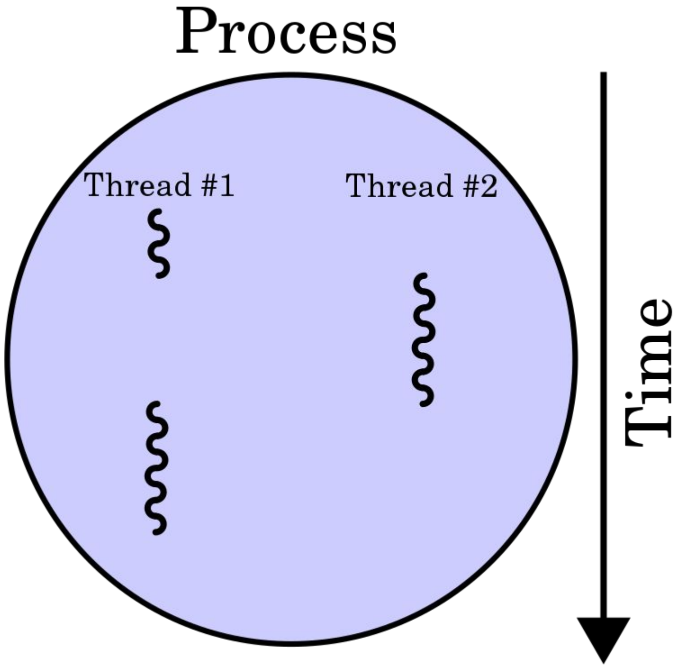
Threading

Objective

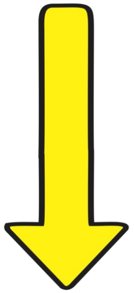
What is a Thread?

1. A thread is a unit of process execution.
2. process can consist the multiple threads.

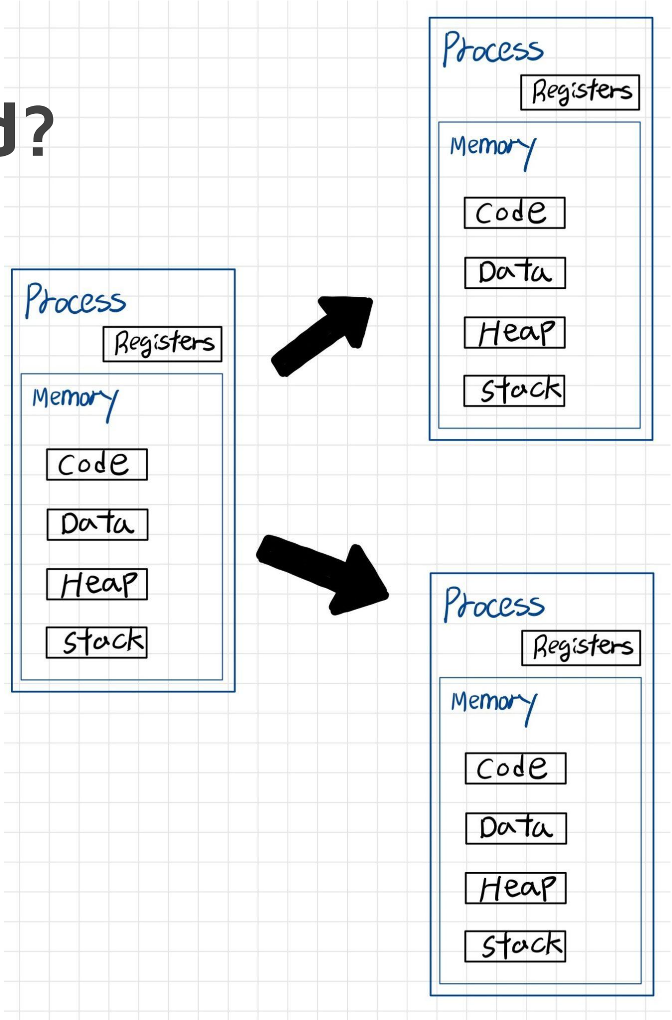


Why are we using Thread?

