



Operating Systems CS F372

Threads


BIJU K RAVEENDRAN



```
#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_t attr;
    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



Thread Creation

- pthread_create
- 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



Thread Creation

- 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.





- Setting attributes for threads is achieved by filling a thread attribute object attr of type pthread_attr_t, then passing it as second argument to pthread_create.
- 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)`

Sched Policy

- 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)

Sched Param

- 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

