



THE UNIVERSITY *of* EDINBURGH
informatics

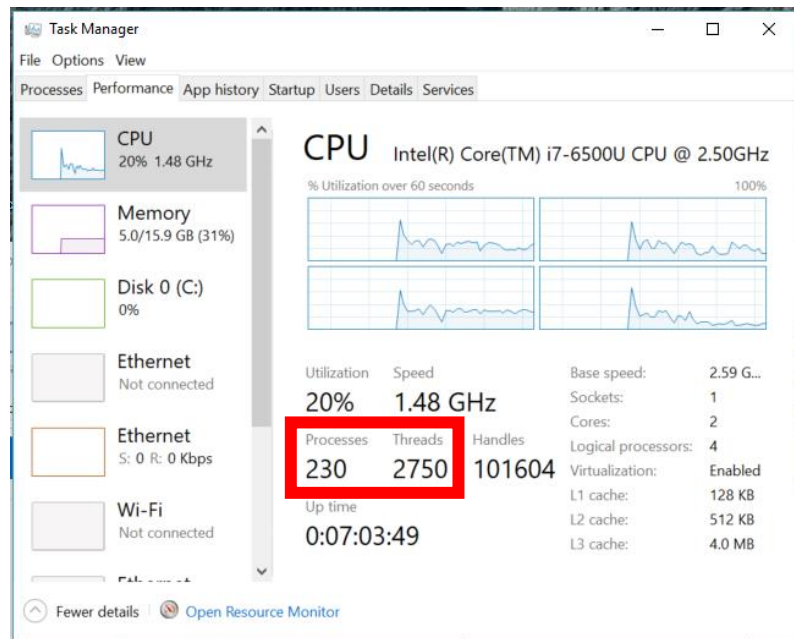
Operating Systems (INFR10079) 2023/2024 Semester 2

Scheduling (Basics)

abarbala@inf.ed.ac.uk

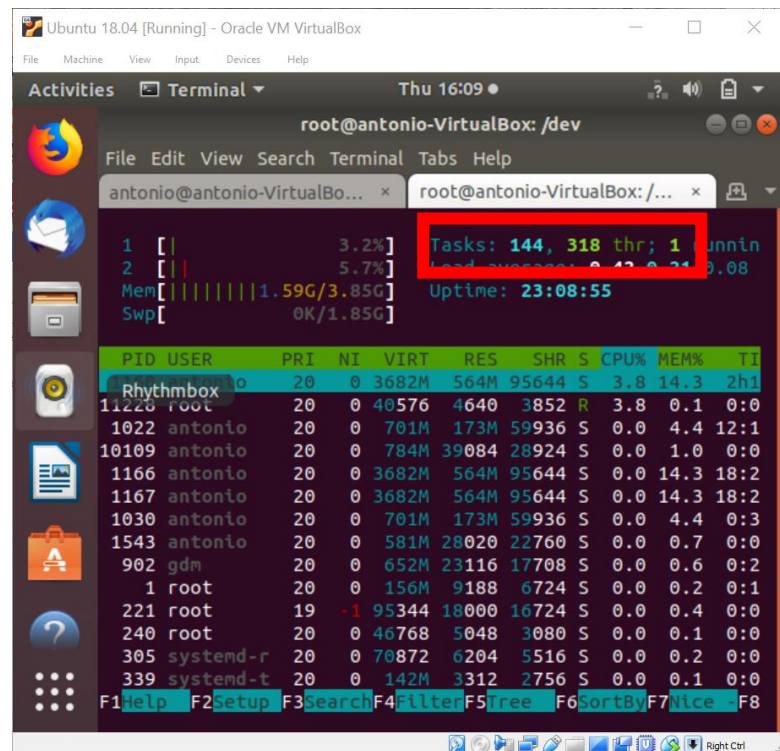
Chapter 5.1, 5.2

How Many Processes or Threads Are Running On My Computer?



Window's Task Manager

UNIX's htop (or top)

