

Amrita Vishwa Vidyapeetham
Amrita School of Engineering, Coimbatore
Department of Computer Science and Engineering
Topic: Threads Practice

19CSE213 Operating Systems Laboratory – Threads Practice

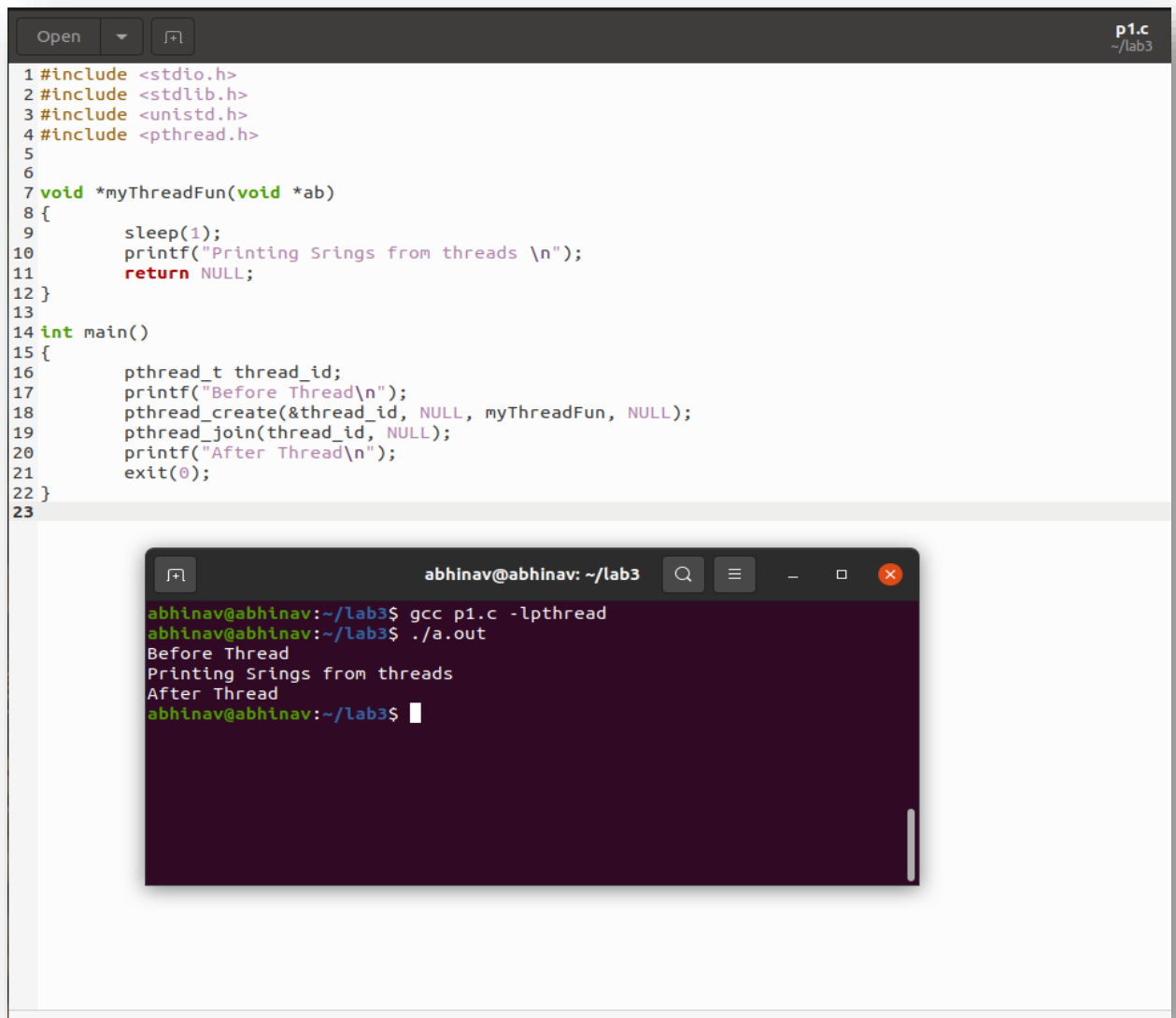
1. Simple Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void *myThreadFun(void *ab)
{
    sleep(1);
    printf("Printing Strings 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);
}
```

- **Explanation :**

- ✓ 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 :



The image shows a code editor window with a file named p1.c. The code defines a function myThreadFun that sleeps for 1 second and prints "Printing Srings from threads \n". The main function declares a pthread_t variable thread_id, prints "Before Thread\n", creates a new thread with pthread_create, joins it with pthread_join, prints "After Thread\n", and exits. Below the code editor is a terminal window showing the compilation and execution of the program. The terminal output shows the sequence of messages printed by the program.

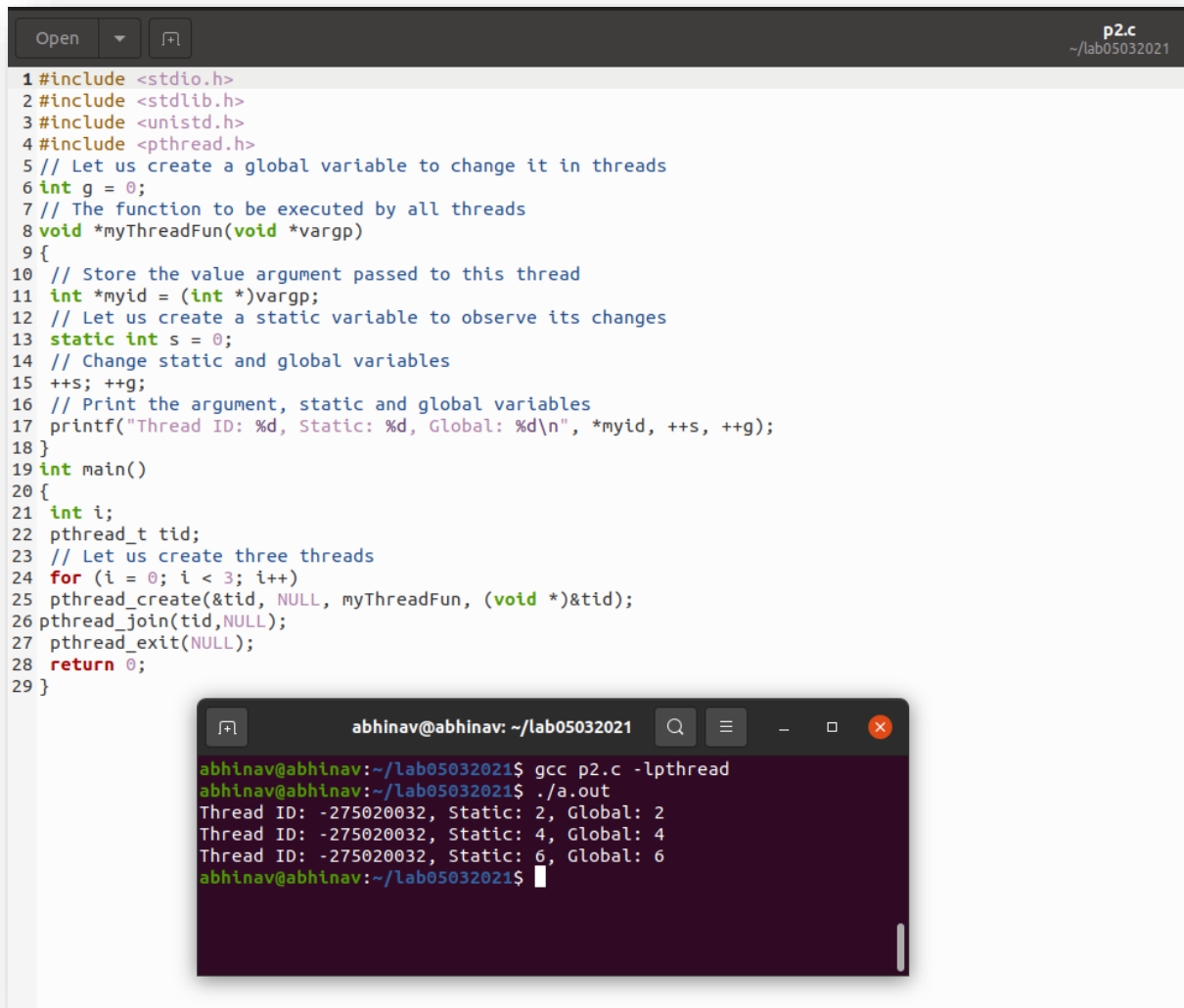
