Parallel and Distributed Computing

Thread

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Outline

- What is thread?
  - Multiple thread application
  - Thread vs Process
  - Advantage and disadvantage of thread
- Thread in Linux

Process review

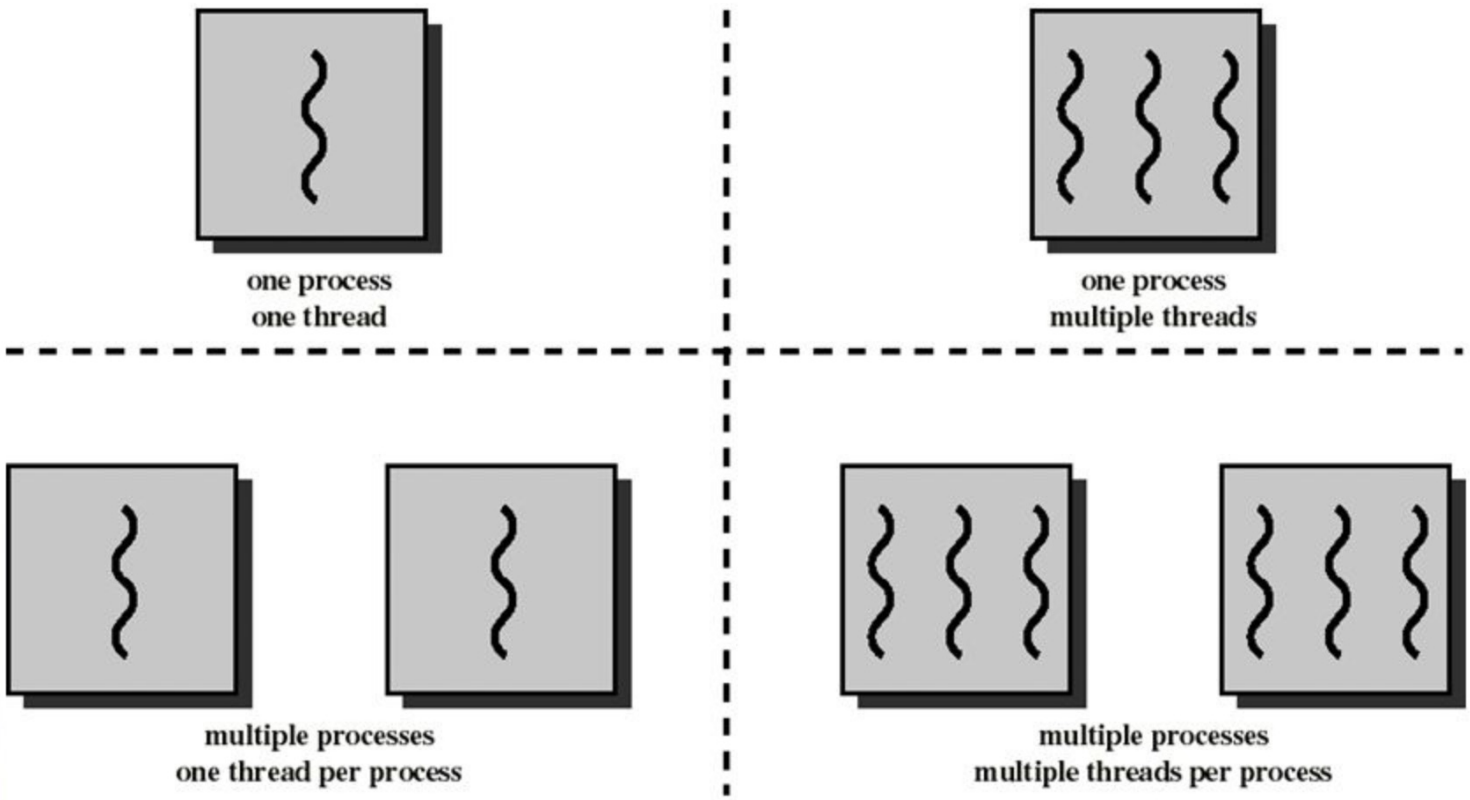
- Definition
  - An instance of a *program* running on a computer
  - An *abstraction* that supports running programs - -> cpu virtualization
  - An *execution stream* in the context of a particular *process state* - -> dynamic unit
  - A *sequential* stream of execution in its *own address space* - -> execution code line by line

What is a thread?

- Thread
  - A finer-granularity entity for execution and parallelism
  - Lightweight process
  - A program in execution without dedicated address space

- Multithreading
  - Running multiple threads within a single process

Finer-granularity entity



Process : thread = 1: N

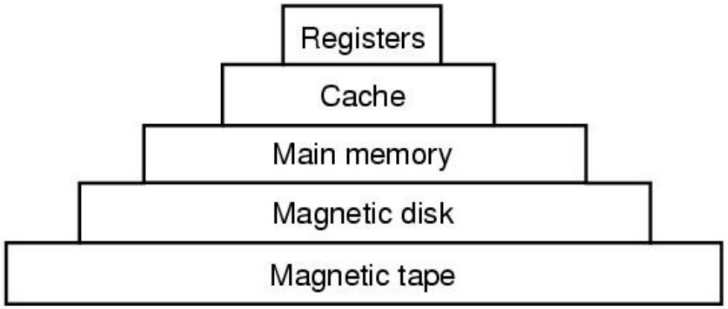
What is a thread?

- Thread
  - A finer-granularity entity for execution and parallelism
  - Lightweight process
  - A program in execution without dedicated address space

- Multithreading
  - Running multiple threads within a single process

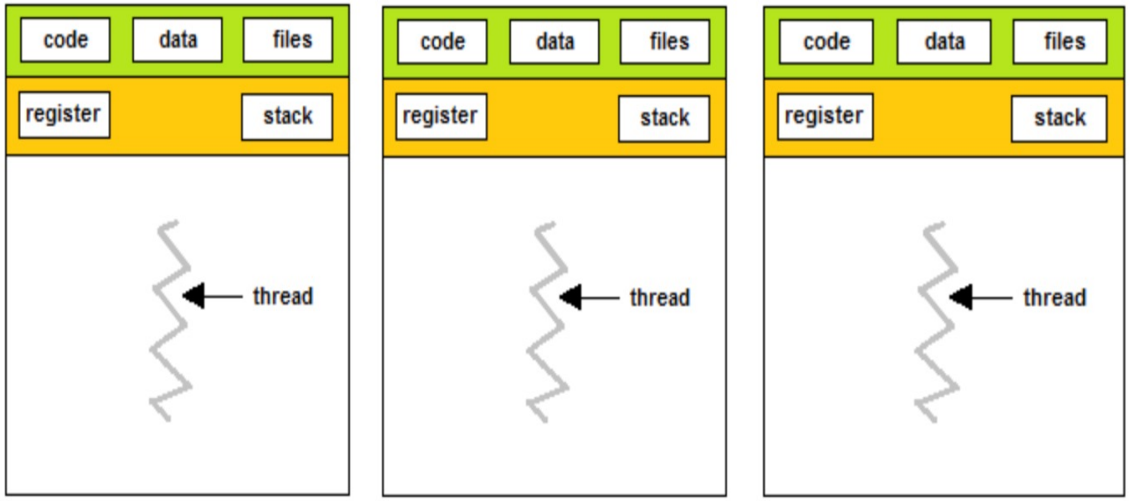
Process review

- Two parts of a process
  - Sequential execution of instructions
  - Process state
    - registers: PC (program counter), SP (stack pointer),…
    - Memory: address space, code, data, stack, heap …
    - I/O status: open files …

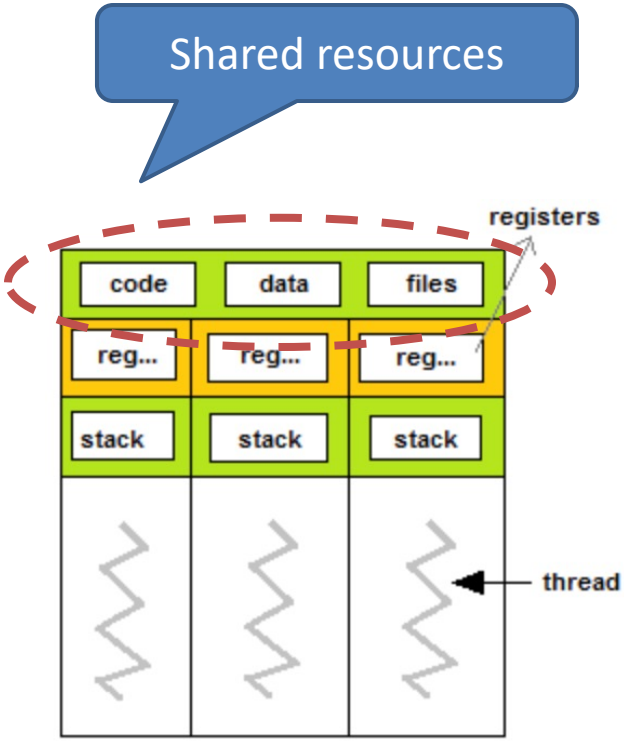


Lightweight process

Occupy more memory,
complex switching (e.g., save old data, switch to new data),
low CPU utilization (e.g., slow context switch)



