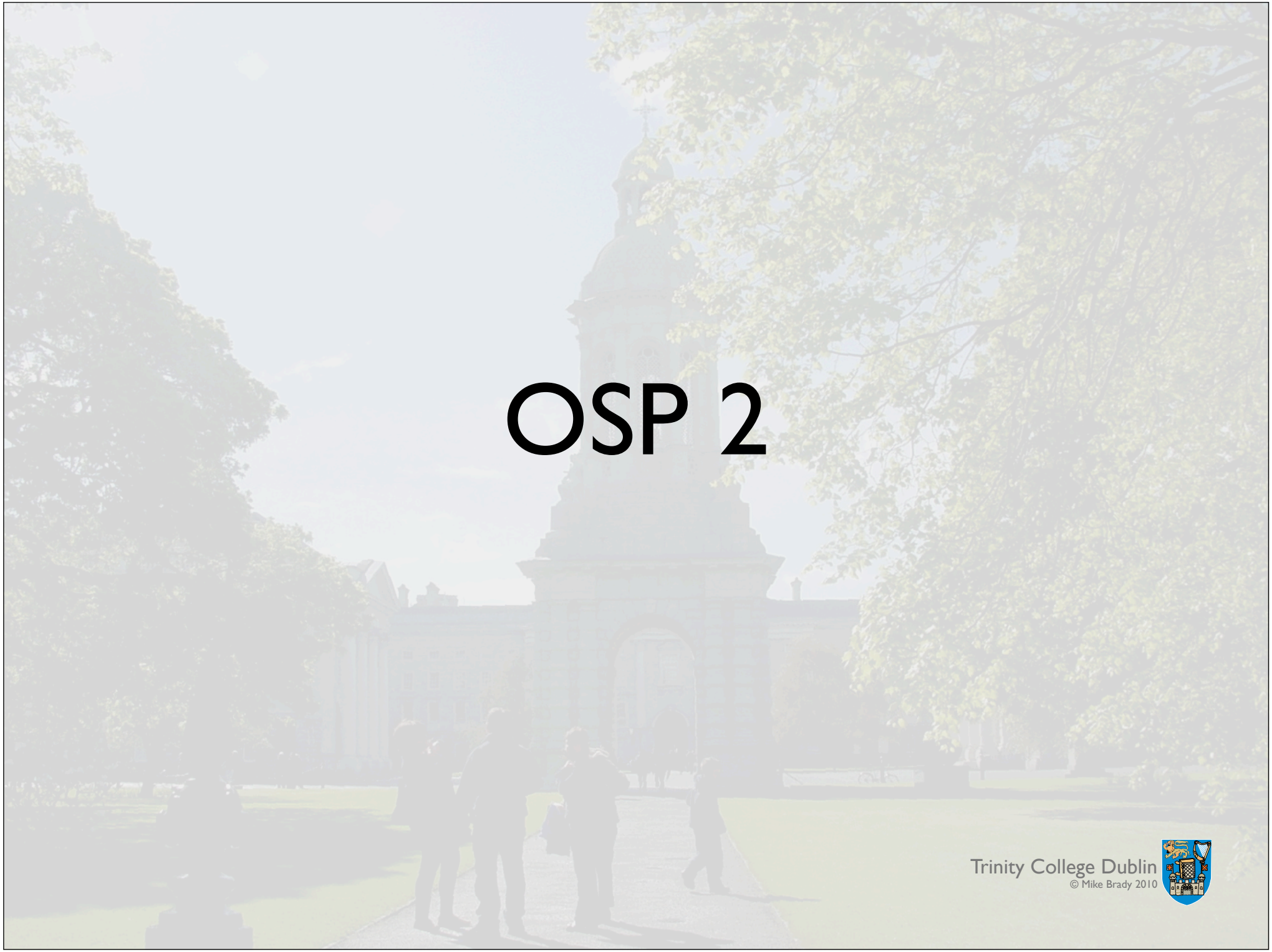
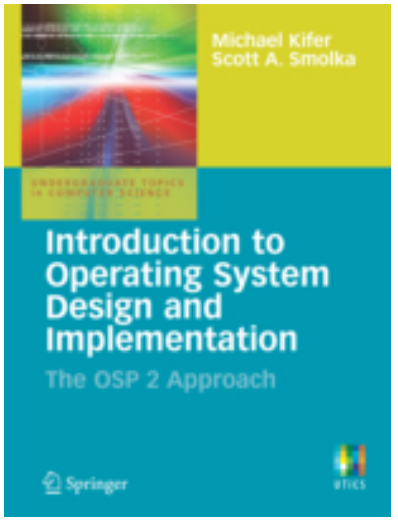
OSP 2



Trinity College Dublin

OSP 2 – A Real Operating System

- OSP 2 is an operating system.
  - Written in Java
  - Designed for teaching
  - Modules for basic OS Services
  - http://www.springer.com/computer/swe/book/978-1-84628-842-5
  - The book is a manual for OSP 2. Not too bad.



