G52APR
Application programming

Processes and Multiple Threading

Colin Higgins – based on material from various sources including the Sun tutorial

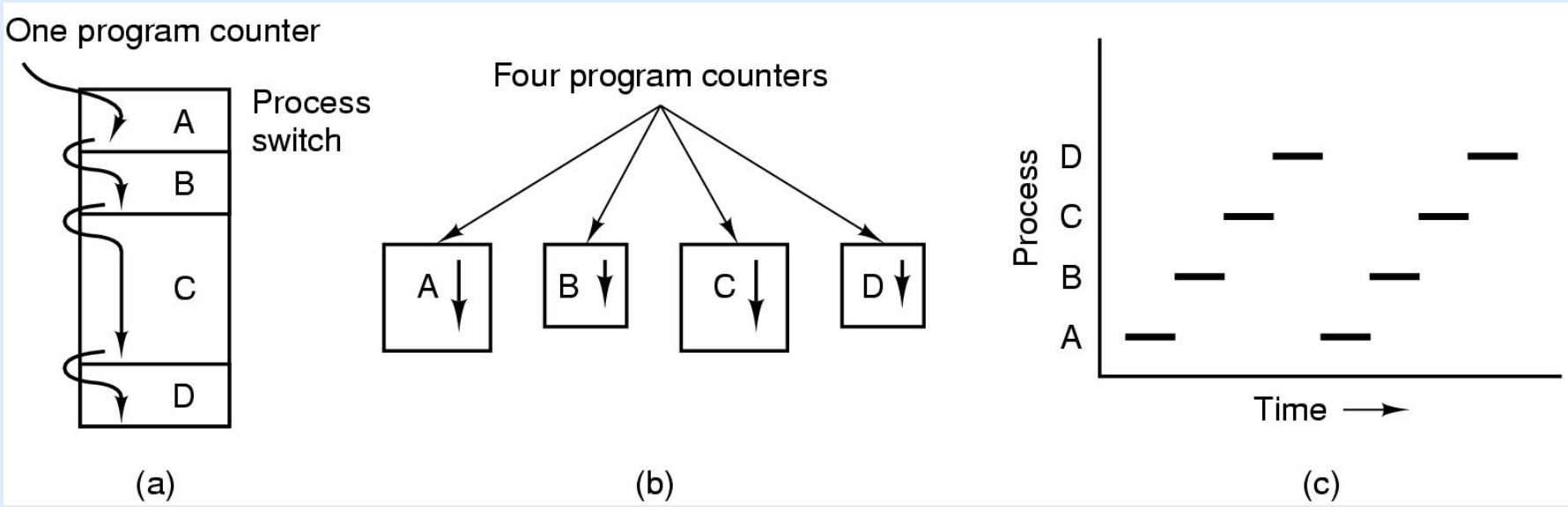
Aims

- To understand the concept of concurrent execution, or concurrency
- To understand the process and thread
- To be able to create, execute, and terminate Threads
- To understand their States and their Priorities

Concurrency

- Multiple tasks for computer
  - Draw & display images on screen
  - Check keyboard & mouse input
  - Send & receive data on network
  - Read & write files to disk
  - Perform useful computation (editor, browser, game)
- How does computer do everything at once?
  - Multiple processing
  - Multitasking

