Concurrency

Synchronous vs. Asynchronous Models
Resources

Topics

- Synchronous vs. Asynchronous Operations
  - Asynchronous event-driven programming
  - Asynchronous message passing
  - Asynchronous network communication
- Resource Management
  - Resource durability: **reusable** vs **consumable**.
  - Resource multiplicity: **static** vs **dynamic**.
  - Resource sharing: **Simultaneous** vs **Sequentially Reusable**

Learning Outcomes

- At the end of this lesson, you will be able to:

1. Describe the differences between synchronous and asynchronous operations and the corresponding methods.
2. Describe resources and their types
3. Apply protection of shared resources to avoid conflicts

Part One!

Part One!

- In this part, we will discuss the differences between synchronous and asynchronous execution of tasks. Try to find answers to the following questions:

1. What is the main difference between synchronous and asynchronous operations?
2. How are the callers informed about the result of an asynchronous operation?
3. What is an event? What is event-driven programming?
4. What is the benefit of multi-threaded synchronous processing?
5. What advantages does asynchronous multi-threaded processing have over synchronous multi-threaded processing?

Operations

Synchronous or Asynchronous?

Synchronous

A synchronous operation **blocks** the process
*until the operation completes (we previously called it **BLOCKING**)*

An operation can be:

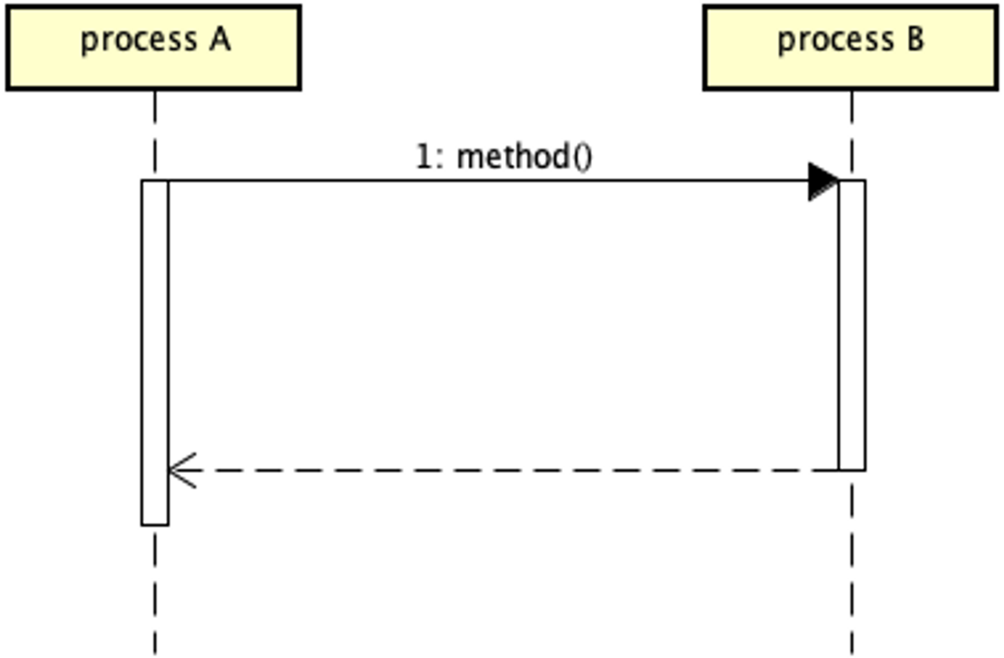
- a computation
- a communication (request/response)

