

```
#include<pthread.h>
#include<stdio.h>
int sum;
void *runner ( void *param );
int main (int argc, char *argv[])
   sum=0;
  pthread_t tid1, tid2;
   pthread attr tattr;
   pthread attr init(&attr);
   pthread create(&tid1, &attr,
          runner, argv[1]);
   pthread create(&tid2, &attr,
          runner, argv[2]);
   pthread join(tid1, NULL);
   pthread join(tid2, NULL);
   printf( "sum = %d \n", sum);
```

```
void *runner ( void *param )
   int upper = atoi (param);
   int i;
   if (upper > 0)
         for ( i=1; i <= upper; i++ )
            sum = sum + i; }
  printf(" The i value is %d and
         the sum value is %d\n",i,
         sum);
   pthread exit(0);
```

## **Threads**

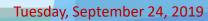
```
#include<pthread.h>
#include<stdio.h>
int value1, value2;
void *runner ( void *param );
void *fact ( void *param );
int main (int argc, char *argv[])
          pthread t tid1,tid2;
          pthread attr t attr;
          pthread_attr_init(&attr);
          pthread_create(&tid1, &attr, runner1, argv[1]);
          pthread create(&tid2, &attr, runner2, argv[2]);
          pthread join(tid1, NULL);
          pthread join(tid2, NULL);
          printf( "value1 = %d\t value2 = %d \n", value1,value2);
```

```
void *runner ( void *param )
   int upper = atoi (param);
   int i; value1=0;
   if (upper > 0)
        for ( i=1; i <= upper; i++ )
        { value1 = value1 + i; }
  printf(" New thread runner:
value1 = %d, value2 =
%d\n",value1, value2);
  pthread exit(0);
```

```
void *fact( void *param )
   int upper = atoi (param);
  int i; value2=0;
  if (upper > = 0)
         value2=1;
         for ( i=1; i <= upper; i++ )
           value2 = value2 * i; }
printf(" New thread fact: value1
= %d, value2 = %d\n",value1,
value2);
   pthread exit(0);
```

# **Thread Related Operations**

- Thread creation
- Thread termination
- Thread synchronization
- Thread scheduling
- Thread data management
- Thread / process interaction







- extern int pthread\_create (pthread\_t \*tid, \_\_const pthread\_attr\_t \*attr, void \*(\*\_\_start\_routine) (void \*), void \*arg)
  - Creates a new thread of control that executes concurrently with the calling thread.
  - The new thread applies the function start\_routine passing it arg as first argument.
  - The new thread terminates either explicitly, by calling pthread\_exit(3), or implicitly, by returning from the start\_routine function.
  - The attr argument specifies thread attributes to be applied to the new thread.
  - The attr argument can also be NULL, in which case default attributes are used





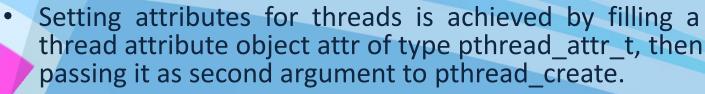
#### Return value

- On success, the identifier of the newly created thread is stored in the location pointed by the thread argument,
- and a 0 is returned.
- On error, a non-zero error code is returned.

### Errors

- EAGAIN not enough system resources to create a process for the new thread.
- EAGAIN more than PTHREAD\_THREADS\_MAX threads are already active.





- Passing NULL is equivalent to passing a thread attribute object with all attributes set to their default values.
- Thread attribute structure is in /usr/include/bits/pthreadtypes.h

```
#define __SIZEOF_PTHREAD_ATTR_T 56 typedef union
```

```
char __size[__SIZEOF_PTHREAD_ATTR_T];
long int __align;
} pthread_attr_t;
```

Detachstate, Schedpolicy, Sched\_param structure, Inheritsched, Scope will be a part of the attribute



## **Attribute Initialization & Destroy**

extern int pthread\_attr\_init (pthread\_attr\_t \*attr)

- Initializes the thread attribute object attr and fills it with default values for the attributes.
- Attribute objects are consulted only when creating a new thread.
- The same attribute object can be used for creating several threads. Modifying an attribute object after a call to pthread\_create does not change the attributes of the thread previously created.
- extern int pthread\_attr\_destroy(pthread\_attr\_t \*attr)
  - Destroys a thread attribute object, which must not be reused until it is reinitialized.
  - pthread\_attr\_destroy does nothing in the LinuxThreads implementation.



## **Detach State**

- Control whether the thread is created in the joinable state (value PTHREAD\_CREATE\_JOINABLE) or in the detached state (PTHREAD\_CREATE\_DETACHED).
- Default value: PTHREAD\_CREATE\_JOINABLE.
- Joinable state
  - Another thread can synchronize on the thread termination and recover its termination code using pthread\_join
  - some of the thread resources are kept allocated after the thread terminates, and reclaimed only when another thread performs pthread\_join on that thread.
- Detached state
  - The thread resources are immediately freed when it terminates
  - pthread\_join cannot be used to synchronize on the thread termination

## **Detach State**

- A thread created in the joinable state can later be put in the detached thread using pthread\_detach.
- extern int pthread\_attr\_setdetachstate
   (pthread\_attr\_t \*attr, int detachstate)

 extern int pthread\_attr\_getdetachstate (\_\_const pthread\_attr\_t \*attr, int \*detachstate)



- Select the scheduling policy for the thread: one of SCHED\_OTHER (regular, non-realtime scheduling), SCHED\_RR (realtime, round-robin) or SCHED\_FIFO (realtime, first-in first-out).
- Default value: SCHED\_OTHER.
- The real time scheduling policies SCHED\_RR and SCHED\_FIFO are available only to processes with super user privileges.
- The scheduling policy of a thread can be changed after creation with pthread setschedparam

# **Sched Policy**

extern int pthread\_attr\_setschedpolicy (pthread\_attr\_t \*attr, int policy)

extern int pthread\_attr\_getschedpolicy (\_\_const pthread\_attr\_t \*attr, int \*policy)



- Contain the scheduling parameters (essentially, the scheduling priority) for the thread.
- See sched\_setparam for more information on scheduling parameters.
- Default value: priority is 0.
- This attribute is not significant if the scheduling policy is SCHED\_OTHER; it only matters for the realtime policies SCHED\_RR and SCHED\_FIFO.
- The scheduling priority of a thread can be changed after creation with pthread\_setschedparam

## **Sched Param**

extern int pthread\_attr\_setschedparam (pthread\_attr\_t \*attr, \_\_const struct sched\_param \*param)

 extern int pthread\_attr\_getschedparam (\_\_const pthread\_attr\_t \*attr, struct sched\_param \*param)

See struct sched\_param in bits/sched.h