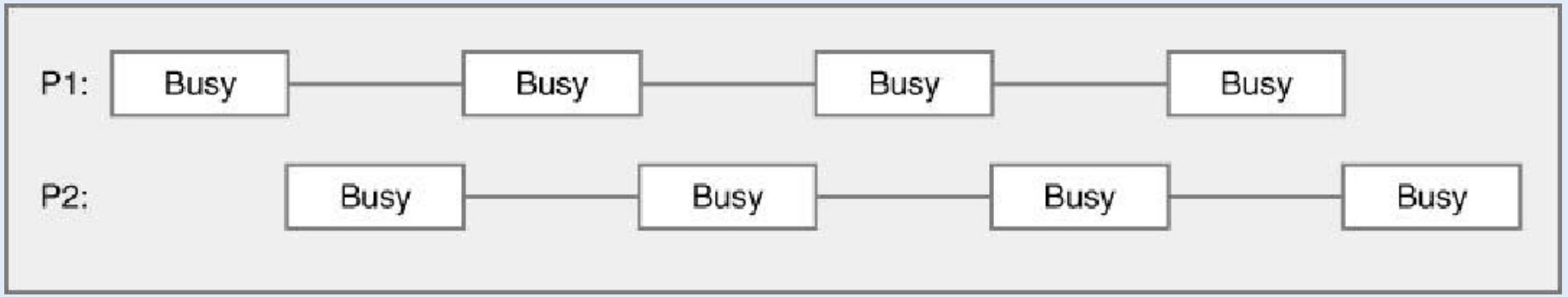
The Process Model



- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant

Multitasking

- Approach
  - Time slicing
  - Computer does some work on a task
  - Computer then quickly switch to next task
  - Tasks managed by operating system (scheduler)
- Context switch



Concurrency

- On a single processor machine
  - Multitasking

  [Can improve performance by reducing waiting]

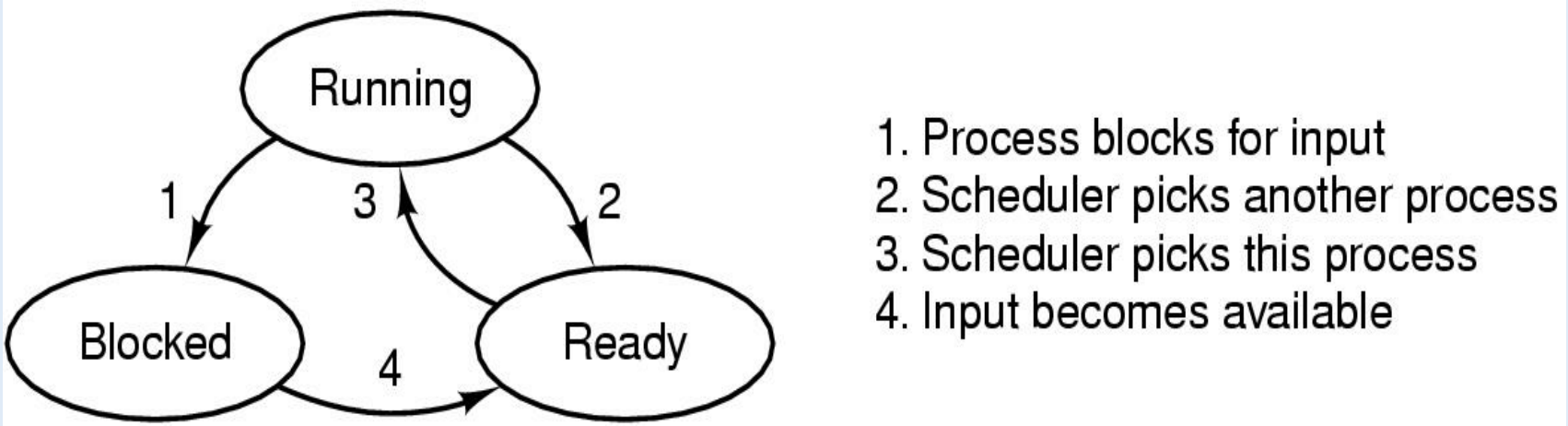
- On a multiple processors machine
  - Multiple processing
  - Multitasking

Perform multi tasks using

➢ Process

- Definition – executable program loaded in memory
- Has own address space
  - Variables & data structures (heap and stack)
- Each process may execute a different program
- Communicate via operating system, files, network
- May contain multiple threads

Process States



- Possible process states
  - running
  - blocked
  - ready
- Transitions between states shown

Process Table

- When a process moves from a running state to a ready or blocked state it must store certain information so that it can restart from the same point when it moves back to a running state
  - Which instruction it was about to execute
  - Which record it was about to read from its input file
  - Values in the registers

Process Control Block

Process management	**Memory management**	**File management**
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Fields of a process table entry

Problems with Processes

- Heavyweight
  - Context switching is expensive (ie takes time/ resources)
- Inter process communication difficult/ inefficient
- Fine grain parallelism therefore impossible

Threads

- “Lightweight process”
- Process have 2 sets of responsibilities =>
  - Resource group (text, data, open files etc)
    - Easy management
  - Thread of execution
    - State (program counter, registers, stack)
- Can separate the two purposes
- See eg Tanenbaum Chapter 2

Perform multi tasks using

➢ Thread

- Definition – sequentially executed stream of instructions
- Shares address space with other threads
- use own stack (local variables)
- Communicate via shared access to data
- Multiple threads in process execute same program
- Also known as “lightweight process

