Operating System Design

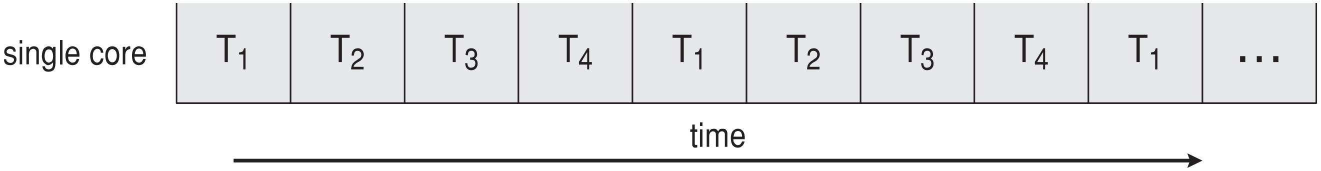
Threads

Neda Nasiriani

Fall 2018

Concurrent vs. Parallel computing

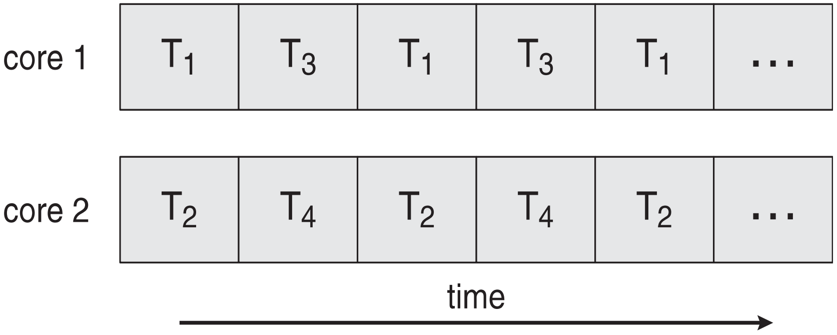
- **Concurrent execution on single-core system:**
  - **Supports more than one task by allowing all the tasks to make progress**



single core	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	…

time

- **Parallelism on a multi-core system:**
  - **Perform more than one task simultaneously**



core 1	T_1	T_3	T_1	T_3	T_1	…
core 2	T_2	T_4	T_2	T_4	T_2	…

time

Multicore or Multiprocessor

- Increasing number of processing cores on computer systems
- Parallelism can be achieved
- Decrease the execution time
- What is the potential performance gain from adding another computing core? (AMDAHL’S LAW)

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

- *S* is serial portion
- *N* processing cores
- If *S=40%* and *N* grows very large what is maximum speed up*?*
  - *2.5* times

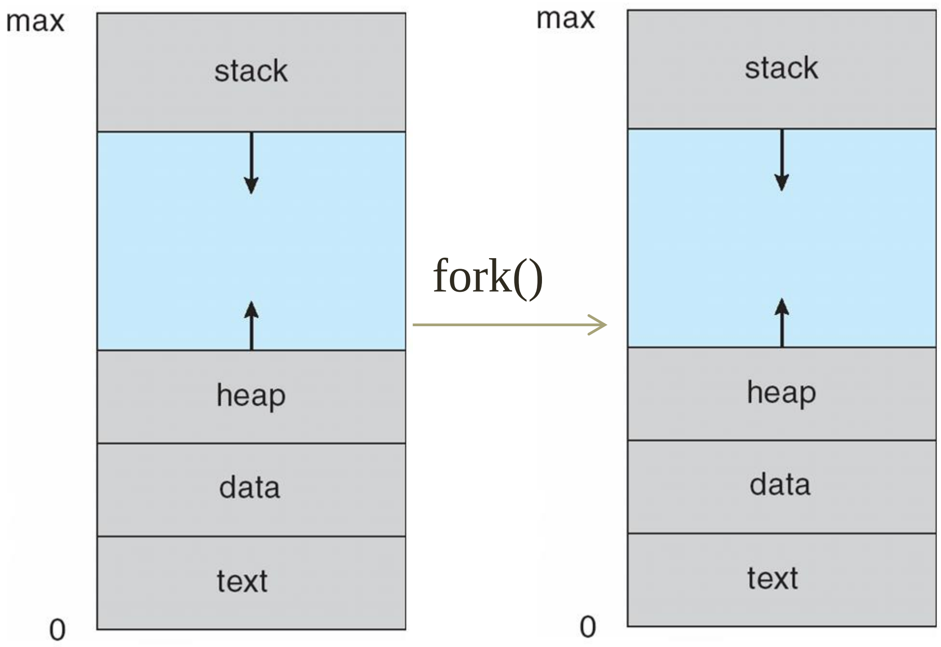
Threads

When multi-process architecture makes sense?

- Web Server Example
  - If you are designing a web server, you need to constantly listen to possible incoming requests
  - Also there could be 1000 of requests every second, how can you address them all in a timely fashion?
- Word Editor Example
  - Allow user to work on a very large file while providing spell checking in the background (without pausing the editing)
  - Apply the keystrokes to the document
  - Automatically saving it without pausing the user work

Word Editor Example

- How can we achieve this based on the things we learnt?
  - Creating a process for spell checking
  - Let’s see how it looks!
  - What happens when the user is entering new data in the parent process? How can the child access it for spell checking?

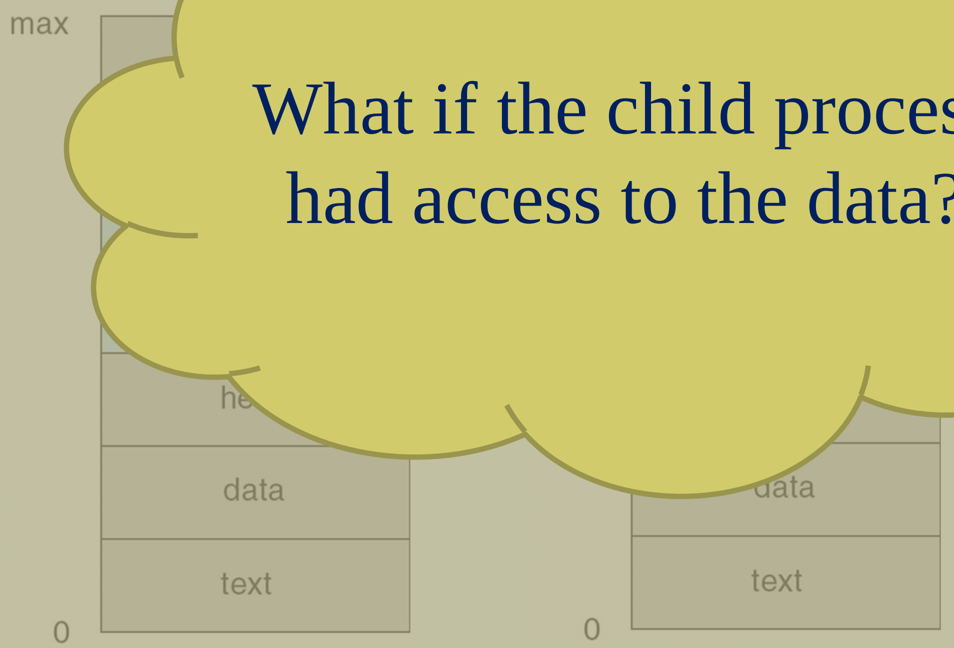


Parent Process

Child Process: Spell Checker

Word Editor Example

- How can we achieve this based on the things we learnt?
  - Creating a process for spell checking
  - Let’s see how it looks!
  - What happens when in the parent process?



