Amrita Vishwa Vidyapeetham

Amrita School of Engineering, Coimbatore

Department of Computer Science and Engineering

Topic: Threads Practice

19CSE213 Operating Systems Laboratory – Threads Practice

---

1. Simple Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void *myThreadFun(void *ab)
{
        sleep(1);
        printf("Printing Srings from threads \n");
        return NULL;
}
int main()
{
        pthread_t thread_id;
        printf("Before Thread\n");
        pthread_create(&thread_id, NULL, myThreadFun, NULL);
        pthread_join(thread_id, NULL);
        printf("After Thread\n");
        exit(0);
}
```

- **Explaination :**

  ✓ In main() we declare a variable called thread_id, which is of type pthread_t. After declaring thread_id, we call pthread_create() function.
  - The first argument is a pointer to thread_id
  - The second argument specifies attributes. If it is NULL, then default attributes shall be used.
  - The third argument is name of function to be executed for the thread to be created.
  - The fourth argument is used to pass arguments to the function.

Output :

Open p1.c ~/lab3

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


void *myThreadFun(void *ab)
{
        sleep(1);
        printf("Printing Srings from threads \n");
        return NULL;
}

int main()
{
        pthread_t thread_id;
        printf("Before Thread\n");
        pthread_create(&thread_id, NULL, myThreadFun, NULL);
        pthread_join(thread_id, NULL);
        printf("After Thread\n");
        exit(0);
}

```

abhinav@abhinav: ~/lab3

```
abhinav@abhinav:~/lab3$ gcc p1.c -lpthread
abhinav@abhinav:~/lab3$ ./a.out
Before Thread
Printing Srings from threads
After Thread
abhinav@abhinav:~/lab3$
```

2.With global and static variables:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
int g = 0;
void *myThreadFun(void *vargp)
{
 int *myid = (int *)vargp;
 static int s = 0;
 ++s; ++g;
 printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}
int main()
{
 int i;
 pthread_t tid;
 for (i = 0; i < 3; i++)
 pthread_create(&tid, NULL, myThreadFun, (void *)&tid);
pthread_join(tid,NULL);
 pthread_exit(NULL);
 return 0;
}
```

Output :

Open

p2.c
~/lab05032021

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
// Let us create a global variable to change it in threads
int g = 0;
// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
 // Store the value argument passed to this thread
 int *myid = (int *)vargp;
 // Let us create a static variable to observe its changes
 static int s = 0;
 // Change static and global variables
 ++s; ++g;
 // Print the argument, static and global variables
 printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}
int main()
{
 int i;
 pthread_t tid;
 // Let us create three threads
 for (i = 0; i < 3; i++)
 pthread_create(&tid, NULL, myThreadFun, (void *)&tid);
pthread_join(tid,NULL);
 pthread_exit(NULL);
 return 0;
}
```

abhinav@abhinav: ~/lab05032021

```
abhinav@abhinav:~/lab05032021$ gcc p2.c -lpthread
abhinav@abhinav:~/lab05032021$ ./a.out
Thread ID: -275020032, Static: 2, Global: 2
Thread ID: -275020032, Static: 4, Global: 4
Thread ID: -275020032, Static: 6, Global: 6
abhinav@abhinav:~/lab05032021$
```

3. 2 Threads :

```c
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>
```

```
 void *SampleThread1(void *vargp)
{
    int i = 0;
    printf("SampleThread(1) is running ... \n");
    for(i = 0; i < 10; i++) {
        sleep(1);
        printf("timer running inside SampleThread(1) = %d\n", i);
    }
    printf("SampleThread(1) is exiting ... \n");
    return NULL;
};

void *SampleThread2(void *vargp)
{
    int i = 0;
    printf("SampleThread(2) is running ... \n");
    for(i = 0; i < 15; i++) {

        sleep(1);
        printf("timer running inside SampleThread(2) = %d\n", i);

    }
    printf("SampleThread(2) is exiting ... \n");
    return NULL;

};
```

```
int main()
{
    int i = 0;
    pthread_t tid1, tid2;
    pthread_create(&tid1, NULL, SampleThread1, NULL);
    pthread_create(&tid2, NULL, SampleThread2, NULL);

    for(i = 0; i < 7; i++) {

        sleep(2);
        printf("timer running outside thread = %d\n", i);
    }

    printf("timer outside Thread is ended ..\n");

    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    exit(0);
}
```