Process vs. thread

- Processes - carry considerable state information, whereas multiple threads within a process Unique
  - Program counter
  - Heap and stack
  - Running by itself, created by other process
  - Own address space
  - Interact via operating system
- Threads (Lightweight Processes) - share state as well as memory and other resources
  - Own Program counter
  - Own Stack
  - Created by a process or another thread
  - Shared address space
  - Interact via shared address space

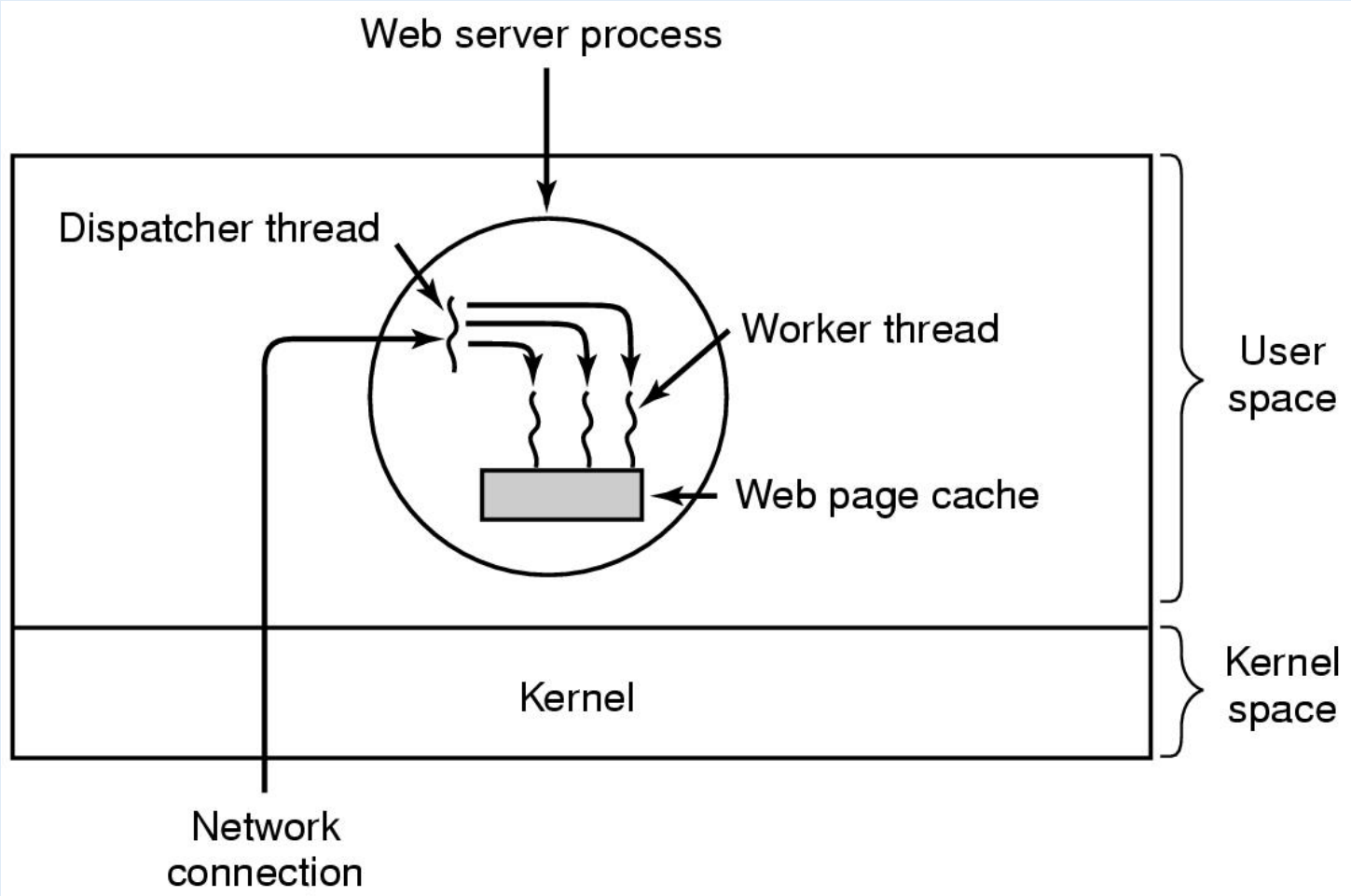
Motivation for Multithreading

- **Speed** – context switching much faster
- Captures **logical** structure of problem
  - May have concurrent interacting components
  - Can handle each component using separate thread
  - Simplifies programming for problem
- Example