1. Processes have their own memory and it's independent.
2. If they want to communicate with each other, IPC is required and there will be context switching



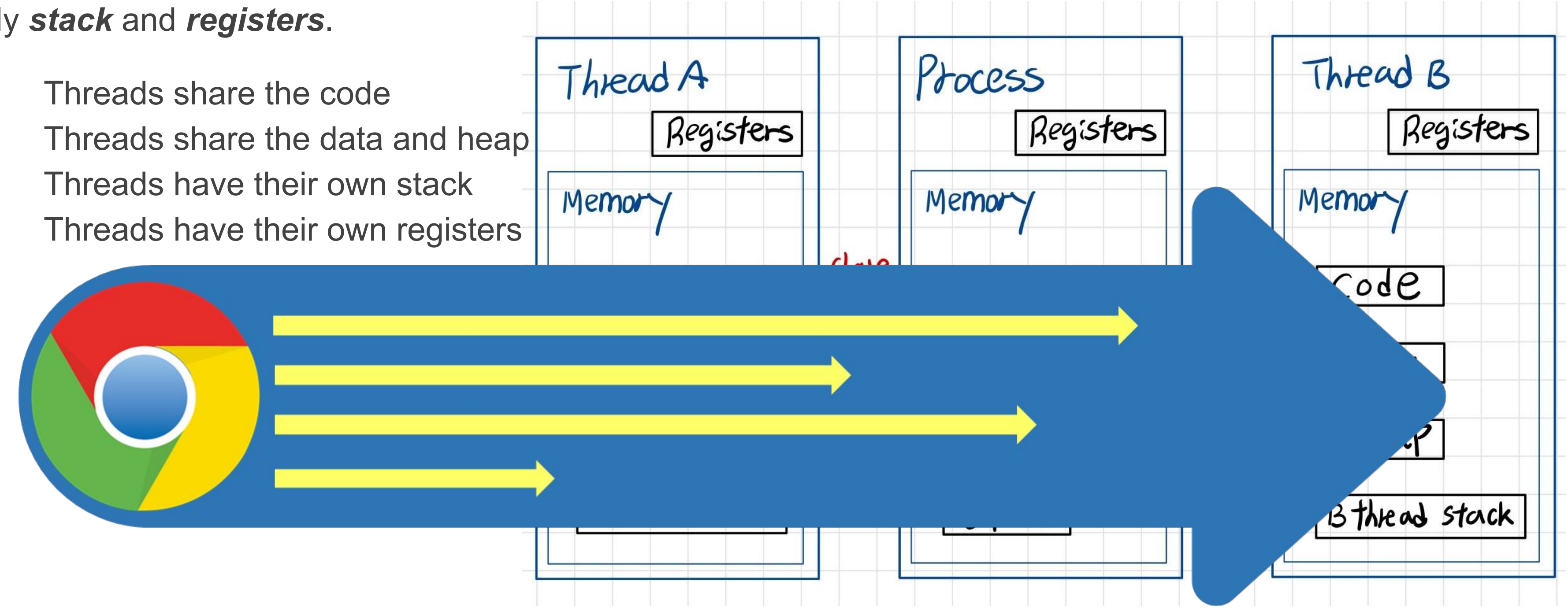
To reduce the communication cost, we are using "threads".



Why are we using Thread?

Threads shares ***code, data, heap*** and creates only ***stack*** and ***registers***.

