National University of Computer and Emerging Sciences

Operating System Lab – 07

**MULTITHREADING**

Contents

[Objective 2](#_bookmark0)

[What are Threads? 2](#_bookmark1)

[Basic System Calls Related to Multithread Programming 2](#_bookmark2)

[‘pthread\_create()’ System Call 3](#_bookmark3)

[‘pthread\_join()’ System Call 3](#_bookmark4)

[Example 1: Two Threads displaying two strings “Hello” and “How are you?” independent of each other 4](#_bookmark5)

[Example 2: Create a function message() that takes threadid as argument and prints the message with thread id.](#_bookmark6) [There should be atleast four independent threads 5](#_bookmark6)

[Attributes in Threads 5](#_bookmark7)

[System Calls related to Attributes of Threads 6](#_bookmark8)

[‘pthread\_attr\_init()’ System Call 6](#_bookmark9)

[‘pthread\_attr\_setdetachstate()’ System Call 6](#_bookmark10)

[‘pthread\_attr\_destroy()’ System Call 6](#_bookmark11)

[Example 3: Create a detached thread for a function infoThread() 7](#_bookmark12)

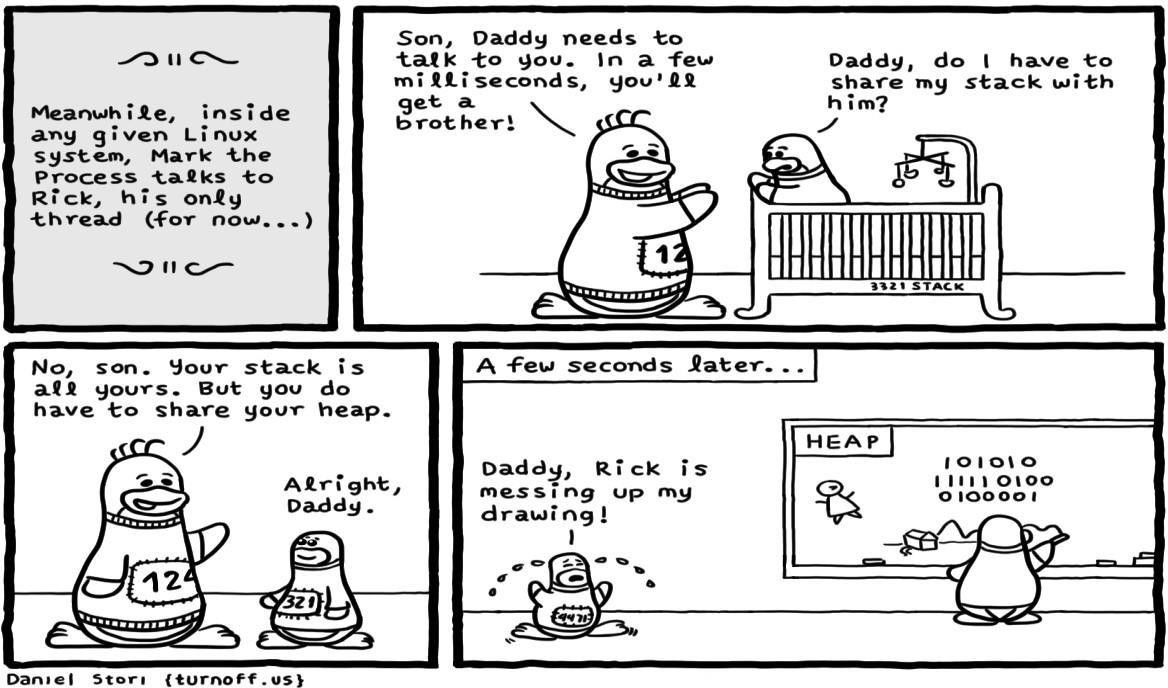
# Objective

Introduction to multithreaded programming using **POSIX thread (pthread) libraries**.

Our main objectives are:

1. Thread creation in Linux
2. Joining of thread in Linux
3. Initializing thread attributes
4. Setting Attribute detach state.
5. Destroying attribute

# Threads

Threads are often described as **lightweight processes**. They can work like two or more processes sharing the same address space i.e., *they will work independently like processes but can share the same global variables*. They are mostly used when two tasks can be done independently without depending much on each other.