Web Server uses threads to handle …

Multiple simultaneous web browser requests

Thread Usage (2)



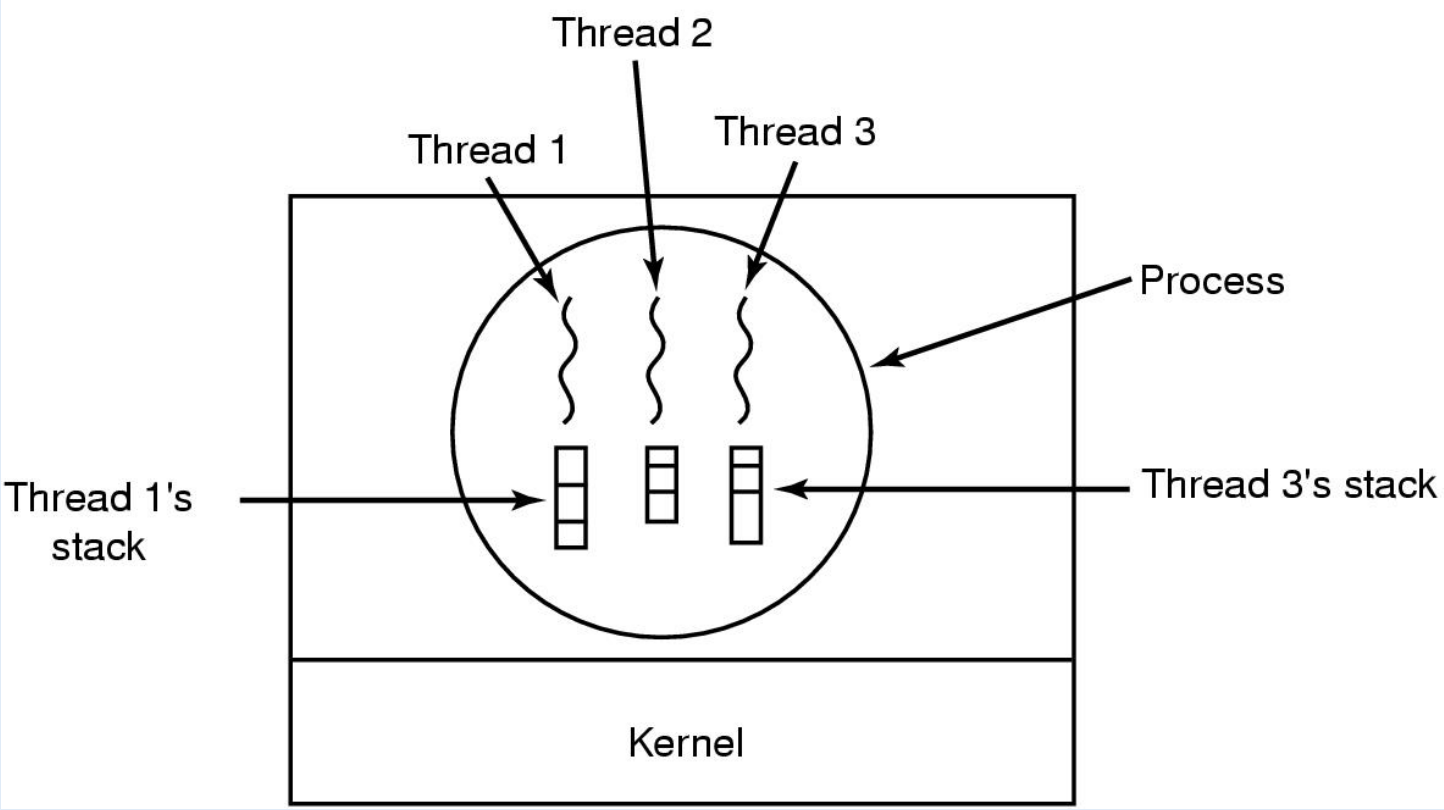
A multithreaded Web server

The Thread Model (2)

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

- Items shared by all threads in a process
- Items private to each thread

The Thread Model (3)



