Threads

Lab 04

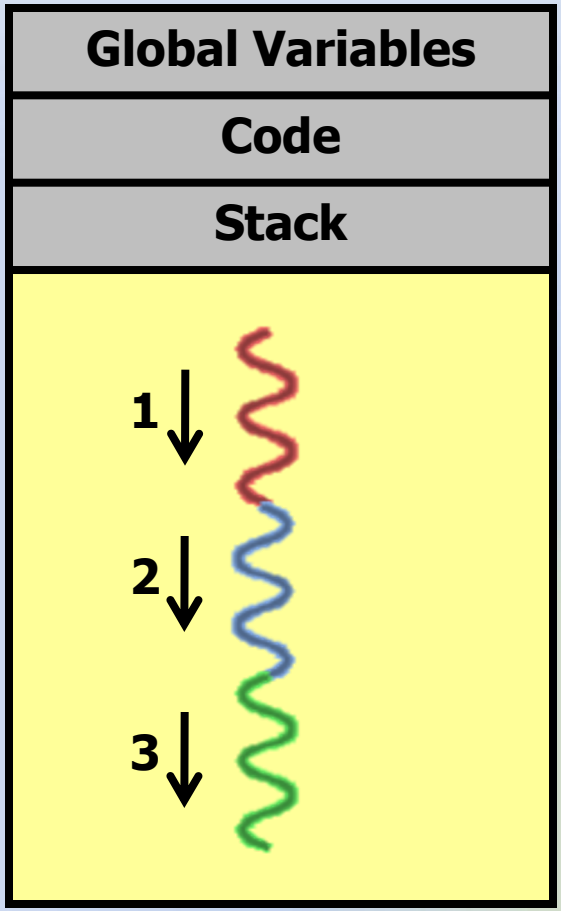
اللهم علمنا ما ينفعنا،،، وانفعنا بما علمتنا،،، وزدنا علماً

Lab Objective

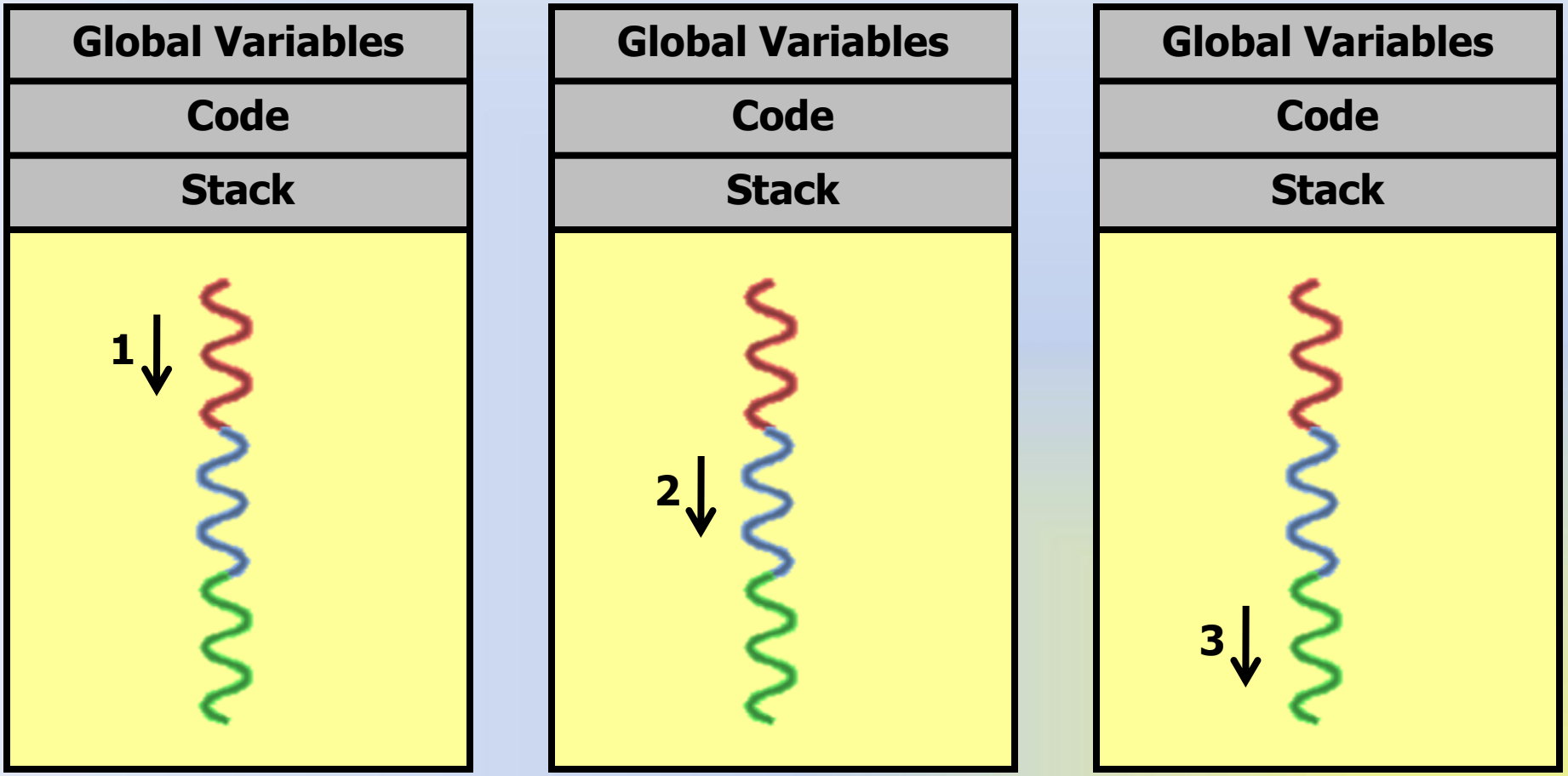
- To practice using threads.

