**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);

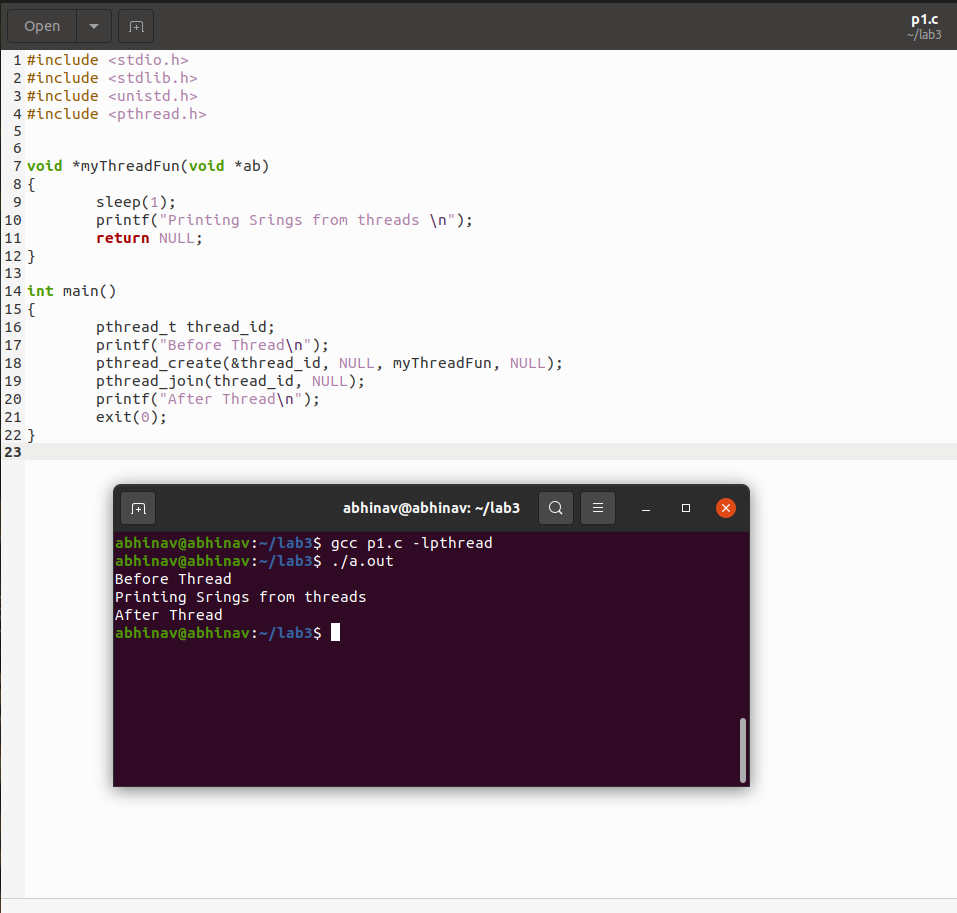
printf("After Thread\n");

