Concurrency

Concurrency

- A sequential system is one where computations - or parts of a computation - are executed to completion, one after the other.
- A concurrent system is one where two or more computations are executing – literally or effectively – “at the same time”.
- A concurrent system is almost the same as a parallel system

Processes and Threads

- A thread or thread of control is a specific sequence of instructions defined by some program, or by some section of a program.
- Any process has one or more threads
- Difference between Parallelism and MultiTasking

Programming with threads

- POSIX Threads
- A low-level library for thread programming, often used from the C programming language
- A parent thread (e.g. “main programme”) calls the library function pthread_create, passing it a pointer to the function with the code for the new thread.
- A parent thread may create any number of threads, and children can create their own children, etc.

Thread creation

- int main(int argc, char* argv [])

```
{
    pthread_t thread ;
     pthread_create(&thread, NULL, run, NULL) ;
     x = 23 ; print x ;
 }
  void* run(void *)
  { y = 42 ;
    print y ; }
```

Multi threaded program

- Thread A:
  - x = 23
  - print x
- Thread B:
  - y = 42
  - print y

Multi threaded program

- Concurrent programs are often non-deterministic: different orders of execution may lead to different outcomes.
- x = 23
- print x
- y = 42
- print y

- x = 23 y = 42 print x print y

Shared counter

- Thread A:
  - x = c
  - c = x + 1
- Thread B:
  - y = c
  - c = y + 1

Issues /Solutions

- Race Condition/Interference:
- the outcome depends which thread gets to a particular point of its programme first.
- Avoiding Interference: make sure that threads never have any variables in common…is that alright?

Additional concepts

- Critical Sections:a code segment in a thread that updates or accesses shared data;
- Mutual Exclusion:methods to ensure only one thread does a particular activity
- Semaphores
- Deadlocks