Multithreading using C++

OOPS Session - Saturday (2-3 pm)

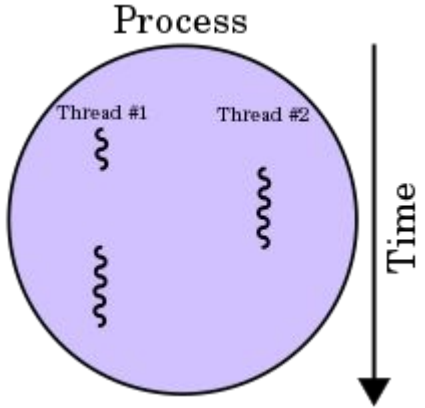
Spot the difference

```
//Code 1
int a[4]={1,2,3,4};
for(i=0;i<4;i++)
    cout<<a[i]<<endl;


//Code 2
int a[4]={1,2,3,4};
cout<<a[0]<<endl;
for(i=1;i<4;i++){
    a[i]+=a[i-1];
    cout<<a[i]<<endl;
}
```

Introduction

- A thread is an independent unit of execution created within the context of a process (or application that is being executed). When multiple threads are executing in a process at the same time, we get the term “multithreading.” Think of it as the application’s version of multitasking.
- Multithreading is a model of program execution that allows for multiple threads to be created within a process, executing independently but concurrently sharing process resources. Depending on the hardware, threads can run fully parallel if they are distributed to their own CPU core.

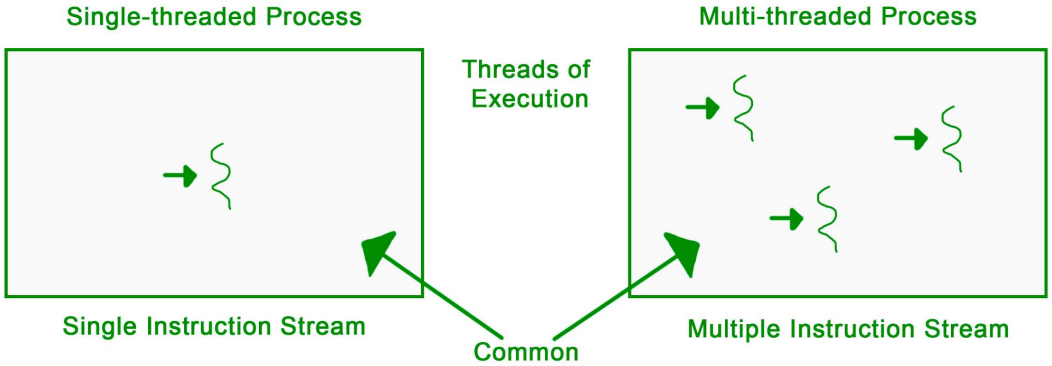


What Is Multithreading Used For?

- Most of the applications that you use on a daily basis have multiple threads running behind the scenes. Consider your internet browser. At any given time, you may have numerous tabs open, each one displaying various types of content. Multiple threads of execution are used to load content, display animations, play a video, and so on.
- Another example of a multithreaded program that we are all familiar with is a word processor. While you are typing, multiple threads are used to display your document, asynchronously check the spelling and grammar of your document, generate a PDF version of the document. These are all happening concurrently, with independent threads performing these tasks internally.

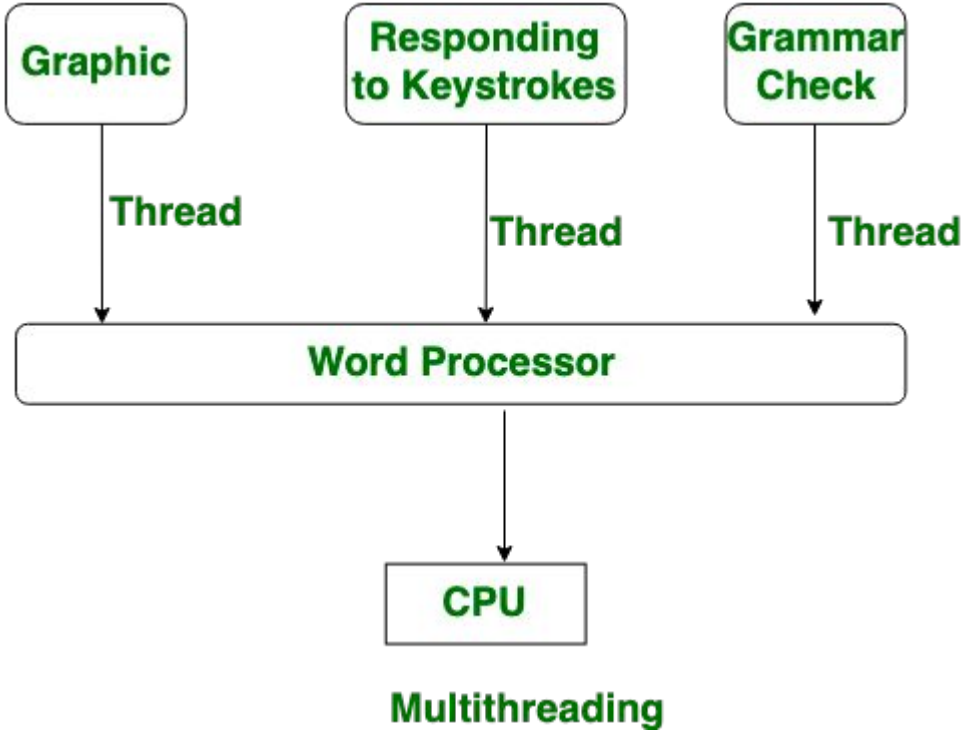
What is Multithreading used for?