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 :**

****

**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);

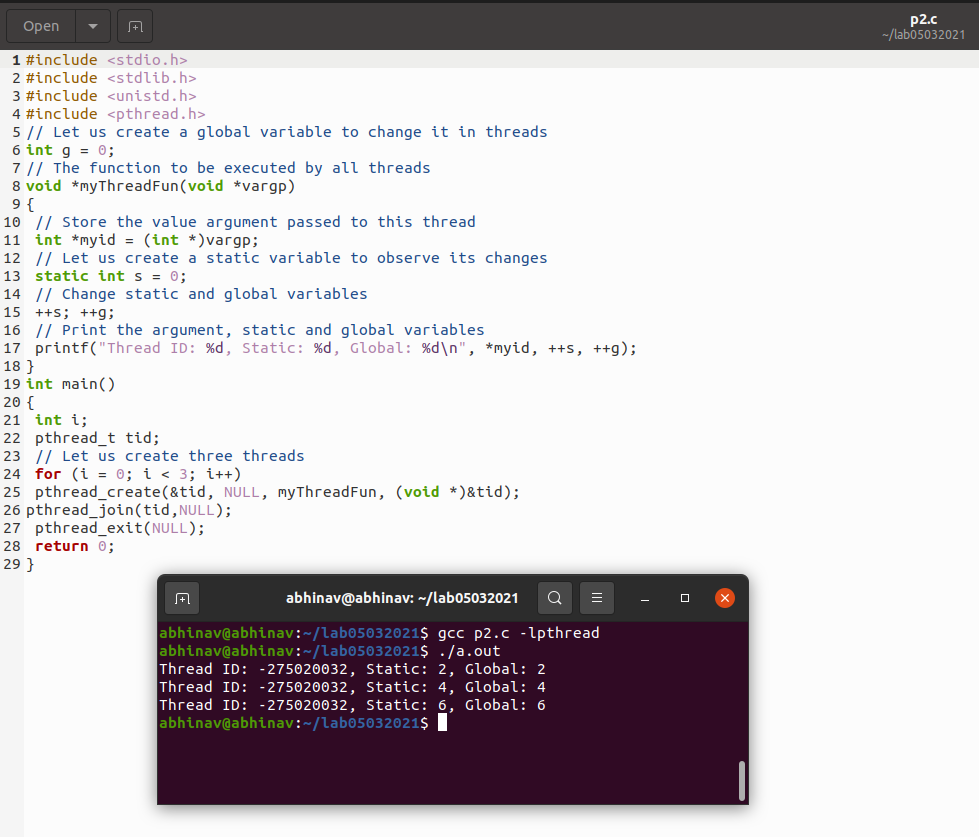
pthread\_join(tid,NULL);

pthread\_exit(NULL);

return 0;

}

**Output :**

****

**3. 2 Threads :**

#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");

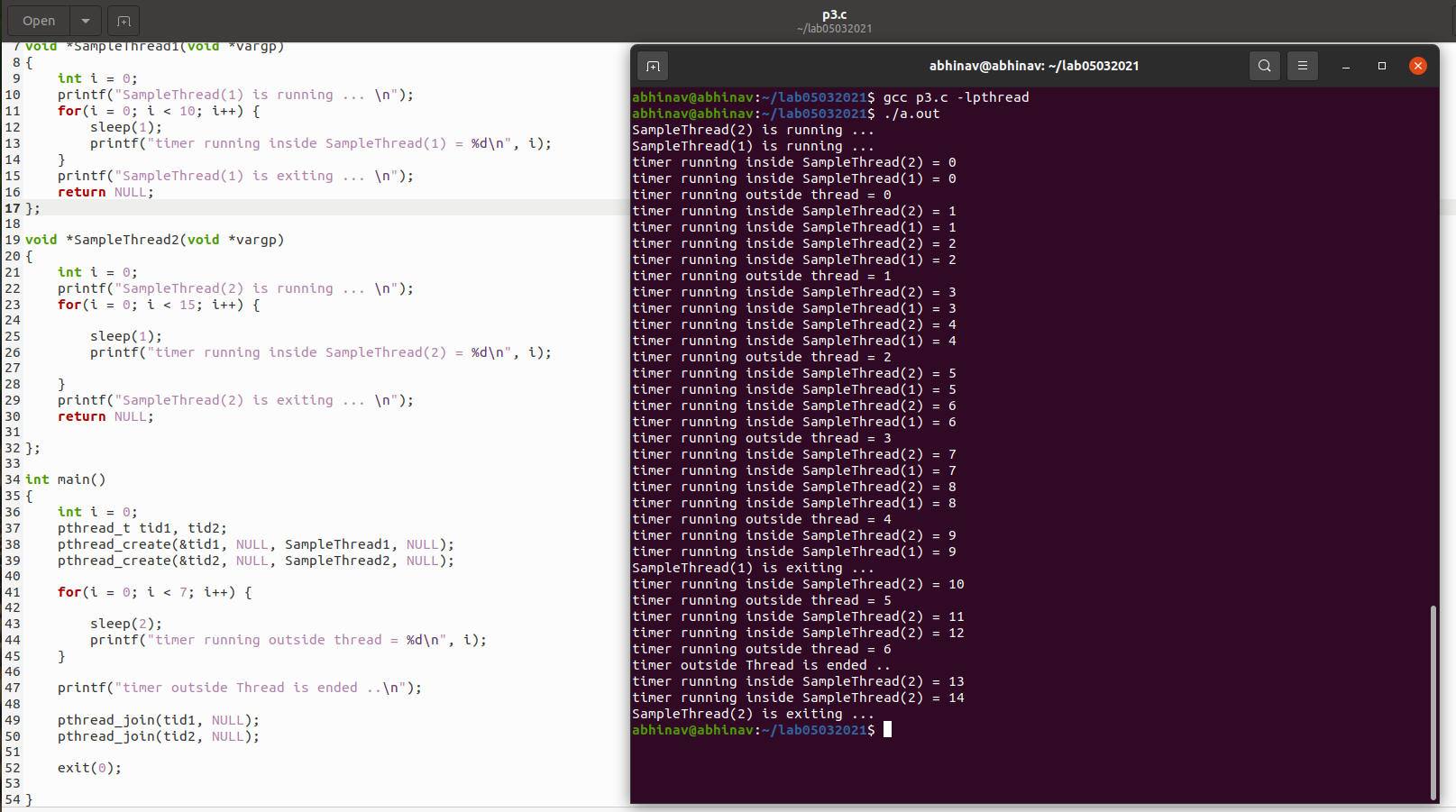
pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

