Threads Programming in Linux: Examples:

pthread_create - create a new thread

#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*.

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.

* 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 process.

The ***attr*** argument points to a ***pthread_attr_t*** structure whose contents are used at thread creation time to determine attributes for the new thread; this structure is initialized using pthread_attr_init(3) and related functions. If *attr* is NULL, then the thread is created with default attributes.

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ vi thread1.c
/* ******************************************************************************
Write a program to create a thread that displays a WELCOME message.* Compiling with libraries: the libraries should follow sources and objects on command line. *
****************************************************************************** */
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>

void *func(void *);
char *mesg = "WELCOME";
int  global_var = 10;

int main()
{
   pthread_t thread;
   int r;
   r = pthread_create(&thread, NULL, func, (void*)mesg);
   //printf("Checking the value of global_var=%d\n",global_var);
```

```
  if (r != 0) {
     printf("\n Failed to create a thread.\n");
     exit(-1);
  }
  sleep(2);
  printf("Checking the value of global_var=%d\n",global_var);

  return 0;
} // main ended....

void *func(void *p) {
   global_var = global_var + 10;
   printf("\n Argument is %s and global_var=%d\n",(char *)p,global_var);
   return 0;
}
```

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ gcc -pthread -o thread1 thread1.c
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ ./thread1
 Argument is WELCOME and global_var=20
Checking the value of global_var=20
```

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ vi multiplethread.c
```

```
/* ***********************************************************************************
* Write a program that creates multiple threads and terminate those. The program should create 5
* threads with the pthread_create()  routine. Each thread prints a "Hello World!" message and then
* terminates with a call to pthread_exit().
* ********************************************************************************** */
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>

int gvar = 0;
void *func(void *a) {
  gvar++;
  printf("\n Thread %d created and the argument is:%s \n",gvar,(char *)a);
  pthread_exit("\nDone\n");
}

int main() {
   pthread_t th[5];
   int ch,i;
   void *arg;
   char *msg = "Hello World";

  for (i=0; i<= 4; i++) {
     ch = pthread_create(&th[i],NULL,func,(void *)msg);
     if (ch!=0) {  printf("\n Failed to create for thread no. %d \n", i); exit(-1);  }
     sleep(2);
  }
  printf("In the main process after creation of %d threads.\n",gvar);
  return 0;
} // end of main
```

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ gcc -pthread -o multiplethread multiplethread.c
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ ./multiplethread
 Thread 1 created and the argument is:Hello World
 Thread 2 created and the argument is:Hello World
 Thread 3 created and the argument is:Hello World
 Thread 4 created and the argument is:Hello World
 Thread 5 created and the argument is:Hello World
In the main process after creation of 5 threads.
```

SYNOPSIS (http://man7.org/linux/man-pages/man3/pthread_join.3.html)

#include <pthread.h>
int pthread_join(pthread_t *thread*, **void *****retval***);**

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 target 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).

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ vi thread3.c
/* ******************************************************************************
* Write a program which implements thread with arguments and thread joining.
* ****************************************************************************** */
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *func (void *a){
  printf("\n Argument is : %s \n", (char *)a);
  pthread_exit("Thread exiting...Done");
}

int main()
{
  pthread_t th;
  int ch;
  void *arg;

  char *msg = "Thread Program";
  ch = pthread_create(&th, NULL, func, (void *)msg);
  if (ch != 0) { printf("\nThread creation failed.\n"); exit(-1); }
  ch = pthread_join(th, &arg);
  if (ch != 0) { printf("\nCreation of joinable thread failed."); exit(-1); }

  printf("\n After thread join successful, return value: %s \n",arg);
  return 0;
}
```

```
nilina@nilina-HP-Pro-3330-MT:~/Desktop/csen3113$ ./thread4
 Argument is: Thread Program
 After thread join successful, return value: Thread exiting...Done
```