1. Threads share the code
2. Threads share the data and heap
3. Threads have their own stack
4. Threads have their own registers



Thread advantage & disadvantage

Advantage

1. Increase resource efficiency
2. Reducing processing cost
3. Shorter program response time due to simple communication method

Disadvantage

1. If one thread destroys resources in the process, all processes can be damaged.
2. Since resources are shared, synchronization errors can be arise

Background

Windows vs. Threads

AGMService.exe	3804	Running	SYSTEM	00	1,888 K	3	Not allowed
AGSService.exe	3776	Running	SYSTEM	00	1,596 K	2	Not allowed
AppleMobileDeviceS...	3648	Running	SYSTEM	00	2,392 K	8	Not allowed
ApplicationFrameHo...	12336	Running		00	6,812 K	24	Not allowed
audiodg.exe	7972	Running	LOCAL SE...	00	6,100 K	9	Not allowed
backgroundTaskHos...	11536	Suspended		00	0 K	9	Not allowed
chrome.exe	3544	Running		00	154,188 K	28	Not allowed
chrome.exe	4772	Running		00	1,392 K	8	Not allowed
chrome.exe	4036	Running		00	196,924 K	48	Not allowed
chrome.exe	1968	Running		00	26,932 K	14	Not allowed
chrome.exe	8300	Running		00	4,848 K	9	Not allowed
chrome.exe	4968	Running		00	10,000 K	17	Not allowed
chrome.exe	3644	Running		00	41,300 K	26	Not allowed
chrome.exe	2464	Running		00	120,044 K	19	Not allowed
chrome.exe	5372	Running		00	9,076 K	17	Not allowed
chrome.exe	9924	Running		00	43,552 K	17	Not allowed
chrome.exe	10680	Running		00	420,032 K	25	Not allowed
chrome.exe	9356	Running		00	269,428 K	27	Not allowed
chrome.exe	6940	Running		00	7,648 K	17	Not allowed
chrome.exe	7020	Running		00	3,336 K	9	llowed
chrome.exe	3164	Running		00	643,216 K	27	ed
chrome.exe	2028	Running		00	7,672 K	17	Not al
chrome.exe	9220	Running		00	68,448 K	17	Not allowe
chrome.exe	11720	Running		00	260,100 K	18	Not allowed
chrome.exe	12704	Running		00	2,584 K	7	Not allowed
chrome.exe	11904	Running		00	6,956 K	16	Not allowed
conhost.exe	7612	Running		00	340 K	2	Not allowed
csrss.exe	756	Running	SYSTEM	00	1,196 K	14	Not allowed
csrss.exe	1000	Running	SYSTEM	01	1,100 K	14	Not allowed
ctfmon.exe	5436	Running		00	3,584 K	13	Not allowed
dasHost.exe	7032	Running	LOCAL SE...	00	4,856 K	5	Not allowed

Utilization	Speed	
5%	4.04 GHz	
Processes	Threads	Handles
157	2134	66377

- chrome running ~20 processes
- 20 processes using ~240 threads

Thread Optimization

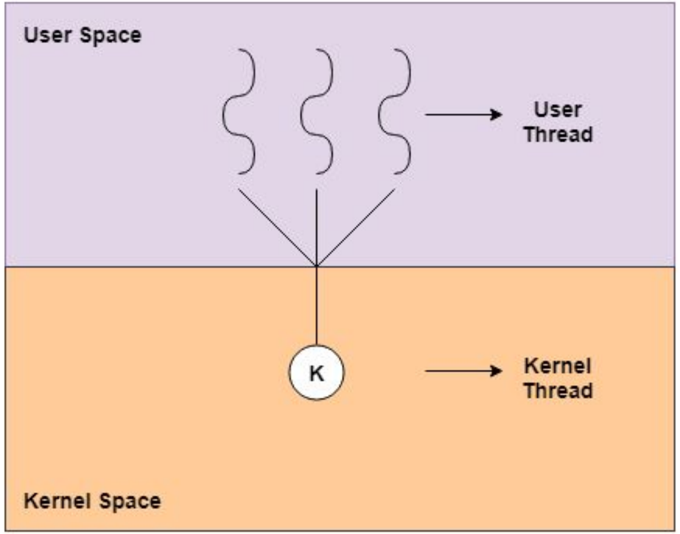
- Processors groups limited to 64 threads
- Can't use more than 50% of CPU towards single application
- Can bypass this limit with custom scheduling