# Basic System Calls Related to Multithread Programming

The following are two basic system calls related to multithreaded programming however, there are many system calls available.

|  |  |  |
| --- | --- | --- |
| **S.NO** | **System Call** | **Description** |
| **1** | Pthread\_create() | For creating threads |
| **2** | Pthread\_join() | Wait of thread termination |

### **pthread\_create()** - System Call

This system call is used to create new thread, a syntax is given below.

#include<pthread.h> int pthread\_create(

pthread\_t \*threaded, *//id of thread*

const pthread\_attr\_t \*attr, *//attributes of thread*

void \*(\*start\_routine) (void\*), *//function that is to assign*

void \*arg *//arguments that have to pass to thread function*

);

Example: pthread\_create(&id[0], NULL, printNumber, &arg);

|  |  |  |
| --- | --- | --- |
| Arguments | Syntax | Description |
| ID | pthread\_t \* | Reference (or pointer) to the ID of the thread. |
| attr | pthread\_attr\_t \* | Used to set the attributes of a thread (e.g., the stack size, scheduling policy, etc.) Passing NULL suffices for most applications. |
| Starting routine | void \* | The name of the function that the thread starts to execute. If the function’s return type is void \*, then its name is simply written; otherwise, it must be type-cast to void \*. |
| arg | void \* | This is the argument that the *starting routine* takes. If it takes multiple arguments, a [struct](https://www.educative.io/edpresso/how-to-use-the-typedef-struct-in-c" \t "_blank) is used. |

##### Return Values:

#### If successful it returns 0 otherwise it generates a nonzero number.

### **pthread\_join()** -System Call

This system call waits for the thread specified by thread to terminate. A syntax is shown below:

Int pthread\_join (

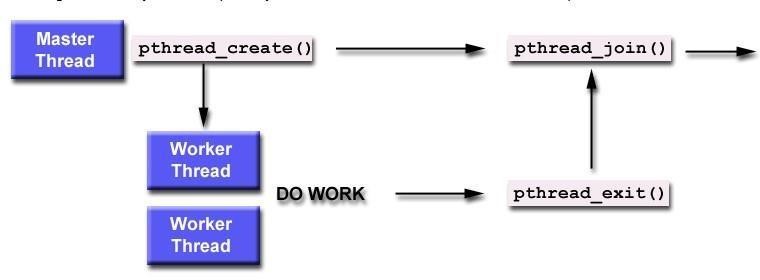
Pthread\_t threaded, *//id of thread which have to join*

void \*\*retval *//return status of thread*

);

##### Return Values:

#### If successful it return 0 otherwise it generates a nonzero number.



## Example 1: Two Threads displaying two strings “Hello” and “How are you?” independent of each other

1. Create a new file thread.c with .c extension using any editor
2. Type the following code.

#include <stdio.h> #include <pthread.h> #include <stdlib.h> void \* thread1()

{

while(1){ printf("Hello!!\n");

}

}

void \* thread2()

{

while(1){

printf("How are you?\n");

}

}

int main()

{

int status; pthread\_t tid1,tid2;

pthread\_create(&tid1,NULL,thread1,NULL); pthread\_create(&tid2,NULL,thread2,NULL); pthread\_join(tid1,NULL); pthread\_join(tid2,NULL);

return 0;}

1. Save and exit.
2. To compile it type the following command on terminal.

gcc thread.c -o thread -lpthread

* **gcc** is the compiler command.
* **thread.c** is the name of c program source file.
* **-o** is option to make object file.
* **thread** is the name of object file.
* **-lpthread** is option to execute pthread.h library file.

1. Run it by using following command.

./thread

The –lpthread at the end to link the pthread library.

## Example 2: Create a function message() that takes threadid as argument and prints the message with thread id. There should be atleast five independent threads

1. Create a new file msgthreads.c with .c extension using any editor
2. Type the following code.

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define NUM\_THREADS 5

void \*PrintHello(void \*threadid)

{

long tid;

tid = (long)threadid;

printf("Hello World! It's me, thread #%ld!\n", tid);

pthread\_exit(NULL);

}

int main (int argc, char \*argv[])

{

pthread\_t threads[NUM\_THREADS];

int rc;

long t;

for(t=0; t<NUM\_THREADS; t++){

printf("In main: creating thread %ld\n", t);

rc = pthread\_create(&threads[t], NULL, PrintHello, (void \*)t);

if (rc){

printf("ERROR; return code from pthread\_create() is %d\n", rc);

exit(-1);