Three processes



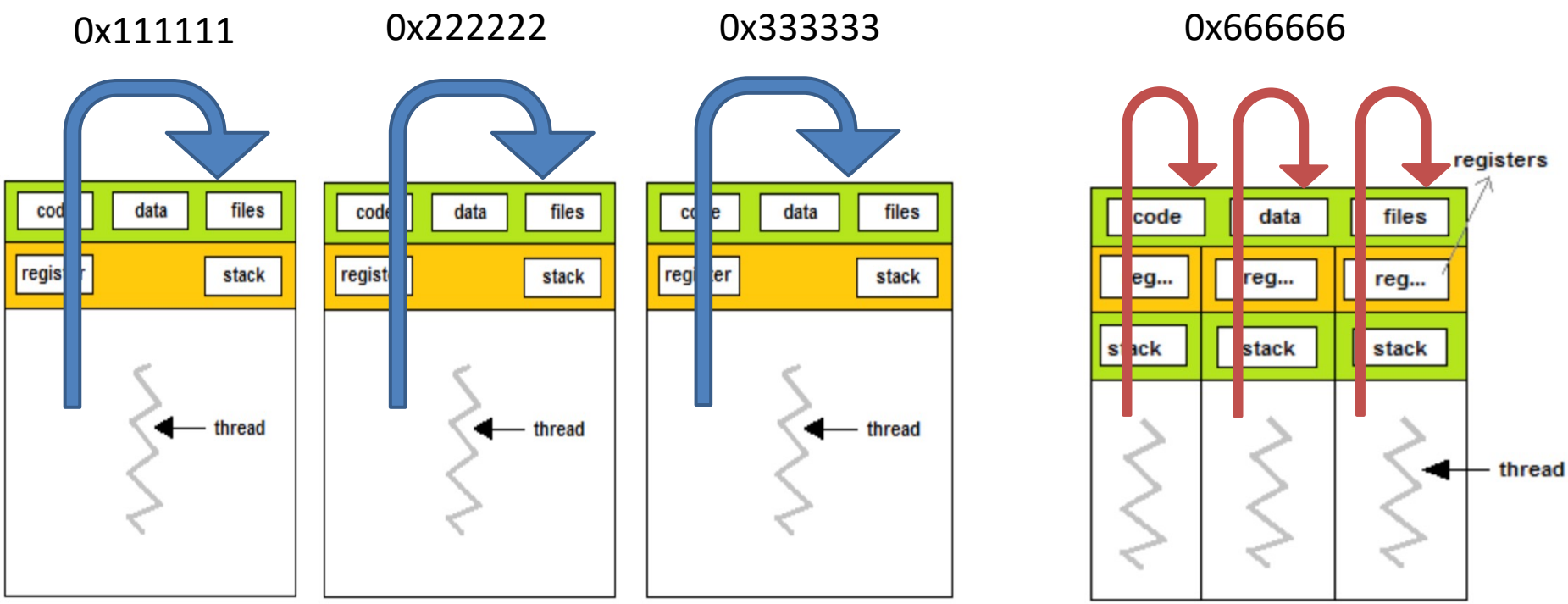
Three threads

What is a thread?

- Thread
  - A finer-granularity entity for execution and parallelism
  - Lightweight process
  - A program in execution without dedicated address space

- Multithreading
  - Running multiple threads within a single process

Dedicated address space



Three processes

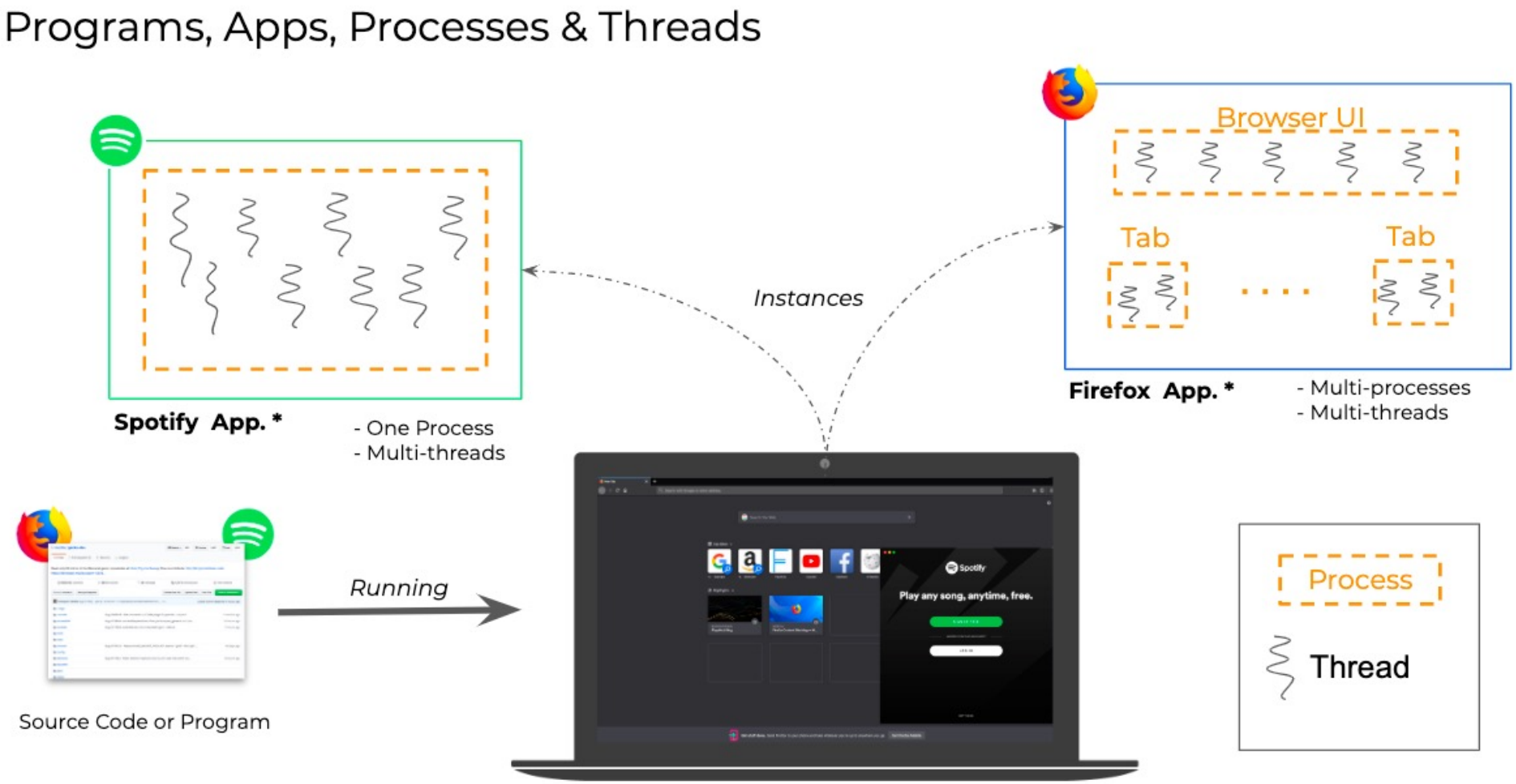
Three threads

What is a thread?