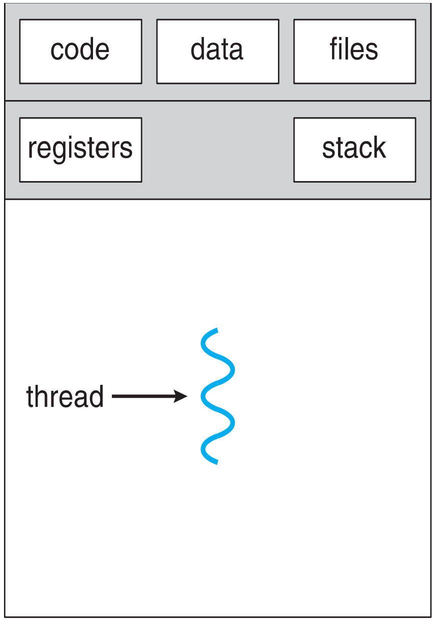
What if the child process had access to the data?

Parent Process

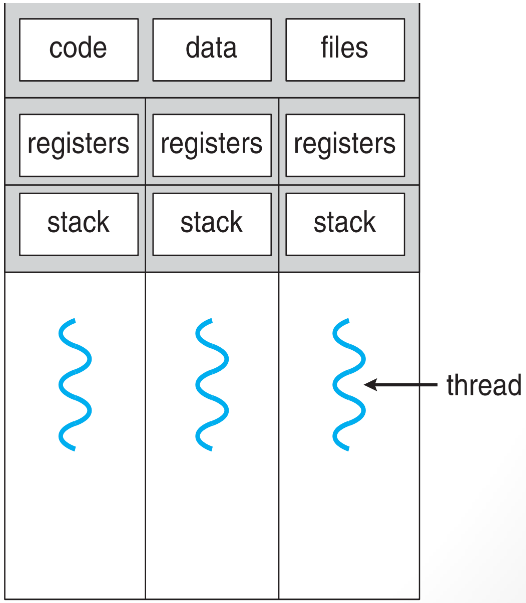
Child Process: Spell Checker

Multi-threaded Execution

- Can we achieve multiple executions using the same data and code?
- How can we have multiple threads of execution (different parts of the code) with access to the same data?
- If we can have multiple executions. Then what info do we need for each thread of execution?
  - Stack
  - Registers



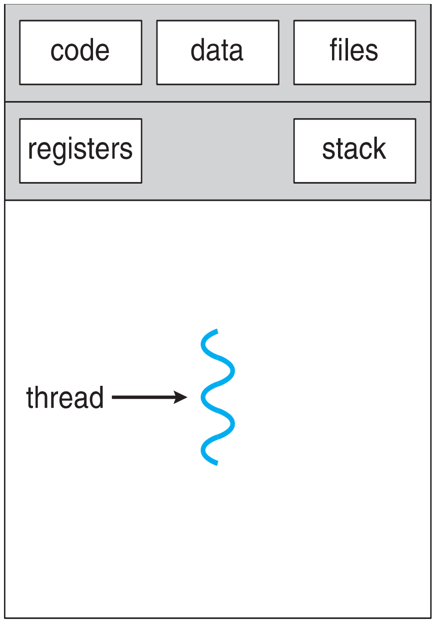
single-threaded process



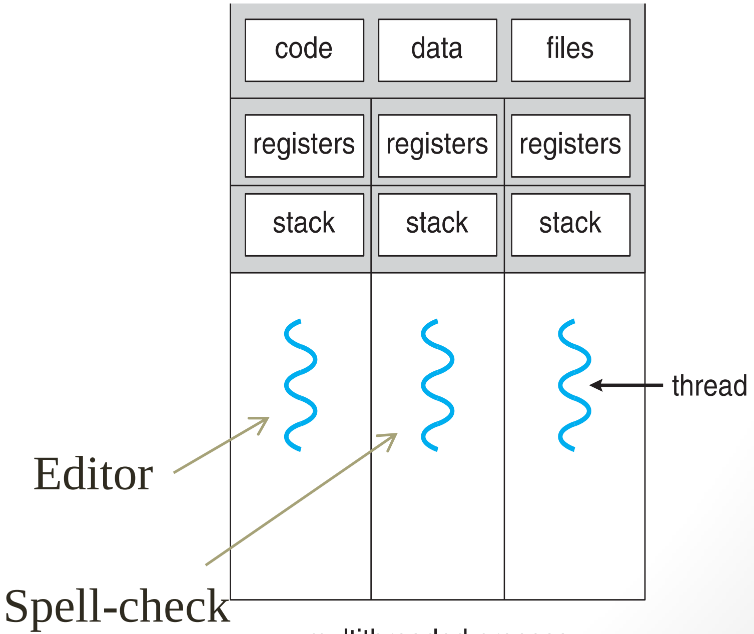
multithreaded process

Multi-threaded Execution

- Let’s think about the word editor example again
  - What if the spell checker thread corrects a word spelling and at the same time the user is changing that word in the editor thread?
  - What can go wrong here?
    - Inconsistency in the data section



single-threaded process

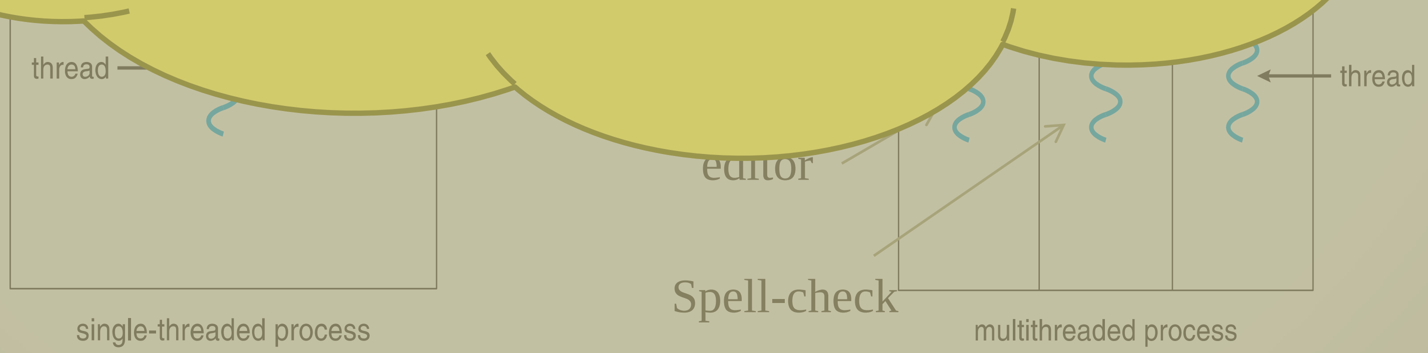


multithreaded process

Multi-threaded Execution

- Let’s think about the word editor example again
  - What if the spell checker thread corrects a word spelling and at the same time the use