Technical

User-Level and Kernel-Level Threads

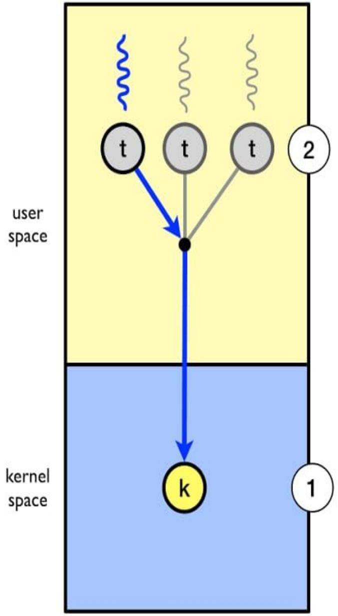
- User-Level threads are implemented by users and the kernel is not aware of their presence

- Kernel-level threads are handled by the OS directly and thread management is done by the kernel



User-Level

Kernel-Level



User-Level

Pros

- Easy and fast to create
- Runs on any OS

Cons

- Multithreaded applications cannot use multiprocessing
- Entire process is blocked if one thread blocks

Kernel-Level

Pros

- Multiple threads can be scheduled on different processors
- Kernel routines can be multithreaded
- If a thread is blocked another thread can still be scheduled

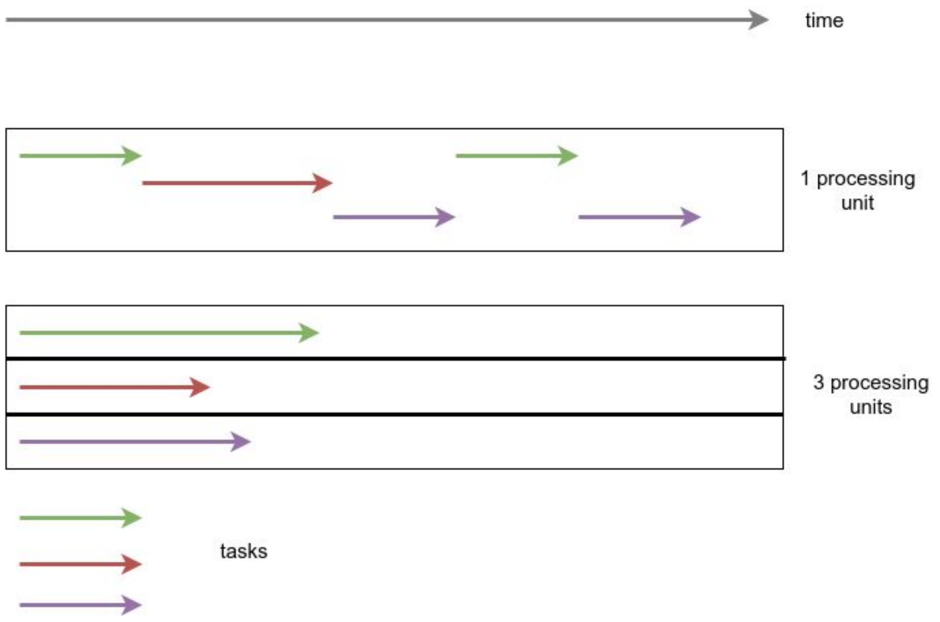
Cons

- Slow to create and manage
- Mode switch is required to transfer control

Concurrency vs Synchronously

- **Synchronously** - Tasks start, run, and stop all at separate intervals

- **Concurrency** - Tasks run simultaneously - Much more efficient



Concurrency vs Synchronously

```
def example():
    print("Function started!")
    sleep(2)
```

```
start = perf_counter()

example()
example()

end = perf_counter()
print(f"{round(end-start, 2)}")
```

```
Function started!
Function started!
4.02
```

34

```
start = perf_counter()

thread1 = threading.Thread(target=example)
thread1.start()

thread2 = threading.Thread(target=example)
thread2.start()

thread1.join()
thread2.join()

end = perf_counter()
print(f"{round(end-start, 2)}")
```

```
Function started!
Function started!
2.01
```