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <pthread.h>
5
6
7 void *myThreadFun(void *ab)
8 {
9     sleep(1);
10    printf("Printing Srings from threads \n");
11    return NULL;
12 }
13
14 int main()
15 {
16    pthread_t thread_id;
17    printf("Before Thread\n");
18    pthread_create(&thread_id, NULL, myThreadFun, NULL);
19    pthread_join(thread_id, NULL);
20    printf("After Thread\n");
21    exit(0);
22 }
23
```

```
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 :



The image shows a code editor window titled 'p2.c' with the file path '~/lab05032021'. The code is a C program that demonstrates thread creation and execution. It includes headers for stdio, stdlib, unistd, and pthread. A global variable 'g' is initialized to 0. A function 'myThreadFun' is defined, which takes a void pointer 'vargp' and increments a static variable 's' and the global variable 'g'. The main function creates three threads using 'pthread_create', each passing its own ID as an argument. After creating the threads, it calls 'pthread_join' to wait for them to finish. Finally, it prints the thread ID, static variable 's', and global variable 'g' for each thread. The terminal window below the code editor shows the command 'gcc p2.c -lpthread' and the execution of './a.out'. The output shows three threads, each with a unique ID, and the static variable 's' and global variable 'g' are incremented by 2 for each thread.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <pthread.h>
5 // Let us create a global variable to change it in threads
6 int g = 0;
7 // The function to be executed by all threads
8 void *myThreadFun(void *vargp)
9 {
10 // Store the value argument passed to this thread
11 int *myid = (int *)vargp;
12 // Let us create a static variable to observe its changes
13 static int s = 0;
14 // Change static and global variables
15 ++s; ++g;
16 // Print the argument, static and global variables
17 printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
18 }
19 int main()
20 {
21 int i;
22 pthread_t tid;
23 // Let us create three threads
24 for (i = 0; i < 3; i++)
25 pthread_create(&tid, NULL, myThreadFun, (void *)&tid);
26 pthread_join(tid, NULL);
27 pthread_exit(NULL);
28 return 0;
29 }
```

```
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 :