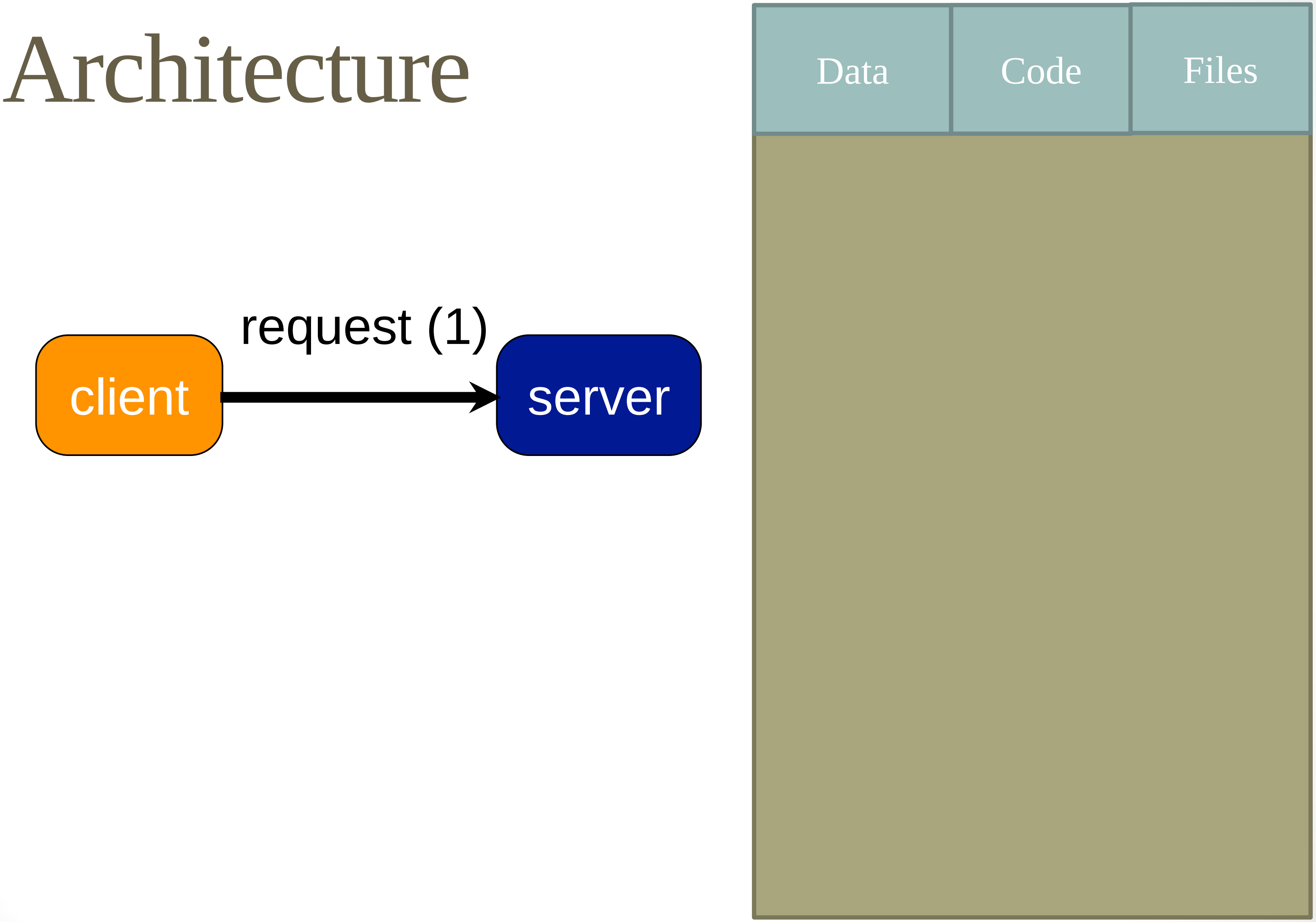
Can we mitigate this inconsistency? YES, We can avoid this using synchronization techniques that we will see in chapter 5.



Web Server Example

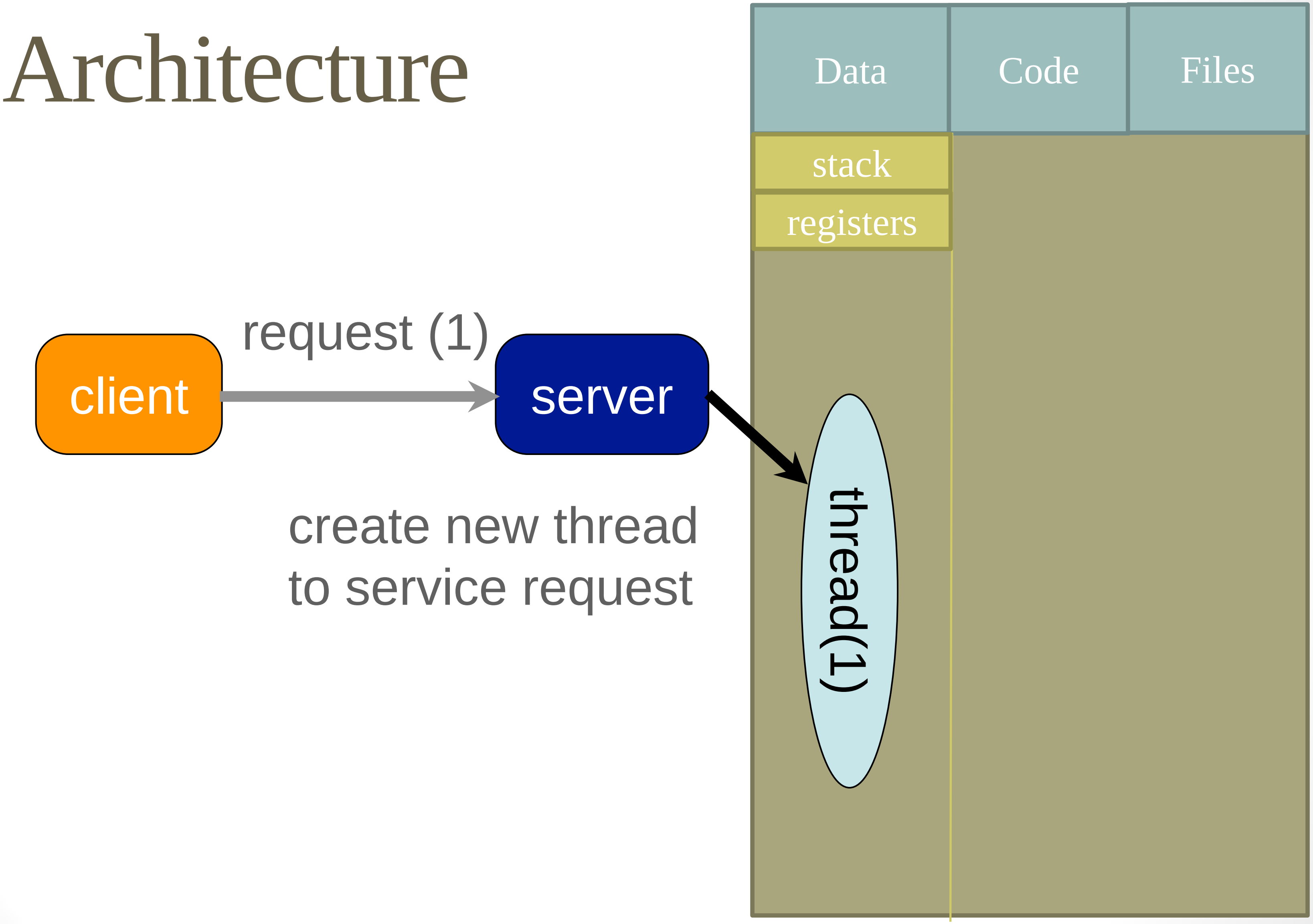
- Assume the web server is servicing search request (google search engine)
- Each request to be served is of similar nature (repetitive code) and has to work on the same information (repetitive data)
- What if we could have multiple executions of the same search code within the same process?
- Let’s see how it looks!