Synchronous

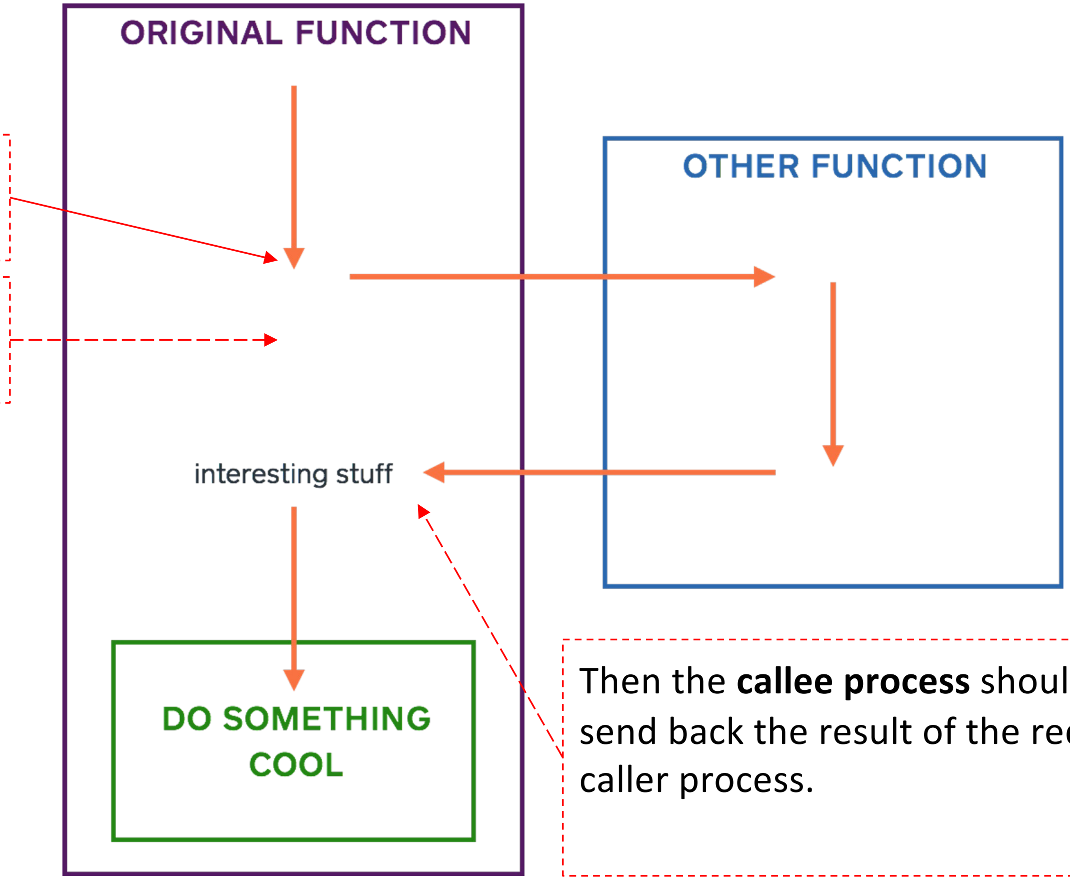
Q: “an operation completes”, what does it mean?

A synchronous execution design: the caller must *wait* until it gets a *return result*

- No further action can be performed
- **Blocking**



Synchronous



The **caller process** can start the other task here

The **caller process** must wait…

Then the **callee process** should find a way to send back the result of the request from the caller process.

Asynchronous

An asynchronous operation is **NOT waiting** for the result

- It is non-blocking
- Only initiates the operation
- The caller should discover the result of the call by other mechanisms
  - ***Polling***
  - ***Interrupts***
  - ***Callback (Events)***

