COMP1521 Tutorial 09

Concurrency

- Running two or more processes at the same time to achieve one goal
- Allows utilisation of multiple CPUs
- E.g. GPUs

Advantages of Concurrency

- Higher throughput
  - Multiple CPUs executing the same program means total time required is less
- Better resource utilisation
  - In multicore systems the extra CPU cores wouldn't be sitting idle

Disadvantages of Concurrency

- Nondeterministic results
  - Same program running on same data but at different points in code can yield unwanted results
- Deadlock
  - Process may block each other from accessing certain resources
- Starvation
  - One process may end up locked out of a resource indefinitely

Semaphores

- Type of locking system to restrict access to resources
- Isn't binary, can have multiple processes access a resource at the same time
- Starts with a value > 0
  - Every time resource access by process, 1 is subtracted from semaphore
  - When process releases resource, 1 is added to semaphore
- When semaphore reaches 0, no more process can access resource

- When do we use semaphores bigger than 1?

Deadlock Example

```
semA: a semaphore allowing only one process at-a-time to access A
semB: a semaphore allowing only one process at-a-time to access B

process1()
{
   wait(semA)
   do some work on A
   wait(semB)
   do some work on B
   signal(semB)
   signal(SemA)
}
```

```
process2()
{
   wait(semB)
   do some work on B
   wait(semA)
   do some work on A
   signal(semA)
   signal(semB)
}
```

Using semaphore to fix code

- >> In github <<

Client/Server rundown

- For a client and server process to communicate, sockets are used
- Both have a socket which allows Interprocess communication
- Sockets can be created using the socket.h library in C

- Example in Github

CURL library

- Aims to make task from previous question simpler
- Example code on Github