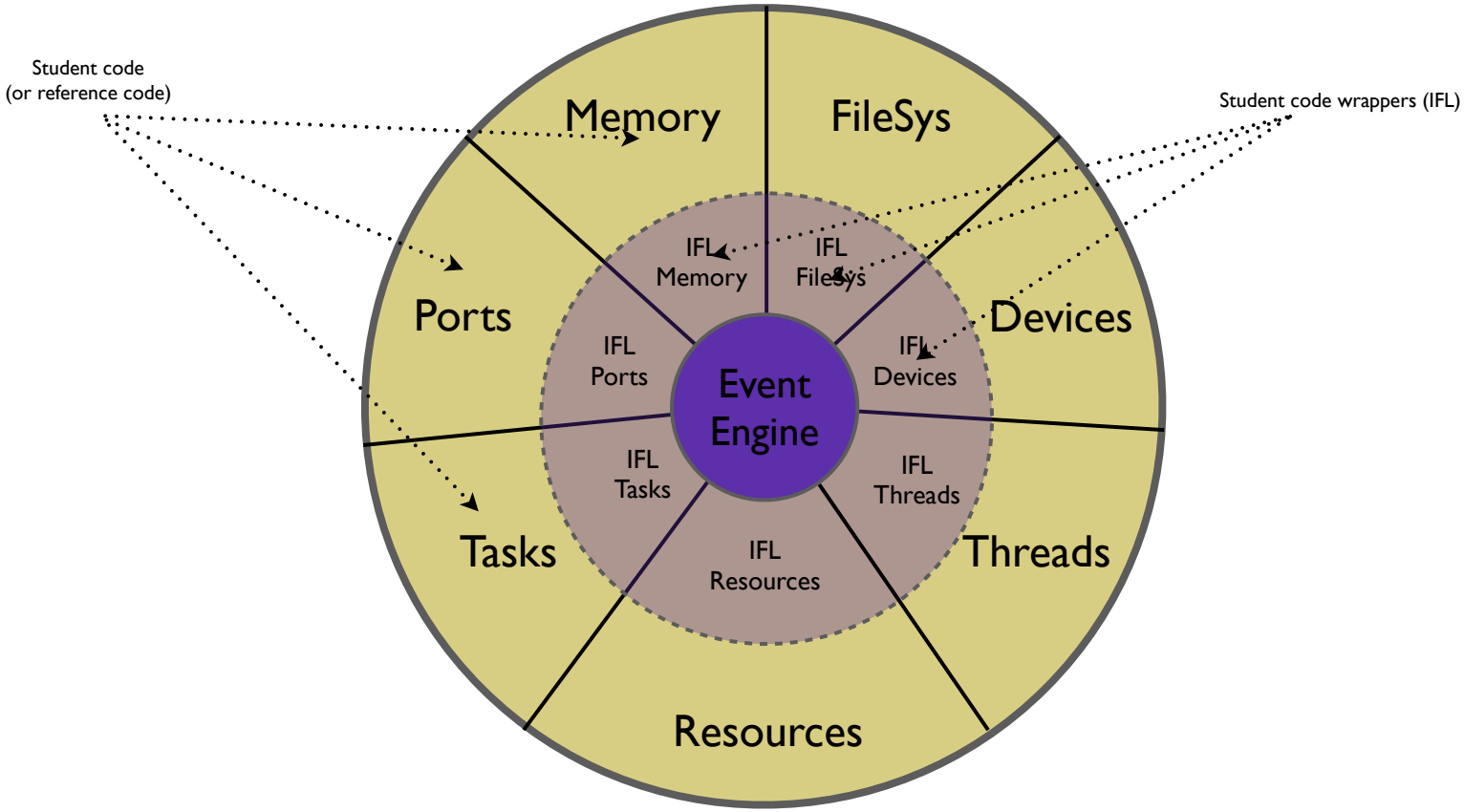
OSP 2 – Simulated User Programs

- User processes – clients of the OS – are simulated.
- The simulator's "event engine" generates events that the OS responds to as if they were real events generated by user programs. Simulation parameters are adjustable.
- Events are passed to the OS through an Interface Layer (IFL).
  - The IFL can monitor results for obvious bugs
  - The IFL can gather statistics.