Output :

p3.c
~/lab05032021

```c
void *SampleThread1(void *vargp)
{
    int i = 0;
    printf("SampleThread(1) is running ... \n");
    for(i = 0; i < 10; i++) {
        sleep(1);
        printf("timer running inside SampleThread(1) = %d\n", i);
    }
    printf("SampleThread(1) is exiting ... \n");
    return NULL;
};

void *SampleThread2(void *vargp)
{
    int i = 0;
    printf("SampleThread(2) is running ... \n");
    for(i = 0; i < 15; i++) {

        sleep(1);
        printf("timer running inside SampleThread(2) = %d\n", i);

    }
    printf("SampleThread(2) is exiting ... \n");
    return NULL;

};

int main()
{
    int i = 0;
    pthread_t tid1, tid2;
    pthread_create(&tid1, NULL, SampleThread1, NULL);
    pthread_create(&tid2, NULL, SampleThread2, NULL);

    for(i = 0; i < 7; i++) {

        sleep(2);
        printf("timer running outside thread = %d\n", i);
    }

    printf("timer outside Thread is ended ..\n");

    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);

    exit(0);

}
```

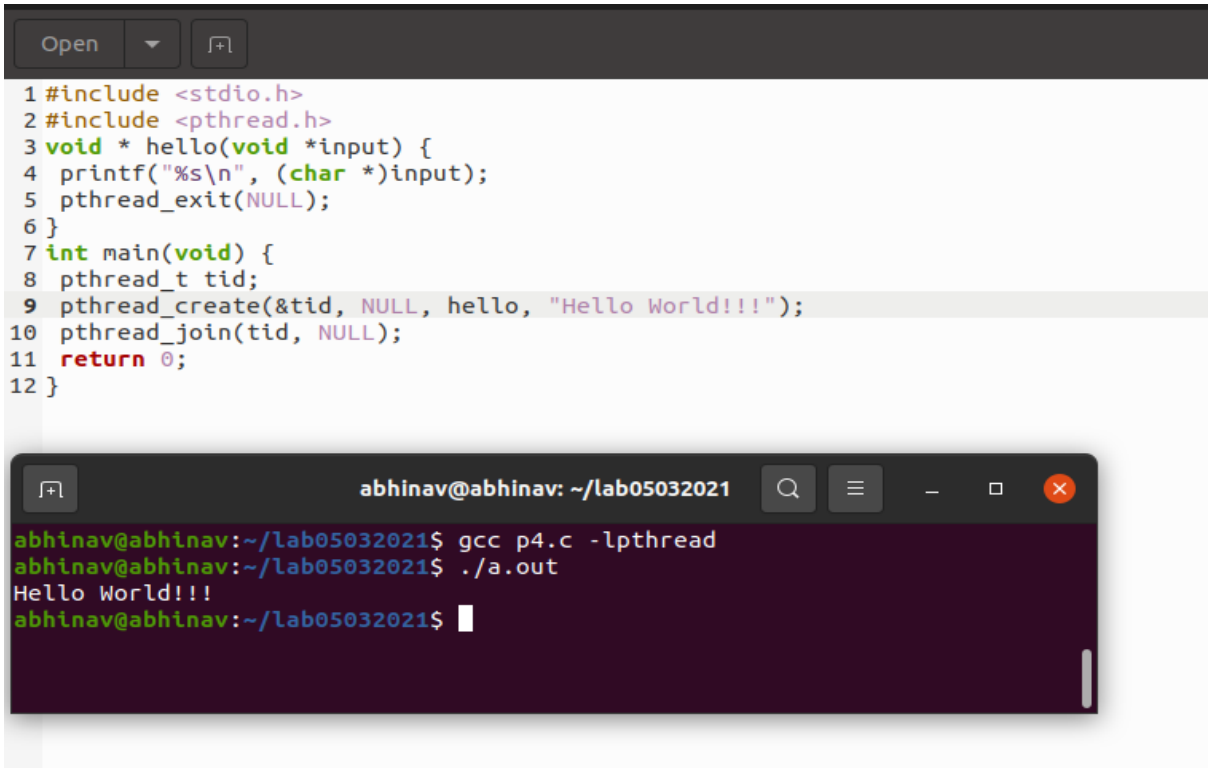
abhinav@abhinav: ~/lab05032021

```
abhinav@abhinav:~/lab05032021$ gcc p3.c -lpthread
abhinav@abhinav:~/lab05032021$ ./a.out
SampleThread(2) is running ...
SampleThread(1) is running ...
timer running inside SampleThread(2) = 0
timer running inside SampleThread(1) = 0
timer running outside thread = 0
timer running inside SampleThread(2) = 1
timer running inside SampleThread(1) = 1
timer running inside SampleThread(2) = 2
timer running inside SampleThread(1) = 2
timer running outside thread = 1
timer running inside SampleThread(2) = 3
timer running inside SampleThread(1) = 3
timer running inside SampleThread(2) = 4
timer running inside SampleThread(1) = 4
timer running outside thread = 2
timer running inside SampleThread(2) = 5
timer running inside SampleThread(1) = 5
timer running inside SampleThread(2) = 6
timer running inside SampleThread(1) = 6
timer running outside thread = 3
timer running inside SampleThread(2) = 7
timer running inside SampleThread(1) = 7
timer running inside SampleThread(2) = 8
timer running inside SampleThread(1) = 8
timer running outside thread = 4
timer running inside SampleThread(2) = 9
timer running inside SampleThread(1) = 9
SampleThread(1) is exiting ...
timer running inside SampleThread(2) = 10
timer running outside thread = 5
timer running inside SampleThread(2) = 11
timer running inside SampleThread(2) = 12
timer running outside thread = 6
timer outside Thread is ended ..
timer running inside SampleThread(2) = 13
timer running inside SampleThread(2) = 14
SampleThread(2) is exiting ...
abhinav@abhinav:~/lab05032021$
```