55

```
start = perf_counter()

threads = []
for _ in range(10):
    thread = threading.Thread(target=example)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()

end = perf_counter()
print(f"{round(end-start, 2)}")
```

```
Function started!
Function started!
Function started!
Function started!
Function started!
Function started!
Function started!
Function started!
Function started!
Function started!
2.04
```

Windows Thread Scheduling

Priority Class

Each process belongs to one of the following priority classes:

IDLE_PRIORITY_CLASS
BELOW_NORMAL_PRIORITY_CLASS
NORMAL_PRIORITY_CLASS
ABOVE_NORMAL_PRIORITY_CLASS
HIGH_PRIORITY_CLASS
REALTIME_PRIORITY_CLASS

Priority Level

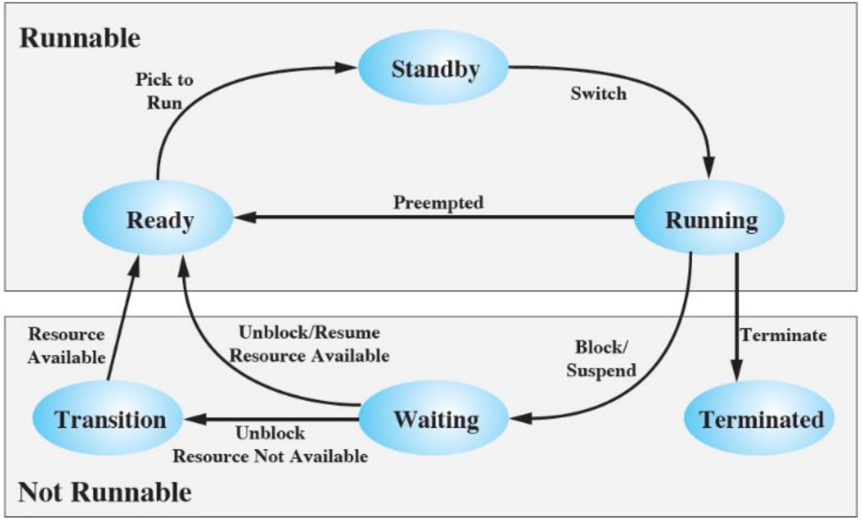
The following are priority levels within each priority class:

THREAD_PRIORITY_IDLE
THREAD_PRIORITY_LOWEST
THREAD_PRIORITY_BELOW_NORMAL
THREAD_PRIORITY_NORMAL
THREAD_PRIORITY_ABOVE_NORMAL
THREAD_PRIORITY_HIGHEST
THREAD_PRIORITY_TIME_CRITICAL

Windows Threading States

Windows has a few different Threading states:

- Ready State
- Standby State
- Running State
- Waiting State
- Transition State
- Terminated State



Reference(APA)

Admin AfterAcademy. (2020, May 1). *What is a thread in OS and what are the differences between a process and a thread?* AfterAcademy. Retrieved March 29, 2022, from https://afteracademy.com/blog/what-is-a-thread-in-os-and-what-are-the-differences-between-a-process-and-a-thread

Baeldung. (2021, October 13). *Process vs. thread*. Baeldung on Computer Science. Retrieved March 29, 2022, from https://www.baeldung.com/cs/process-vs-thread

Bauer, R., & Bauer, R. (2021, August 9). *Threads vs. processes: A look at how they work within your program*. Backblaze Blog | Cloud Storage & Cloud Backup. Retrieved March 29, 2022, from https://www.backblaze.com/blog/whats-the-diff-programs-processes-and-threads/

Javatpoint. (n.d.). *Process vs. thread: Difference between process and thread - javatpoint*. www.javatpoint.com. Retrieved March 29, 2022, from https://www.javatpoint.com/process-vs-thread

MKS075. (2021, December 24). *Difference between process and Thread*. GeeksforGeeks. Retrieved March 29, 2022, from https://www.geeksforgeeks.org/difference-between-process-and-thread/