Some OS Basics

Some notes adopted from Bryant and O'Hallaron

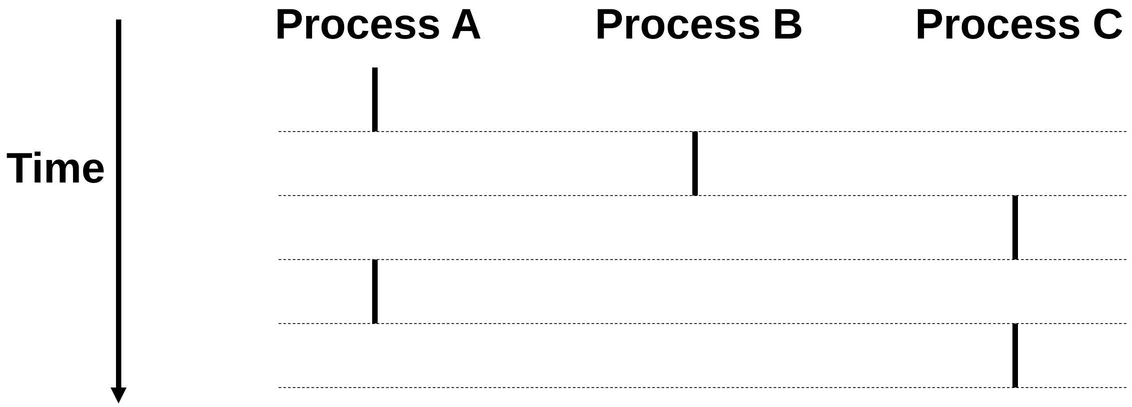
Role of The Operating System?

Processes

- Def: A process is an instance of a running program.
  - One of the most profound ideas in computer science.
  - Not the same as “program” or “processor”
- Process provides each program with two key abstractions:
  - Logical control flow
    - Each program seems to have exclusive use of the CPU.
  - Private address space
    - Each program seems to have exclusive use of main memory.
- How are these Illusions maintained?
  - Process executions interleaved (multitasking)
  - Address spaces managed by virtual memory system