4.Passing 1 Argument :

```
#include <stdio.h>
#include <pthread.h>
void * hello(void *input) {
 printf("%s\n", (char *)input);
 pthread_exit(NULL);
}
int main(void) {
 pthread_t tid;
 pthread_create(&tid, NULL, hello, "Hello World!!!");
 pthread_join(tid, NULL);
 return 0;
}
```

Output :



5.Structure :

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
struct args {
 char* name;
 int age;
};
void *hello(void *input) {
 printf("name: %s\n", ((struct args*)input)->name);
 printf("age: %d\n", ((struct args*)input)->age);
}
int main() {
 struct args *Allen = (struct args *)malloc(sizeof(struct args));
 char allen[] = "Allen";
 Allen->name = allen;
 Allen->age = 20;
 pthread_t tid;
 pthread_create(&tid, NULL, hello, (void *)Allen);
 pthread_join(tid, NULL);
return 0;
}
```

Output :

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
struct args {
 char* name;
 int age;
};
void *hello(void *input) {
 printf("name: %s\n", ((struct args*)input)->name);
 printf("age: %d\n", ((struct args*)input)->age);
}
int main() {
 struct args *Allen = (struct args *)malloc(sizeof(struct args));
 char allen[] = "Allen";
 Allen->name = allen;
 Allen->age = 20;
 pthread_t tid;
 pthread_create(&tid, NULL, hello, (void *)Allen);
 pthread_join(tid, NULL);
return 0;
}
```

```
abhinav@abhinav: ~/lab05032021
abhinav@abhinav:~/lab05032021$ gcc p5.c -lpthread
abhinav@abhinav:~/lab05032021$ ./a.out
name: Allen
age: 20
abhinav@abhinav:~/lab05032021$
```

6.Arrays :

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#define NUM_THREADS 8
char *messages[NUM_THREADS];
void *PrintHello(void *threadid)
{
 int *id_ptr, taskid;
 sleep(1);
 id_ptr = (int *) threadid;
 taskid = *id_ptr;
 printf("Thread %d: %s\n", taskid, messages[taskid]);
 pthread_exit(NULL);
}
int main(int argc, char *argv[])
{
 pthread_t threads[NUM_THREADS];
 int *taskids[NUM_THREADS];
 int rc, t;
 messages[0] = "English: Hello World!";
 messages[1] = "French: Bonjour, le monde!";
 messages[2] = "Spanish: Hola al mundo";
 messages[3] = "Klingon: Nuq neH!";
 messages[4] = "German: Guten Tag, Welt!";
 messages[5] = "Russian: Zdravstvytye, mir!";
```

```
messages[6] = "Japan: Sekai e konnichiwa!";
messages[7] = "Latin: Orbis, te saluto!";
for(t=0;t<NUM_THREADS;t++)
{ taskids[t] = (int *) malloc(sizeof(int));
*taskids[t] = t;
printf("Creating thread %d\n", t);
rc = pthread_create(&threads[t], NULL, PrintHello, (void *) taskids[t] );
if (rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
pthread_exit(NULL);
}
```

Output :