Computer Operating Systems

Threads

T. Tolga Sarı (sarita@itu.edu.tr)
Sultan Çoğay (cogay@itu.edu.tr)
Doğukan Arslan (arslan.dogukan@itu.edu.tr)

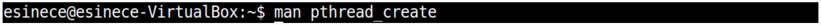
March 16, 2022

Today

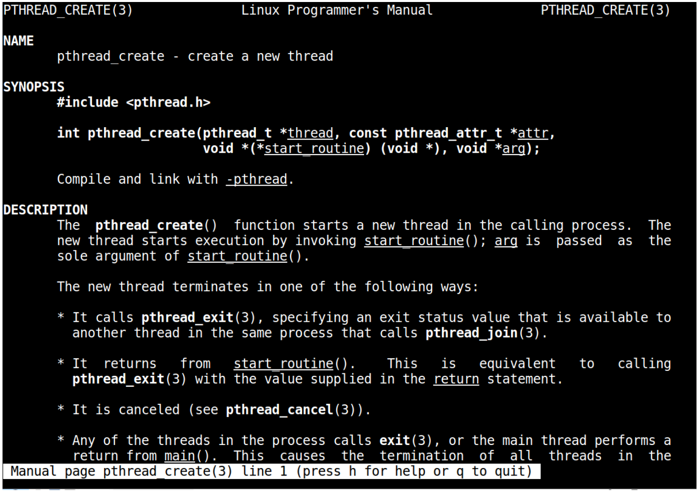
Thread Creation and Termination

- Thread libraries provide functions for creating threads and other operations.
- To create a thread, `pthread_create()` function can be used which is defined in `<pthread.h>`
- To terminate threads, `pthread_exit()` function defined in the same library can be used.

man pthread_create

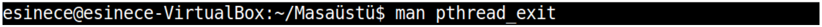


```
esinece@esinece-VirtualBox:~$ man pthread_create
```

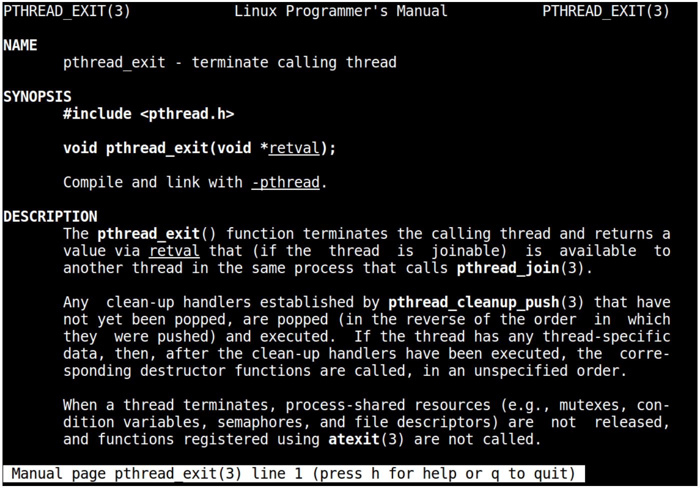


```
PTHREAD_CREATE(3)          Linux Programmer's Manual          PTHREAD_CREATE(3)

NAME
       pthread_create - create a new thread

SYNOPSIS
       #include <pthread.h>

       int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                          void *(*start_routine) (void *), void *arg);

       Compile and link with -pthread.

DESCRIPTION
       The  pthread_create()  function starts a new thread in the calling process.  The
       new thread starts execution by invoking start_routine(); arg is  passed  as  the
       sole argument of start_routine().

       The new thread terminates in one of the following ways:

       * It calls pthread_exit(3), specifying an exit status value that is available to
         another thread in the same process that calls pthread_join(3).

       * It  returns  from  start_routine().   This  is  equivalent  to  calling
         pthread_exit(3) with the value supplied in the return statement.

       * It is canceled (see pthread_cancel(3)).

       * Any of the threads in the process calls exit(3), or the main thread performs a
         return from main().  This  causes  the  termination  of  all  threads  in  the
 Manual page pthread_create(3) line 1 (press h for help or q to quit)
```