}

}

pthread\_exit(NULL);

}

1. Save and exit.
2. To compile it type the following command on terminal.

gcc msgthreads.c –o msgthreads -lpthread

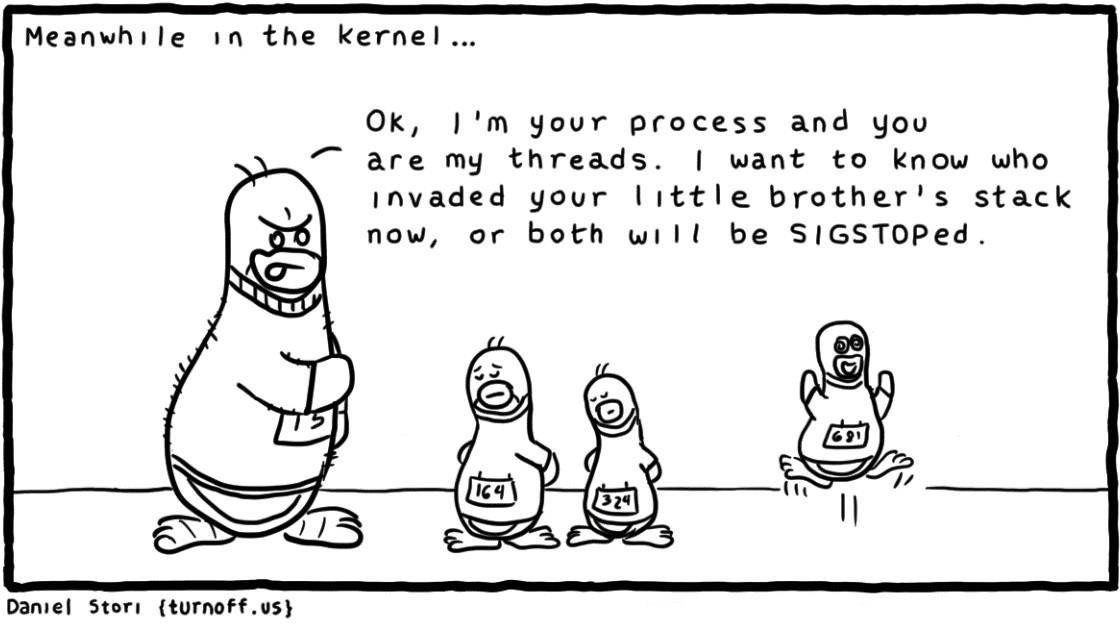
1. Run it by using following command.

./msgthreads

**Note:** remove pthread\_join system call and then observe the changes

# Attributes in Threads

Thread attributes are thread characteristics that affect the behavior of the thread. **NULL** is passed in above examples in place of thread attributes, now we start to place thread attributes that uses default attributes of threads. We may create and customize a thread attribute object to specify other values for the attribute.



# System Calls related to Attributes of Threads

The following are the system calls related to threads’ attribute.

|  |  |  |
| --- | --- | --- |
| S.NO | System Call | Description |
| 1 | pthread\_attr\_init() | Initializes a thread attributes object |
| 2 | pthread\_attr\_setdetachstate() | Controls detach state of a thread |
| 3 | pthread\_attr\_destroy() | Destroys attribute objects |

### 1. **pthread\_attr\_init()** -System Call

This initializes a thread attributes object attr with the default value. The syntax is shown below:

int pthread\_attr\_init(pthread attr t \*attr)

##### Return Values:

#### If successful completion, it will return a 0 otherwise, an error number is returned to indicate the error.

### 2. **pthread\_attr\_setdetachstate()** -System Call

The detachstate attribute controls whether the thread is created in a detached state.

int pthread\_attr\_setdetachstate(pthread\_attr\_t \*attr, int detachstate)

##### PTHEAD\_CREATE\_DETACHED

Thread state is detached means it cannot be joined with other threads.