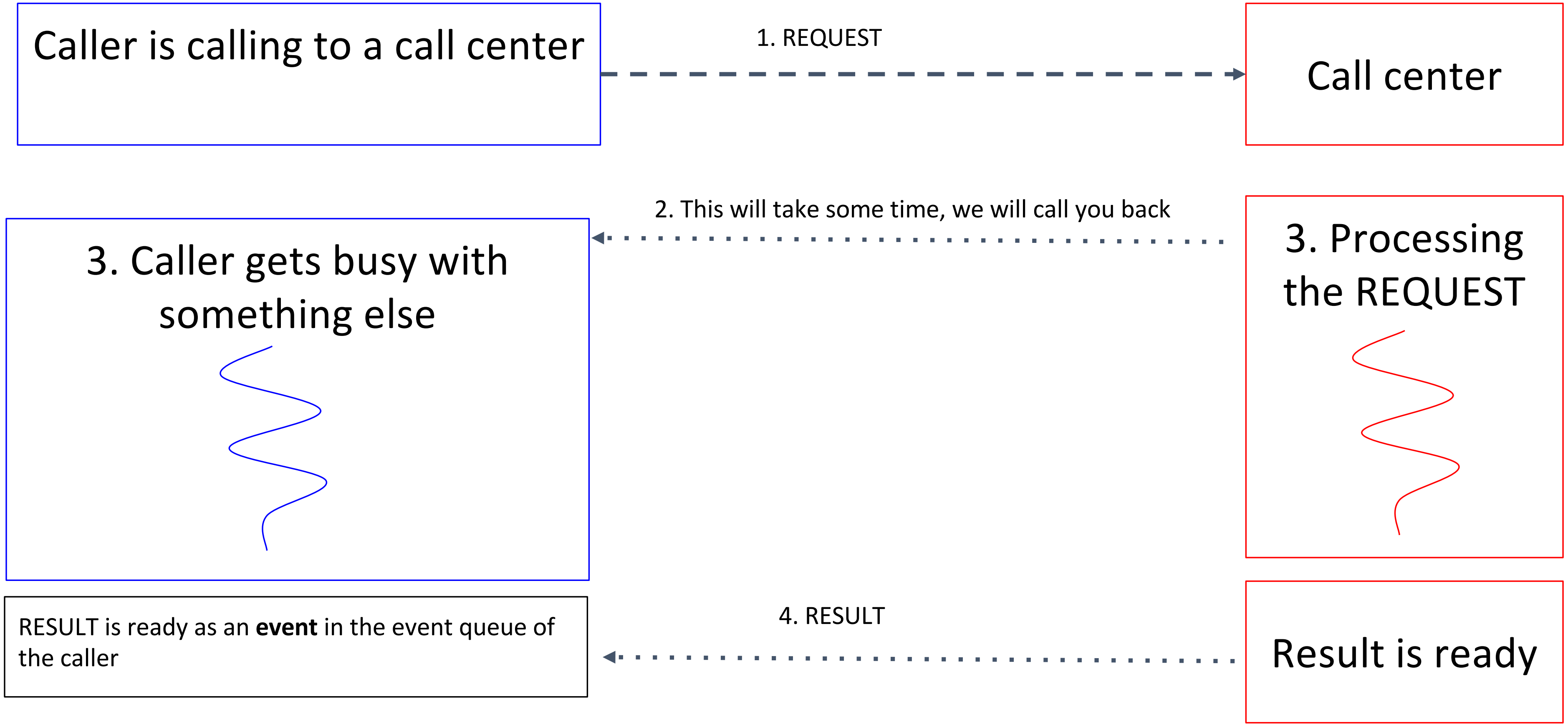
Polling

- ***Polling***: the process checks other processes/threads/devices regularly to receive their data whenever it is ready:
  - For example, polling a parallel printer port to check whether it is ready for another character.
  - Or, polling a shared queue to see if the result is ready.

Interrupts

- ***Interrupt***: The process continues performing its task until another process/thread/device stops it to send data
  - Example: keyboard
  - when you press a key, the current process is stopped, and the code of the key just pressed is read and put in a buffer. Then the process continues from the point left.