man pthread_exit

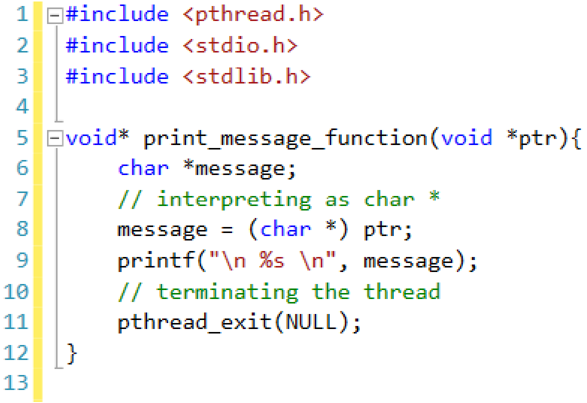


```
esinece@esinece-VirtualBox:~/Masaüstü$ man pthread_exit
```



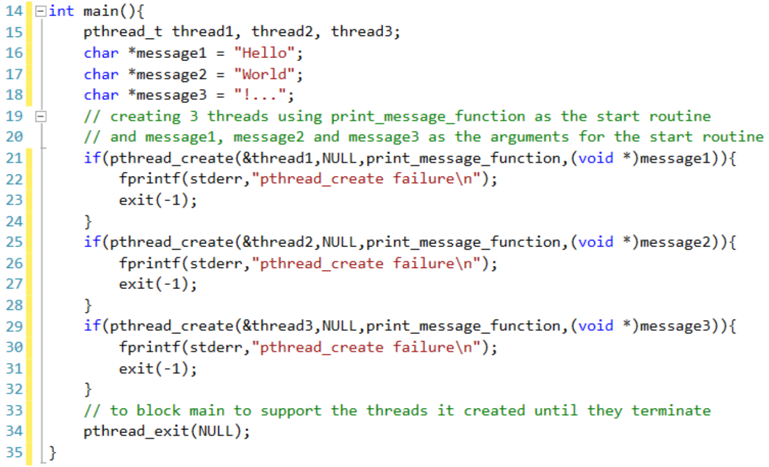
```
PTHREAD_EXIT(3)            Linux Programmer's Manual           PTHREAD_EXIT(3)

NAME
       pthread_exit - terminate calling thread

SYNOPSIS
       #include <pthread.h>

       void pthread_exit(void *retval);

       Compile and link with -pthread.

DESCRIPTION
       The pthread_exit() function terminates the calling thread and returns a
       value via retval that (if the  thread  is  joinable)  is  available  to
       another thread in the same process that calls pthread_join(3).

       Any  clean-up handlers established by pthread_cleanup_push(3) that have
       not yet been popped, are popped (in the reverse of the order  in  which
       they  were pushed) and executed.  If the thread has any thread-specific
       data, then, after the clean-up handlers have been executed, the  corre-
       sponding destructor functions are called, in an unspecified order.

       When a thread terminates, process-shared resources (e.g., mutexes, con-
       dition variables, semaphores, and file descriptors) are  not  released,
       and functions registered using atexit(3) are not called.

 Manual page pthread_exit(3) line 1 (press h for help or q to quit)
```

Example Program 1



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

void* print_message_function(void *ptr){
    char *message;
    // interpreting as char *
    message = (char *) ptr;
    printf("\n %s \n", message);
    // terminating the thread
    pthread_exit(NULL);
}

```

Example Program 1



