TIỂU TRÌNH

I. BIÊN DỊCH

gcc –o example example.c –lpthread

II. VÍ DỤ

1. Ví dụ 1

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

void *print_message_function( void *arg );

main()
{
    pthread_t thread1, thread2;
    char *message1 = "Thread 1";
    char *message2 = "Thread 2";
    int  iret1, iret2;

    /* Create independent threads each of which will execute function */

    iret1 = pthread_create( &thread1, NULL, print_message_function, (void*) message1);
    if(iret1)
    {
        fprintf(stderr,"Error - pthread_create() return code: %d\n",iret1);
        exit(EXIT_FAILURE);
    }

    iret2 = pthread_create( &thread2, NULL, print_message_function, (void*) message2);
    if(iret2)
    {

        fprintf(stderr,"Error - pthread_create() return code: %d\n",iret2);
        exit(EXIT_FAILURE);
    }

    printf("pthread_create() for thread 1 returns: %d\n",iret1);
    printf("pthread_create() for thread 2 returns: %d\n",iret2);


    pthread_join( thread1, NULL);
    pthread_join( thread2, NULL);

    exit(EXIT_SUCCESS);
}

void *print_message_function( void *arg )
{
    char *message;
    message = (char *) arg;
    printf("%s \n", message);
}
```

2. Ví dụ 2

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

/* This is our thread function.  It is like main(), but for a thread */
void *threadFunc(void *arg)
{
    char *str;
    int i = 0;

    str=(char*)arg;

    while(i < 10 )
    {
        usleep(1);
        printf("threadFunc says: %s %d\n",str,i);
        ++i;
    }

    return NULL;
}

int main(void)
{
    pthread_t pth;   // this is our thread identifier
    int i = 0;

    /* Create worker thread */
    pthread_create(&pth,NULL,threadFunc,"processing...");

    /* wait for our thread to finish before continuing */
    pthread_join(pth, NULL);

    while(i < 10 )
    {
        usleep(1);
        printf("main() is running... %d\n",i);
        ++i;
    }

    return 0;
}
```

3. Ví dụ 3

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

/* This is our thread function.  It is like main(), but for a thread*/
void *threadFunc(void *arg)
{
    char *str;
    int i = 0;

    str=(char*)arg;

    while(i < 14 )
    {
        usleep(1);
        printf("threadFunc says: %s %d\n",str,i);
        ++i;
    }

    return NULL;
}

int main(void)
{
    pthread_t pth;   // this is our thread identifier
    int i = 0;

    pthread_create(&pth,NULL,threadFunc,"processing...");

    while(i < 10)
    {
        usleep(1);
        printf("main is running... %d\n",i);
        ++i;
    }

    printf("main waiting for thread to terminate...\n");
    pthread_join(pth,NULL);

    return 0;
}
```

4. Ví dụ 4

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

/* This is our thread function.  It is like main(), but for a thread*/
void *threadFunc(void *arg)
{
    int mul = 0;
    int* t;
    t = (int*)arg;
    mul = t[0] * t[1];
    printf("threadFunc says: %d\n",mul);
    return (void*) mul;
}

int main(void)
{
    pthread_t pth1, pth2;    // this is our thread identifier
    int sum = 0;
    int a[2]={4,6};
    int b[2]={5,7};
    void* result;

    pthread_create(&pth1,NULL,threadFunc,(void*)a);
    pthread_create(&pth2,NULL,threadFunc,(void*)b);

    pthread_join(pth1,&result);
    sum = (int)result;
    pthread_join(pth2,&result);
    sum += (int)result;

    printf("Sum = %d \n",sum);
    return 0;
}
```