##### PTHREAD\_CREATE\_JOINABLE

Thread state is joinable means it can be joined with other threads

### 3. **pthread\_attr\_destroy()** -System Call

When a thread attributes objects is no longer required, it should be destroyed using this system call.

int pthread\_attr\_destroy(pthread\_attr\_t \*attr)

##### Return Values:

#### If successful completion, it will return a 0 otherwise, an error number is returned to indicate the error.

[pthread\_self](http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_self.html)()

Syntax

pthread\_t tid;

tid = pthread\_self();

#### **DESCRIPTION** The pthread\_self() function shall return the thread ID of the calling thread.

#### **RETURN VALUE**

#### Refer to the DESCRIPTION.

#### **ERRORS**

#### No errors are defined. The pthread\_self() function shall not return an error code.

## Example 3: Create a detached thread for a function infoThread()

1. Create a new file detachthread.c with .c extension using any editor
2. Type the following code.

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

void \*theThread(void \*parm) {

printf("Entered the thread\n");

return NULL;

}

int main(int argc, char \*\*argv) { pthread\_attr\_t attr; pthread\_t thread;

printf("Create a default thread attributes object\n"); pthread\_attr\_init(&attr);

printf("Set the detach state thread attribute\n"); pthread\_attr\_setdetachstate(&attr,PTHREAD\_CREATE\_DETACHED);

printf("Create a thread using the new attributes\n");

pthread\_create(&thread, &attr, theThread, NULL);

printf("Destroy thread attributes object\n");

pthread\_attr\_destroy(&attr);

int rc;

rc = pthread\_join(thread, NULL);

printf("Join now fails because the detach state attribute was changed\n pthread\_join returns non zero value %d",rc);

printf("Main completed\n"); return 0;

}

1. Save and exit.
2. To compile it type the following command on terminal.

gcc detachthread.c –o detachthread –lpthread

1. Run it by using following command.

./detachthread

You can get much information about these attributes and more information about system calls related to thread attributes: follow the links below

1. <https://docs.oracle.com/cd/E19455-01/806-5257/6je9h032j/index.html>
2. <http://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html>
3. <https://vcansimplify.wordpress.com/2013/03/08/pthread-tutorial-simplified/>

**Lab Activity**

* 1. Write a program that create 3 threads

a) On successful creation, print “Thread #” in its starting routine and terminate themselves by showing their return value.

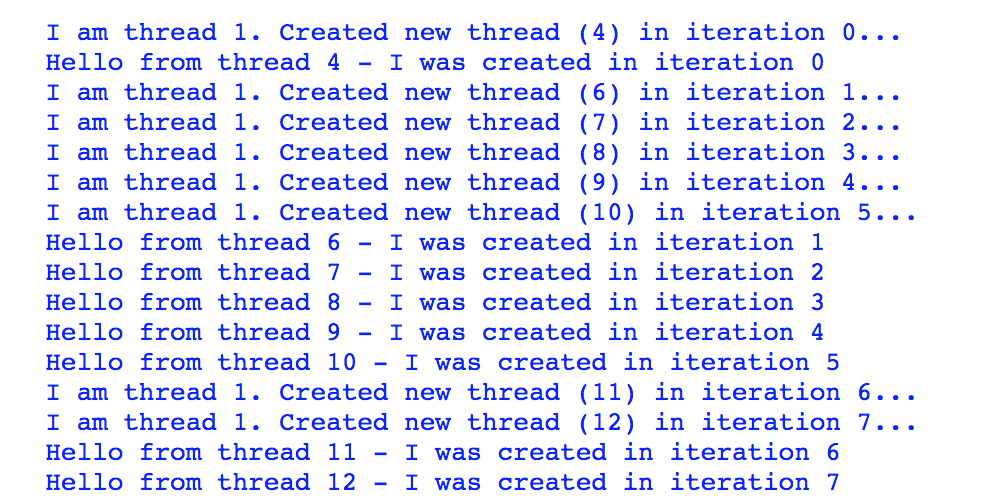
b) On unsuccessful creation, Print “Error”.

Thread 1  
Thread 2

Thread 3  
Thread 1 returns: 0  
Thread 2 returns: 0

Thread 3 returns: 0

* 1. Write a program, which make 4 threads. Each thread will print one table out of [5678] up to 1000.
  2. Write a program that creates N number of threads specified in the command line, each prints “hello message and its own thread ID”. Sleep the main thread for 1 second and create every 4 or 5 threads. The output of your code should alike:



* 1. Write a program to sum 10 elements of an array by multithreading.
  2. Procom has 4 volunteers on their front desk.
     + Volunteer 1 manages On day registration
     + Volunteer 2 handles announcements
     + Volunteer 3 handles sponsors
     + Volunteer 4 resolve queries of participants Implement this system using pthread for 100 participants.