3/1 CS 3420 Lecture: Concurrency

1. Concurrency
   1. Running in parallel

X

T1 T2

y b

z c

Task 1 is telling what T2 should do

1. Basic Assumptions
   1. Non-interference: processes care about their own variables
   2. Atomicity: from POV of other tasks it’s an all or nothing execution
      1. Ex.

ISR ->

hour = Minutes / 60;

minutes = Minutes %60;

->

Make interrupt see ISR process is atomic (then 1:00 would be impossible) because calculating hour and minutes is all or nothing

* 1. Assignments cannot “collide” to produce

1. Weak Fairness
   1. Enabled action: one that is ready to execute
   2. If an action is continuously enable, it will get a change to execute
   3. Everyone has a chance to do something at some point (who is eligible defines strong vs weak fairness)
2. Mutual Exclusion
   1. NCS: non-critical section- need not terminate
   2. CS: crit section- always terminates
   3. Requirements
   4. Safety: at most one process is inside CS
      1. Processes don’t spend all their time trying to access CS
   5. Progress: at least one process that want to be in CS is guaranteed access at some point
      1. Deadlock: ex. Traffic jam
      2. Live lock: trying to get past someone but keep blocking each other…
      3. Much harder to detect livelock because processes keep running but no results
   6. Fairness: Access infinite times and get access at least once to CS
   7. Examples: Implementing mutual exclusion in C

While (1) {

NCS;

While (turn!=1)

CS

Turn = 2;

}

While (1) {

NCS;

While (turn!=2)

CS

Turn = 1;

}

* 1. Only allow other to go into CS once you yourself go through the CS
     1. Guarantees safety: assume NCS/CS doesn’t mess with turn. Break safety: more than one process is in CS simultaneously (safety guaranteed by !=1/2 if CPU switches between processes)
     2. Progress violated: NCS doesn’t terminate

While (1) {

NCS;

While (x2)

X1=1;

CS

X1 = 0;

}

While (1) {

NCS;

While (x1)

X2=1;

CS

X2 = 0;

}

* 1. General: x1,x2 tell if they are interested
  2. Not safe: CPU can switch back and forth if able to get past while, then both CS will be executed

3/9 Locking