```
#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 :

```
Open  p3.c
~/lab05032021

1 void *SampleThread1(void *vargp)
2 {
3     int i = 0;
4     printf("SampleThread(1) is running ... \n");
5     for(i = 0; i < 10; i++) {
6         sleep(1);
7         printf("timer running inside SampleThread(1) = %d\n", i);
8     }
9     printf("SampleThread(1) is exiting ... \n");
10    return NULL;
11};
12
13 void *SampleThread2(void *vargp)
14 {
15     int i = 0;
16     printf("SampleThread(2) is running ... \n");
17     for(i = 0; i < 15; i++) {
18         sleep(1);
19         printf("timer running inside SampleThread(2) = %d\n", i);
20     }
21     printf("SampleThread(2) is exiting ... \n");
22    return NULL;
23};
24
25 int main()
26 {
27     pthread_t tid1, tid2;
28     pthread_create(&tid1, NULL, SampleThread1, NULL);
29     pthread_create(&tid2, NULL, SampleThread2, NULL);
30
31     for(i = 0; i < 7; i++) {
32         sleep(2);
33         printf("timer running outside thread = %d\n", i);
34     }
35     printf("timer outside Thread is ended ..\n");
36
37     pthread_join(tid1, NULL);
38     pthread_join(tid2, NULL);
39
40     exit(0);
41 }
42
43
44
45
46
47
48
49
50
51
52
53
54
```

```
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 :

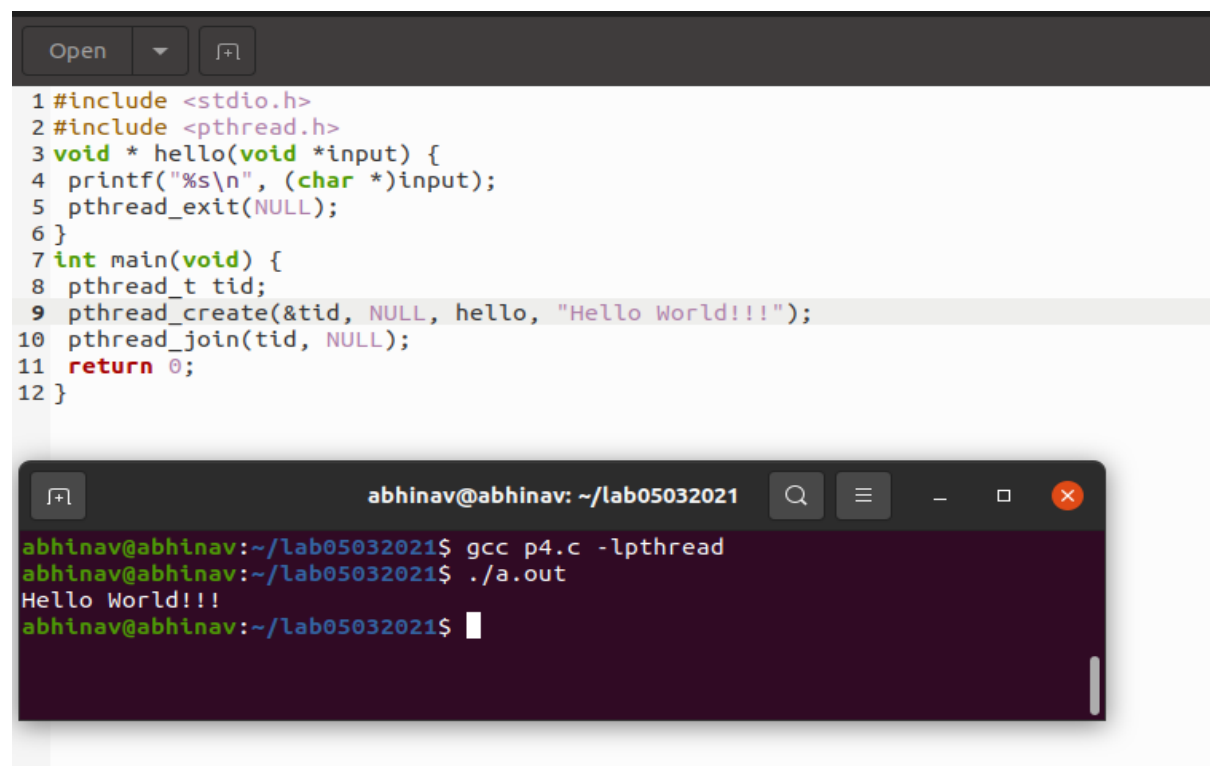
```
#include <stdio.h>

