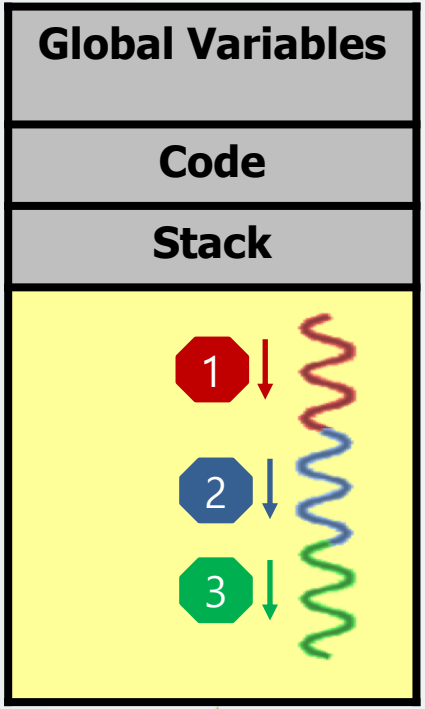
جامعة الأمير مقرن بن عبد العزيز
University of Prince Mugrin

Lab 5
Threads

Objective

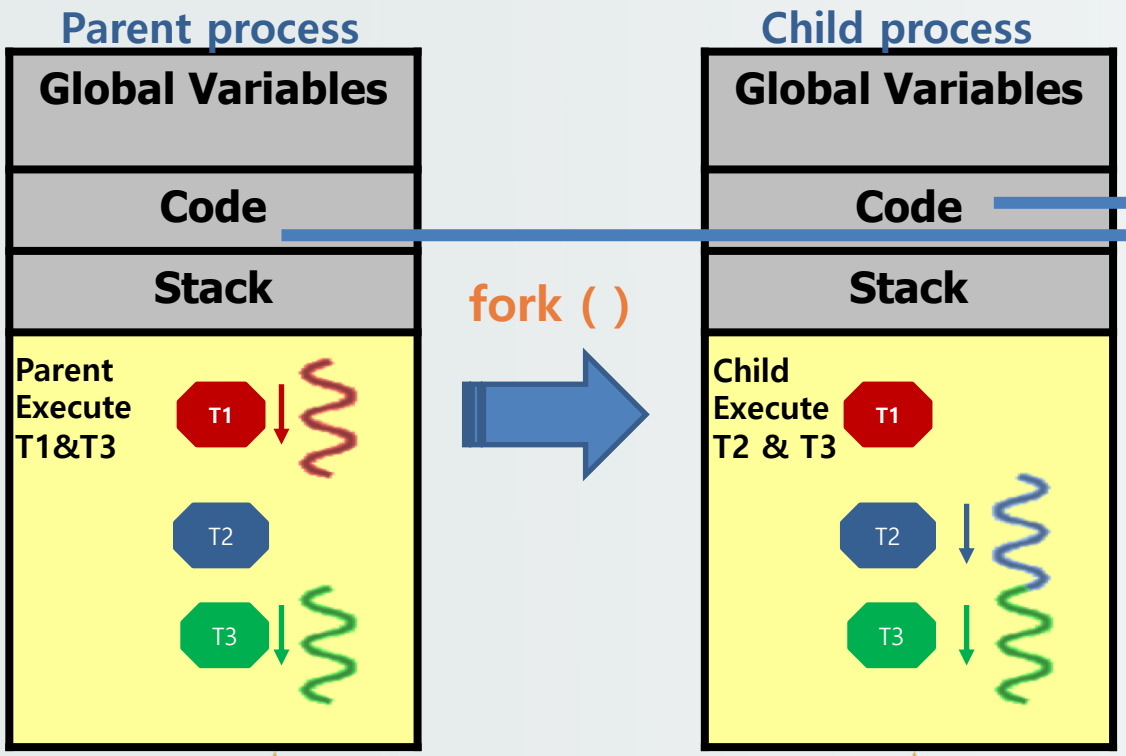
- To review the definition of **Threads**
- To practice using **Threads**.
  - To create and terminate threads
  - To pass argument to threads
  - To create multiple threads

Revision: A Single-Threaded Process



Single thread of execution (with 3 different tasks; Task 1, 2 and 3. The three tasks being executed sequentially)