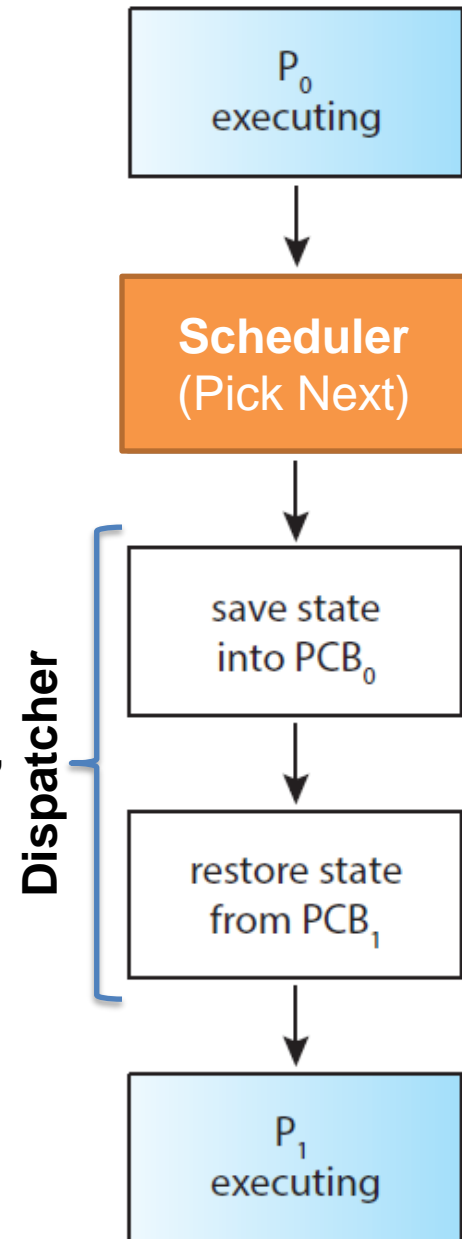


Who will run next?