THREADS VS. PROCESSES

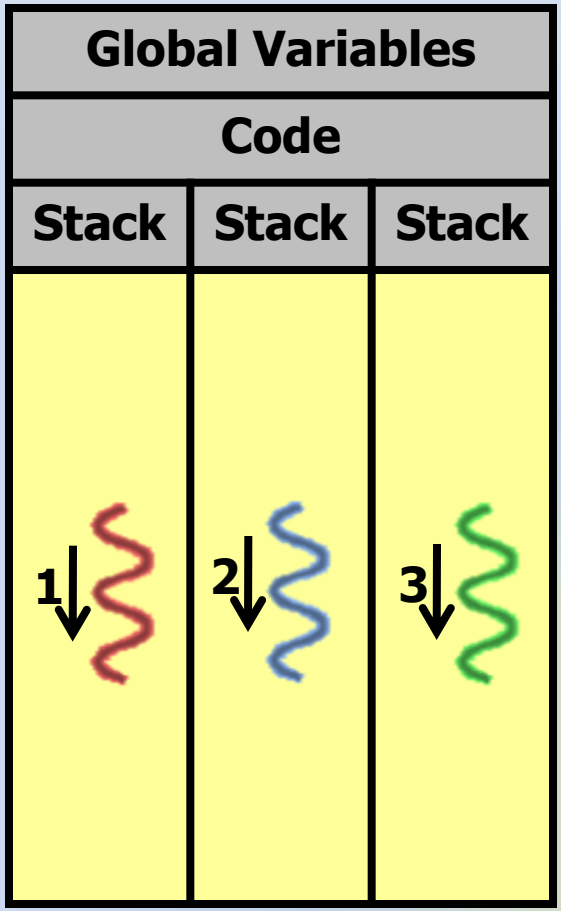
Single Process



Multiple Processes using `fork()`



Single Process with Multiple Threads



Thread Creation

- When a program is started, a single thread is created, called the *initial thread* or *main thread*.
- Additional threads are created by:

```
int pthread_create (pthread_t * tid,1
                    const pthread_attr_t *attr,2
                    void *(*func) (void*),3
                    void *arg);4
```

- Returns 0 if OK, positive Exxx value on error

1. `tid` → The newly-created thread ID
2. `attr` → the new thread attributes, use NULL to get system default
3. `func` → Pointer to a function to execute when the thread starts
4. `arg` → Pointer to `func` argument (multiple arguments can be passed by creating a structure and passing the address of the structure)

Thread Management

- Each thread has a unique ID, a thread can find out its ID by calling:

```
pthread_t pthread_self();
```

- A thread can be terminated by calling:

```
void pthread_exit();
```

- The main thread can wait for a thread to terminate by calling:

```
int pthread_join(pthread_t tid, void **status);
```

  - **Note:** with `pthread_join`, we must specify the `tid` of the thread.

Practice

- In the following C++ program, the main process creates two threads of the function `doit`
- The function has a loop to increment the global variable `counter` by 1 for 10 times.
- Within every iteration of the loop, the function prints out the ID of the thread that is running and the current value of `counter`
- Write, compile and run the program in Linux then answer the questions in the check-off section.

```
#include <iostream>
#include "pthread.h"
using namespace std;
#define NLOOP 10 //Constant value
int counter = 0;
void * doit(void *);
int main()
{
        pthread_t tidA, tidB;
        pthread_create(&tidA, NULL, doit, NULL);
        pthread_create(&tidB, NULL, doit, NULL);
        pthread_join(tidA, NULL);
        pthread_join(tidB, NULL);
        exit(0);
}//end main
void * doit(void *vprt)
{
        int i, val;
        for( i = 0; i<NLOOP; i++) {
                val = counter;
                cout<<"Thread = "<<pthread_self();
                cout<<" Counter = "<<dec<<counter<<endl;
        //dec ->To display numbers in decimal format
                sleep(2);
                counter = val+1;
        }
        return (NULL);
} //end doit function
```

#include <unistd.h>
#include <stdlib.h>

Global variable incremented by the threads

Create two threads to run the function `doit`

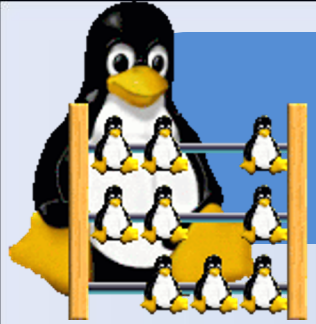
Wait for both treads to terminate

Each thread increments the global variable `counter` by 1 for 10 times

Very Important Note

- Use the option `-pthread` or `-lpthread` with the compilation command to enable the support of multithreading with the pthread library.
- Your command line should look something like this:

```
$ g++ lab4.C -o lab4 -pthread
```



Check Off

1) Why the final value of `counter` is 10 and not 20?
2) Run the program again. while it' running, use the command `ps -all` in a separate window. Write down the `PID` of the process(es) related to the program. Explain the difference between this program and the program you had in the previous lab in terms of number of PIDs.
3) modify the loop in the `doit` function to be as follows:

```
for( i = 0; i<NLOOP; i++) {
        cout<<"Thread = "<<pthread_self();
        cout<<" Counter = "<<dec<<counter<<endl;
        counter++;
```

Recompile the program and run it. what is the maximum value of `counter` ?

1) Briefly explain the behavior of the program based on the results you obtain from the previous questions.

??? ANY QUESTIONS ???

☺