- Thread
  - A finer-granularity entity for execution and parallelism
  - Lightweight process
  - A program in execution without dedicated address space

- Multithreading
  - Running multiple threads within a single process

What is a thread? Multithreading apps



** this image may not reflect the reality for the show-cased apps*

What is a thread? Multithreading apps

Activity Monitor (All Processes)

CPU | Memory | Energy | Disk | Network | Search

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID	User
Activity Monitor	7.0	1:21.83	7	17	691	ksuo
hidd	3.8	13:18.76	5	1	99	_hidd
coreaudiod	1.9	1:44.86	6	55	151	_coreaudiod
Google Chrome	1.3	37:38.24	32	4	683	ksuo
sysmond	0.9	9:24.21	3	1	260	root
launchd	0.8	2:04.06	2	1	1	root
Dock	0.7	30.94	3	0	696	ksuo
iconservicesagent	0.6	12.97	5	0	351	ksuo
Google Chrome Helper	0.6	30.03	8	19	1951	ksuo
Microsoft Outlook	0.4	12:19.10	23	16	686	ksuo
Backup and Sync from Google	0.4	7:55.60	46	17	830	ksuo
Google Chrome Helper	0.2	5:56.29	8	5	739	ksuo
欧路词典	0.2	26.01	8	3	740	ksuo
Microsoft PowerPoint	0.2	7:24.65	19	5	689	ksuo
powerd	0.1	40.42	2	1	57	root
Google Chrome Helper	0.1	6.01	14	1	9082	ksuo
Finder	0.1	1:57.51	7	1	698	ksuo
mds	0.1	1:58.49	8	3	67	root
fseventsd	0.0	43.58	10	3	47	root
launchservicesd	0.0	29.57	4	0	79	root
mds_stores	0.0	2:51.27	5	0	238	root
smcFanControl	0.0	54.43	7	1	831	ksuo

System:	8.14%	CPU LOAD	Threads:	2105
User:	12.30%		Processes:	519
Idle:	79.56%			

Multithreading apps

https://youtu.be/Me0mL44TxW0

Activity Monitor (All Processes)

CPU | Memory | Energy | Disk | Network

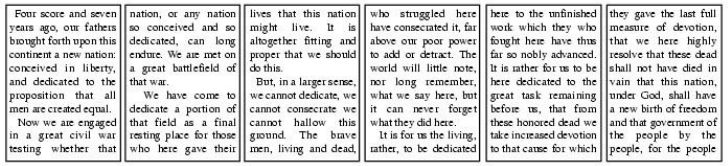
Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID	User
launchservicesd	0.6	39.79	4	1	79	root
sysmond	0.6	12:19.64	5	1	260	root
Backup and Sync from Google	0.5	13:25.83	45	13	830	ksuo
Microsoft Outlook	0.3	14:14.27	24	13	686	ksuo
Microsoft PowerPoint	0.2	20:47.90	17	3	689	ksuo
smcFanControl	0.2	1:11.42	7	0	831	ksuo
Finder	0.2	2:58.42	8	1	698	ksuo
mds	0.2	2:23.95	7	9	67	root
iTerm2	0.2	45.79	5	1	2145	ksuo
Google Chrome Helper	0.2	7:55.58	9	7	739	ksuo
VirtualBox	0.2	23.40	10	2	9676	ksuo
distnoted	0.1	32.92	2	0	293	ksuo
contextstored	0.1	29.99	4	1	150	root
Google Chrome Helper	0.1	34.33	15	1	9220	ksuo
mds_stores	0.1	3:21.65	6	1	238	root
Notification Center	0.1	16.96	4	0	320	ksuo
SystemUIServer	0.1	51.09	4	1	697	ksuo
Google Chrome Helper	0.1	11.04	13	1	9722	ksuo
iconservicesagent	0.1	0.04	4	2	10338	root
VBoxSVC	0.1	9.58	13	6	9681	ksuo
fseventsd	0.1	55.73	10	3	47	root
mdworker_shared	0.1	2.75	4	0	10132	ksuo
Microsoft Word	0.1	3:35.92	17	2	693	ksuo
powerd	0.1	51.69	2	0	57	root
Texpad	0.1	2:22.32	6	0	6606	ksuo
loginwindow	0.1	32.48	3	0	93	ksuo
com.apple.appkit.xpc.openA...	0.1	6.90	4	0	3785	ksuo
Google Chrome Helper	0.1	2.97	13	0	9723	ksuo
storeuid	0.1	5.53	5	1	846	ksuo
Spotlight	0.1	12.77	6	0	701	ksuo
spindump	0.1	4.39	3	1	365	root
com.apple.PressAndHold	0.1	6.63	4	1	453	ksuo
CoreServicesUIAgent	0.1	1.18	5	0	10290	ksuo

System: 17.71%
User: 21.57%
Idle: 60.72%

CPU LOAD

Threads: 2018
Processes: 420

Example 1: A word process with three threads



Four score and seven years ago, our fathers brought forth upon this continent a new nation: conceived in liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war.

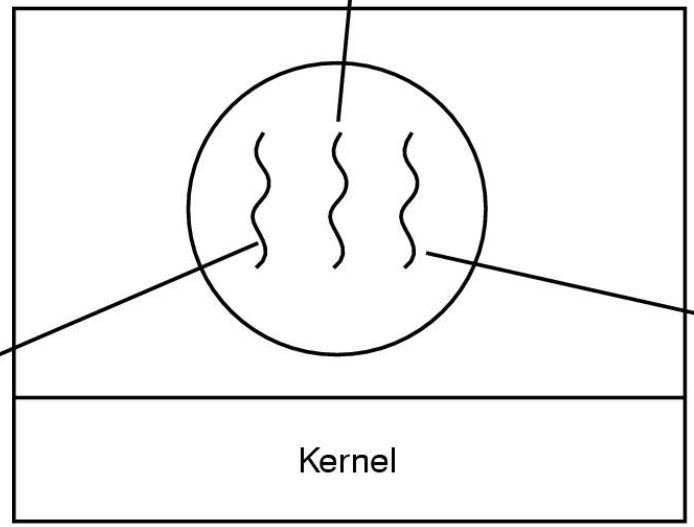
We have come to dedicate a portion of that field as a final resting place for those who here gave their lives that this nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we cannot dedicate, we cannot consecrate we cannot hallow this ground. The brave men, living and dead, who struggled here have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember, what we say here, but it can never forget what they did here.

It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us, that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion, that we here highly resolve that these dead shall not have died in vain that this nation, under God, shall have a new birth of freedom and that government of the people by the people, for the people

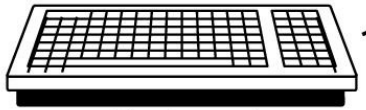
Second thread handles screen display

Why not multiple processes ?