Callback



Caller is calling to a call center

1. REQUEST

Call center

2. This will take some time, we will call you back

3. Caller gets busy with something else

3. Processing the REQUEST

RESULT is ready as an **event** in the event queue of the caller

4. RESULT

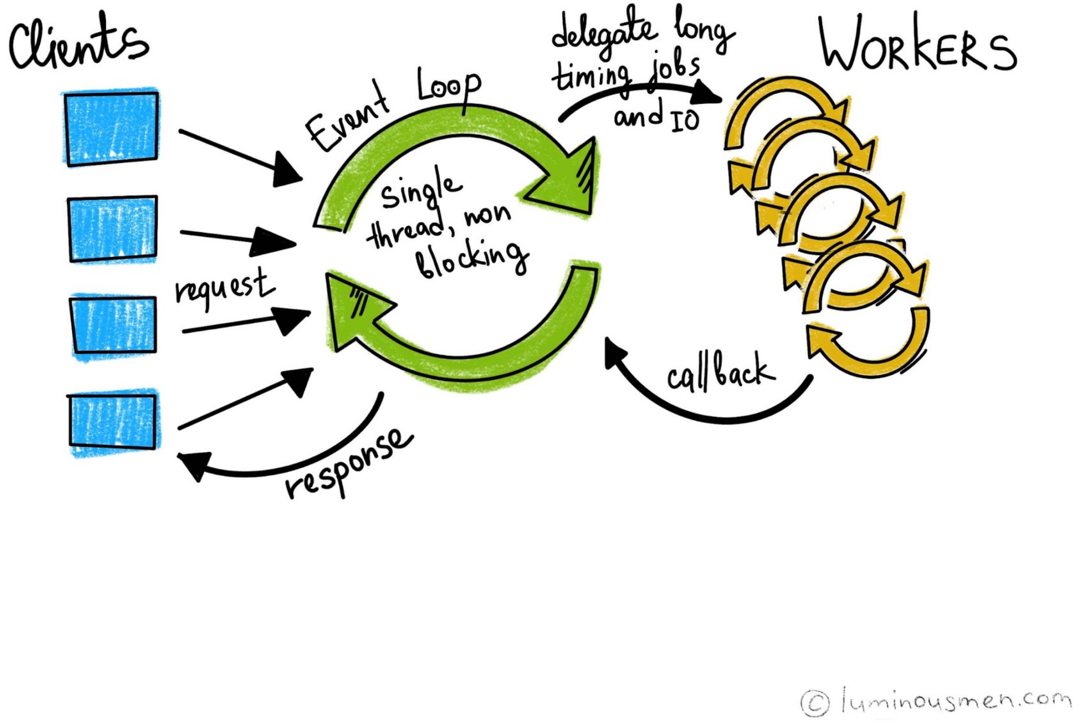
Result is ready

Event-Driven Programming

- ***Event-Driven Programming*** focuses on the generation and handling of event notifications.
    - ***Events*** are often actions performed by the user during the execution of a program such as clicking on a button, pressing a key, etc.
    - ***Events*** can also be messages generated by the operating system or another process/thread, or by a peripheral device.

Event Driven Programming

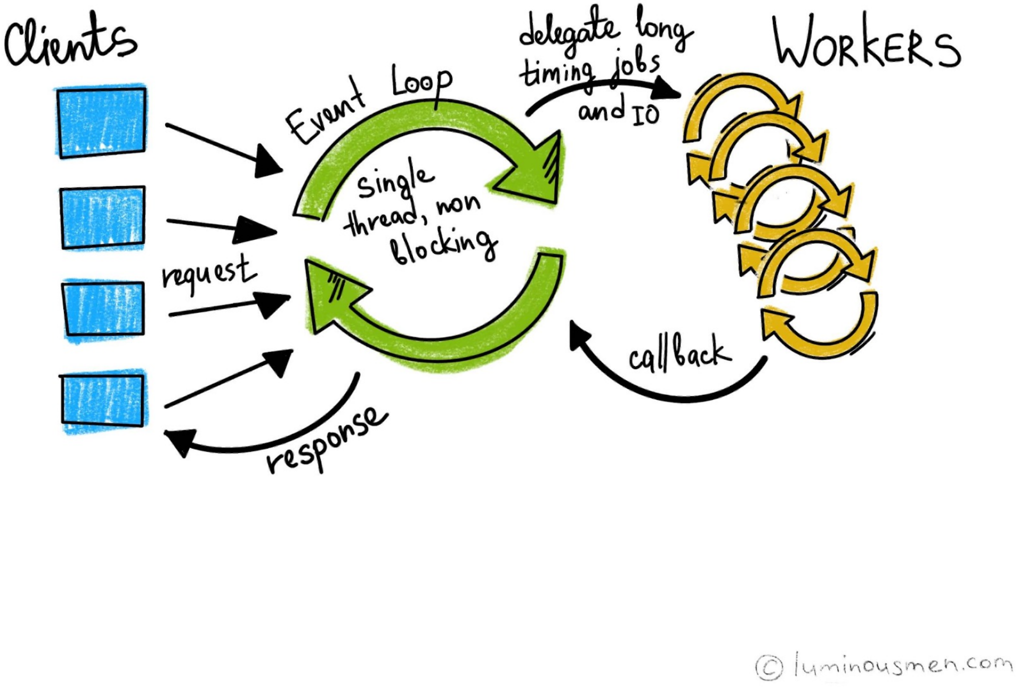
The central element of an event-driven application is a *scheduler* that *receives* a stream of events and *passes* each event to the relevant event handler.



Event Driven Programming

Event-handler receives a function as an argument and calls it back when the event occurs.