Multithreaded Server Architecture



Based on slides from Luiz F. Perrone

Multithreaded Server Architecture



client

request (1)

server

create new thread to service request

Data

Code

Files

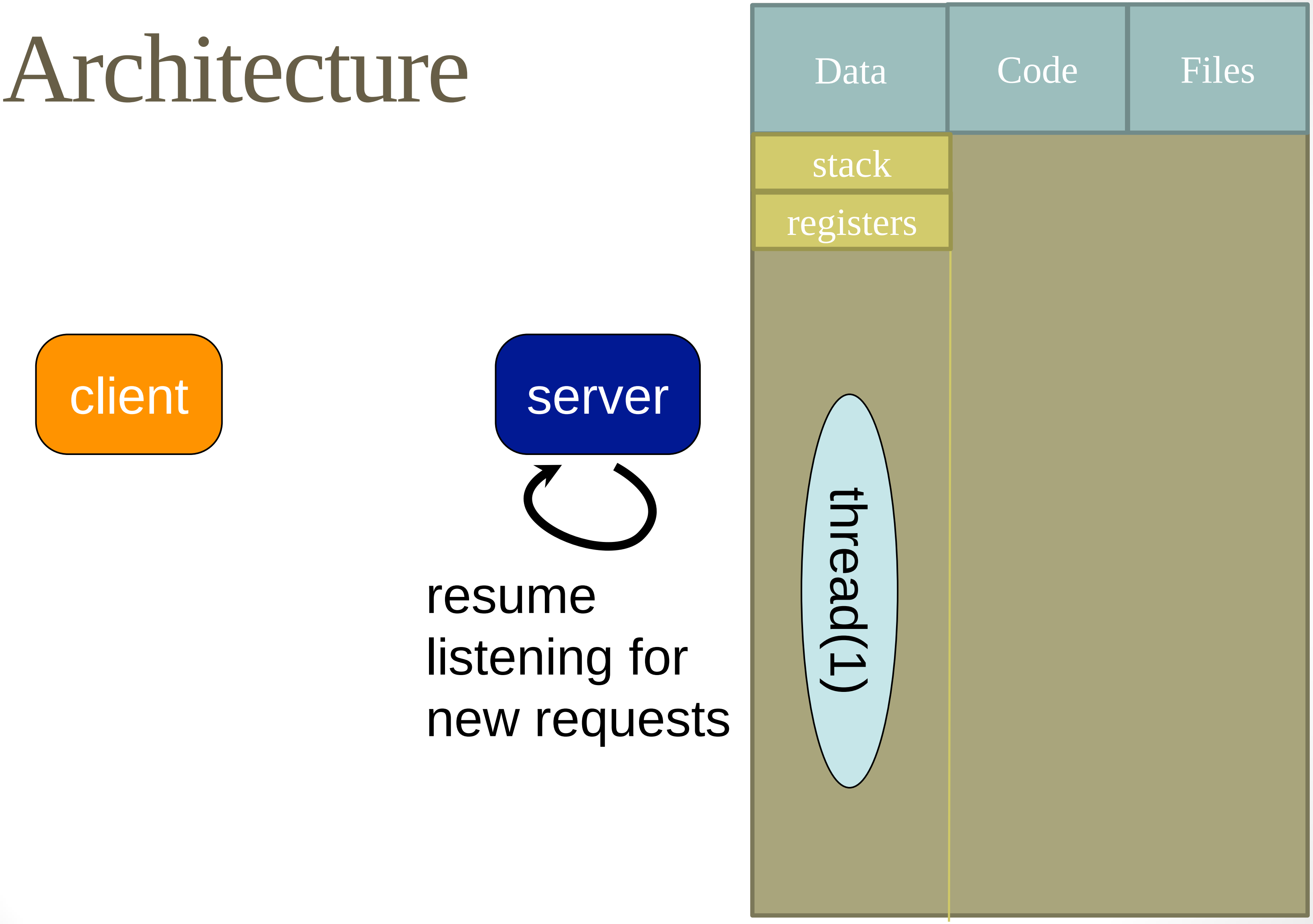
stack

registers

thread(1)