```c
int main(){
    pthread_t thread1, thread2, thread3;
    char *message1 = "Hello";
    char *message2 = "World";
    char *message3 = "!...";
    // creating 3 threads using print_message_function as the start routine
    // and message1, message2 and message3 as the arguments for the start routine
    if(pthread_create(&thread1,NULL,print_message_function,(void *)message1)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
    }
    if(pthread_create(&thread2,NULL,print_message_function,(void *)message2)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
    }
    if(pthread_create(&thread3,NULL,print_message_function,(void *)message3)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
    }
    // to block main to support the threads it created until they terminate
    pthread_exit(NULL);
}
```

Compiling a Program Including Thread/s

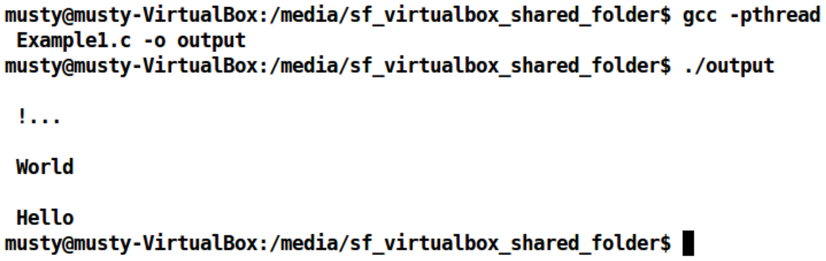
- These applications should be linked with thread library. Sample, proper compilation:
  ```
  gcc source.c -o output -pthread
  ```

Compiling a Program Including Thread/s

- These applications should be linked with thread library. Sample, proper compilation:

```
gcc source.c -o output -pthread
```

Output of the Example Program 1

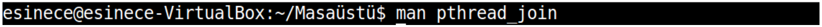


```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc -pthread
 Example1.c -o output
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./output

 !...

 World

 Hello
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ █
```

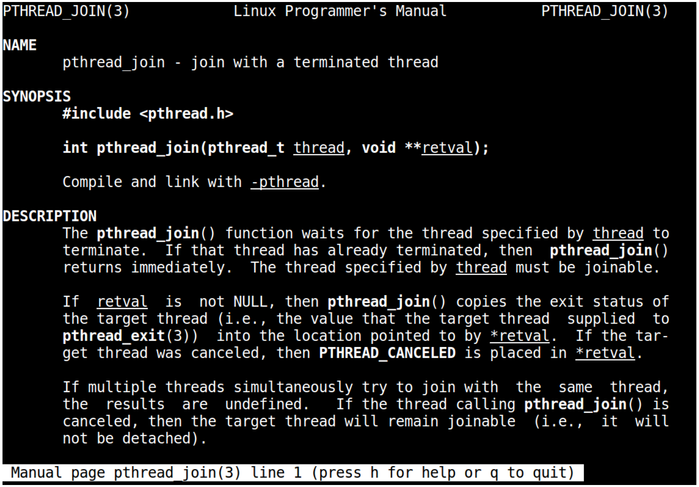
Join

- Sometimes a thread is created to do a specific part of a job. Another thread should wait until that thread completed. Waiting mechanism can be done with the pthread_join() function.
- "Join" is a way to achieve synchronization between threads
- pthread_join() stops the calling thread from executing until the thread at the specified id ends.

man pthread_join

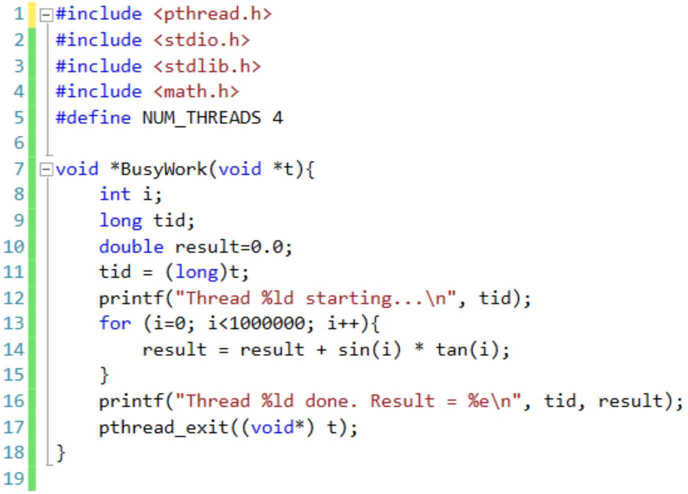


