OPERATING SYSTEMS

Lab-5: Thread Programs

Name – Vinay Santosh Menon
Registration Number - 20BAI1103

TASK:

To understand how threads work using C programs.

Program1:

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

void* first_thread(void* vargp)
{
   printf("Hello World\nFirst thread process\n");

}
int main()
{
   pthread_t pthread_id;
   printf("Before thread creation\n");
   pthread_create(&pthread_id,NULL,first_thread,NULL);
   pthread_join(pthread_id,NULL);
   printf("After thread creation\n");

}
```

Output:

```
vanitas@vinay:~/Documents/Code/OS/Thread$ gcc thread1.c
vanitas@vinay:~/Documents/Code/OS/Thread$ ./a.out
Before thread creation
Hello World
First thread process
After thread creation
vanitas@vinay:~/Documents/Code/OS/Thread$
```

Explanation:
Here we just create threads using pthread_create, we pass it to the function, and use join so that only after the thread function is executed, the main function continues.

Program 2:

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

void* trial_thread(void* vargp)
{
   int m = 0;
   while(m<1000)
   {
       printf("I am a thread function\n");
       m++;
   }
}
int main()
{
   pthread_t pthread_id;
   int ret , n=0, m= 0;
   ret = pthread_create(&pthread_id,NULL,&trial_thread,NULL);
   //pthread_join(pthread_id,NULL);
   if(ret==0)
   {
       printf("Thread process created successfully.\n");
   }
   else
   {
       printf("Thread process not created.\n");
       return 0;
```

```
    }
    while(n<1000)
    {
        printf("I am main function.\n");
        n++;
    }
}
```

Output:

```
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am main function.
I am a thread function
I am a thread function
I am main function.
I am main function.
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
I am a thread function
```

Explanation:

The print statements are randomly getting executed by thread and main functions as we have not used join function, if join function is used only after the thread function is over, the main function will continue.

Program-3:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void* printmsg(void* ptr)
{
   char *txt;
   txt = (char *)ptr;
   printf("%s\n",txt);
}
int main()
{
   pthread_t pthread1,pthread2;
   char* msg1 = "Thread 1";
   char* msg2 = "Thread 2";
   int iret1,iret2;
   iret1 = pthread_create(&pthread1, NULL, printmsg, (void*)
msg1);
   iret2 = pthread_create(&pthread2, NULL, printmsg, (void*)
msg2);
   pthread_join(pthread1,NULL);
   pthread_join(pthread2,NULL);
   printf("Thread 1 returns: %d\n",iret1);
   printf("Thread 2 returns: %d\n",iret2);
   exit(0);
```

```
}
```

Output:

```
vanitas@vinay:~/Documents/Code/OS/Thread$ gcc thread3.c
vanitas@vinay:~/Documents/Code/OS/Thread$ ./a.out
Thread 2
Thread 1
Thread 1 returns: 0
Thread 2 returns: 0
vanitas@vinay:~/Documents/Code/OS/Thread$
```

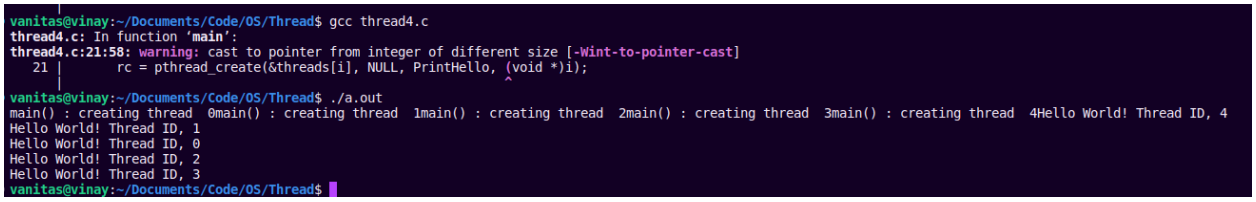
Program-4:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define NUM_THREADS 5
void *PrintHello(void *threadid)
{
  long tid;
  tid = (long)threadid;
  printf("Hello World! Thread ID, %ld\n", tid);
  pthread_exit(NULL);
   }

int main ()
{
  pthread_t threads[NUM_THREADS];
  int rc;
  int i;
  for( i = 0; i < NUM_THREADS; i++ )
{
     printf ( "main() : creating thread  %d" ,i );
     rc = pthread_create(&threads[i], NULL, PrintHello, (void
*)i);
     if (rc)
{
```

```
        printf("Error:unable to create thread, %d\n", rc);
        exit(-1);
      }
   }
   pthread_exit(NULL);
}
```

Output:



Explanation:

We follow the same procedure, but for multiple threads using an array.