Cooperating Threads

Sarah Diesburg
Operating Systems
CS 3430
Independent Threads

- No states shared with other threads
- Deterministic computation
  - Output depends on input
- Reproducible
  - Output does not depend on the order and timing of other threads
- Scheduling order does not matter
- e.g., compilers
Cooperating Threads

- Shared states
- Nondeterministic
- Nonreproducable
- Example: 2 threads sharing the same display
  
  Thread A
  printf("ABC");

  Thread B
  printf("123");

- You may get "A12BC3"
So, Why Allow Cooperating Threads?
So, Why Allow Cooperating Threads?

- Shared resources
  - e.g., a single processor

- Speedup
  - Occurs when threads use different resources at different times

- Modularity
  - An application can be decomposed into threads
Some Concurrent Programs

If threads work on separate data, scheduling does not matter.

Thread A
x = 1;

Thread B
y = 2;
Some Concurrent Programs

- If threads share data, the final values are not as obvious

<table>
<thead>
<tr>
<th>Thread A</th>
<th>Thread B</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 1;</td>
<td>y = 2;</td>
</tr>
<tr>
<td>x = y + 1;</td>
<td>y = y * 2;</td>
</tr>
</tbody>
</table>

- What are the indivisible operations?
Atomic Operations

- An *atomic operation* always runs to completion; it’s all or nothing
  - e.g., memory loads and stores on most machines
- Many operations are not atomic
  - Double precision floating point store on 32-bit machines
Suppose…

- Each C statement is atomic
- Let’s revisit the example…
All Possible Execution Orders

Thread A
\[ x = 1; \]
\[ x = y + 1; \]

Thread B
\[ y = 2; \]
\[ y = y \times 2; \]

A decision tree
All Possible Execution Orders

Thread A:
\[ x = 1; \]
\[ x = y + 1; \]

Thread B:
\[ y = 2; \]
\[ y = y \times 2; \]
Another Example

- Assume each C statement is atomic
  - Both threads are in the same address space

Thread A
j = 0;
while (j < 10) {
    ++j;
}
printf("A wins");

Thread B
j = 0;
while (j > -10) {
    --j;
}
printf("B wins");
So…

- Who wins?
- Can the computation go on forever?

- *Race conditions* occur when threads share data, and their results depend on the timing of their executions…