Based on slides from Luiz F. Perrone

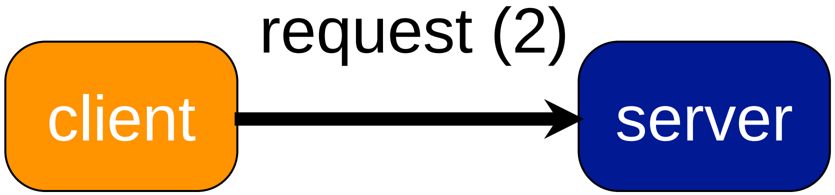
Multithreaded Server Architecture

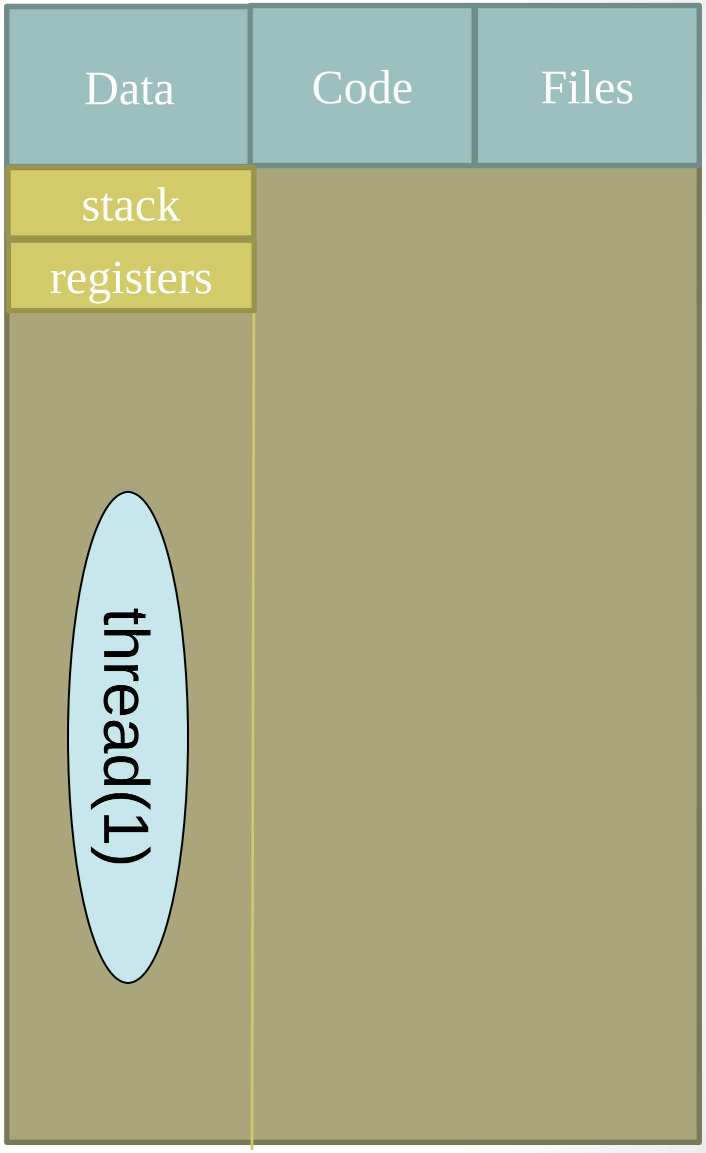


resume listening for new requests

Based on slides from Luiz F. Perrone

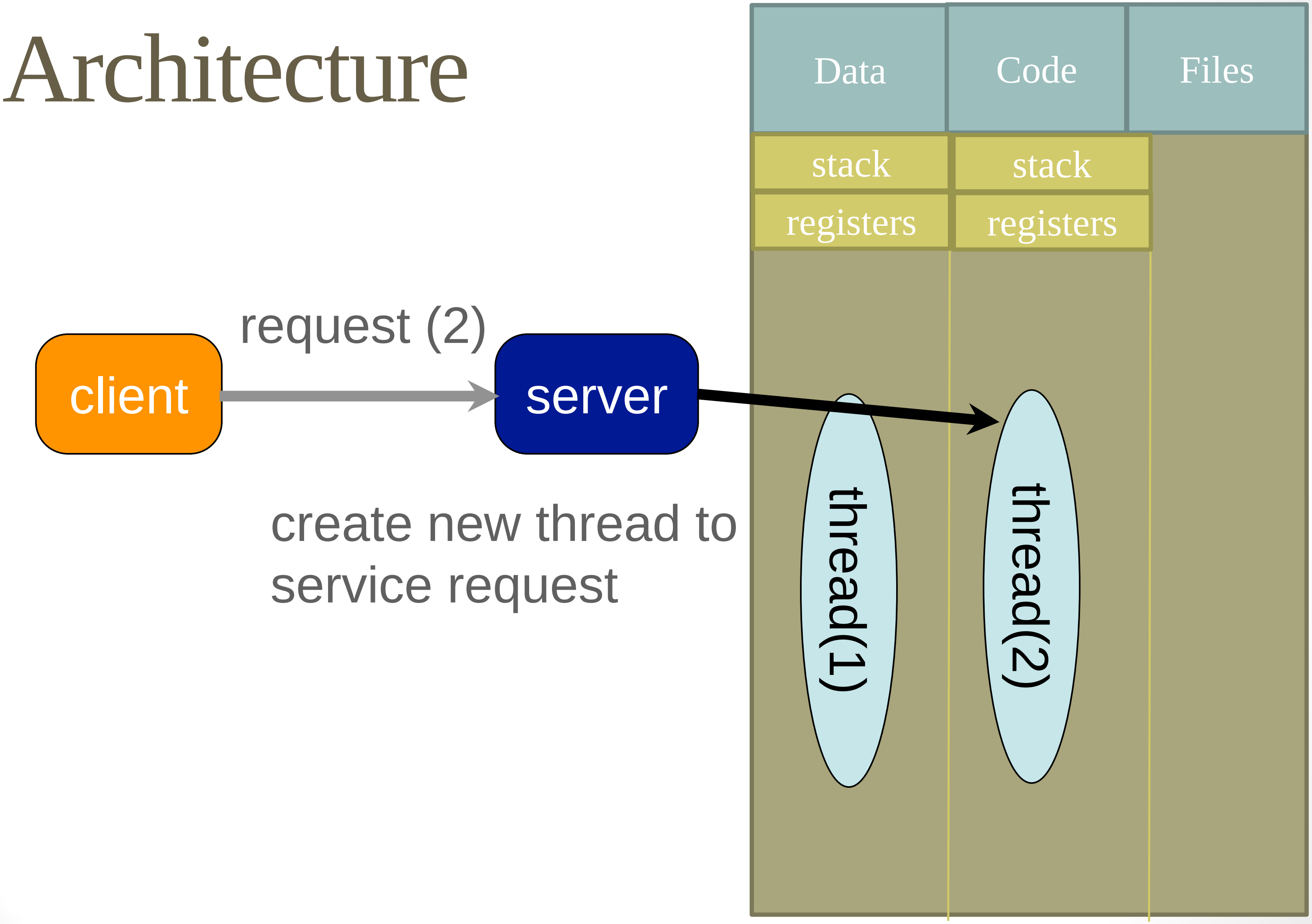
Multithreaded Server Architecture





Based on slides from Luiz F. Perrone

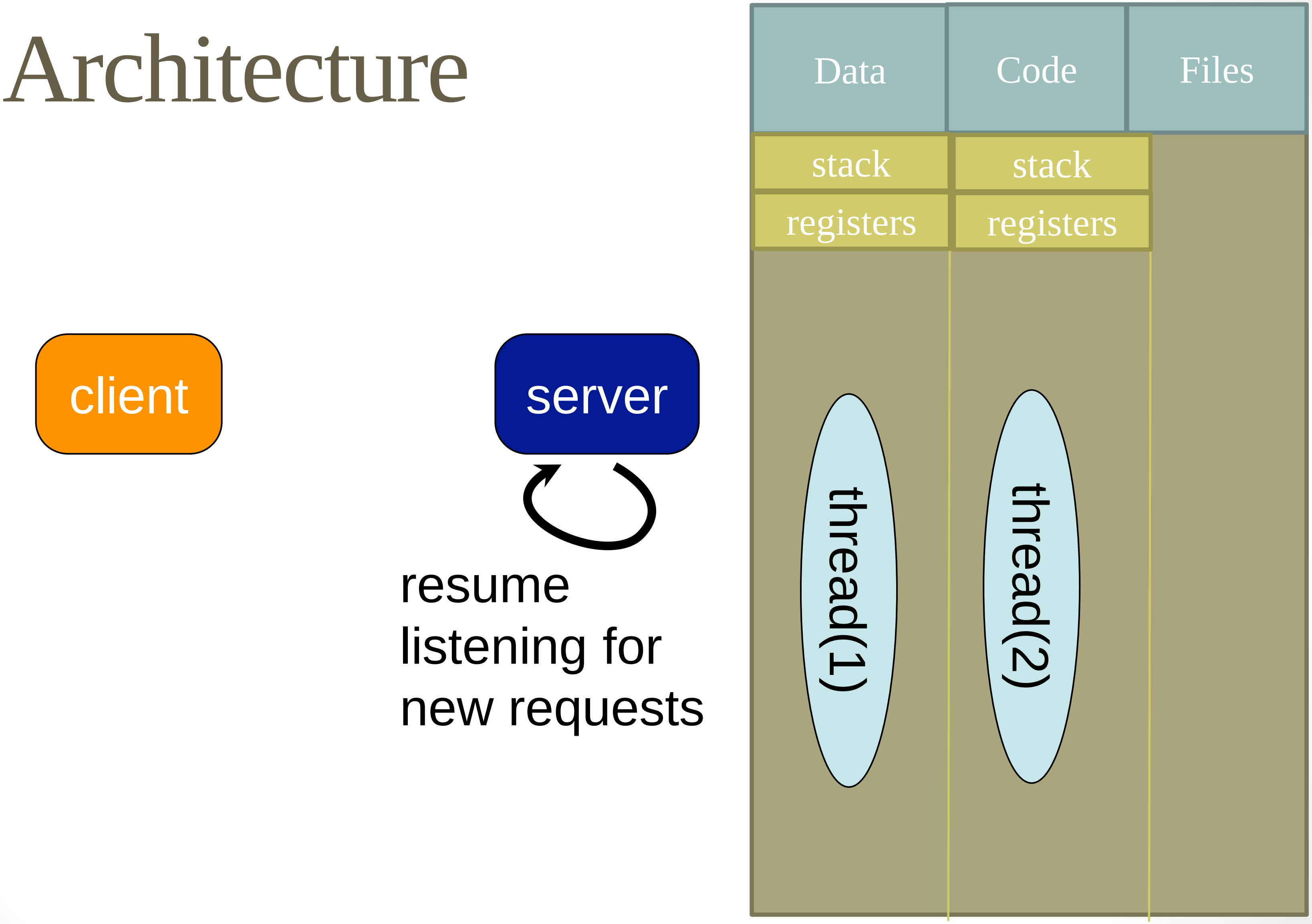
Multithreaded Server Architecture



create new thread to service request

Based on slides from Luiz F. Perrone

Multithreaded Server Architecture



resume listening for new requests

Based on slides from Luiz F. Perrone

Threads vs. Processes

- What are the advantages of Threads over Processes
  - Light weight
  - Processes are costly to create (around 30X time more time)
  - Context switching processes can take up to 5X more time
  - More Efficient
  - Sharing data is easier
  - PCBs are large data types while TCB are way smaller
- What are the disadvantages and challenges of Threads
  - We need consistency when accessing the shared data
  - We should implement synchronization method among threads accessing the shared data
  - Can complicates execution

FYR: Why Threads?

- **Responsiveness:** multiple threads can be executed in parallel (in multi-core machines)
- **Resource sharing:** multiple threads have access to the same data, sharing made easier
- **Economy:** the overhead in creating and managing threads is smaller