- Multithreading also leads to minimization and more efficient use of computing resources. Application responsiveness is improved as requests from one thread do not block requests from other threads.
- Additionally, multithreading is less resource-intensive than running multiple processes at the same time. There is much more overhead, time consumption, and management involved in creating processes as compared to creating and managing threads.



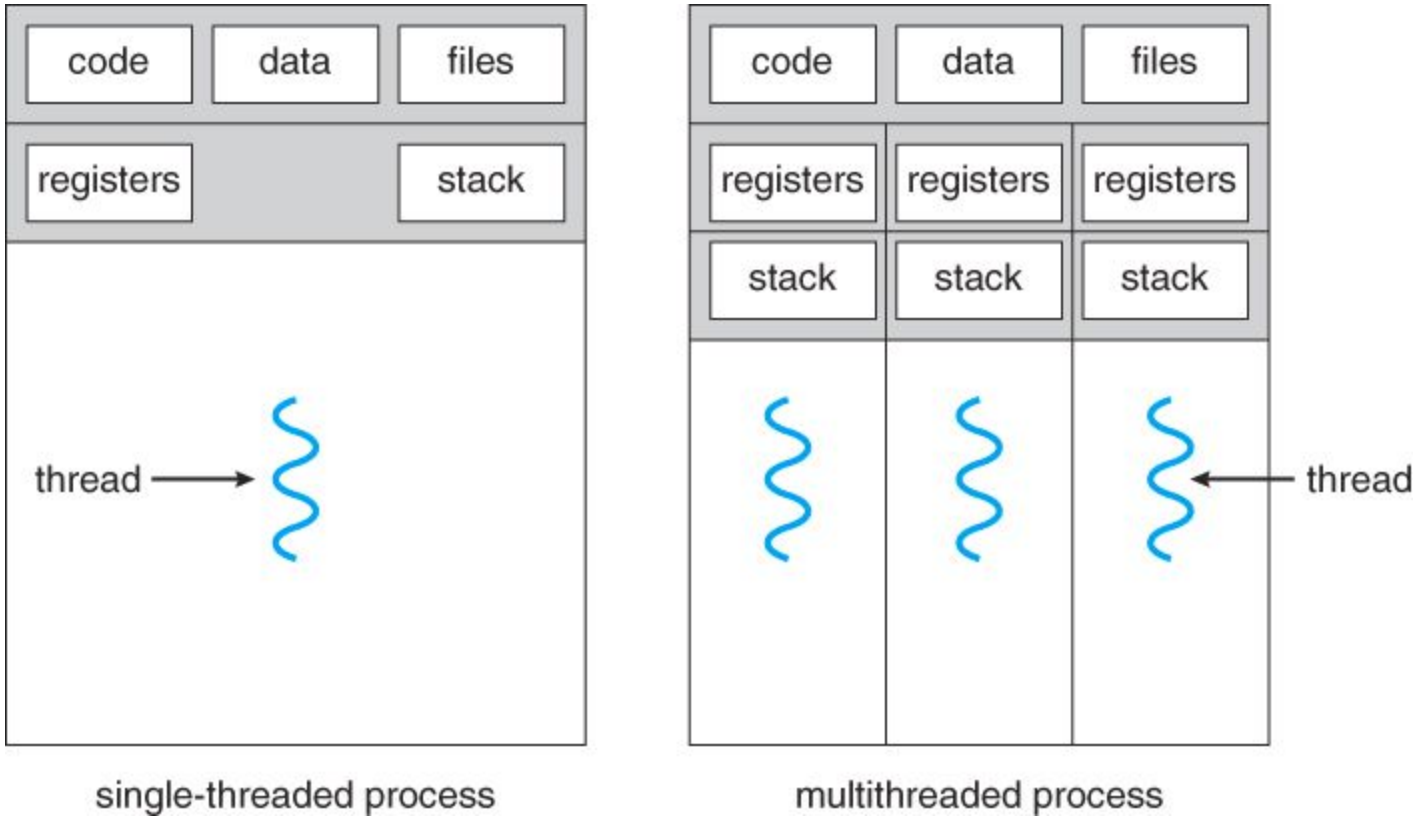
Single Thread and Multi Thread Process

Example


Graphic
Responding to Keystrokes
Grammar Check
Thread
Thread
Thread
Word Processor
CPU
Multithreading