#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 :



The image shows a code editor window with a dark theme. The code is a C program that uses pthreads to create a thread that prints "Hello World!!!". The code is as follows:

```
1 #include <stdio.h>
2 #include <pthread.h>
3 void * hello(void *input) {
4     printf("%s\n", (char *)input);
5     pthread_exit(NULL);
6 }
7 int main(void) {
8     pthread_t tid;
9     pthread_create(&tid, NULL, hello, "Hello World!!!");
10    pthread_join(tid, NULL);
11    return 0;
12 }
```

Below the code editor is a terminal window. The terminal shows the command to compile the program using gcc and the output of the program.

```
abhinav@abhinav: ~/lab05032021
abhinav@abhinav:~/lab05032021$ gcc p4.c -lpthread
abhinav@abhinav:~/lab05032021$ ./a.out
Hello World!!!
abhinav@abhinav:~/lab05032021$
```


5.Structure :

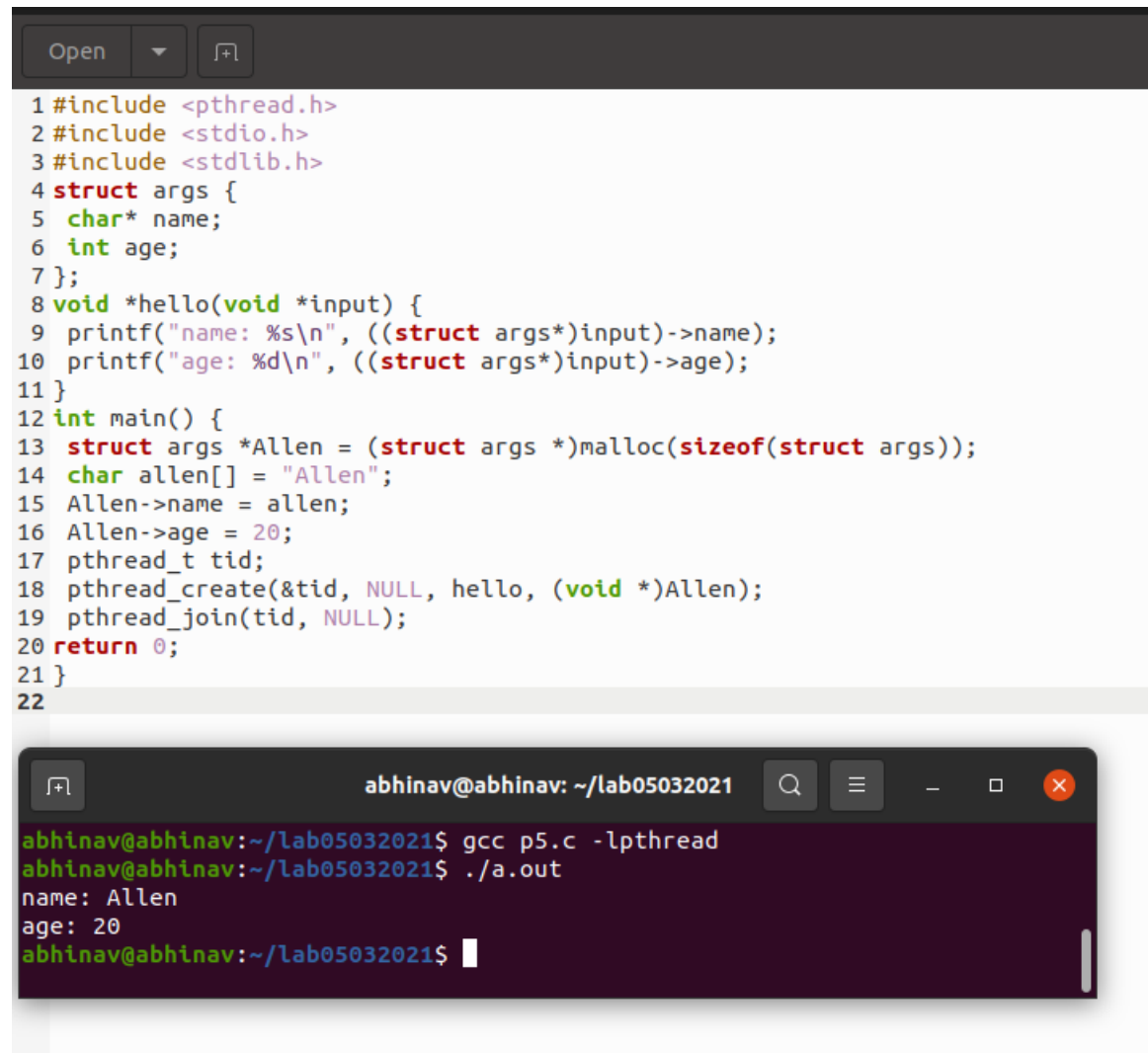
```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

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 :



The image shows a code editor window with a C program and a terminal window below it. The C program defines a struct 'args' with 'name' and 'age' fields, a 'hello' function that prints these fields, and a 'main' function that creates a thread 'Allen' and prints its details. The terminal shows the compilation and execution of the program, resulting in the output 'name: Allen' and 'age: 20'.