- **Scalability:** more processors (or cores), more threads running in parallel

Create Threads

```
NAME
    pthread_create - create a new thread

SYNOPSIS
    #include <pthread.h>

    int pthread_create(pthread_t *thread,
                       const pthread_attr_t *attr,
                       void *(*start_routine) (void *),
                       void *arg);

    Compile and link with -pthread.
```

Based on slides from Luiz F. Perrone

Example

```
/* COMPILE WITH: gcc thread-ex.c -lpthread -o thread-ex */
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 5
#define SLEEP_TIME 3

void *sleeping(void *); /* forward declaration to thread routine */

int main(int argc, char *argv[]) {
int i;
pthread_t tid[NUM_THREADS]; /* array of thread IDs */
for ( i = 0; i < NUM_THREADS; i++)
  pthread_create(&tid[i], NULL, sleeping,(void *)SLEEP_TIME);

for ( i = 0; i < NUM_THREADS; i++)
  pthread_join(tid[i], NULL);

printf("main() reporting that all %d threads have terminated\n", i);
return (0);
} /* main */
```

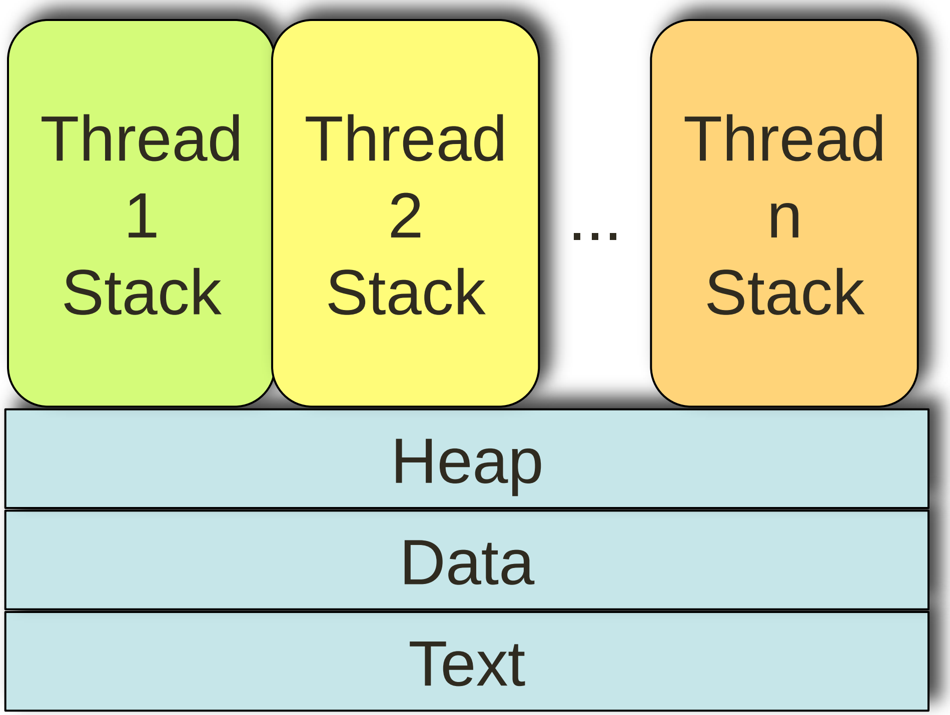
Passing arguments to threads

```
void * sleeping(void *arg) {
  int sleep_time = (int)arg;
  printf("thread %ld sleeping %d seconds ...\n",
  pthread_self(), sleep_time);
  sleep(sleep_time);
  printf("\nthread %ld awakening\n", pthread_self());
  return (NULL);
}
```

- A thread can take parameter(s) pointed by its **arg** and
- can return a pointer to some memory location that stores
- its results. Gotta be careful with these pointers!!!