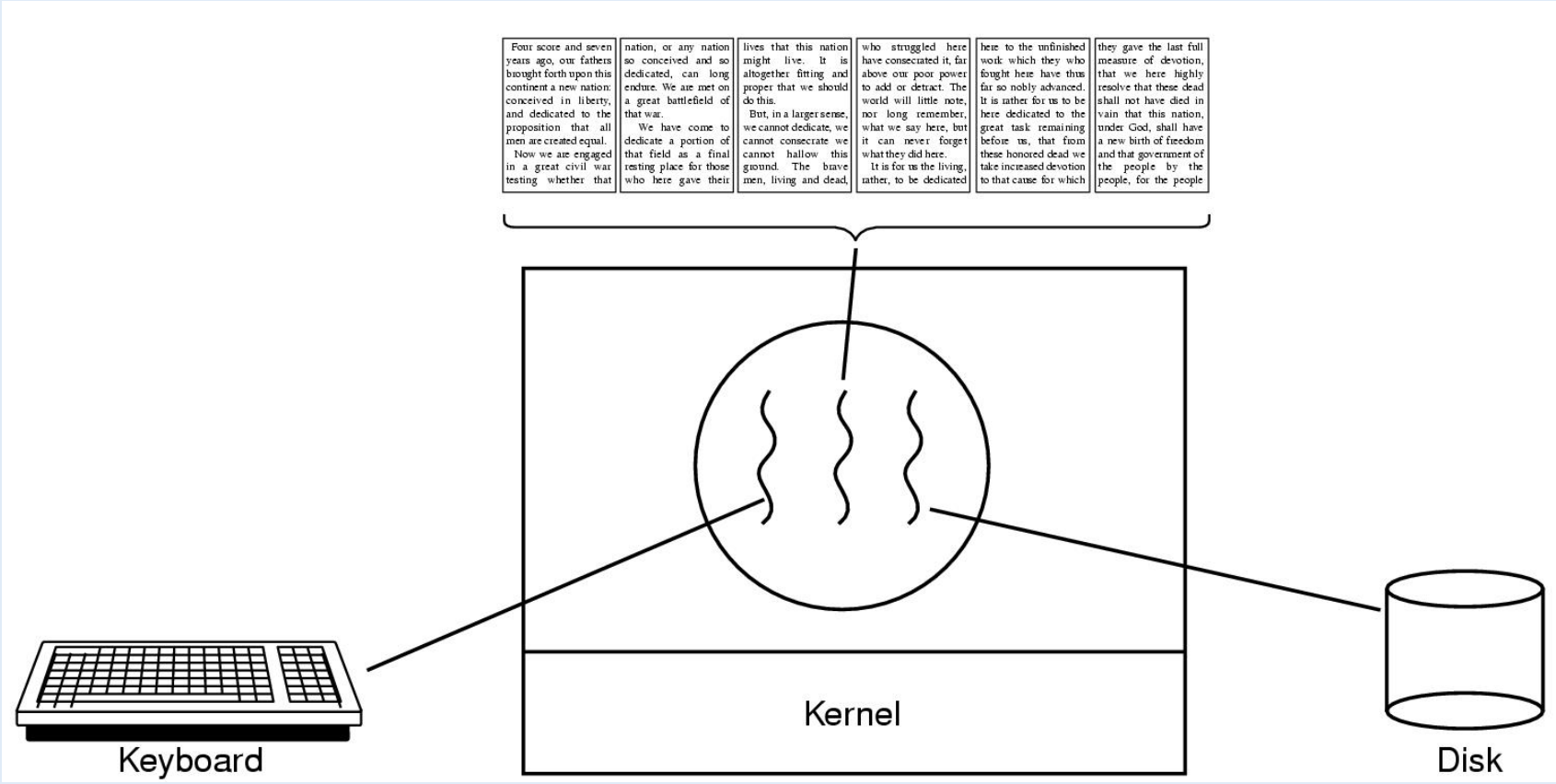
Each thread has its own stack

Thread Usage (0)

- Multiple activities which may block
  - Multi threading
    - can still do useful work
    - can share resources
- Easier to switch (& create)

  ~100 times faster than process switch
- Speed
  - Given CPU bound I/O bound mix

Thread Usage (1)



A word processor with three threads

Word processor details

- Can’t have 3 processes
  - All working on same document in same workspace
- But if we have 3 threads
  - One thread interacts with user
  - One thread reformats document
  - One thread auto-saves