

Module 2: Concurrency Basics

Topic 1.1: Processes

Processes

- An instance of a running program
 - Things unique to a process
1. Memory
 - Code, stack, heap, shared libraries
 - Virtual address space
 2. Registers
 - Program counter, data regs, stack ptr, ...

Operating Systems

- Allows many processes to execute concurrently
- Processes are switched quickly
 - 20ms
- User has the impression of parallelism
- Operating system must give processes fair access to resources

Task Manager

Task Manager

File

Options

View

Processes

Performance

App history

Startup

Users

Details

Services

Name	1% CPU	46% Memory	3% Disk	0% Network	0% GPU	GPU Engine
Apps (4)						
> Google Chrome (10)	0.5%	774.1 MB	0.1 MB/s	0 Mbps	0%	
> Microsoft PowerPoint (4)	0%	125.7 MB	0 MB/s	0 Mbps	0%	
> Task Manager	0.5%	18.3 MB	0.1 MB/s	0 Mbps	0%	
> Windows Explorer	0.1%	46.6 MB	0.1 MB/s	0 Mbps	0%	
Background processes (70)						
AcroTray (32 bit)	0%	1.4 MB	0 MB/s	0 Mbps	0%	
> Adobe Acrobat Update Service ...	0%	0.9 MB	0 MB/s	0 Mbps	0%	
> Adobe Genuine Software Integri...	0%	1.5 MB	0 MB/s	0 Mbps	0%	
Application Frame Host	0%	4.6 MB	0 MB/s	0 Mbps	0%	
Avast Antivirus (32 bit)	0%	9.8 MB	0 MB/s	0 Mbps	0%	
> Avast Behavior Shield	0%	37.8 MB	0 MB/s	0 Mbps	0%	
> Avast Service (32 bit)	0.1%	27.3 MB	0.1 MB/s	0 Mbps	0%	

<

>

<

Fewer details

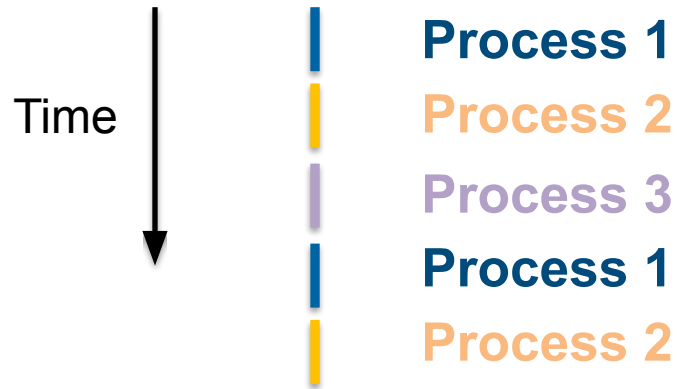
End task

Module 2: Concurrency Basics

Topic 1.2: Scheduling

Scheduling Processes

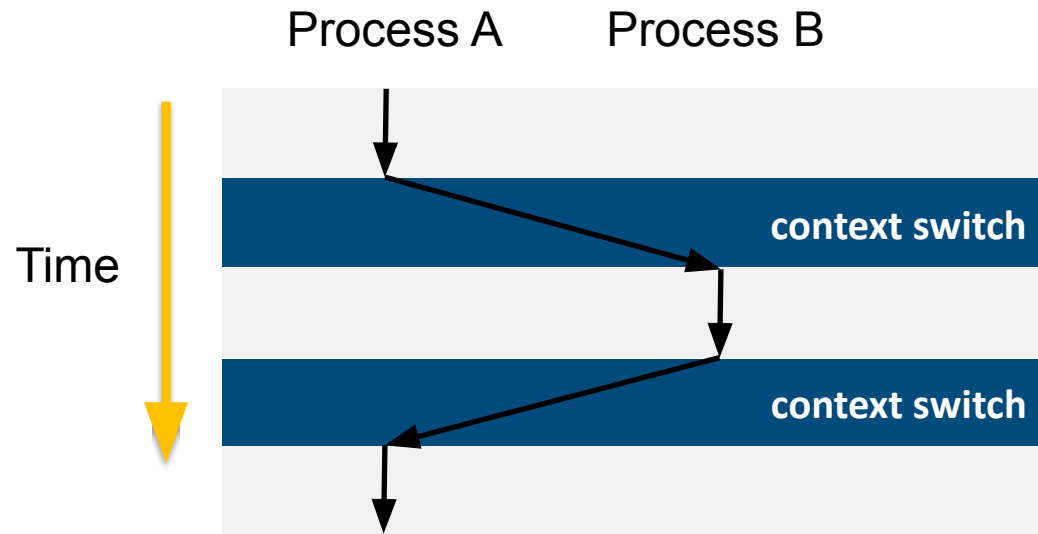
- Operating system schedules processes for execution
- Gives the illusion of parallel execution



- OS gives fair access to CPU, memory, etc.