exit(0);

}

**Output :**



**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;

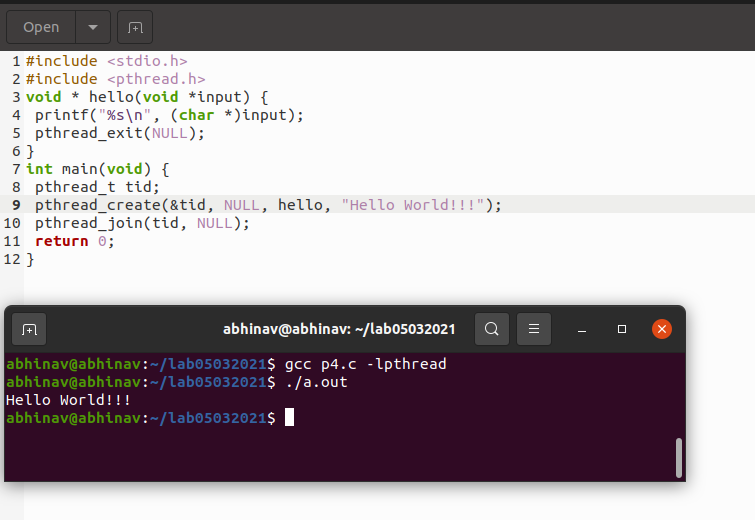
pthread\_create(&tid, NULL, hello, "Hello World!!!");

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;

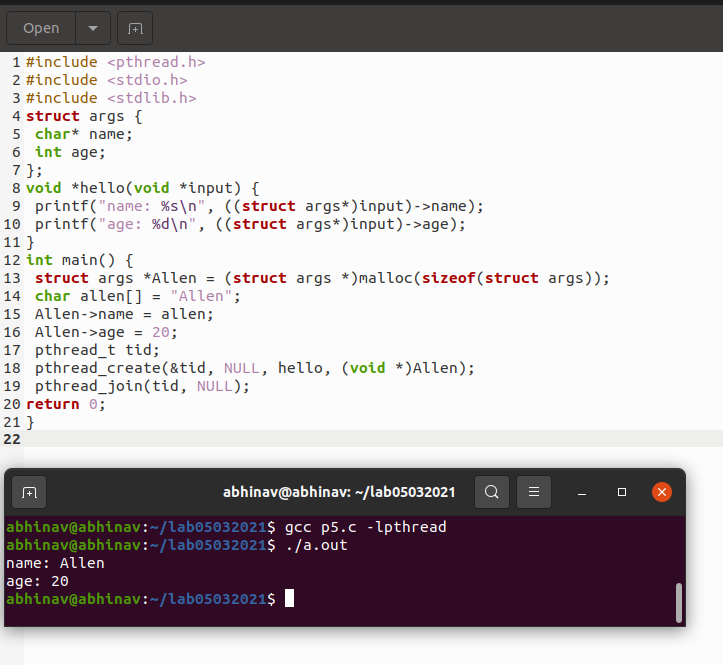
pthread\_create(&tid, NULL, hello, (void \*)Allen);

pthread\_join(tid, NULL);

return 0;