```
esinece@esinece-VirtualBox:~/Masaüstü$ man pthread_join
```



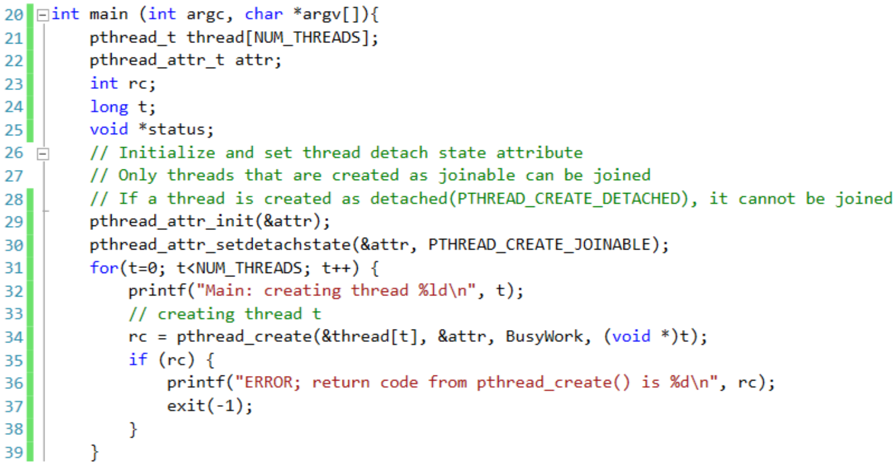
```
PTHREAD_JOIN(3)          Linux Programmer's Manual         PTHREAD_JOIN(3)

NAME
       pthread_join - join with a terminated thread

SYNOPSIS
       #include <pthread.h>

       int pthread_join(pthread_t thread, void **retval);

       Compile and link with -pthread.

DESCRIPTION
       The pthread_join() function waits for the thread specified by thread to
       terminate.  If that thread has already terminated, then  pthread_join()
       returns immediately.  The thread specified by thread must be joinable.

       If  retval  is  not NULL, then pthread_join() copies the exit status of
       the target thread (i.e., the value that the target thread  supplied  to
       pthread_exit(3))  into the location pointed to by *retval.  If the tar-
       get thread was canceled, then PTHREAD_CANCELED is placed in *retval.

       If multiple threads simultaneously try to join with  the  same  thread,
       the  results  are  undefined.   If the thread calling pthread_join() is
       canceled, then the target thread will remain joinable  (i.e.,  it  will
       not be detached).

 Manual page pthread_join(3) line 1 (press h for help or q to quit)
```

Example Program 2



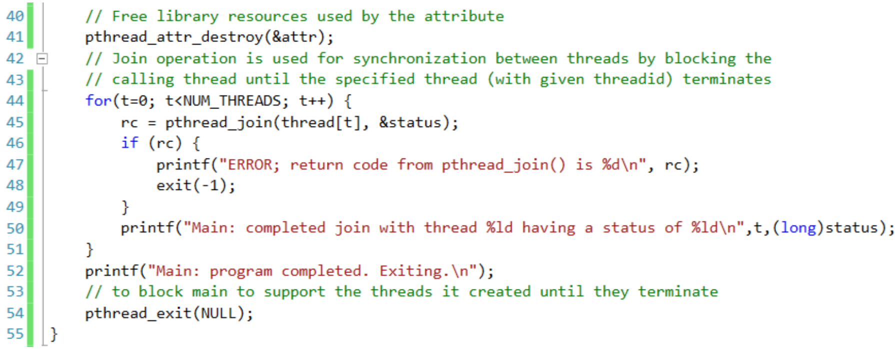
```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_THREADS 4