Context Switch

- Control flow changes from one process to another
- Process “context” must be swapped

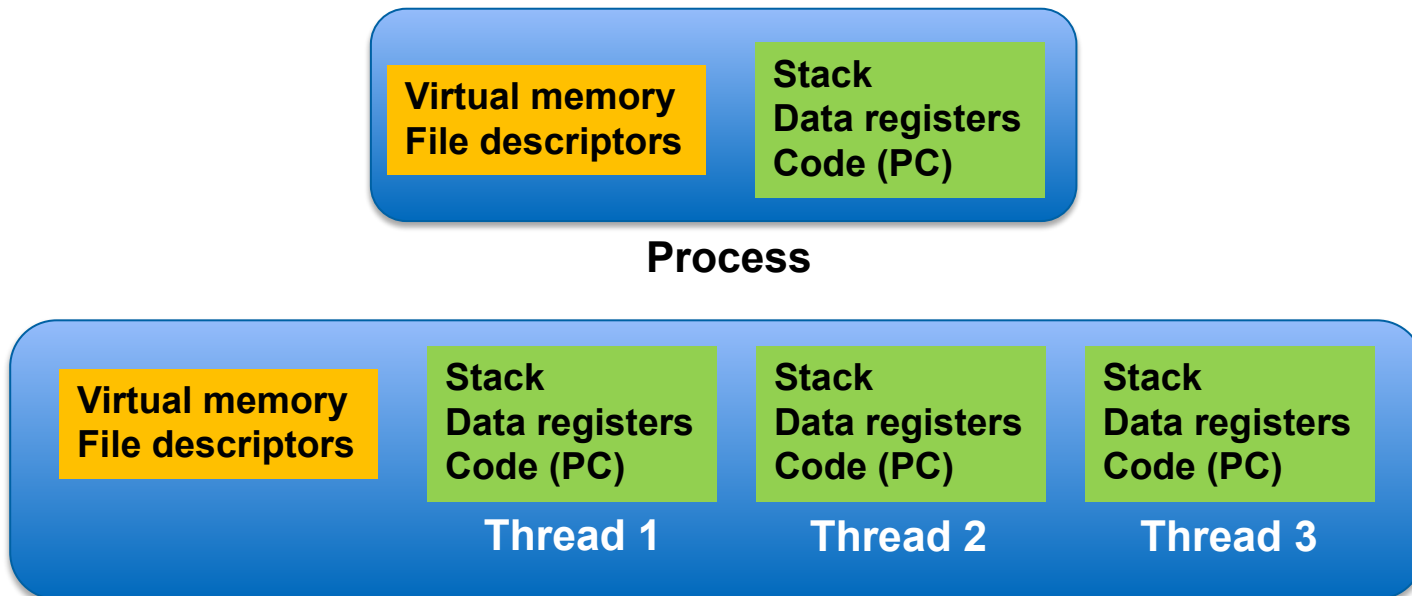


Module 2: Concurrency Basics

Topic 1.3: Threads and Goroutines

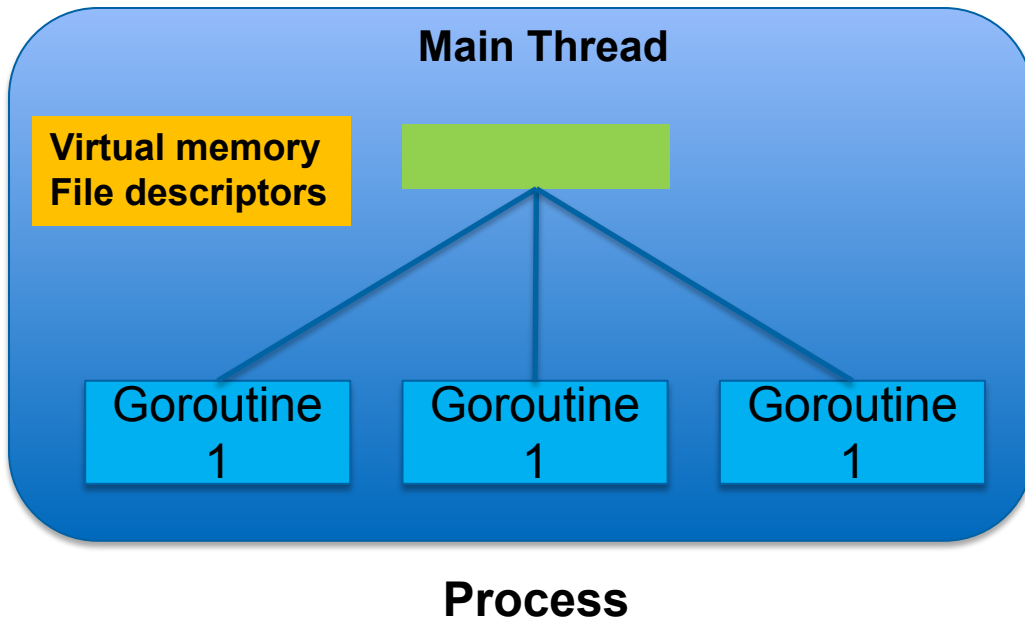
Threads vs. Processes

- Many threads can exist in one process
- Threads share some context
- OS schedules threads rather than processes