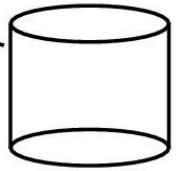


Third thread handles saving the document to disk

First thread handles keyboard input



Keyboard



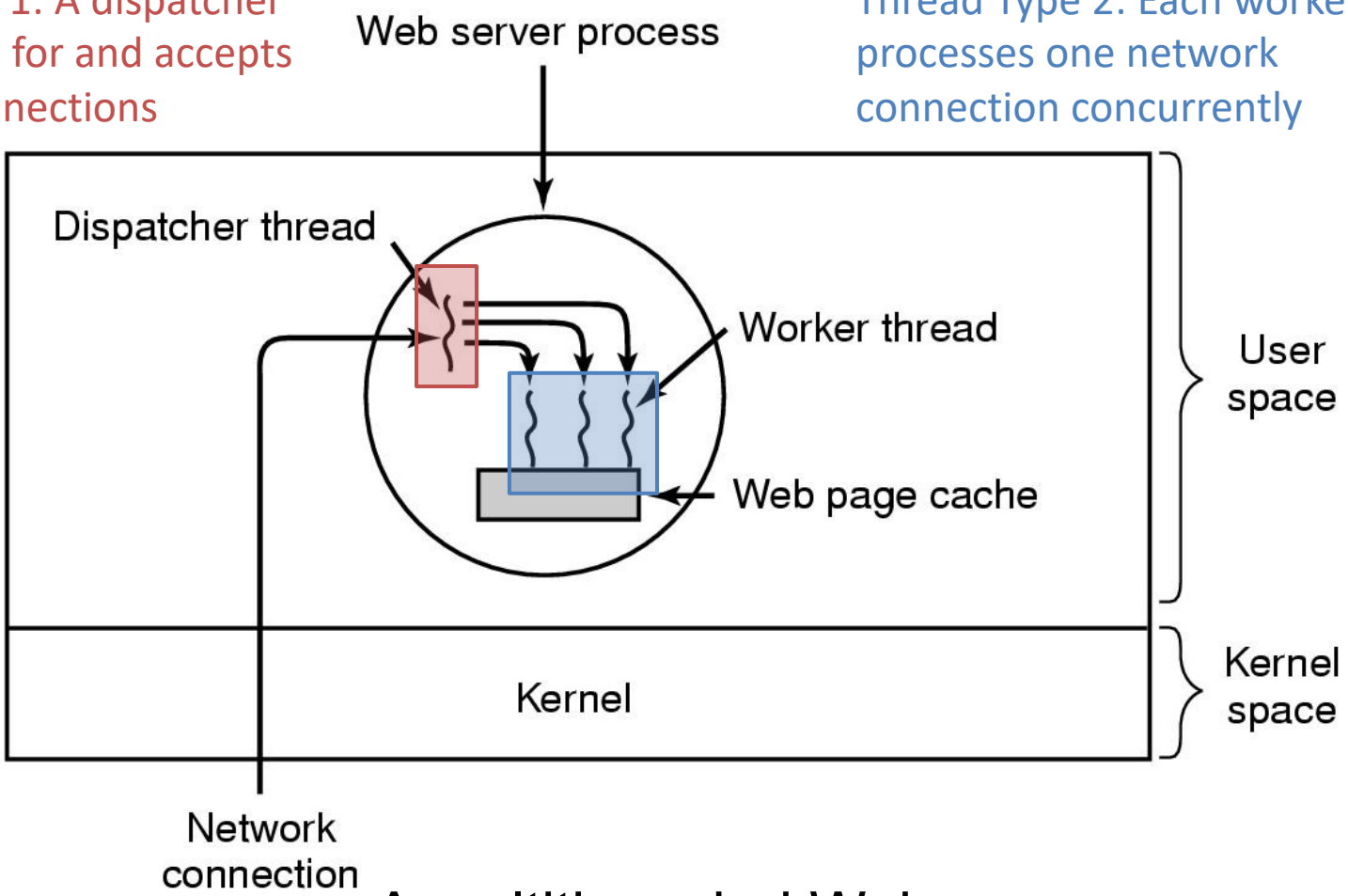
Disk

A word process with three threads.

Example 2: a multi-threaded web server

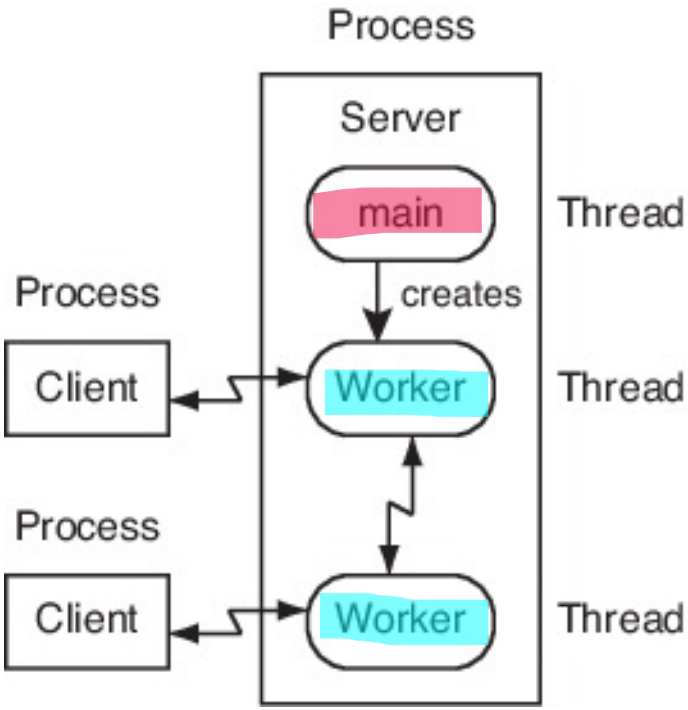
Thread Type 1: A dispatcher thread waits for and accepts network connections

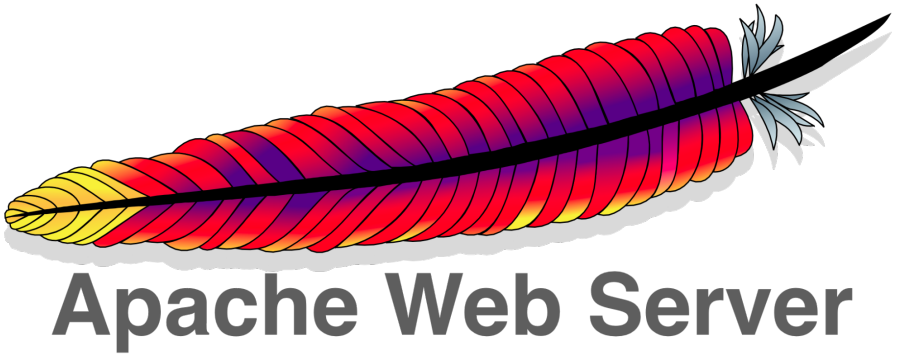
Thread Type 2: Each worker processes one network connection concurrently



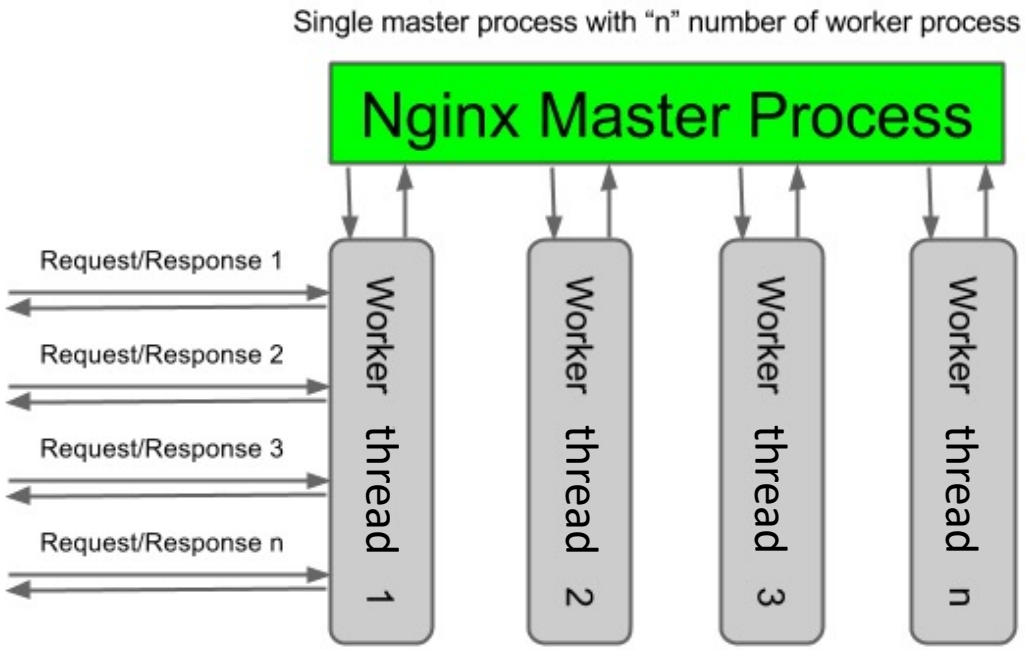
A multithreaded Web server.

Example 2: a multi-threaded web server



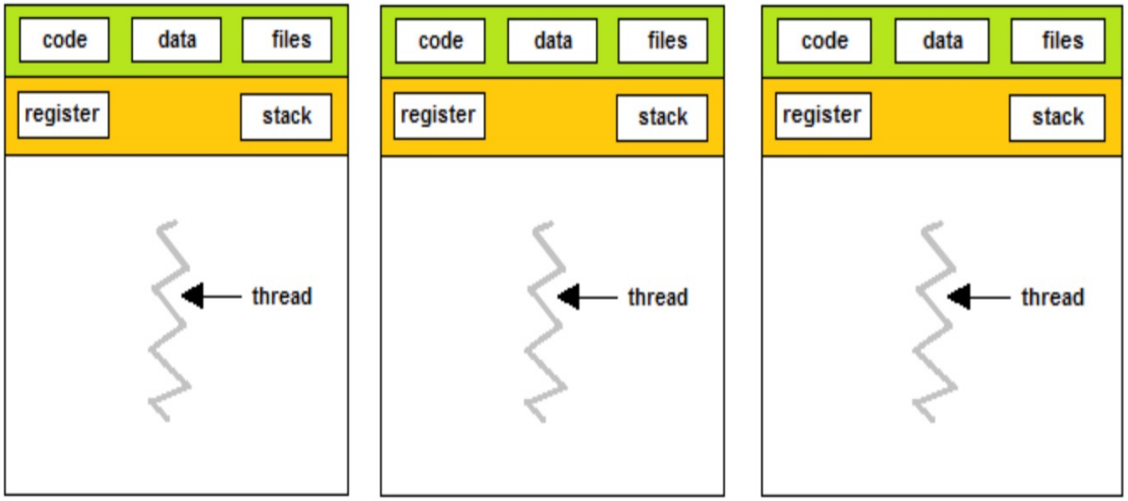


Example 2: a multi-threaded web server

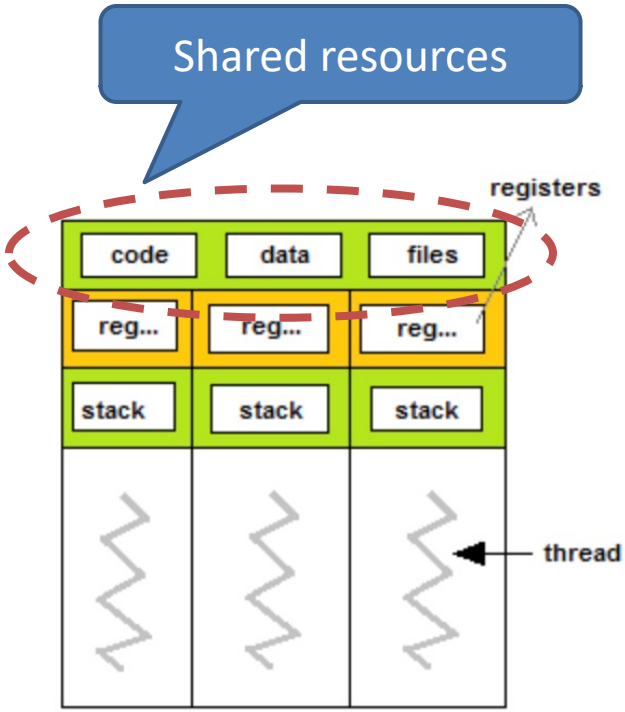


Why not multiple processes ?