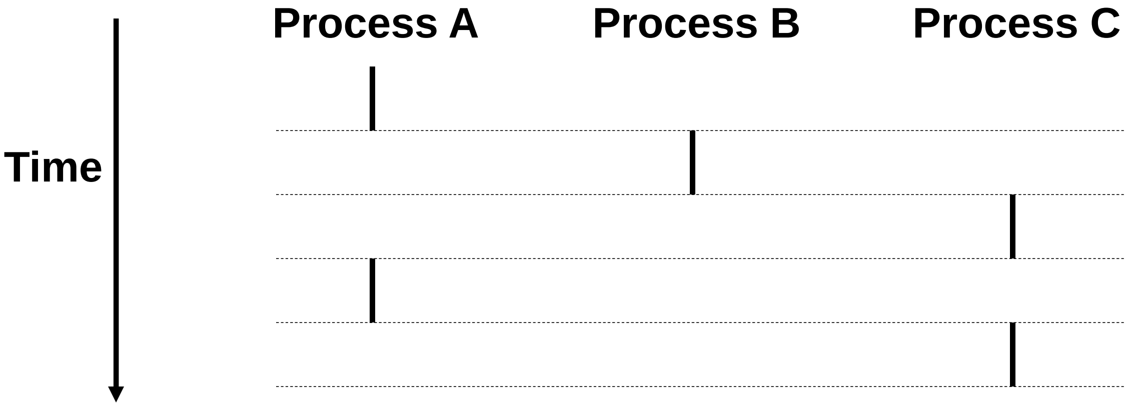
Logical Control Flows

Each process has its own logical control flow



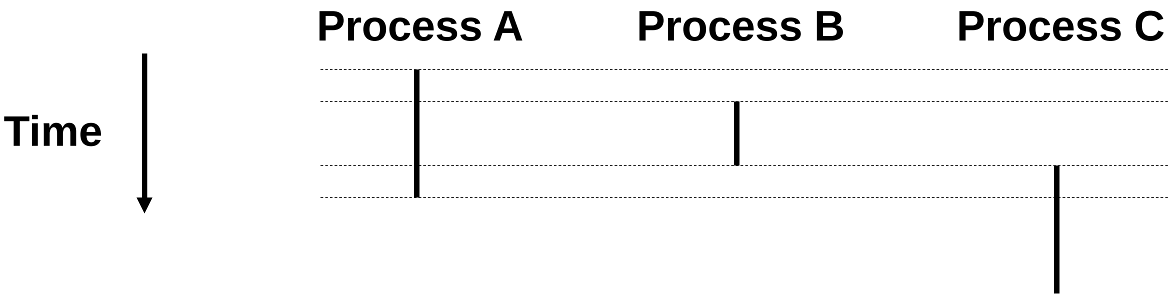
Concurrent Processes

- Two processes *run concurrently* (*are concurrent)* if their flows overlap in time.
- Otherwise, they are *sequential.*
- Examples:
  - Concurrent: A & B, A & C
  - Sequential: B & C



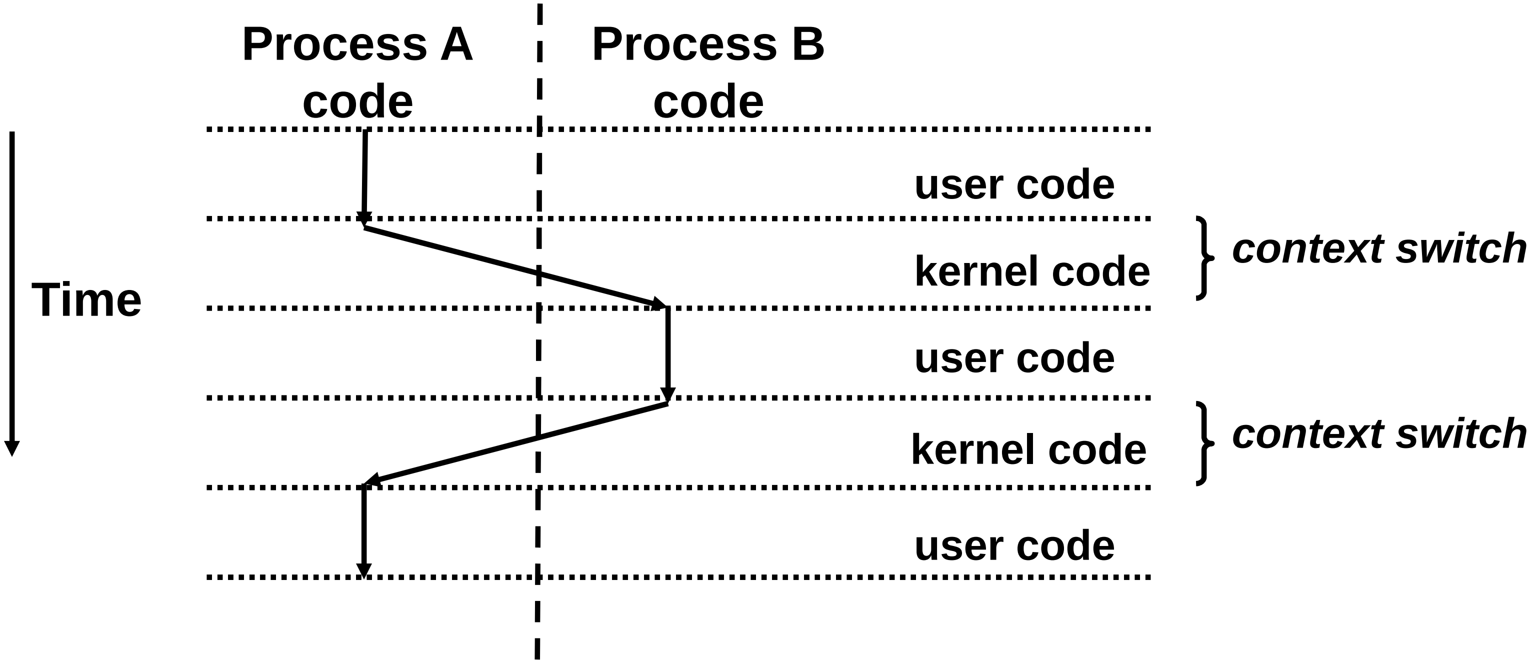
User View of Concurrent Processes

- Control flows for concurrent processes are physically disjoint in time.
- However, we can think of concurrent processes are running in parallel with each other.