void *BusyWork(void *t){
     int i;
     long tid;
     double result=0.0;
     tid = (long)t;
     printf("Thread %ld starting...\n", tid);
     for (i=0; i<1000000; i++){
         result = result + sin(i) * tan(i);
     }
     printf("Thread %ld done. Result = %e\n", tid, result);
     pthread_exit((void*) t);
}

```

Example Program 2



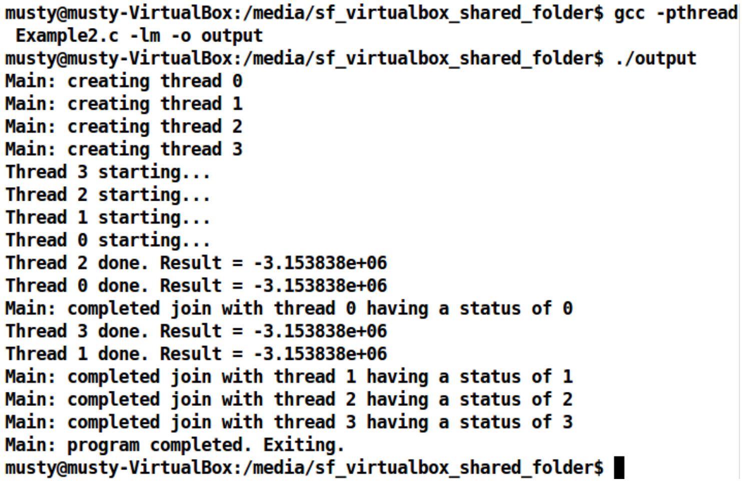
```c
int main (int argc, char *argv[]){
    pthread_t thread[NUM_THREADS];
    pthread_attr_t attr;
    int rc;
    long t;
    void *status;
    // Initialize and set thread detach state attribute
    // Only threads that are created as joinable can be joined
    // If a thread is created as detached(PTHREAD_CREATE_DETACHED), it cannot be joined
    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
    for(t=0; t<NUM_THREADS; t++) {
        printf("Main: creating thread %ld\n", t);
        // creating thread t
        rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }
```

Example Program 2



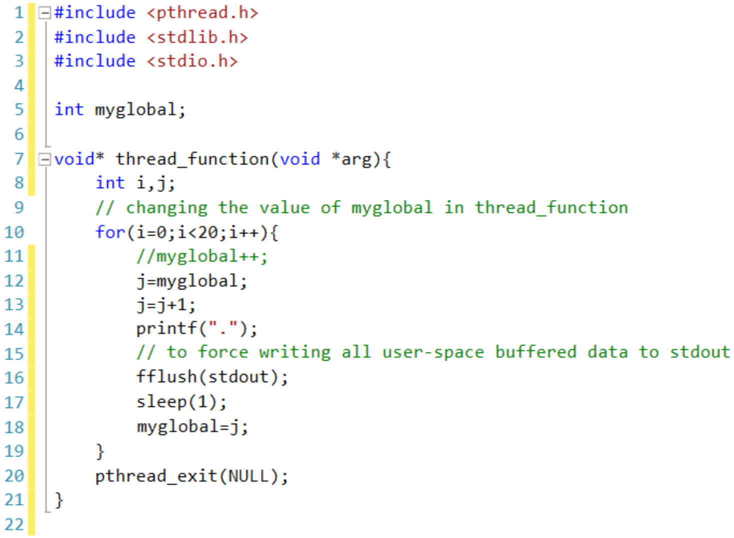
```
    // Free library resources used by the attribute
    pthread_attr_destroy(&attr);
    // Join operation is used for synchronization between threads by blocking the
    // calling thread until the specified thread (with given threadid) terminates
    for(t=0; t<NUM_THREADS; t++) {
        rc = pthread_join(thread[t], &status);
        if (rc) {
            printf("ERROR; return code from pthread_join() is %d\n", rc);
            exit(-1);
        }
        printf("Main: completed join with thread %ld having a status of %ld\n",t,(long)status);
    }
    printf("Main: program completed. Exiting.\n");
    // to block main to support the threads it created until they terminate
    pthread_exit(NULL);
}
```

Output of the Example Program 2



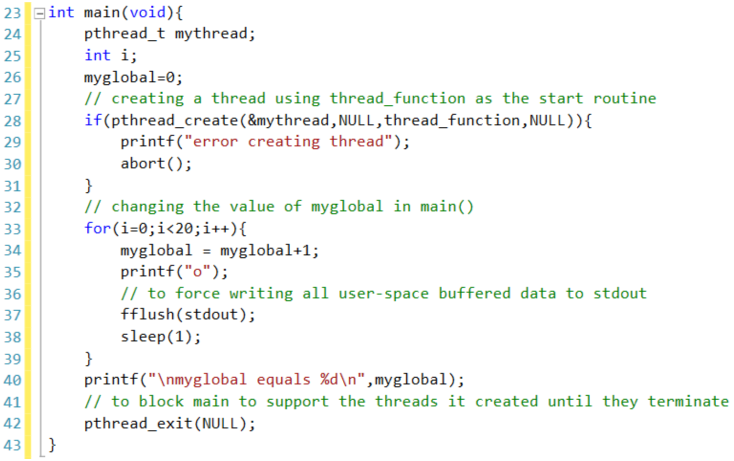
```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc -pthread
 Example2.c -lm -o output
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./output
Main: creating thread 0
Main: creating thread 1
Main: creating thread 2
Main: creating thread 3
Thread 3 starting...
Thread 2 starting...
Thread 1 starting...
Thread 0 starting...
Thread 2 done. Result = -3.153838e+06
Thread 0 done. Result = -3.153838e+06
Main: completed join with thread 0 having a status of 0
Thread 3 done. Result = -3.153838e+06
Thread 1 done. Result = -3.153838e+06
Main: completed join with thread 1 having a status of 1
Main: completed join with thread 2 having a status of 2
Main: completed join with thread 3 having a status of 3
Main: program completed. Exiting.
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$
```

Example Program 3



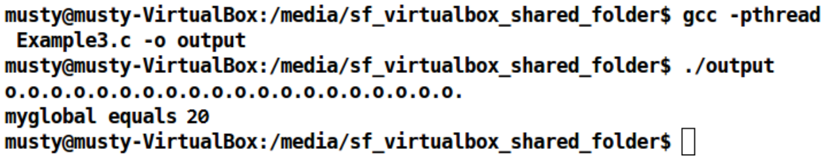
```c
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>