Revision: Multiple Processes using fork()



fork ( )

Code

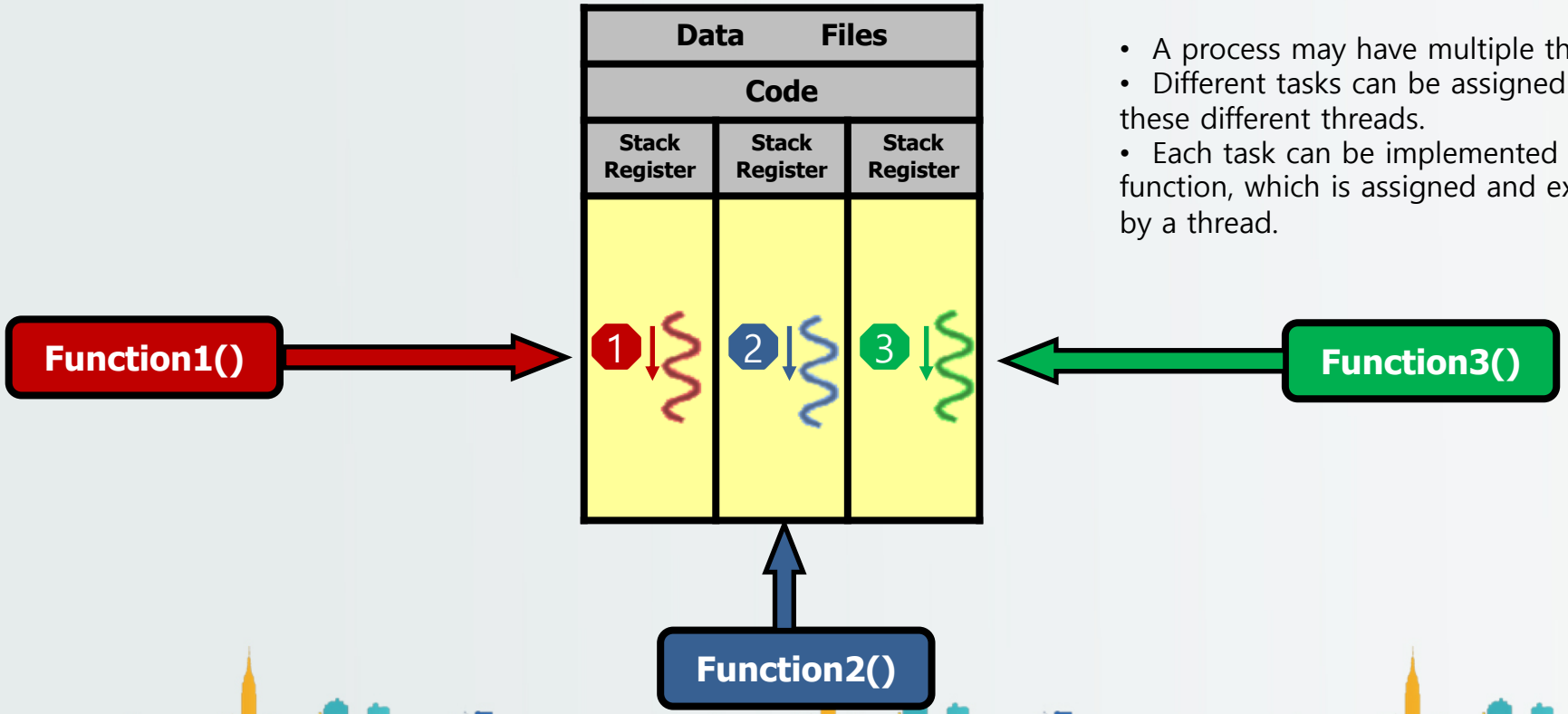
```
int main()
{
      pid_t  pid;

      pid = fork();

      if (_______) {
            /* do Task1*/
       }
       else if (_______) {
            /* do Task2*/
       }
       /* do Task 3*/
}
```

- Both, the parent and child have the same code.
- But they may follow different flow of execution based on the return value of fork().
- Child and parent execute concurrently

Single Process with Multiple Threads



- A process may have multiple threads.
- Different tasks can be assigned to these different threads.
- Each task can be implemented as a function, which is assigned and executed by a thread.