Scheduling

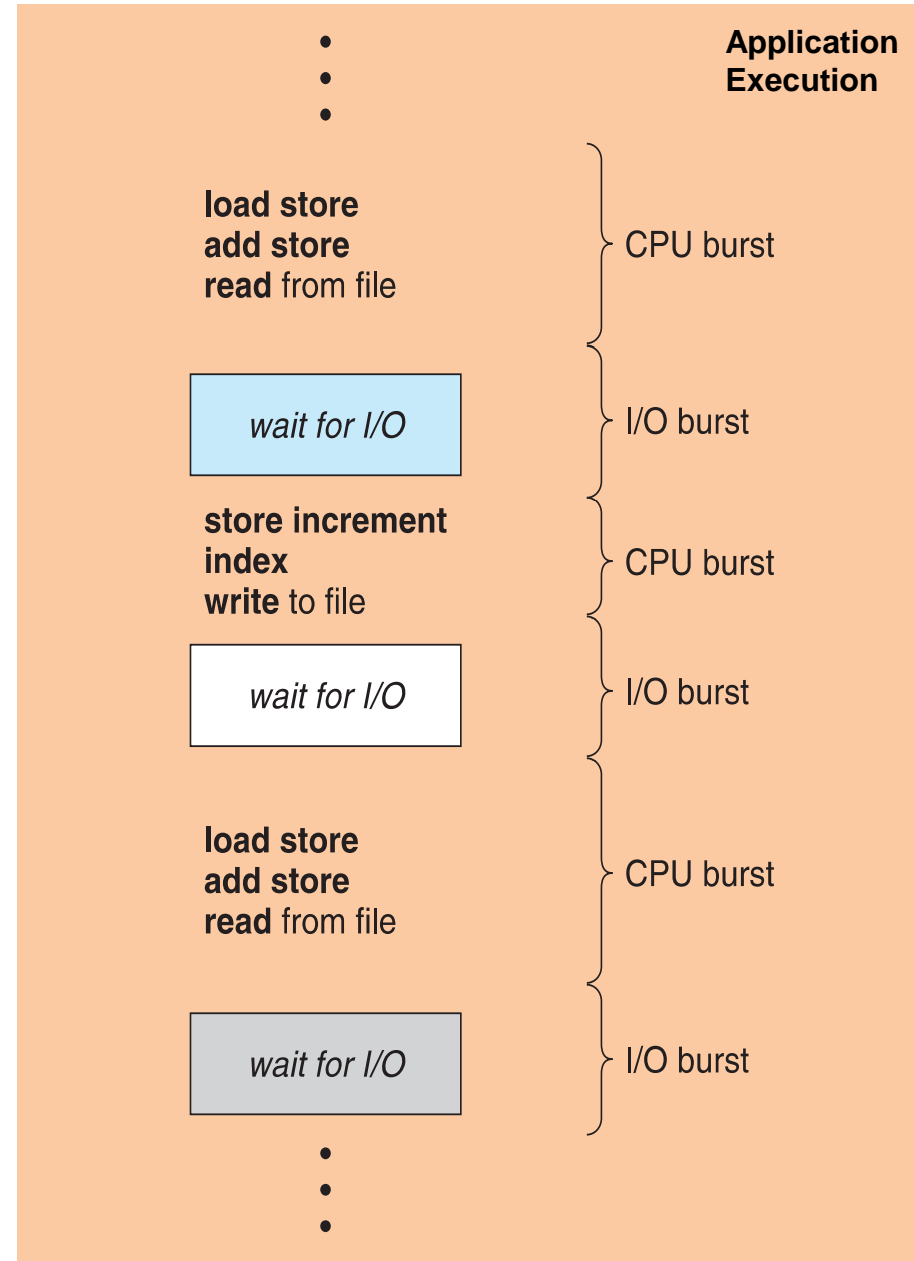
Single CPU/core

- Decision of who will **run next**
 - Process or thread
 - **In kernel for proc and kernel-level thr** (slideset' focus)
- **Pick among** the ones in ready queue(s)
- *When?*
 - Potentially, **all times we switch** to the OS
 - Interrupt (device completion, timer interrupt, etc.)
 - Syscall (including voluntary process/thread termination, or yield)
 - Exception (including involuntary termination)
- *How?*
 - **Scheduler** decides
 - **Policy** (implemented by an algorithm)
 - Task switched by **dispatcher**
 - **Mechanism** (kernel or user code)



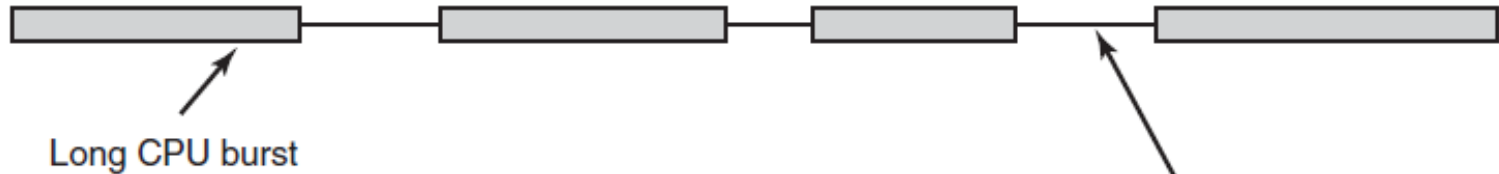
Process or Thread Behavior #1

- Process or thread execution consists of cycles of
 - CPU execution: **CPU burst**
 - I/O wait: **IO burst**
- **CPU bursts** distribution is **application dependent**
- **Maximum CPU utilization** with multiprogramming
 - Don't leave CPU idle
- While a process or thread is waiting for IO **another can run** on the CPU
 - Focus on **single** CPU/core



Process or Thread Behavior #2

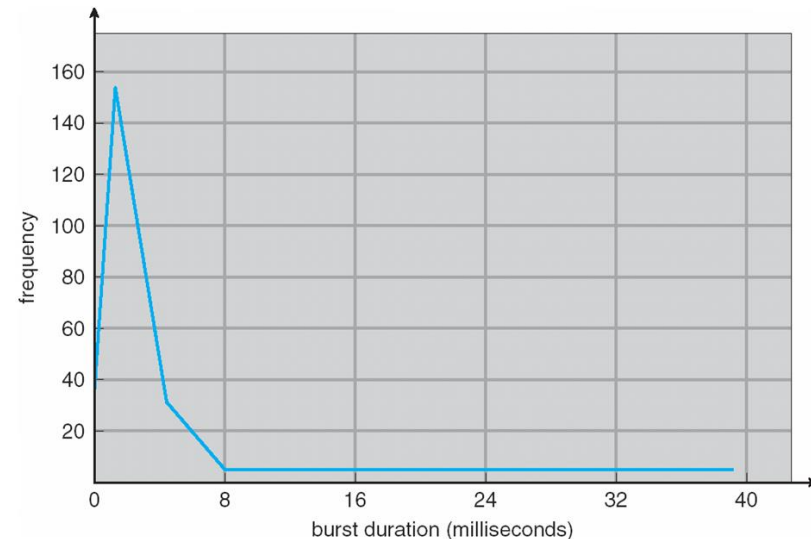
CPU
bound



I/O
bound



Time



While a process or thread is waiting for I/O **another can run** on the CPU

Scheduling Goals: Performance

- Many performance goals (which may conflict)
 - Maximize **CPU utilization**
 - Maximize **throughput** (processes completed per time unit)
 - Minimize **turnaround time** (time from submission of task to completion)
 - Minimize **waiting time** (all periods spent waiting in the ready queue from submission)
 - Minimize **response time** (time from submission of request to response is produced)
 - Minimize **energy** (joules per instruction) subject to some constraint (e.g., frames/second)
- In most cases we optimize the **average metric**
 - But sometimes minimize the worst case (e.g., response time)