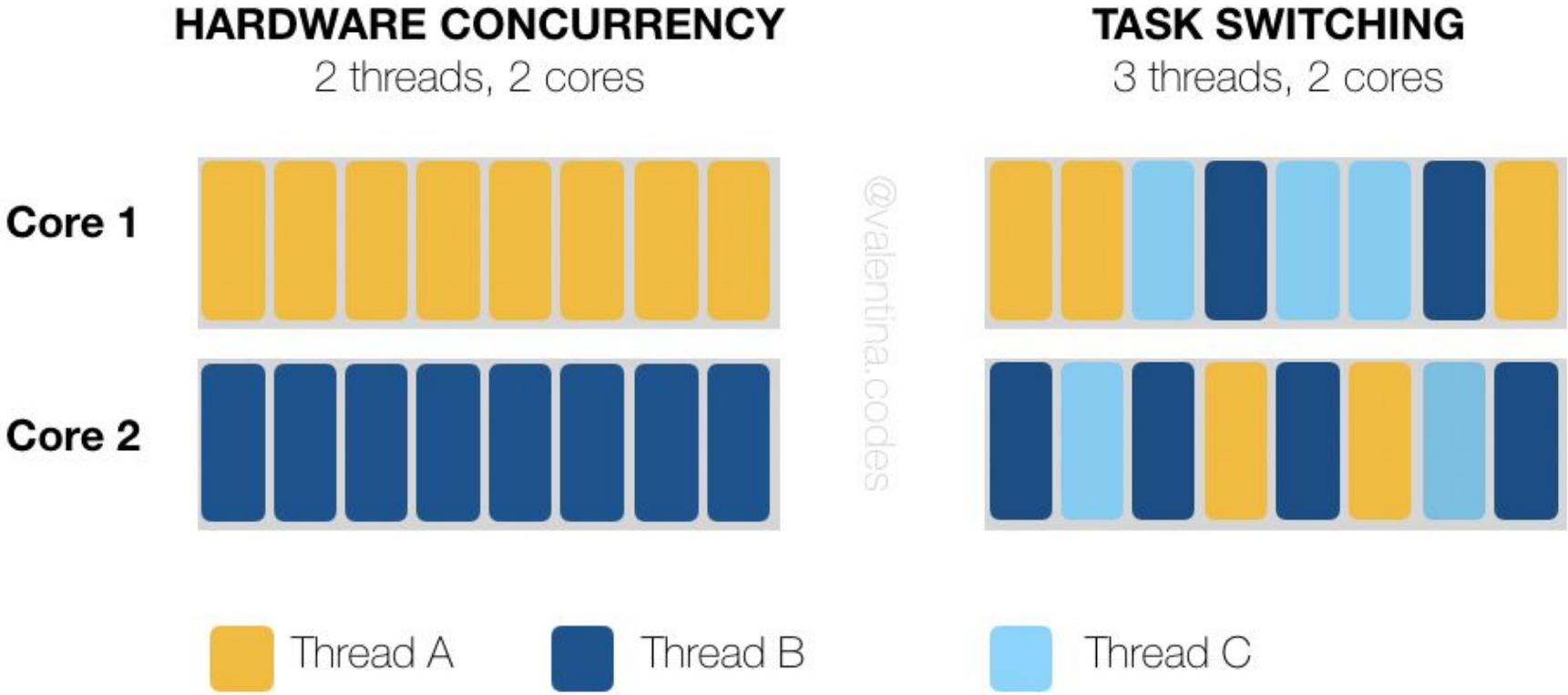
Multithreading vs Multiprocessing

- In Multiprocessing, every process owned a separate address space.While in Multithreading, a common address space is shared by all the threads.
- In Multiprocessing, Many processes are executed simultaneously.While in multithreading, many threads of a process are executed simultaneously.
- In Multiprocessing, CPUs are added for increasing computing power.While In Multithreading, many threads are created of a single process for increasing computing power.



single-threaded process

multithreaded process


HARDWARE CONCURRENCY
2 threads, 2 cores
TASK SWITCHING
3 threads, 2 cores
Core 1
Core 2
@valentina.codes
Thread A
Thread B
Thread C

Multithreading in C++?

- POSIX Pthread Library.
- It allows us to create multiple threads for concurrent process flow.
- It is most effective on multiprocessor or multi-core systems where threads can be implemented on a kernel-level for achieving the speed of execution.
- We must include the pthread.h header file at the beginning of the script to use all the functions of the pthreads library.
- To execute the c file, we have to use the -pthread or -lpthread in the command line while compiling the file.

Syntax

- pthread_create(&tid,&attr,runner,data);
- Tid - pointer to an unsigned integer value that returns the thread id of the thread created.
- Attr - pointer to a structure that is used to define thread attributes like detached state, scheduling policy, stack address, etc. Set to NULL for default thread attributes.
- Runner - Function to be executed.
- Data - Data passed to the function.

Syntax:

- pthread_exit(0) -
- pthread_join(tid,NULL) - waits for the thread to be terminated.

```
pthread_t tid;
pthread_attr_t attr;
pthread_attr_init(&attr);
pthread_create(&tid,&attr,runner,data);
pthread_join(tid,NULL);
pthread_exit(0);
```

Code:

Link - https://github.com/YogaVicky/OOPS

Thanks Guys!