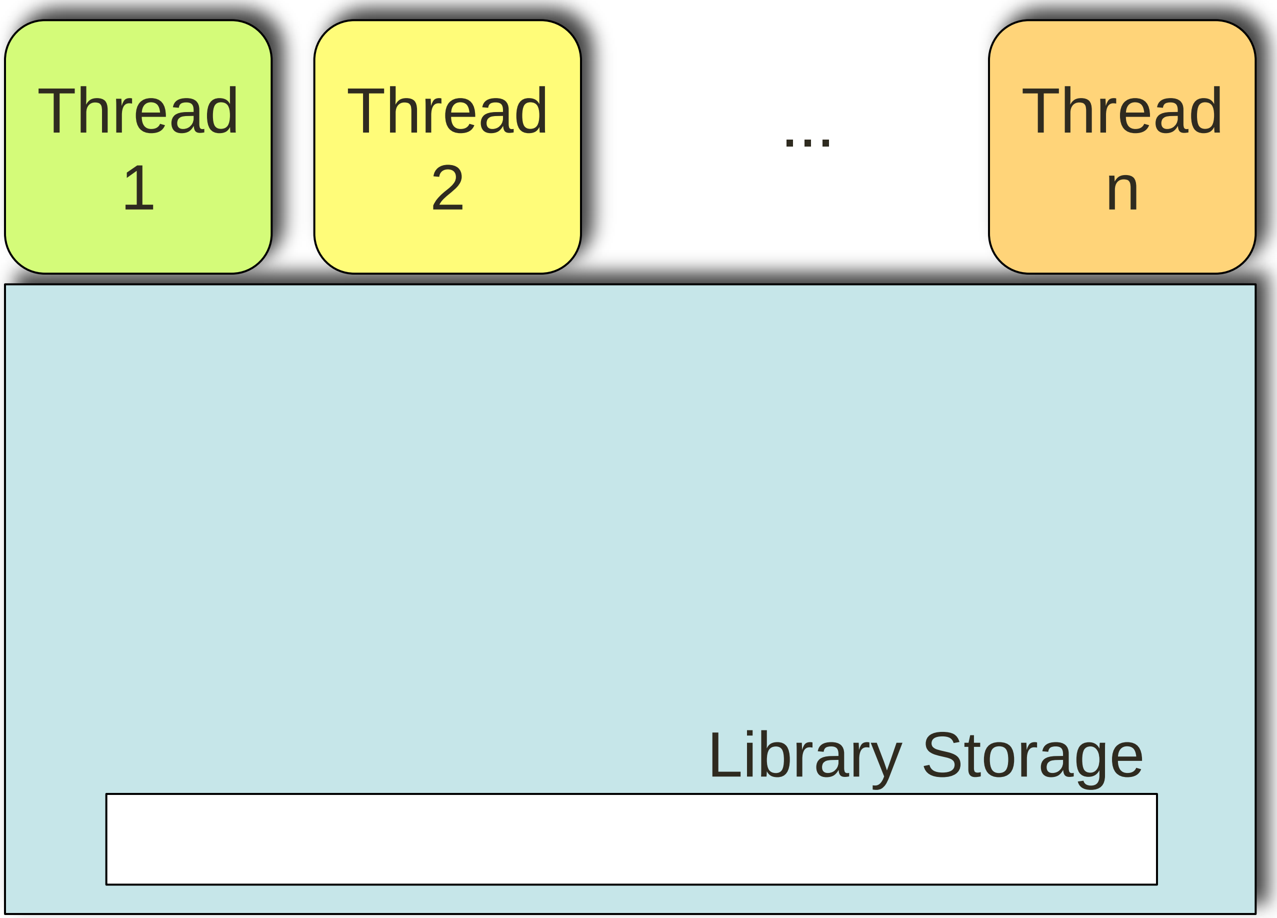
Activity!

Shared Memory Model

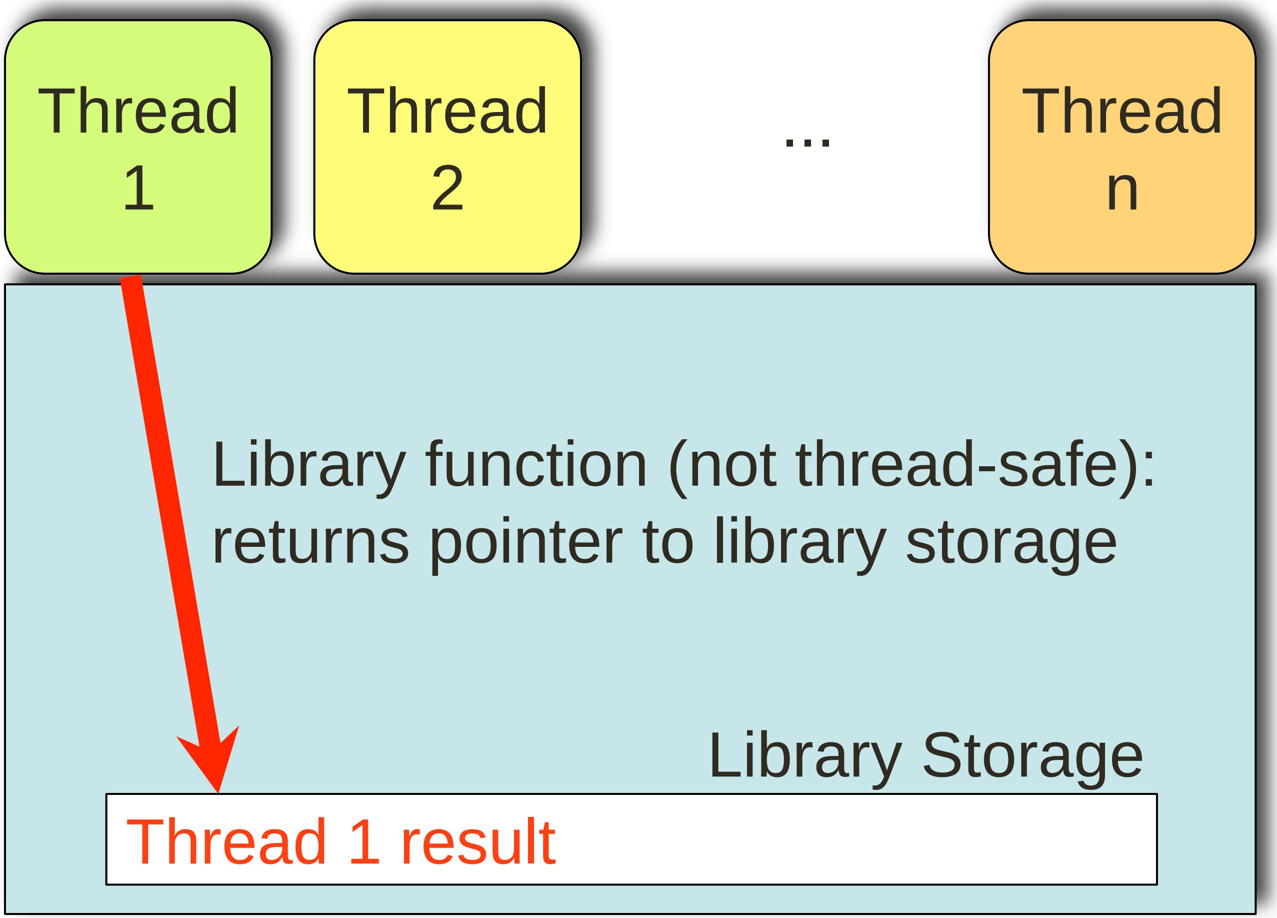


- All threads have access to the same global, shared memory
- Threads also have their own private data (how?)
- Programmers are responsible for protecting globally shared data