Context Switching

- Processes are managed by a shared chunk of OS code called the *kernel*
- Control flow passes from one process to another via a *context switch.*



Process: Traditional View

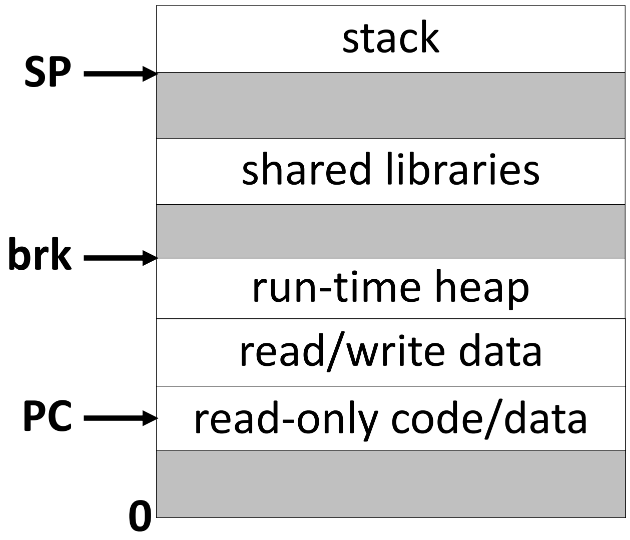
- Process = process context + code, data, and stack

Process context

Program context:
Data registers
Condition codes
Stack pointer (SP)
Program counter (PC)

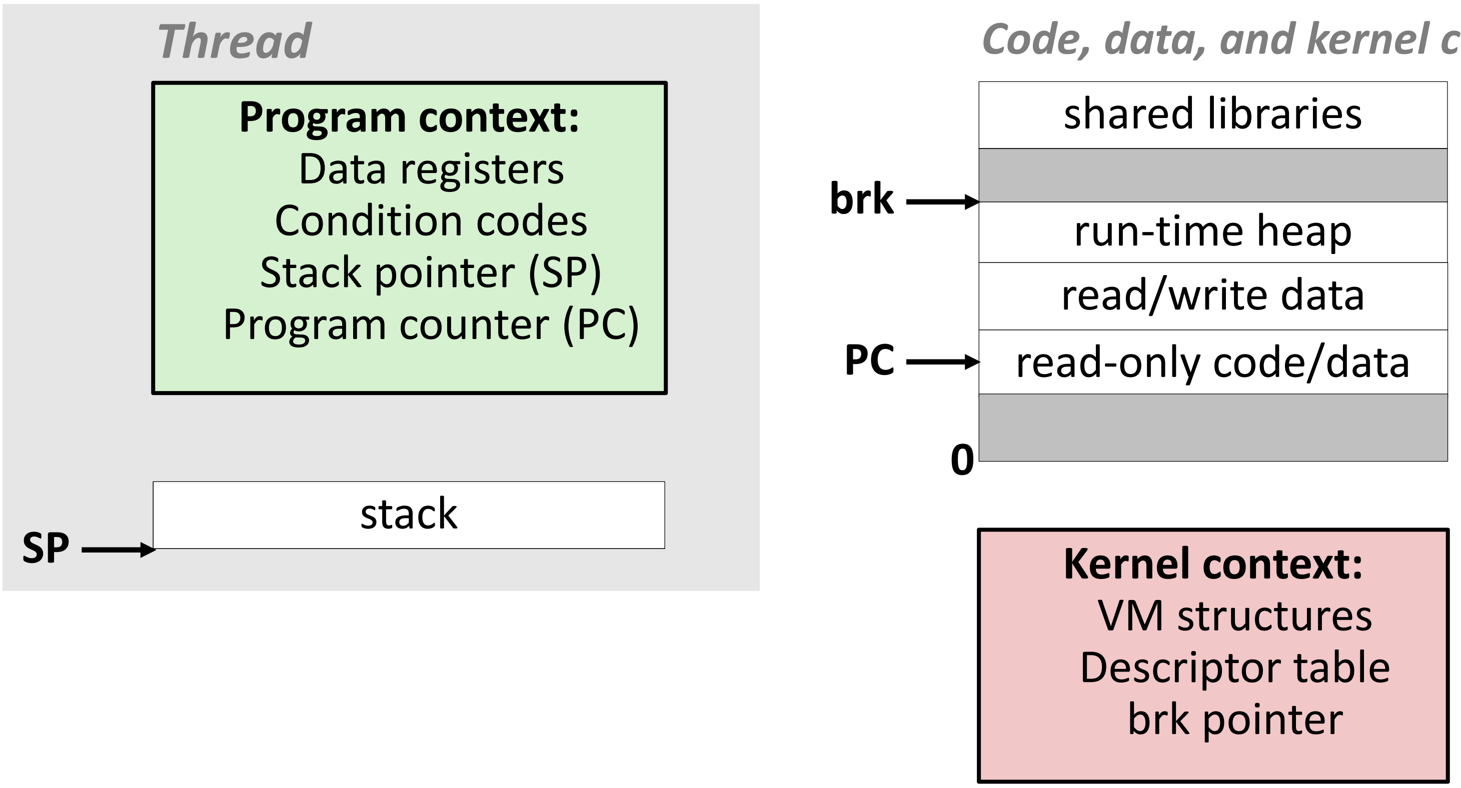
Kernel context:
VM structures
Descriptor table
brk pointer