Creating Threads

- A **thread** is the **smallest unit of processing (or computation)**.
- A **thread exists within a process** - that is, **a single process may contain multiple threads**
- **Creating a multithreaded program will require :**
  - **# include <pthread.h>** ← Library of Threads
  - **pthread_t** ← Thread Type
  - **pthread_create()** ← Function to Create Thread

Creating Threads (cont.)

- When a program starts, **a single thread is created**, **called** the **initial thread** or **main thread**.
- Then, **additional threads** can be created using the following function :

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

tid
The newly-created thread ID

The new thread attributes, use NULL to get system default.

Pointer to a function to execute when the thread starts.

Pointer to func argument (multiple arguments can be passed by creating a structure and passing the address of the structure). NULL if no argument passed.

An example of a call to **pthread_create() :**

```
pthread_create ( &thread1 , NULL , function1 , NULL );
```

Passing Arguments to Threads

- **pthread_create()** allows programmer **to pass an argument** from the command line e.g. `argv [1]` to the thread call "`runner`".

```
pthread_create ( &thread, NULL, runner, argv[1] );
```

- Details of the parameters to **pthread_create()** :

```
int   pthread_create ( thread, attr, start_routine, arg );
```

Parameter	Description
thread	A unique identifier for the new thread returned by the subroutine.
attr	An attribute object that may be used to set thread attributes. You can specify a thread at tributes object, or NULL for the default values.
start_routine	The C routine that the thread will execute once it is created.
arg	A single argument that may be passed to start_routine. It must be passed by reference a s a pointer cast of type void. NULL may be used if no argument is to be passed.

Thread Management

- The pthread_join () suspends the execution of the calling thread unt il the target thread terminates.

```
pthread_t   pthread_join (pthread, void *status);
```

- The pthread_self () is used to get the ID of the calling thread.

```
pthread_t   pthread_self (void);
```

- A thread can be terminated by calling:

```
void   pthread_exit (void *value_ptr);
```

- pthread_attr_init () initialises a thread attributes object *attr* with the default value for all of the individual attributes

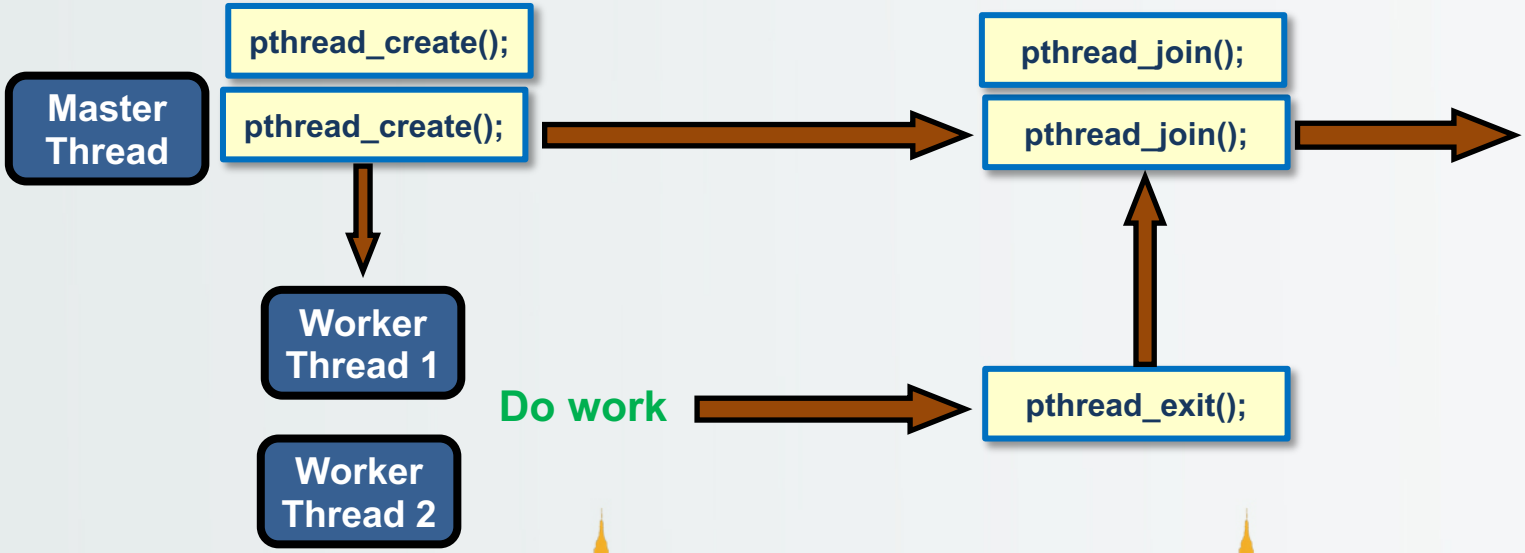
```
int   pthread_attr_init (pthread_attr_t *attr);
```