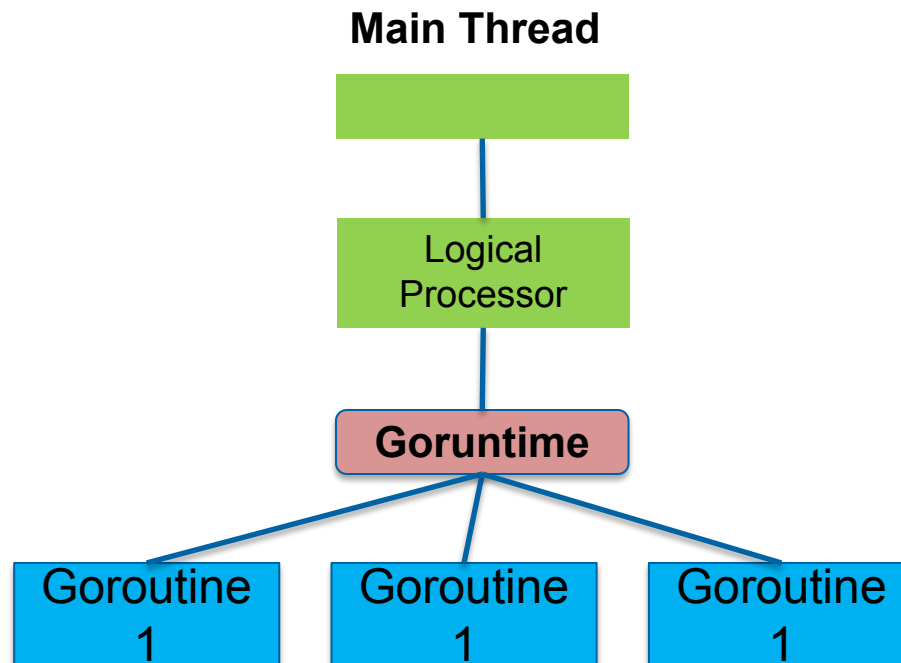
Goroutines

- Like a thread in Go
- Many Goroutines execute within a single OS thread



Go Runtime Scheduler

- Schedules goroutines inside an OS thread
- Like a little OS inside a single OS thread
- **Logical processor** is mapped to a thread



Module 2: Concurrency Basics

Topic 2.1: Interleavings

Interleavings

- Order of execution within a task is known
- Order of execution **between** concurrent tasks is unknown
- Interleaving of instructions between tasks is unknown

Task 1

1: $a = b + c$

2: $d = e + f$

3: $g = h + i$

Task 2

1: $r = s + t$

2: $u = v + w$

3: $x = y + z$

Possible Interleavings

1: $a = b + c$	
	1: $r = s + t$
2: $d = e + f$	
	2: $u = v + w$
3: $g = h + i$	
	3: $x = y + z$

1: $a = b + c$	
2: $d = e + f$	
3: $g = h + i$	
	1: $r = s + t$
	2: $u = v + w$
	3: $x = y + z$

- Many interleavings are possible
- Ordering is **non-deterministic**
- Must consider all possibilities

Race Conditions

- Outcome depends on non-deterministic ordering

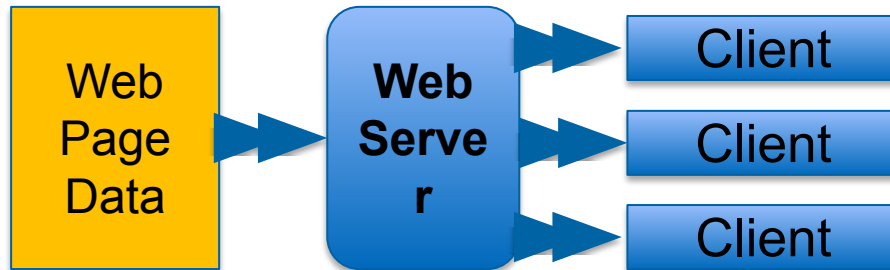
1: x = 1	
	1: print x
2: x = x + 1	

1: x = 1	
2: x = x + 1	
	1: print x

- Races occur due to **communication**

Communication Between Tasks

- Threads are largely independent but not completely independent
- Web server, one thread per client



- Image processing, 1 thread per pixel block

