Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Process and Threads

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Sequential vs Concurrent Blocks

Sequential Block:

- Simulator executes statements in a sequential block in sequence
- It finishes the current statement, then begins the next
- You always know the order in which it actually executes the statements
- The simulator exits the block after finishing the last statement

- Concurrent Block
  - Simulator executes statements in a concurrent block in parallel
  - It starts executing all statements simultaneously
  - You can not know the order in which it actually executes statements scheduled for the same simulation time
  - The simulator exits the block after finishing the latest statement.
  - A **return** statement in the context of fork..join is illegal.

Process and Threads

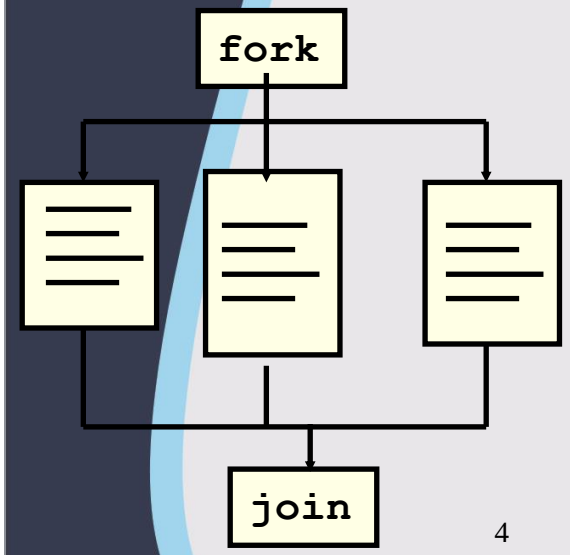
Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

- Threads are created via fork…join
- Threads execute until a blocking statement
  - wait for: (event, mailbox, semaphore, variable, etc.)

Multiple Independent Threads Maximize Stimulus Interactions

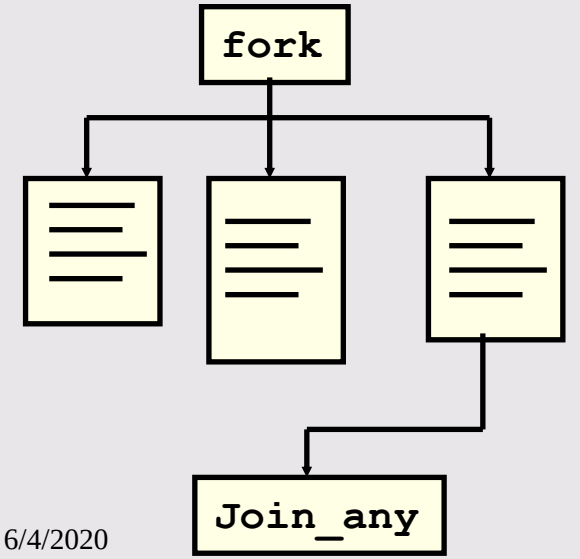
fork-join

```
fork
   a;b;c;
join
```



fork-join_any

```
fork
   a;b;c;
join_any
disable fork //kill all child
                // process
wait fork   //wait till child
            //process over
```



fork-join_none

```
fork
   a;b;c;
join_none
```