Threads Synchronisation

- **"Joining"** is one way to **accomplish synchronization** between threads.

```
pthread_join( thread1 , NULL );
```

For example:



Compiling a pthread program

- Use the option **-pthread** with the **compilation command** to enable the support of **multithreading** with the **pthread library**.

- **The command line :**

```
$ gcc –pthread –o code code.c
```

DEMO: Code1

No thread vs One Thread

What is the output?

```c
#include <stdio.h>
#include <unistd.h>
#define MAX_ITER 5

void MyTurn () {

  for (int i = 0; i < MAX_ITER; i++) {
    sleep (1);
    printf ("MyTurn: i=%d\n",i);
  }
}

void YourTurn () {

  for (int i = 0; i < MAX_ITER; i++) {
    sleep (2);
    printf ("  YourTurn: i=%d\n",i);
  }
}

int main () {

  MyTurn ();
  YourTurn ();
  return 0;
}
```

```
~/Lab5$ gcc code1.c
~/Lab5$ ./a.out
MyTurn: i=0
MyTurn: i=1
MyTurn: i=2
MyTurn: i=3
MyTurn: i=4
  YourTurn: i=0
  YourTurn: i=1
  YourTurn: i=2
  YourTurn: i=3
  YourTurn: i=4
~/Lab5$
```

```
~/Lab5$ gcc code1.c  -pthrea
~/Lab5$ ./a.out
MyTurn: i=1
  YourTurn: i=1
MyTurn: i=2
MyTurn: i=3
  YourTurn: i=2
MyTurn: i=4
MyTurn: i=5
  YourTurn: i=3
  YourTurn: i=4
  YourTurn: i=5
~/Lab5$
```

How to create one thread that execute MyTurn()?

Code1.c

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#define MAX_ITER 5

void* MyTurn (void* arg) {
  for (int i = 1; i <= MAX_ITER; i++) {
    sleep (2);
    printf ("MyTurn: i=%d\n",i);
  }
  pthread_exit (0);
}

void YourTurn () {

  for (int i = 1; i <= MAX_ITER; i++) {
    sleep (1);
    printf ("  YourTurn: i=%d\n",i);
  }
}

int main () {
  pthread_t thread;

  pthread_create (&thread,NULL, MyTurn, NULL);
  YourTurn ();
  pthread_join (thread, NULL);
  return 0;
}
```

DEMO: Code2

Passing Argument to Thread

```
~/Lab5$ ./a.out
MyTurn: i=1
  YourTurn: i=1
MyTurn: i=2
MyTurn: i=3
  YourTurn: i=2
MyTurn: i=4
MyTurn: i=5
  YourTurn: i=3
  YourTurn: i=4
  YourTurn: i=5
~/Lab5$
```

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

void* MyTurn (void* arg) {

  int max_iter = *((int*)arg);

  for (int i = 1; i <= max_iter; i++) {
    sleep (1);
    printf ("MyTurn: i=%d\n",i);
  }
}

void YourTurn (int max_iter) {
  for (int i = 1; i <= max_iter; i++) {
    sleep (2);
    printf ("  YourTurn: i=%d\n",i);
  }
}

int main () {
  int val=5;
  pthread_t thread;

  pthread_create (&thread,NULL, MyTurn, (void*) &val);
  YourTurn (val);
  pthread_join (thread, NULL);
  return 0;
}
```