```
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct args {
5     char* name;
6     int age;
7 };
8 void *hello(void *input) {
9     printf("name: %s\n", ((struct args*)input)->name);
10    printf("age: %d\n", ((struct args*)input)->age);
11 }
12 int main() {
13     struct args *Allen = (struct args *)malloc(sizeof(struct args));
14     char allen[] = "Allen";
15     Allen->name = allen;
16     Allen->age = 20;
17     pthread_t tid;
18     pthread_create(&tid, NULL, hello, (void *)Allen);
19     pthread_join(tid, NULL);
20     return 0;
21 }
22
```

```
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: Zdravstvyye, mir!";
```

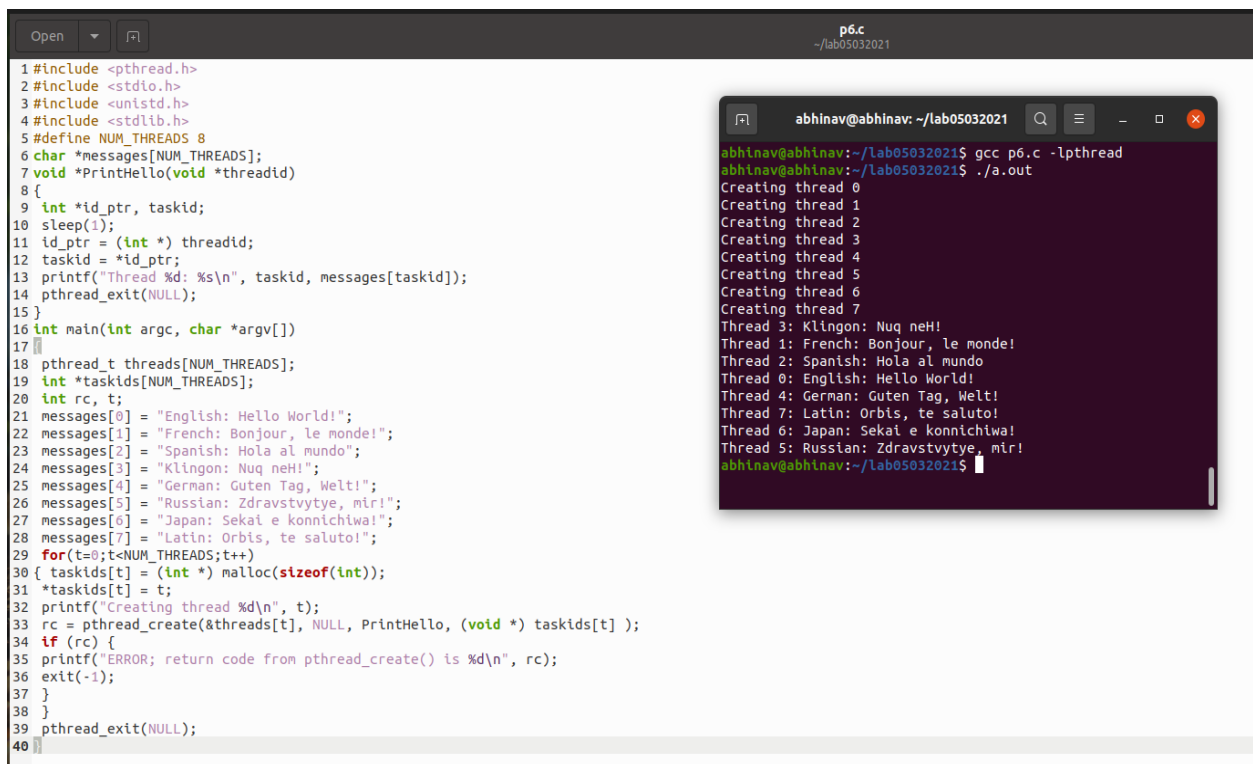
```