Code, data, and stack



Process: Alternative View

- Process = thread + code, data, and kernel context



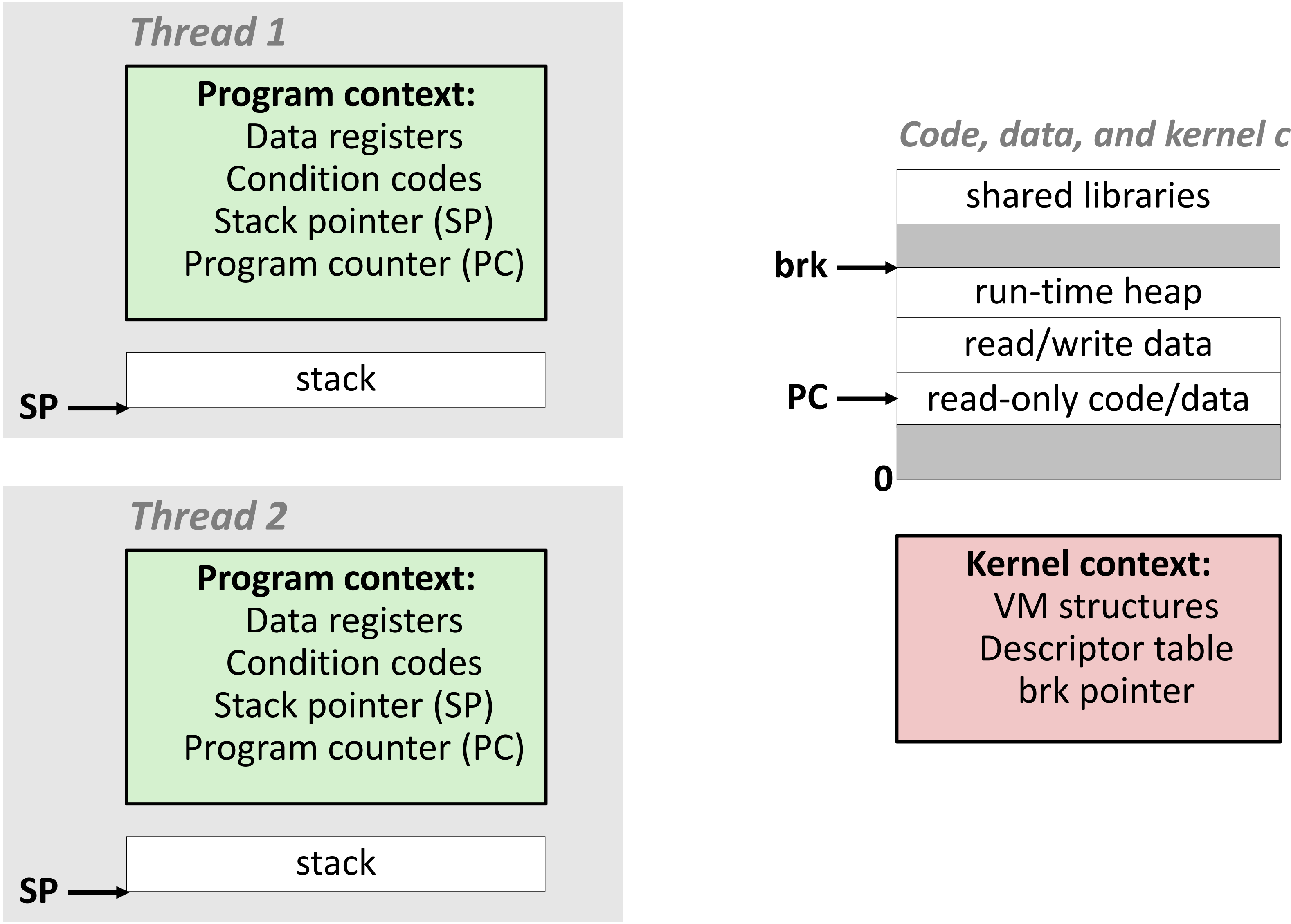
Thread

Program context:
Data registers
Condition codes
Stack pointer (SP)
Program counter (PC)

Code, data, and kernel context

Kernel context:
VM structures
Descriptor table
brk pointer

Process with Two Threads



Thread 1

Program context:
Data registers
Condition codes
Stack pointer (SP)
Program counter (PC)

stack

SP

Thread 2

Program context:
Data registers
Condition codes
Stack pointer (SP)
Program counter (PC)

stack

SP

Code, data, and kernel context

shared libraries

brk

run-time heap

read/write data

PC

read-only code/data

0

Kernel context:
VM structures
Descriptor table
brk pointer

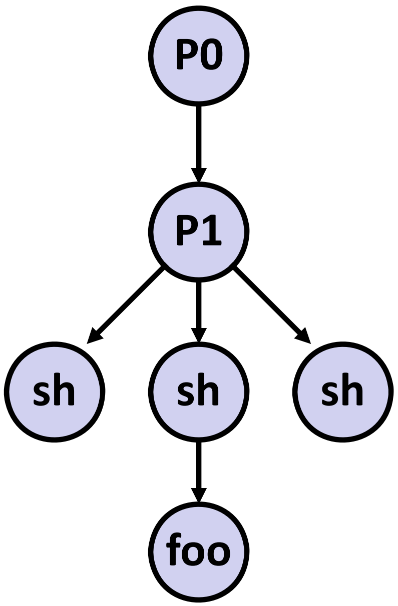
Threads vs. Processes