Occupy more memory,
complex switching
low CPU utilization
Difficult to communicate
Expensive data sharing



Three processes



Three threads

Why multiple threads

- Good example from Wikipedia: multiple threads within a single process are like multiple cooks trying to prepare the same meal together.

  - Each one is doing one thing.
  - They are probably doing different things.
  - They all share the same recipe but may be looking at different parts of it.
  - They have private state but can communicate easily.
  - They must coordinate!

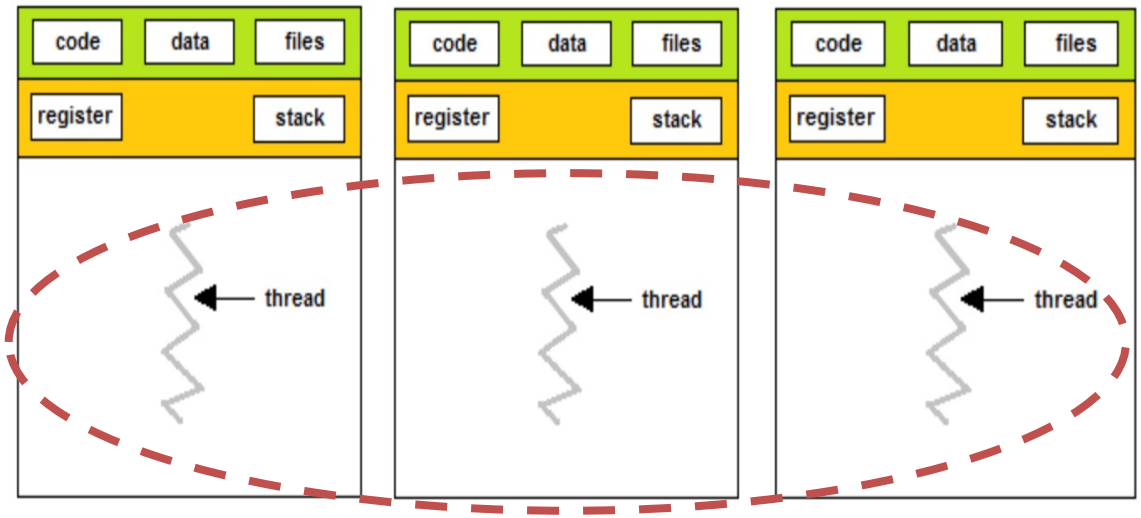
Outline

- What is thread?
  - Multiple thread application
  - Thread vs Process
  - Advantage and disadvantage of thread
- Thread in Linux

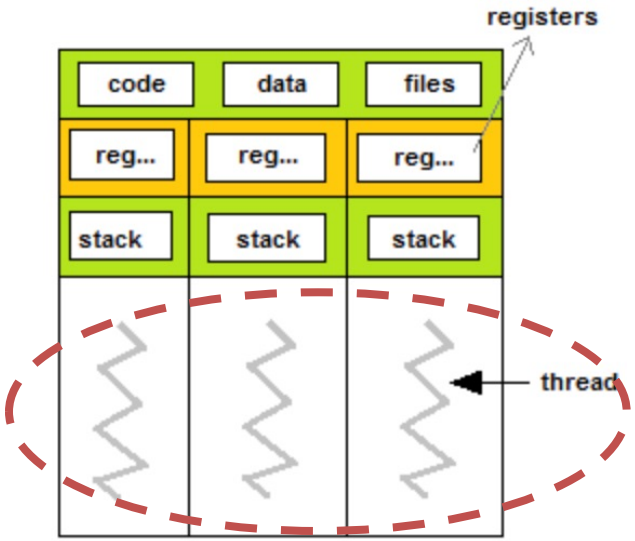
Process vs. Thread comparison

	Process	Thread
Concurrency and protection	?	?
Data structure	?	?
Performance	?	?

Processes vs. Threads (Concurrency and protection)



Three processes

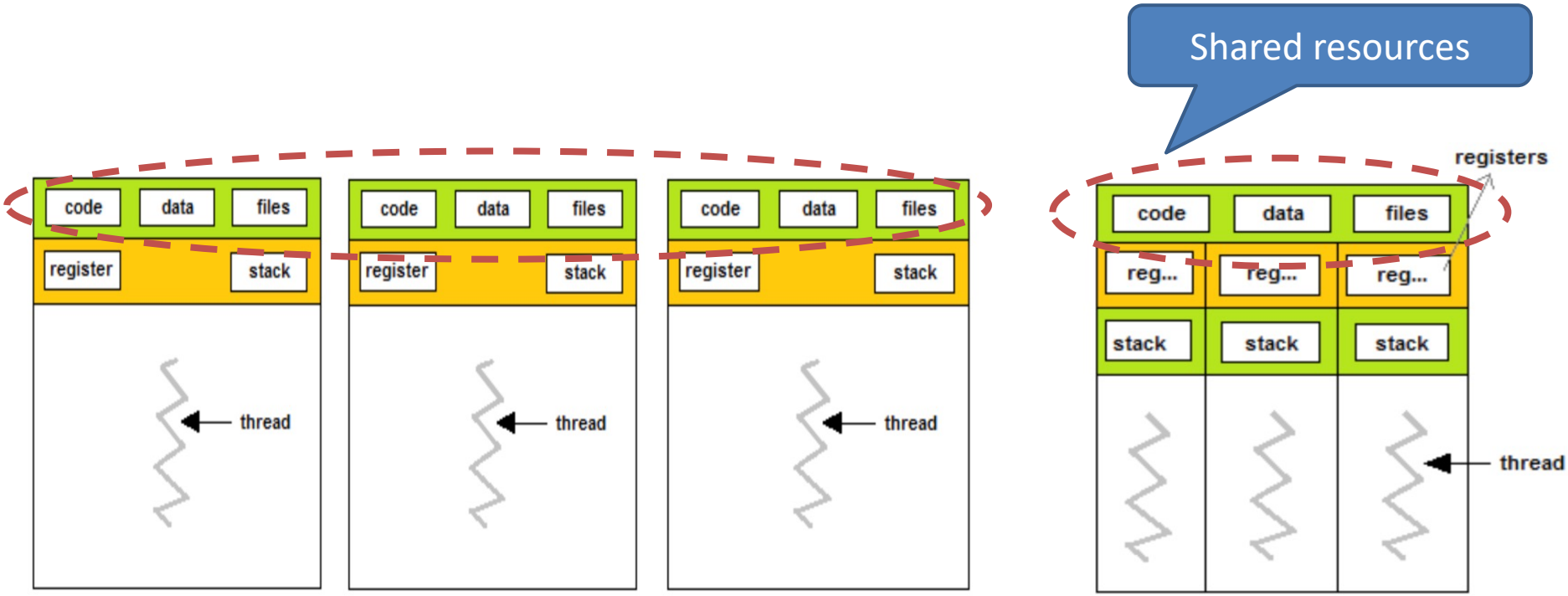


Three threads

Processes vs. Threads (Concurrency and protection)

	Process	Threads
Concurrency	Parallel execution stream of instructions ✓	Maintain parallel execution stream of instructions ✓
Protection		

Processes vs. Threads (Concurrency and protection)