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 :



The screenshot shows a code editor on the left and a terminal window on the right. The code editor displays the source code for a C program named `p6.c`, which uses `pthread` to create multiple threads. The code includes headers for `pthread.h`, `stdio.h`, `unistd.h`, and `stdlib.h`. It defines `NUM_THREADS` as 8 and creates an array of messages in various languages. The `main` function creates an array of threads and calls `pthread_create` for each thread, passing a task ID. The `PrintHello` function (defined in the code but not shown in the terminal output) prints the message for each thread ID. The terminal window shows the output of the program, which lists the creation of 8 threads and their respective messages.

```

1 #include <pthread.h>
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5 #define NUM_THREADS 8
6 char *messages[NUM_THREADS];
7 void *PrintHello(void *threadid)
8 {
9     int *id_ptr, taskid;
10    sleep(1);
11    id_ptr = (int *) threadid;
12    taskid = *id_ptr;
13    printf("Thread %d: %s\n", taskid, messages[taskid]);
14    pthread_exit(NULL);
15 }
16 int main(int argc, char *argv[])
17 {
18     pthread_t threads[NUM_THREADS];
19     int *taskids[NUM_THREADS];
20     int rc, t;
21     messages[0] = "English: Hello World!";
22     messages[1] = "French: Bonjour, le monde!";
23     messages[2] = "Spanish: Hola al mundo";
24     messages[3] = "Klingon: Nuq neH!";
25     messages[4] = "German: Guten Tag, Welt!";
26     messages[5] = "Russian: Zdravstvuyte, mir!";
27     messages[6] = "Japan: Sekai e konnichiwa!";
28     messages[7] = "Latin: Orbis, te saluto!";
29     for(t=0;t<NUM_THREADS;t++)
30     { taskids[t] = (int *) malloc(sizeof(int));
31       *taskids[t] = t;
32       printf("Creating thread %d\n", t);
33       rc = pthread_create(&threads[t], NULL, PrintHello, (void *) taskids[t] );
34       if (rc) {
35         printf("ERROR; return code from pthread_create() is %d\n", rc);
36         exit(-1);
37       }
38     }
39     pthread_exit(NULL);
40 }

```

```

abhinav@abhinav: ~/lab05032021$ gcc p6.c -lpthread
abhinav@abhinav: ~/lab05032021$ ./a.out
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Creating thread 4
Creating thread 5
Creating thread 6
Creating thread 7
Thread 3: Klingon: Nuq neH!
Thread 1: French: Bonjour, le monde!
Thread 2: Spanish: Hola al mundo
Thread 0: English: Hello World!
Thread 4: German: Guten Tag, Welt!
Thread 7: Latin: Orbis, te saluto!
Thread 6: Japan: Sekai e konnichiwa!
Thread 5: Russian: Zdravstvuyte, mir!
abhinav@abhinav: ~/lab05032021$

```