OSP 2 – Simulated Hardware

- CPU
- Disk
- System Clock
- Hardware Timer
- Interrupt Vector

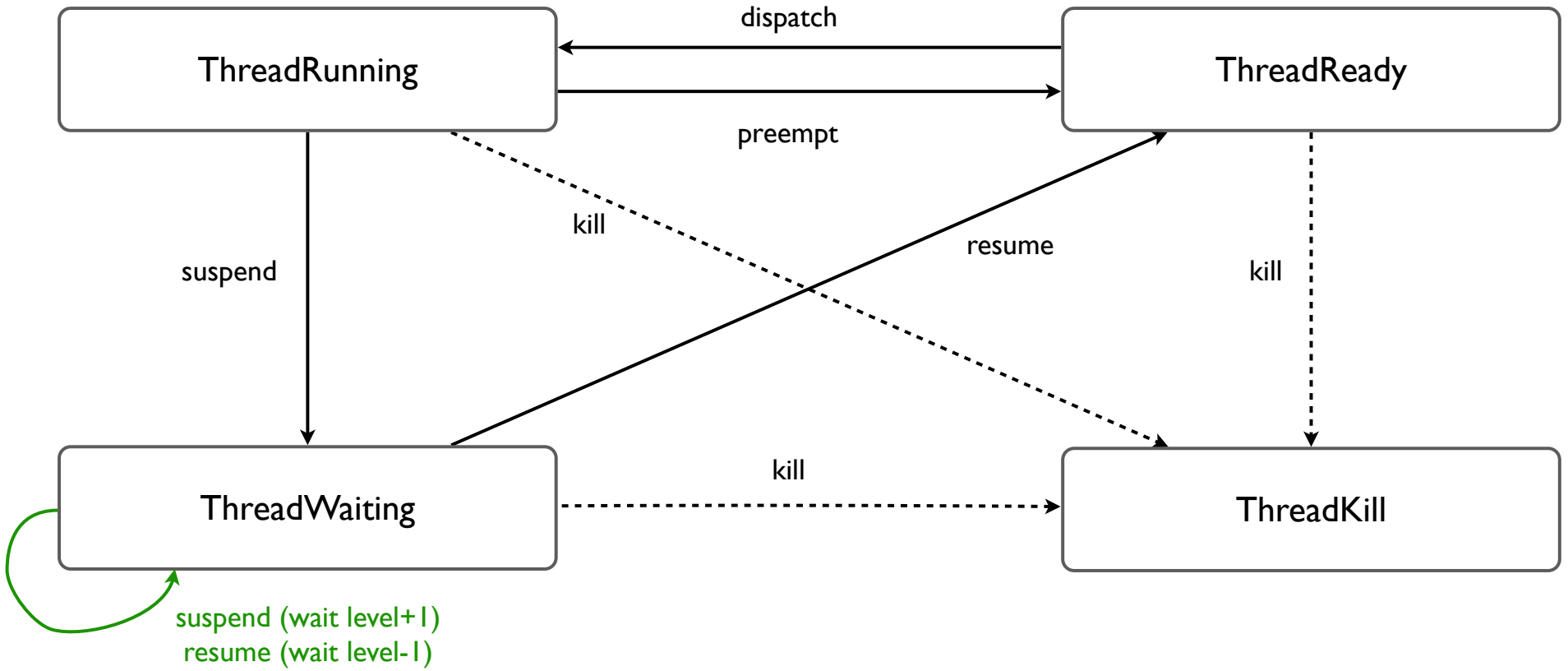
Logical Structure of OSP2



Student Packages

- Two versions of each "Student Package"
- Reference Code implements a package's required functionality
- Student Packages leave some methods empty, for the student to develop.
- You can develop code for one package and use the reference code for all the other packages.
- Thus, you can experiment with aspects of a real OS.