- Threads and processes: similarities
  - Each has its own logical control flow
  - Each can run concurrently with others
  - Each is context switched (scheduled) by the kernel
- Threads and processes: differences
  - Threads share code and data, processes (typically) do not
  - Threads are much less expensive than processes
    - Process control (creating and reaping) is more expensive as thread control
    - Context switches for processes much more expensive than for threads

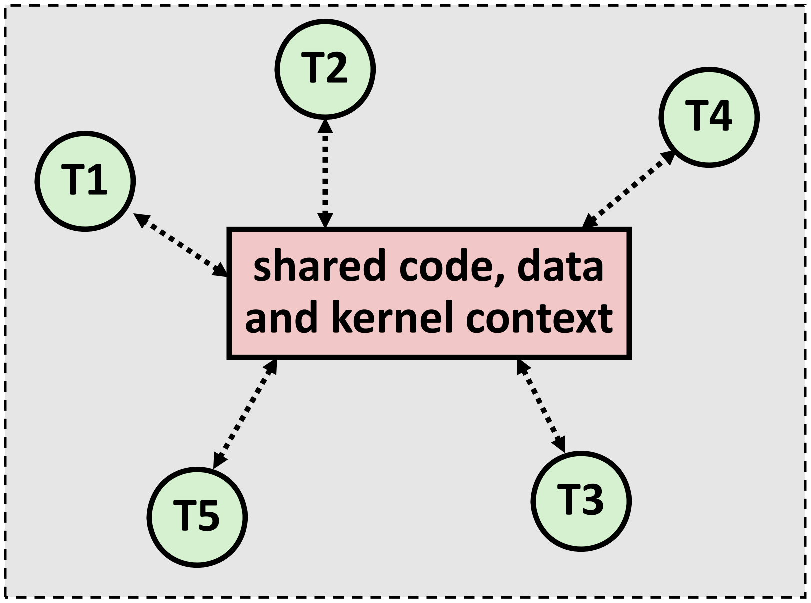
Threads vs. Processes (contd.)

- Processes form a tree hierarchy
- Threads form a pool of peers
  - Each thread can kill any other
  - Each thread can wait for any other thread to terminate
  - Main thread: first thread to run in a process

Process hierarchy



Thread pool


T2
T4
T1
shared code, data and kernel context
T5
T3