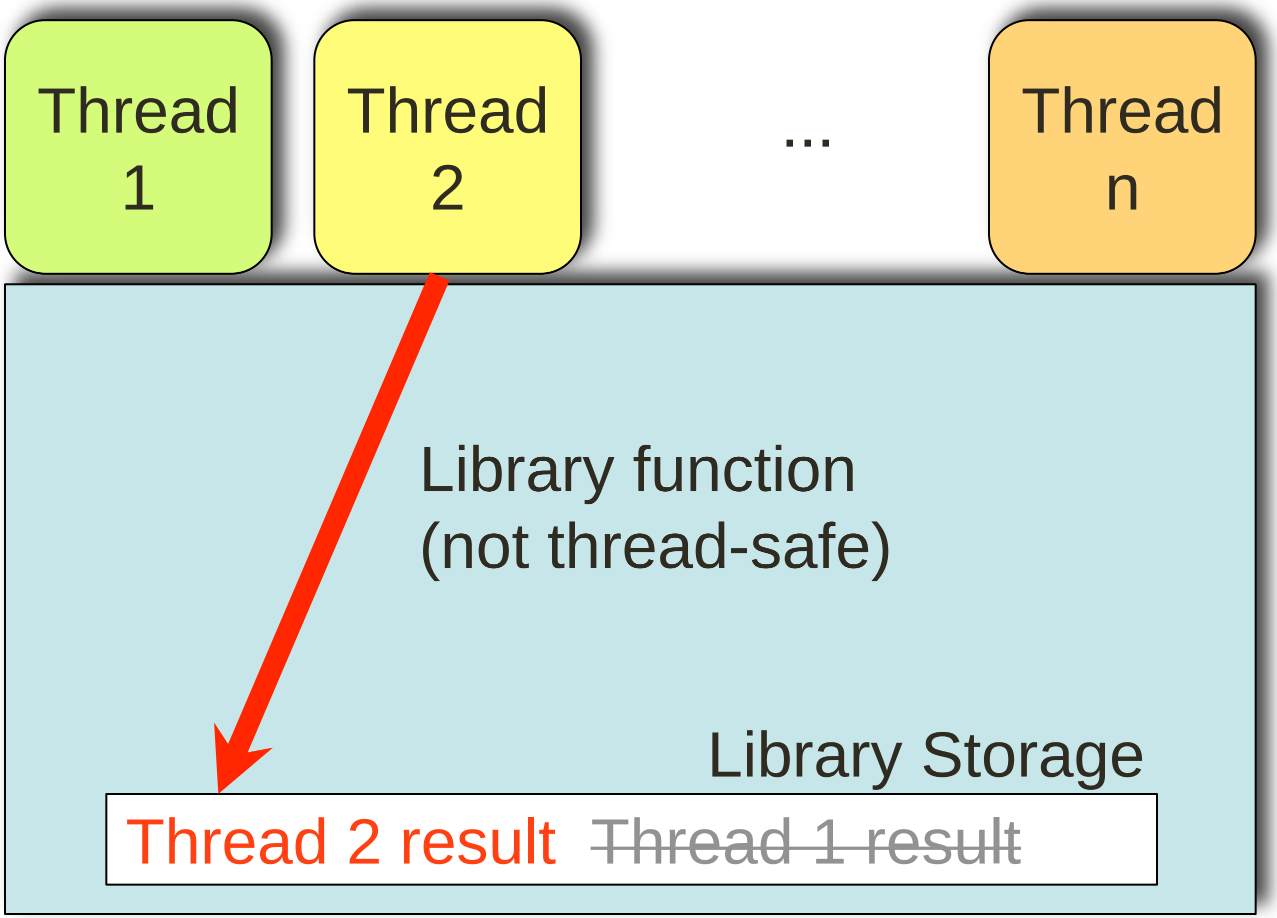
Thread Safeness



Thread Safeness


Thread 1
Thread 2
...
Thread n
Library function (not thread-safe): returns pointer to library storage
Library Storage
Thread 1 result

Thread Safeness



Thread 1

Thread 2

...

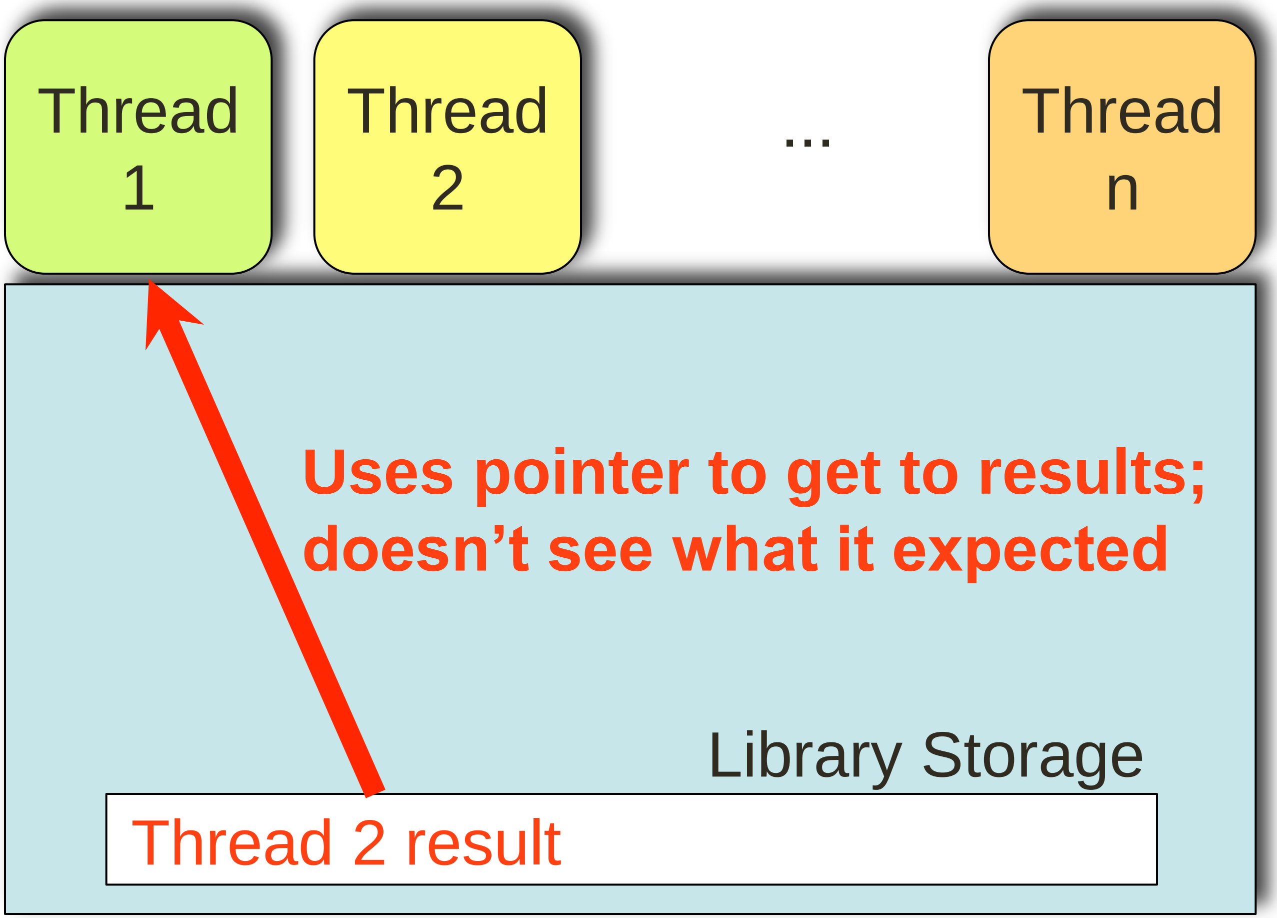
Thread n

Library function (not thread-safe)

Library Storage

Thread 2 result ~~Thread 1 result~~

Thread Safeness



Thread 1

Thread 2

…

Thread n

Uses pointer to get to results; doesn’t see what it expected

Library Storage

Thread 2 result