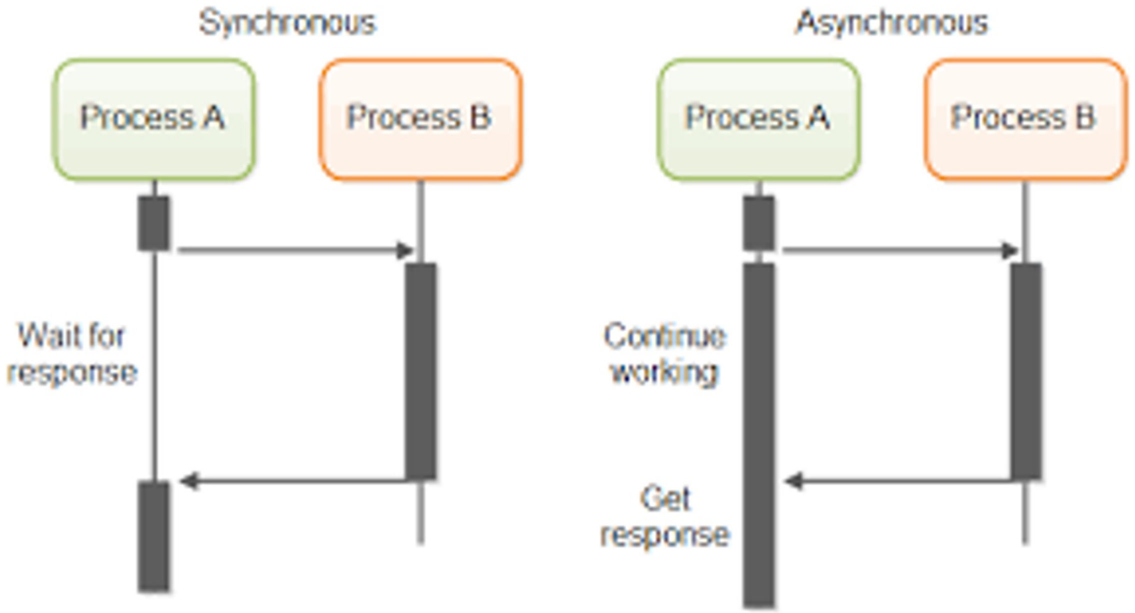
- These functions are named ***callback*** functions



Asynchronous vs. Synchronous

Applying asynchronous techniques can help concurrency

- The caller *continues* its execution while the operation is preparing the result.
- Improves performance and **responsiveness**



Asynchronous Programming: applications

- Very helpful in **GUI** tasks (multithreading in GUI programming can be very difficult)
- While the GUI thread interacts with the user, the asynchronous operation prepares the result
- A function can handle each request of the user

Asynchronous Programming: tips

- Don’t apply for simple computational tasks, you will not gain much

- Make a balance between *simplicity* and *efficiency*

Asynchronous Programming: tips

Usually, the best is to apply when you are communicating with another system, component, device, GUI…

- *Example*: Requesting a URL to download some content
- *Example*: The task is performing lots of I/O (file/database read/write), and then the main application can utilise the CPU
- *Example*: Message Passing

(**does not use synchronous send/receive**)

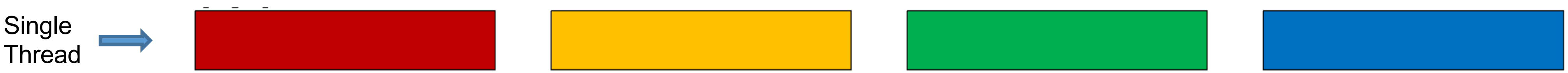
Multithreaded Vs Asynchronous Models

- Multithreaded processing can be synchronous or asynchronous

  - In the *synchronous* multithreaded model, a thread waits for other thread(s) to complete their tasks (using join)

  - In the *asynchronous* model, the threads perform additional tasks while waiting for the results from other threads