Example – Threads



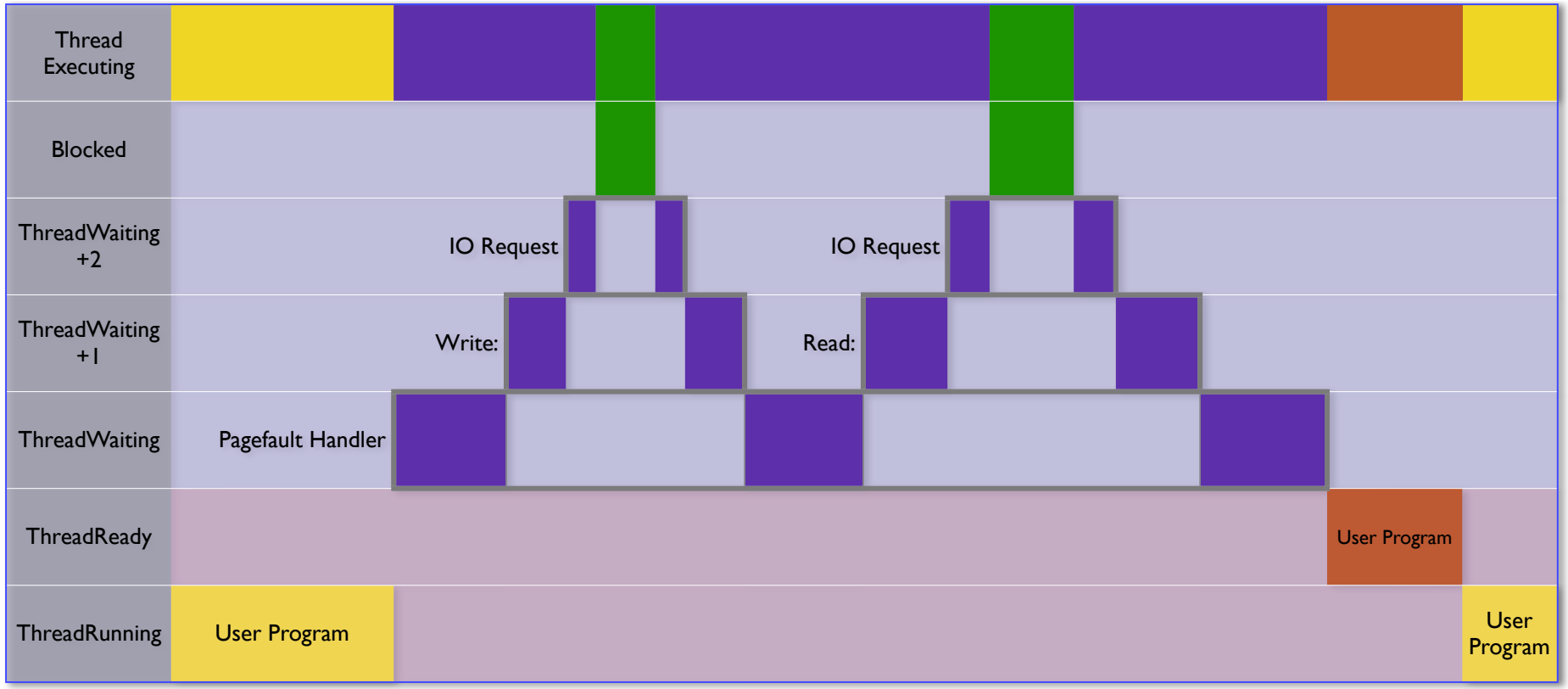
ThreadWaiting!

- Not quite what you might expect!
  - When a user thread enters ThreadWaiting, it doesn't stop executing!
  - Instead, it becomes a system thread, and will do the work requested.
  - So, what "waiting" means in ThreadWaiting is that the user program is waiting for a page fault or a system call to complete.
  - If a ThreadWaiting thread itself makes a system call, its state becomes ThreadWaiting+1, and so on…

ThreadWaiting

- ThreadWaiting states start at Waiting Level 0 and go upwards when further (nested) system calls or pagefaults occur…
- …and when those calls or pagefaults are finished, the Waiting Level drops.
- At Waiting Level 0, a resume sets the Thread Status to ThreadReady and it should be put into the pool of user threads ready to execute.

Example – Dirty Page Fault



Note: Blocked Section not to scale: usually blocked time is ***much*** larger.

More about the Scheduler

- Upwards movements are made by suspend events
- Downward movements by resume events
- The system [seems to] always prioritize threads in any ThreadWaiting state over any user level threads.
- When a dispatch() is made in the system, it seems to dispatch ThreadWaiting-state threads before allowing the do_dispatch() to run (?).
- Thus, only user-level threads can be scheduled by us.