Three processes

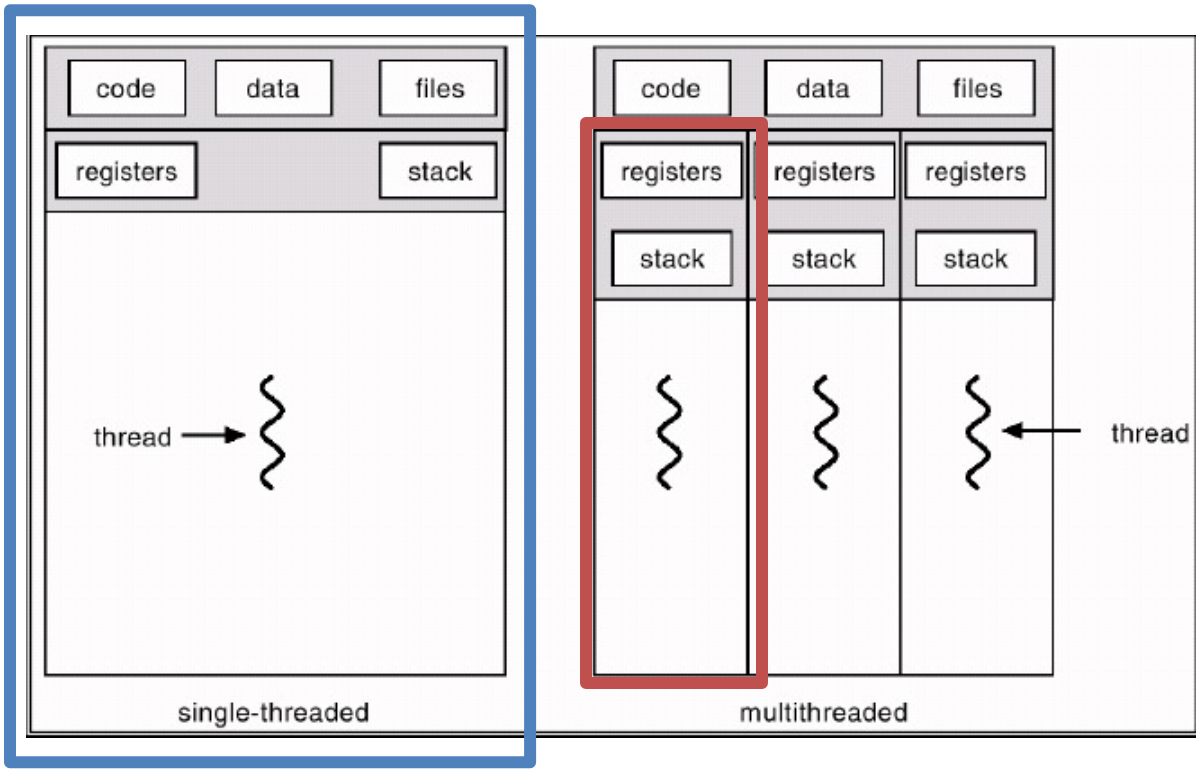
Three threads

Processes vs. Threads (Concurrency and protection)

Separate concurrency from protection

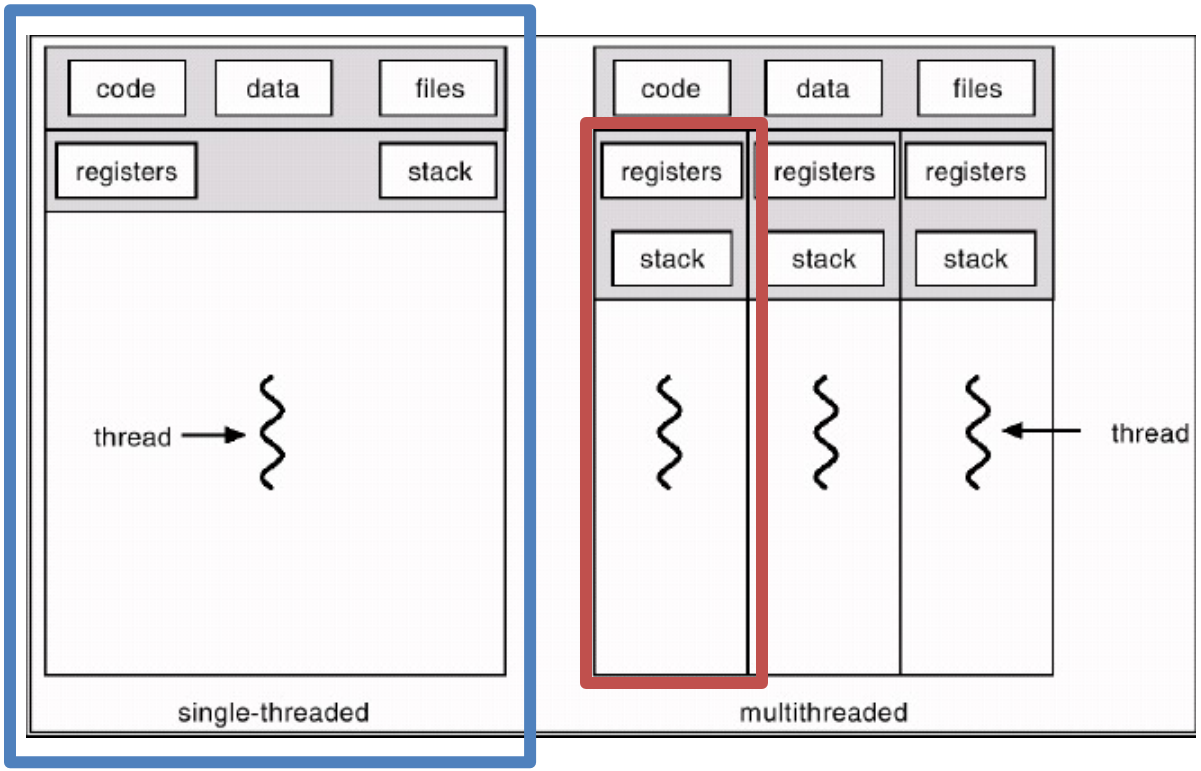
	Process	Threads
Concurrency	Parallel execution stream of instructions ✓	Maintain parallel execution stream of instructions ✓
Protection	A dedicated address space ✓	Share address space with other threads ✗

Processes vs. Threads (Data structure)



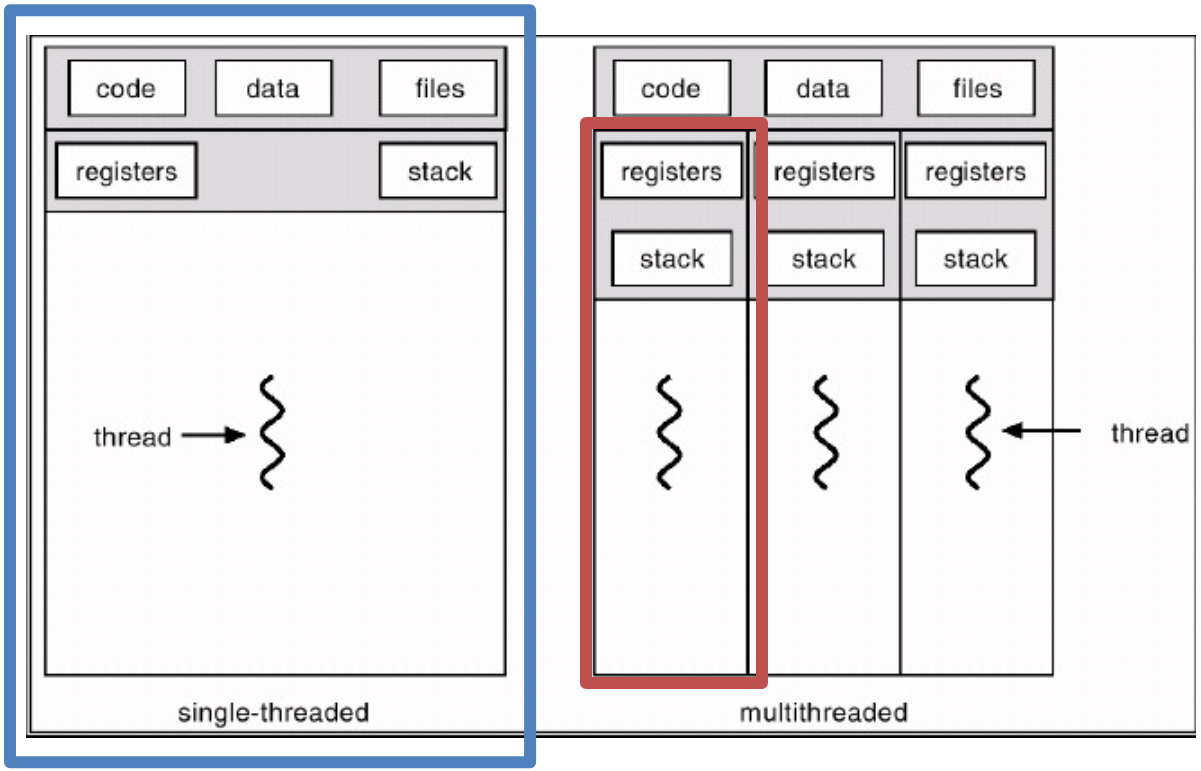
Process	Thread
Have data/code/heap	
Include at lease one thread	
Have own address space, isolated from other processes	