}

**Output :**



**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);

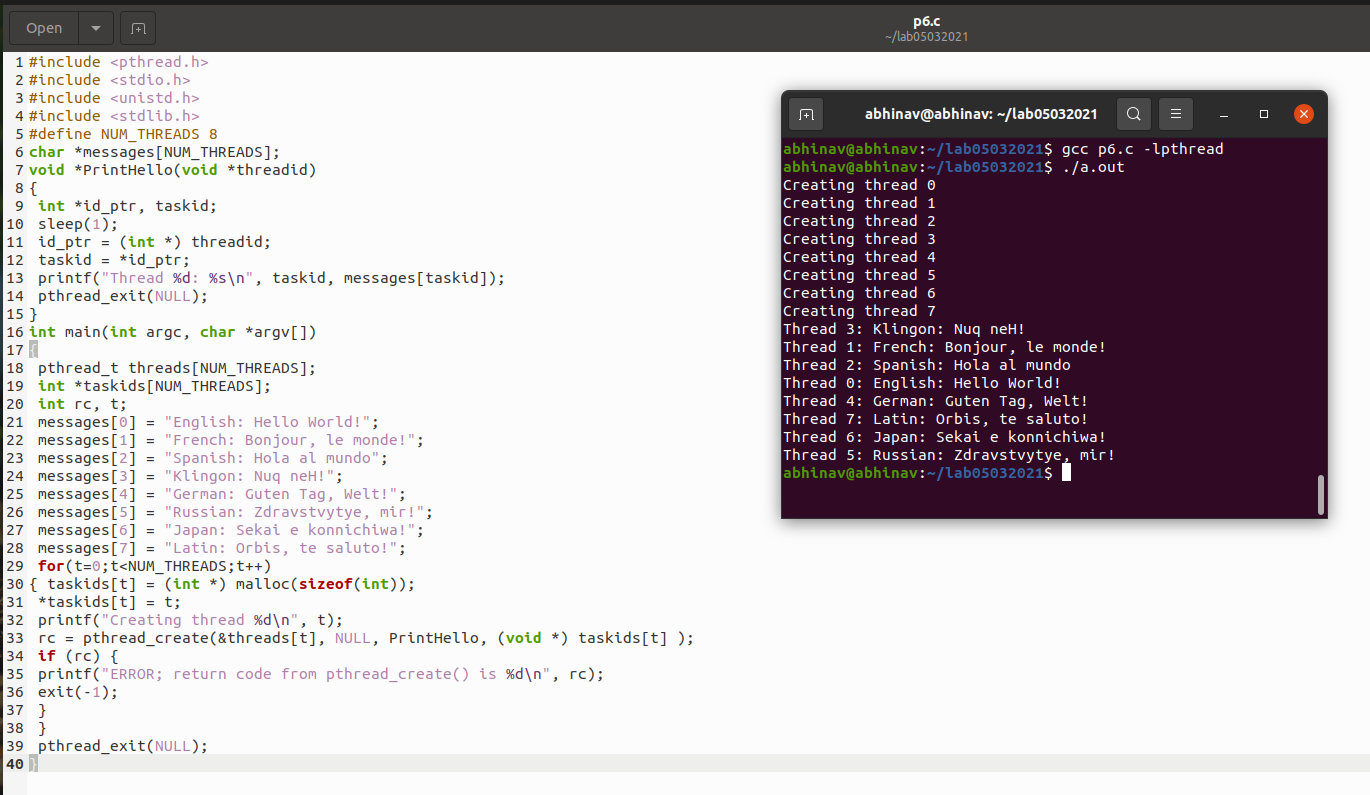
}

}

pthread\_exit(NULL);

}

**Output :**

****