int myglobal;

void* thread_function(void *arg){
    int i,j;
    // changing the value of myglobal in thread_function
    for(i=0;i<20;i++){
        //myglobal++;
        j=myglobal;
        j=j+1;
        printf(".");
        // to force writing all user-space buffered data to stdout
        fflush(stdout);
        sleep(1);
        myglobal=j;
    }
    pthread_exit(NULL);
}

```

Example Program 3



```c
int main(void){
    pthread_t mythread;
    int i;
    myglobal=0;
    // creating a thread using thread_function as the start routine
    if(pthread_create(&mythread,NULL,thread_function,NULL)){
        printf("error creating thread");
        abort();
    }
    // changing the value of myglobal in main()
    for(i=0;i<20;i++){
        myglobal = myglobal+1;
        printf("o");
        // to force writing all user-space buffered data to stdout
        fflush(stdout);
        sleep(1);
    }
    printf("\nmyglobal equals %d\n",myglobal);
    // to block main to support the threads it created until they terminate
    pthread_exit(NULL);
}
```

Output of the Example Program 3



```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc -pthread
 Example3.c -o output
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./output
o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.o.
myglobal equals 20
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ 
```

References

1. https://computing.llnl.gov/tutorials/pthreads/
2. https://docs.google.com/presentation/d/1nfUGmM9W8tA4GSh4Uucb0rZDwO01udlf/edit?usp=sharing&ouid=116515393822328287464&rtpof=true&sd=true