Processes vs. Threads (Data structure)



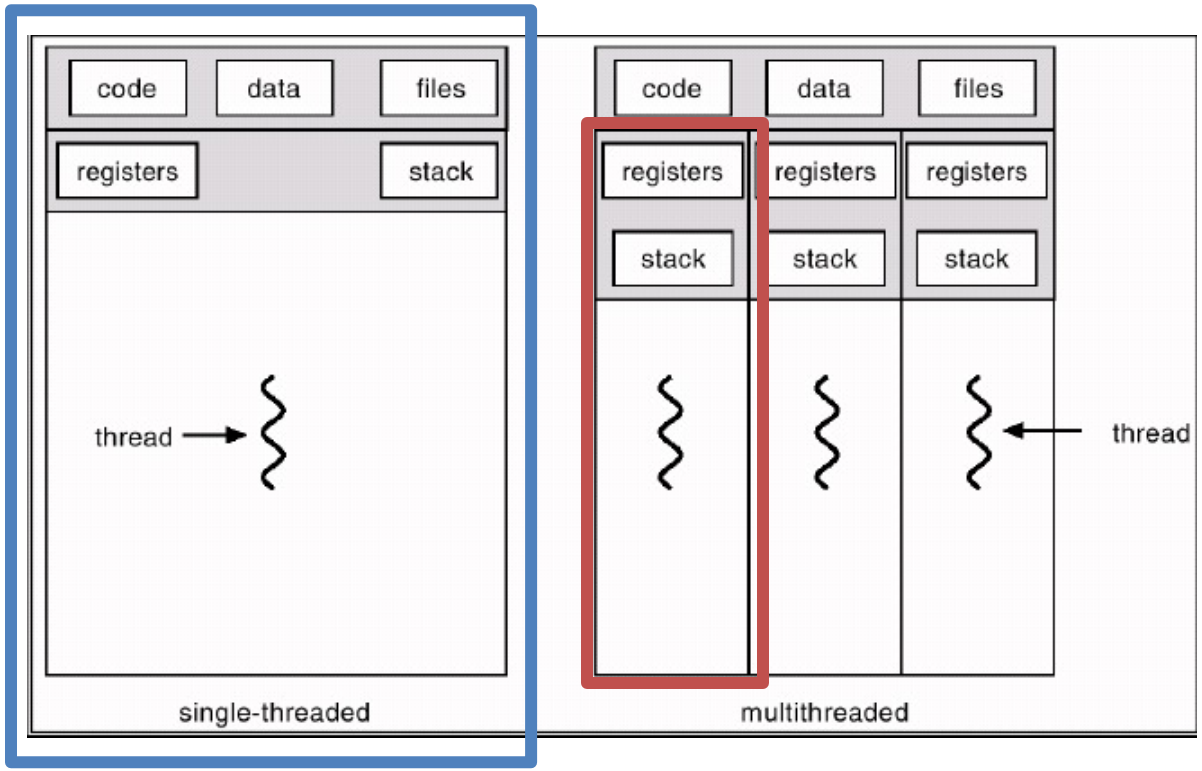
Context	Process	Thread
File pointer	*	
Stack	*	*
Memory	*	
State	*	*
Priority	*	*
I/O state	*	
Authority	*	

Processes vs. Threads (Data structure)



Context	Process	Thread
Scheduling	*	
Statistics	*	
File description	*	
Read/Write pointer	*	
Event/ Signal	*	
Registers	*	*

Processes vs. Threads (Performance)

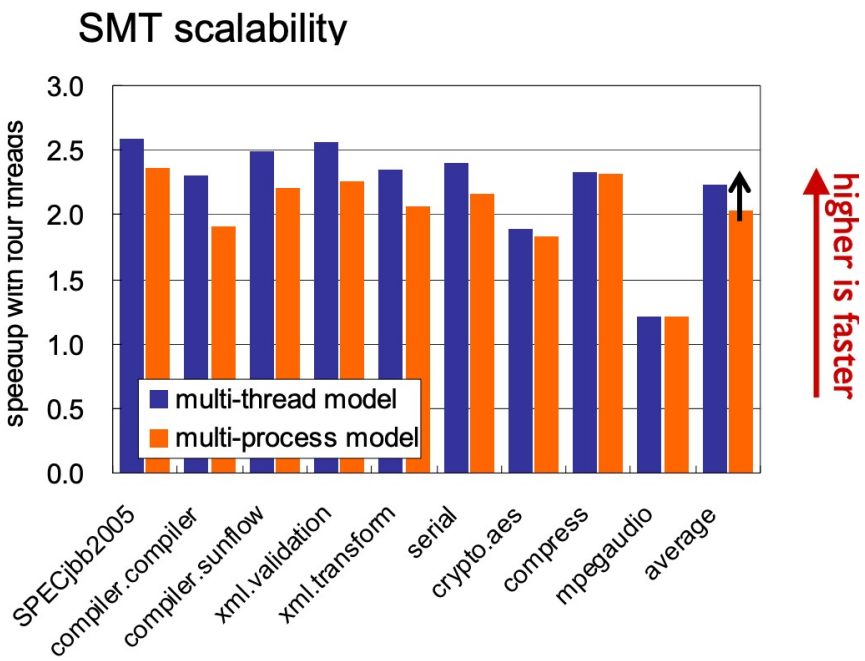


Process	Thread
Expensive to create	Inexpensive to create
Expensive context switching	Inexpensive context switching
IPC can be expensive	Efficient communication

Paper: Performance of Multi-Process and Multi-Thread Processing on Multi-core SMT Processors

- Our results showed that both models (multi-process vs. multi-thread) achieved almost comparable performance, whereas the multi-thread model achieved much better SMT scalability and higher performance.

https://pdfs.semanticscholar.org/dd54/a215131c5eacd997708c5c4612345fe989c7.pdf



multi-thread model was 9.6% faster on average

Outline

- What is thread?
  - Multiple thread application
  - Thread vs Process
  - Advantage and disadvantage of thread
- Thread in Linux

Advantages of Threads

- Efficient creation
  - Only create the thread context
- Express concurrency
  - Lightweight, better performance, higher scalability
- Efficient communication
  - Communication can be carried out via shared data objects within the shared address space

Disadvantages of Threads

- Shared data - -> Security
  - Global variables are shared between threads.
  - Accidental data changes can cause errors.
- Lack of robustness
  - Crash in one thread will crash the entire process.
- Some library functions may not be thread-safe
  - Library Functions that return pointers to static internal memory. E.g. gethostbyname()

Outline

- What is thread?
  - Multiple thread application
  - Thread vs Process
  - Advantage and disadvantage of thread
- Thread in Linux

Processes vs. Threads in Linux

- On Mon, 5 Aug 1996, Peter P. Eiserloh wrote:
  We need to keep a clear the concept of threads. Too many people seem to confuse a thread with a process. The following discussion does not reflect the current state of linux, but rather is an attempt to stay at a high level discussion.

- http://lkml.iu.edu/hypermail/linux/kernel/9608/0191.html
- The way Linux thinks about this (and the way I want things to work) is that there _is_ no such thing as a “process” or a “thread”. There is only the totality of the COE (called "task" by Linux).