Scheduling Goals: Fairness

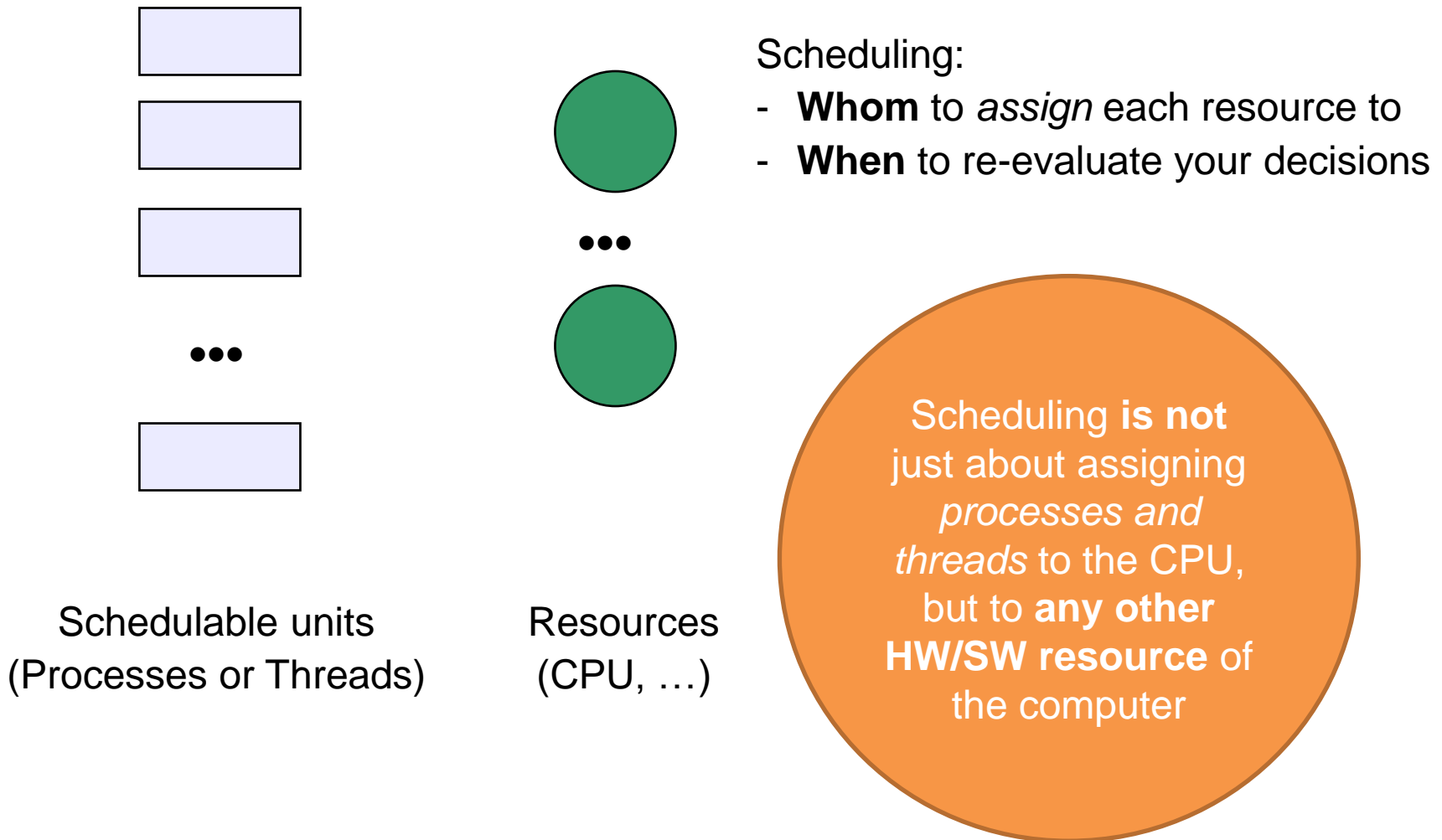
- No single, compelling definition of **“fair”**
 - How to **measure** fairness?
 - Equal CPU consumption? (over what time scale?)
 - Fair per-user? per-process? per-thread?
 - What if one process is CPU bound and one is I/O bound?
- Sometimes the goal is to be **unfair**
 - Explicitly favor some particular class of requests
 - **Priority system**
 - Avoid starvation
 - Be sure everyone gets at least some service

Classes of Schedulers

- Batch
 - Throughput / utilization oriented
 - *Example:* audit inter-bank funds transfers each night, Pixar rendering, Hadoop/MapReduce jobs
- Interactive
 - Response time oriented
 - *Example:* window-based operating system
- Real time
 - Deadline driven
 - *Example:* embedded systems (cars, airplanes, etc.)

We will **not talk** about real-time scheduling

Modeling General Scheduling Problem



When to Re-evaluate the Decision?

- **Non-preemptive scheduling**
 - Processes/threads execute until completion or until they want
 - **Voluntary process switch**
 - **Process/thread switch on blocking calls**
 - The scheduler gets involved only at exit or on request
 - For every clock interrupt, running process keeps going
- **Preemptive scheduling**
 - While a process/thread executes, its execution may be paused, and another process/thread resumes its execution, etc.
 - **Involuntary process switch**
 - For every clock interrupt, running process may be suspended and switched with another process (if there is any)