Single Threaded Synchronous Processing

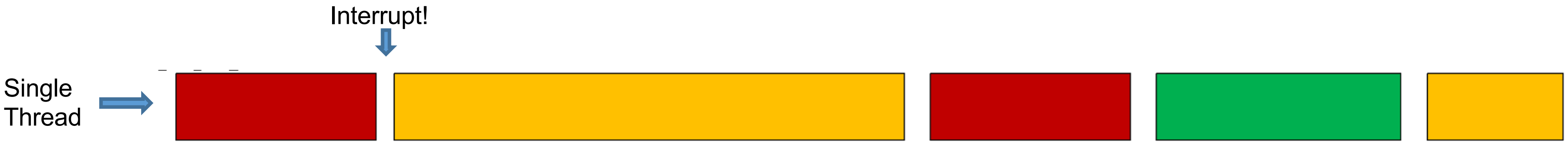


No efficiency*, **No** responsiveness

Each box represents a single task, as “a GUI interactive element” or “reading/writing from/to the disk”

*Efficiency: maxmizing resource (ex.: CPU) utilization

Single Threaded Asynchronous Processing

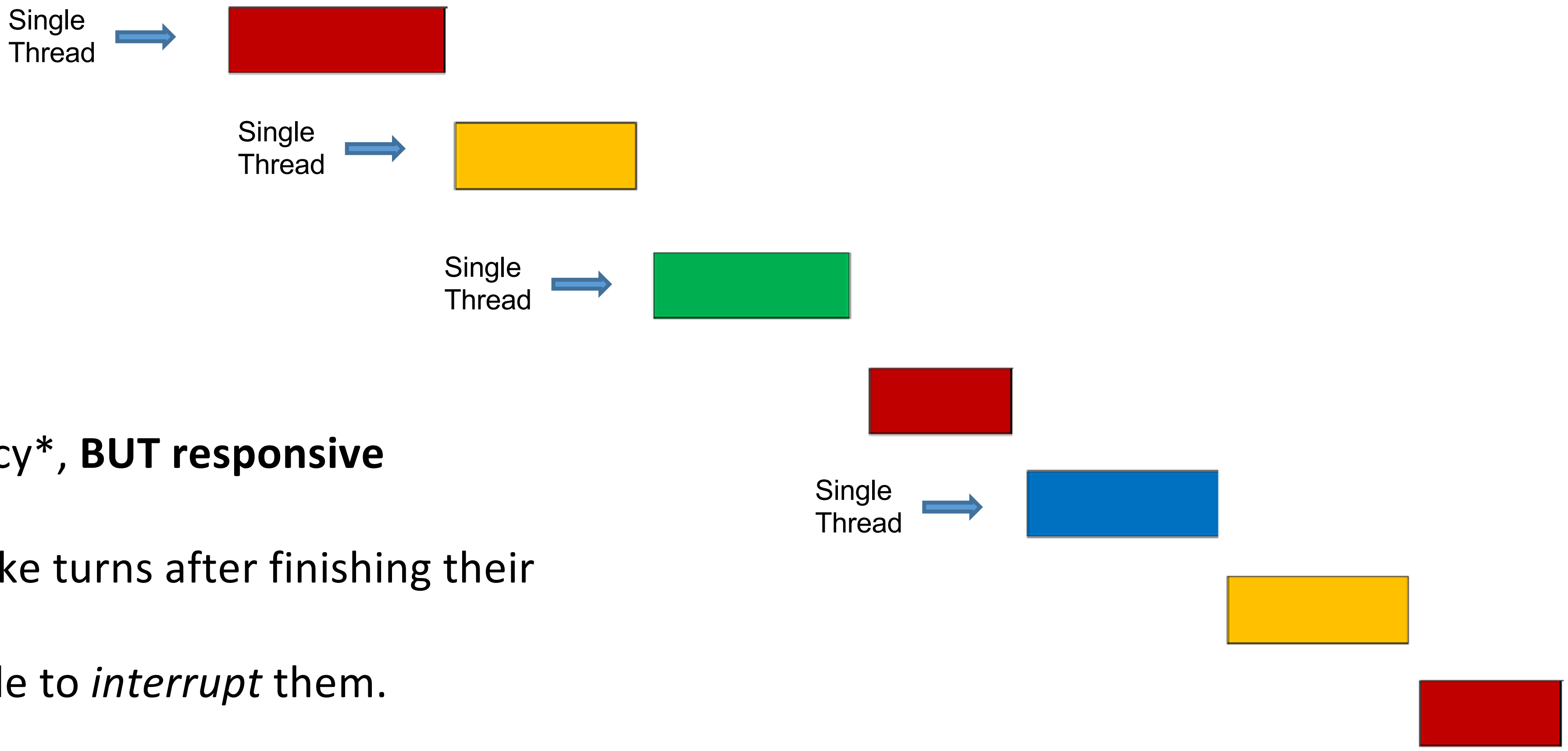


No efficiency*, **BUT responsive**

RED has been *interrupted* by a GUI task that has been invoked.
After RED goes back to its execution.