Passing Command Line Argument to Thread

Argument count (`argc`) keep tracks of total count of the command line arguments while argument vector (`argv[]`) is an array of pointers to the arguments.

```
gcc -o myprog myprog.c
```

```
argc
      4
argv[0]
      gcc
argv[1]
      -o
argv[2]
      myprog
argv[3]
      myprog.c
```

```
~/Lab5$ ./a.out 3
MyTurn: i=1
  YourTurn: i=1
MyTurn: i=2
MyTurn: i=3
  YourTurn: i=2
  YourTurn: i=3
  YourTurn: i=4
  YourTurn: i=5
~/Lab5$ _
```

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>    //atoi() is defined here
#include <pthread.h>

void* MyTurn (void* arg) {
  int max_iter = *((int*)arg);
  for (int i = 1; i <= max_iter; i++) {
    sleep (1);
    printf ("MyTurn: i=%d\n",i);
  }
  pthread_exit(0);
}

void YourTurn (int max_iter) {
  for (int i = 1; i <= max_iter; i++) {
    sleep (2);
    printf ("  YourTurn: i=%d\n",i);
  }
}

int main (int argc, char *argv[]) {
  int val=5,  num=atoi(argv[1]);
  pthread_t thread;

  pthread_create (&thread,NULL, MyTurn, (void*) &num);
  YourTurn (val);
  pthread_join (thread, NULL);
  return 0;
}
```

DEMO: Code3

Single Thread and Variable “sum”

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#define MAX_ITER 5

int sum=0;

void* add_one (void* arg) {
  for (int i = 1; i <= MAX_ITER; i++) {
    sum++;
    printf (" i=%d  sum=%d\n", i, sum);
  }
  pthread_exit (NULL);
}

int main () {
  pthread_t thread;

  pthread_create (&thread,NULL, add_one, NULL);
  pthread_join (thread, NULL);
  return 0;
}
```

```
~/Lab5$ gcc code3.c
~/Lab5$ ./a.out
 i=1  sum=1
 i=2  sum=2
 i=3  sum=3
 i=4  sum=4
 i=5  sum=5
~/Lab5$
```

Multi-thread & Shared Variable “sum”

```
~/Lab5$ gcc code3-manythread.c -pthread
~/Lab5$ ./a.out
Thread 140420897011264: i=1  sum=1
Thread 140420897011264: i=2  sum=3
Thread 140420897011264: i=3  sum=4
Thread 140420897011264: i=4  sum=5
Thread 140420897011264: i=5  sum=6

Thread 140420888618560: i=1  sum=2
Thread 140420888618560: i=2  sum=7
Thread 140420888618560: i=3  sum=8
Thread 140420888618560: i=4  sum=9
Thread 140420888618560: i=5  sum=10

~/Lab5$ _
```

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#define MAX_ITER 5
#define MAX_THREAD 2

int sum=0;          /* shared variable */

void* add_one (void* arg) {

  for (int i = 1; i <= MAX_ITER; i++) {
    sum++;
    printf ("Thread %ld: i=%d  sum=%d\n", pthread_self(), i, sum);
  }
  printf("\n");
  pthread_exit (NULL);
}

int main () {
  pthread_t thread[MAX_THREAD];

  for (int t=0; t<MAX_THREAD; t++) {
    pthread_create (&thread[t],NULL, add_one,NULL);
  }

  for (int t=0; t<MAX_THREAD; t++) {
    pthread_join (thread[t], NULL);
  }
  return 0;
}
```

Resources:

1. What is Multithreading? https://www.youtube.com/watch?v=0KAGazeMZ2o
2. Introduction to Threads https://www.youtube.com/watch?v=LOfGJcVnvAk
3. Process Management (Processes and Threads) https://www.youtube.com/watch?v=OrM7nZcxXZU
4. How to create and join threads in C (pthreads) https://www.youtube.com/watch?v=uA8X5zNOGw8
5. Multithreading Using pthreads in C language (Part 1) https://www.youtube.com/watch?v=qPhP86HIXgg
6. What are command line arguments (argc and argv)? https://www.youtube.com/watch?v=decAHMKIo_A&list=RDCMUC6qj_bPq6tQ6hLwOBpBQ42Q&index=27
7. What is pthread_t? https://www.yo[illegible]Qmuj1RrCk&list=RDCMUC6qj_bPq6tQ6hLwOBpBQ42Q&index=27