Thread creation: clone(), not fork()

- Create new threads in Linux
  - Use `clone()` to create threads instead of using fork()
  - `clone()` is usually not called directly but from some threading libraries, such as pthread.
  - http://man7.org/linux/man-pages/man2/clone.2.html

```c
int main(int argc, char *argv[])
{
        char *stack;                    /* Start of stack buffer */
        char *stackTop;                 /* End of stack buffer */
        pid_t pid;
        struct utsname uts;

        if (argc < 2) {
                fprintf(stderr, "Usage: %s <child-hostname>\n", argv[0]);
                exit(EXIT_SUCCESS);
        }

        /* Allocate stack for child */

        stack = malloc(STACK_SIZE);
        if (stack == NULL)
                errExit("malloc");
        stackTop = stack + STACK_SIZE;  /* Assume stack grows downward */

        /* Create child that has its own UTS namespace;
           child commences execution in childFunc() */

        pid = clone(childFunc, stackTop, CLONE_NEWUTS | SIGCHLD, argv[1]);
        if (pid == -1)
                errExit("clone");
        printf("clone() returned %ld\n", (long) pid);

        /* Parent falls through to here */
```

Clone system call

- No. 56, https://filippo.io/linux-syscall-table/

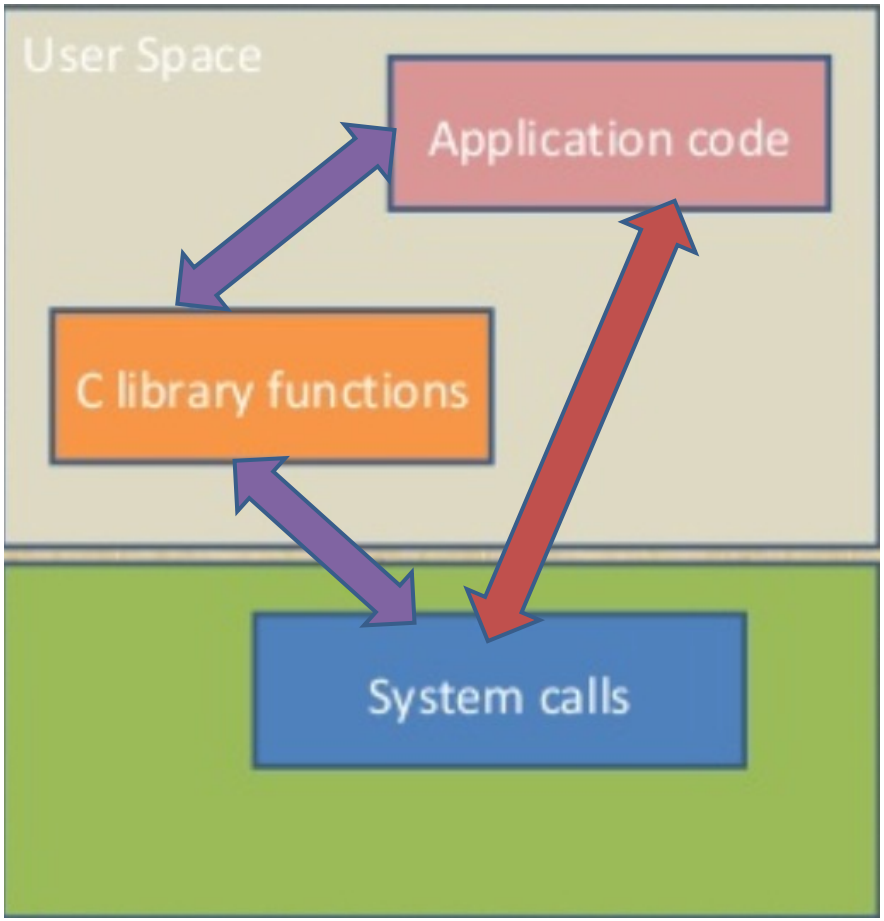
filippo.io

52	getpeername	sys_getpeername	net/socket.c
53	socketpair	sys_socketpair	net/socket.c
54	setsockopt	sys_setsockopt	net/socket.c
55	getsockopt	sys_getsockopt	net/socket.c
56	clone	stub_clone	kernel/fork.c
57	fork	stub_fork	kernel/fork.c
58	vfork	stub_vfork	kernel/fork.c
59	execve	stub_execve	fs/exec.c
60	exit	sys_exit	kernel/exit.c
61	wait4	sys_wait4	kernel/exit.c
62	kill	sys_kill	kernel/signal.c

Clone system call

Pthread_create()
in the user space

Clone()
in the kernel



Pthread Creation Example

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

void *myThread1(void)
{
        int i;
        for (i=0; i<3; i++)
        {
                printf("This is the 1st pthread.\n");
                sleep(1);
        }
}

int main()
{
        int ret=0;
        pthread_t id1;

        printf("This is main thread!\n");

        ret = pthread_create(&id1, NULL, (void*)myThread1, NULL);
        if (ret)
        {
                printf("Create pthread error!/n");
                return 1;
        }

        pthread_exit(NULL);

        return 0;
}
```

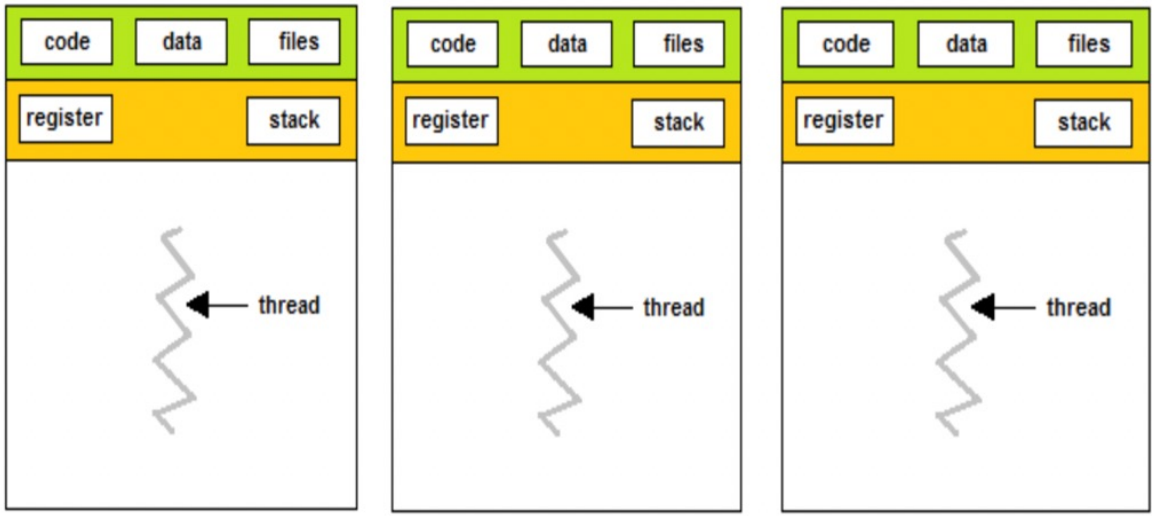
https://github.com/kevinsuo/CS7172/blob/master/pthread_create.c

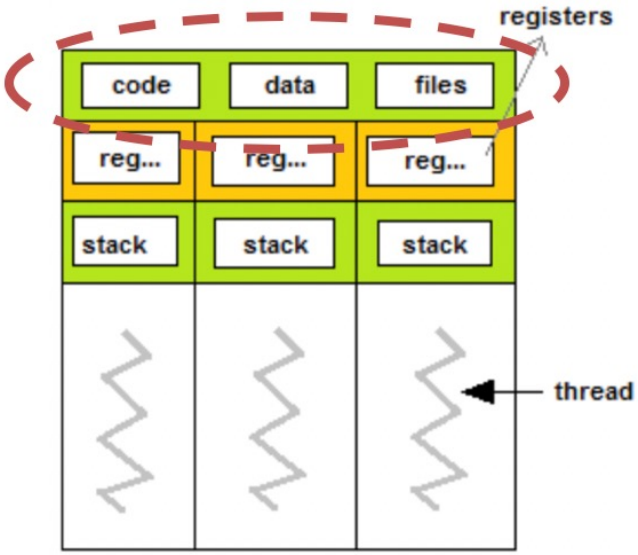
```
pi@raspberrypi ~/Downloads> ./test.o
This is main thread!
This is the 1st pthread.
This is the 1st pthread.
This is the 1st pthread.
```

Clone() in the kernel

Fork vs clone

- Fork:
  - **All resources** in PCB are **copied** from parent process to child process
- Clone:
  - The resources in PCB are **partly copied** from one thread to another thread (the copied context is different in vfork)





Multiple threads → Pandora's Concurrency Box

- Multiple threads → Concurrency
- The illusion of concurrency is both **powerful** and **useful**:
  - It helps us think about how to structure our applications - -> multiple threads apps
  - It hides latencies caused by hardware devices - -> accelerate execution

- Unfortunately, concurrency also creates **problems**:
  - **Coordination**: how do we enable efficient communication between the multiple threads involved in performing a single task?
  - **Correctness**: how do we ensure that shared information remains consistent when being accessed by multiple threads concurrently?

Concurrency and multi-threads

- Unless precisely synchronized, threads may:
  - Be run in **any order**,
  - Be stopped and restarted at **any time**,
  - Remain stopped for **arbitrary lengths of time**.

} **Problems**

- Generally these are **good things** - - the operating system is responsible for how to allocate resources.

Talk about CPU scheduling, memory management, lock, synchronization in future talks

Conclusion

- What is thread?
  - Multiple thread application
  - Thread vs Process
  - Advantage and disadvantage of thread
- Thread in Linux