*Efficiency: maxmizing resource (ex.: CPU) utilization

Multithreaded Synchronous Processing



Single Thread

Single Thread

Single Thread

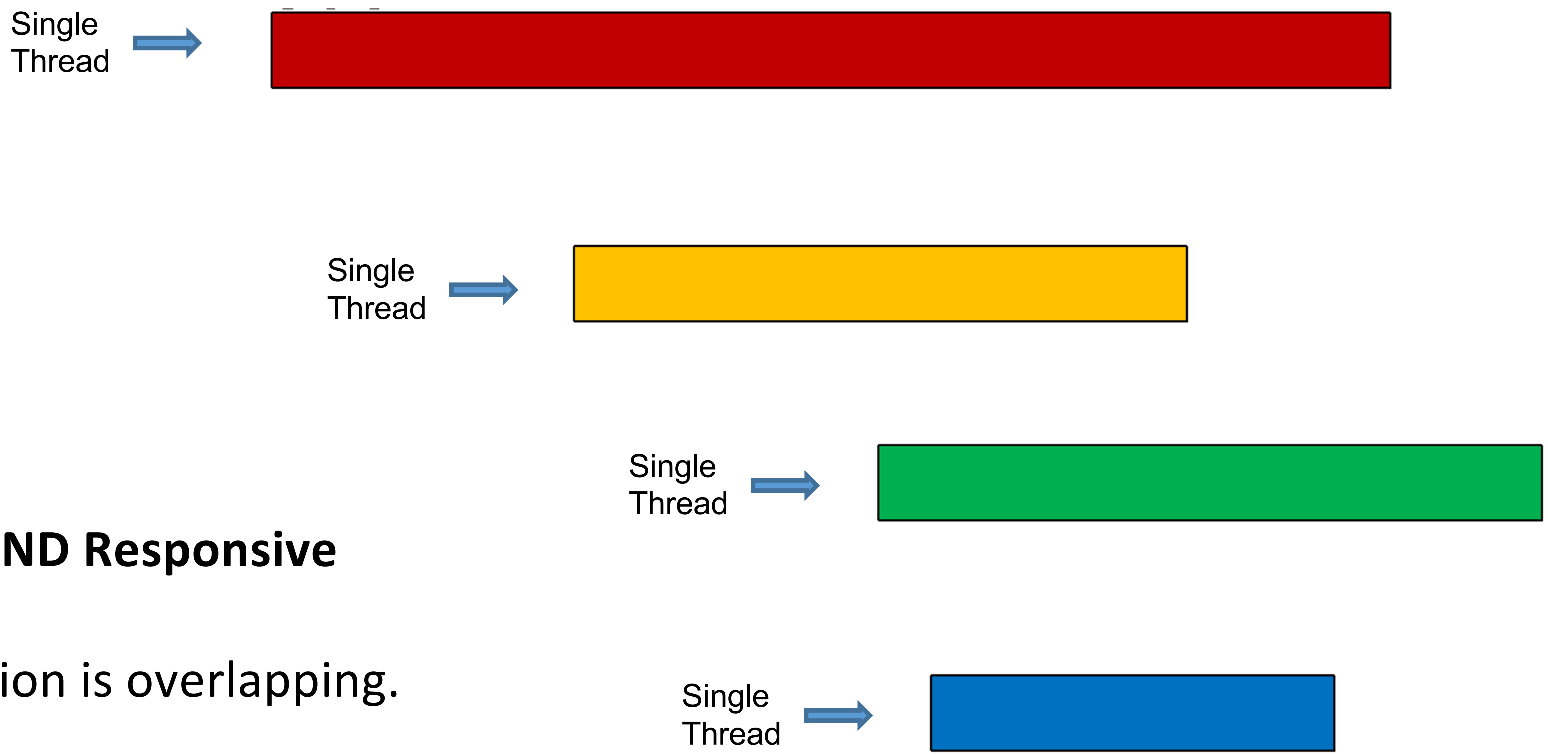
Single Thread

No efficiency*, **BUT responsive**

Threads take turns after finishing their quantum.
It is possible to *interrupt* them.

