Looking deeper into processes...

- Process embodies *two* primary characteristics
  - Resource ownership (memory image)
  - Execution of a *sequence of instructions*
    - This characteristic is captured by abstraction called *thread*
- *Process* provides environment & resources for *thread* of execution

- Thread *is the basic unit of execution and scheduling*
- Every process has a thread of execution
  - Process not limited to *single* thread of execution

Consider an example...

Making roasted vegetable pasta!

- Come up with sequence of operations for it

What we learn from the example...

- Process could contain *multiple instruction sequences*
- Sequences can be modeled as *threads* in *single* process

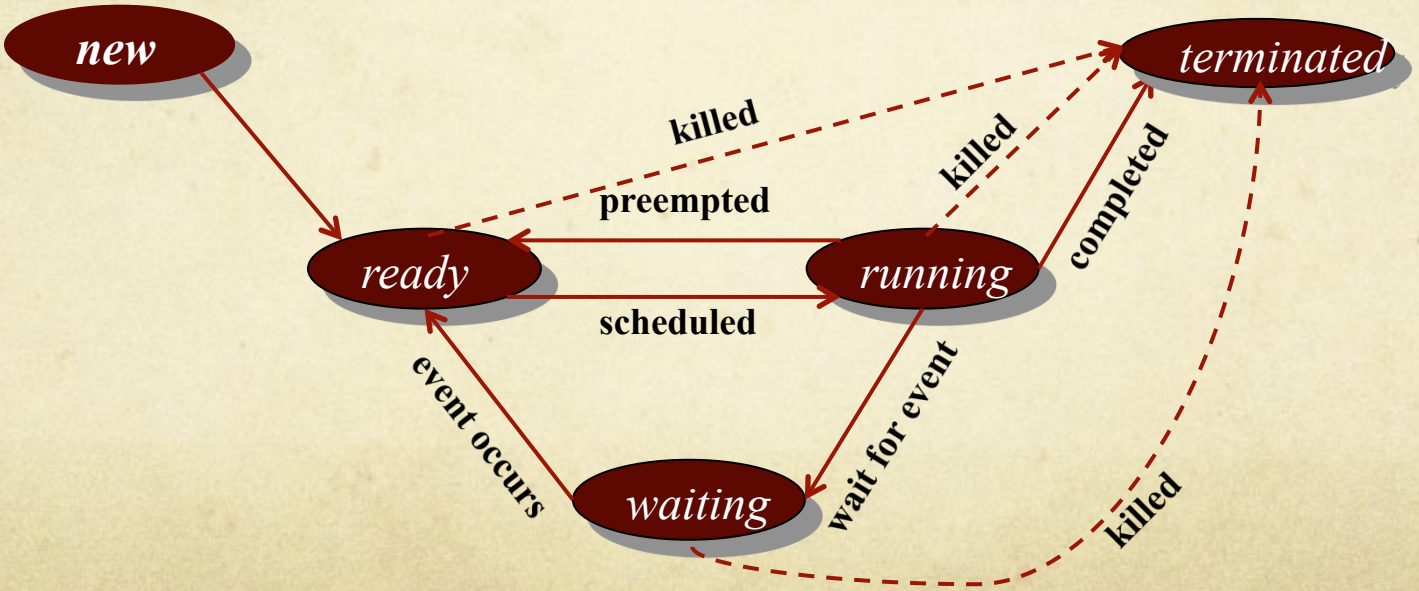
  - Allows *concurrency* within single process
- Threads of a given process are closely related
  - Threads may *share* process resources
  - Work in *co-operative* manner
- This concept is called *multi-threading*

Examples of multi-threading

- Word processor
  - *Thread* to interface with user
  - *Thread* to reformat document in the background
  - *Thread* to perform periodic disk backup (auto save)
  - ...
- Web server
  - *Dispatcher thread* listens for requests on given connection
  - Requests serviced by different *worker threads*

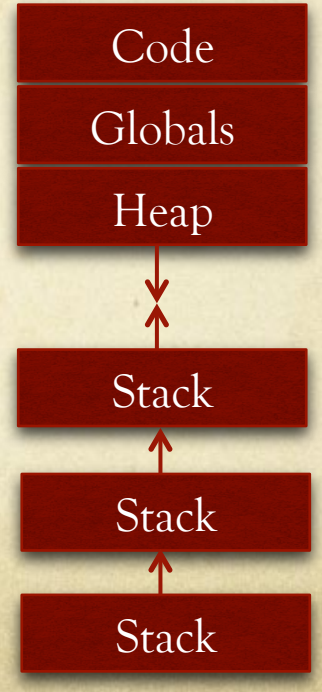
Thread states

- Similar to processes, threads also can be in different states
  - States & transitions are similar to processes



What information is per-thread?

- Program counter
- Stack & stack pointer
- Registers
- Process with multiple threads looks like this…



Thread information

- Just like *Process Control Blocks* are used for processes...
  - ...*Thread Control Blocks* are used to maintain thread info

Important points to note

- Process → instance of *single* program owned by *single* user
- OS provides no protection among threads of *same* process
  - Assumption → threads within process *cooperate*, not *compete*
- All thread specific data (TCBs & stacks for all threads)
  - Part of *process* memory image…
  - …and hence accessible to *all* threads
- Programs must be *carefully* designed & implemented to ensure correctness
- Every process has at least *one* thread of execution (e.g., main)
  - Processes start with this one default thread
  - This thread can *create* other threads, which can create more, …

Advantages of multi-threading

- More efficient CPU utilization within process
- Creation/termination of threads is *cheap*
- Context switch time between threads of process is minimal
  - Only info *exclusive* to threads needs to be "switched"
- Supports effective communication due to shared memory