*Efficiency: maxmizing resource (ex.: CPU) utilization

Multithreaded Asynchronous Processing



Efficient* AND Responsive

The execution is overlapping.

*Efficiency: maxmizing resource (ex.: CPU) utilization

Part One!

- In this part, we will discuss the differences between synchronous and asynchronous execution of tasks. Try to find answers to the following questions:

1. What is the main difference between synchronous and asynchronous operations?
2. How are the callers informed about the result of an asynchronous operation?
3. What is an event? What is event-driven programming?
4. What is the benefit of multi-threaded synchronous processing?
5. What advantages does asynchronous multi-threaded processing have over synchronous multi-threaded processing?

Part Two!

Part Two!

- In this part, we will discuss resources, their types and their differences. Try to find the answer to the following questions:

1. What are some examples of resources in a computer system?
2. What is a reusable resource? Give examples
3. What is a consumable resource? Give examples
4. What is a simultaneous resource?
5. What is a static resource?
6. Which resources are more important in discussing concurrency?

Resource Management

Concurrency as Resource Sharing (recap)

- **Concurrent:** Multiple programs (or threads) accessing a shared resource at the same time.
  - Example: Many threads trying to make changes to the same data structure (a global list, map, etc.).

Resources

- A **resource** is anything required by a process.
  - In fact, there can be no process needing no resources.

- Operating Systems function as a **resource manager**.
- Resources can have:
  durability, multiplicity, and shareability.

Resource Management

- To properly allocate resources the management considers:
  - **Protection,**
  - **Economy,**
  - **Convenience,**
  - **and Fairness.**

- The management should also *avoid deadlock* problems.

Resource Characteristics

- A resource's characteristics determine how (in part) it's managed.

- The main characteristics are:
  - Resource durability: **reusable** vs **consumable**.
  - Resource multiplicity: **static** vs **dynamic**.
  - Resource shareability:

    simultaneous reusable vs **Sequentially Reusable**

Does any of this ring any bell?

Durability of Resources

- A **consumable resource** disappears after begin used.
  - Network packets and Inter-process Communication (IPC) messages are consumable resources.

- A **reusable resource** continues to exist after usage.

- Q: Is data stored on a hard disk a Consumable Resource? Why?

Simultaneous Reusable Resources

- A **simultaneous reusable resource** can be used by more than one process **at the same time.**
  - Input devices tend to be simultaneous reusable resources. (memory pages, keyboard, etc…)
- A CPU is a simultaneous reusable resource if processes are preemptive. Why?

Sequentially Reusable Resources

- Sequentially reusable resources can be used **by at most one process at a time**.

- Output devices tend to be sequentially-reusable devices.

- Example:
  - Printers: We cannot interrupt a print task and switch to another one (Why?) but it is reused by many.

Multiplicity: Static Vs Dynamic Resources

- A **static resource** has a fixed or slowly changing number of units.
  - Disks and CPUs are static resources.
  - Reusable resources tend to be static.

- A **dynamic resource** has a varying number of units.
  - Consumable resources are necessarily dynamic.
  - They have to be created and consumed.

Discussion

- Which properties of resources are important in concurrency?

  - Resource durability: **reusable** vs **consumable**.
  - Resource multiplicity: **static** vs **dynamic**.
  - Resource shareability: **simultaneous reusable** vs **Sequentially Reusable**

Part Two!

- In this part, we will discuss resources, their types and their differences. Try to find the answer to the following questions:

1. What are some examples of resources in a computer system?
2. What is a reusable resource? Give examples
3. What is a consumable resource? Give examples
4. What is a simultaneous resource?
5. What is a static resource?
6. Which resources are more important in discussing concurrency?

Summary

- A synchronous operation blocks the process until the operation completes
- An asynchronous operation is non-blocking and only initiates the operation
- Asynchronous processing is another way of concurrency
- Resources have different types and characteristics

Time for a quiz…