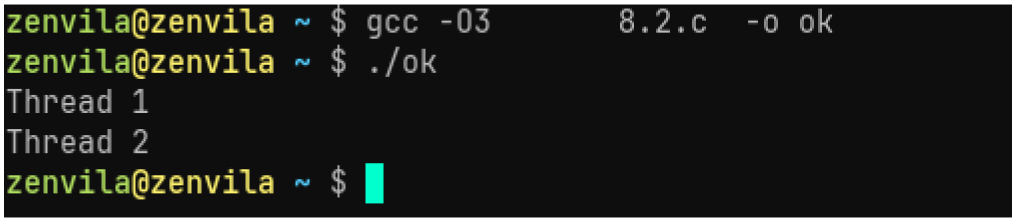
Name : haris Roll # 23P-0573



```
zenvila@zenvila ~ $ gcc -O3       8.2.c  -o ok
zenvila@zenvila ~ $ ./ok
Thread 1
Thread 2
zenvila@zenvila ~ $
```

we have to use library pthread.h

Q1 What is the pthread_create() and the pthread_join() calls doing?

Actually, pthread_join() is the opposite of pthread_create(). Pthread_create() will split our single threaded process into two-threaded process. Pthread_join() will join back the two-threaded process into a single threaded process.
The simple pthread create for creating the thread and pthred join is for the to make joinable threads btw threds is also created as joinable tread and make it again join

Q2 In the pthread_create() call, what are the 4 parameters?

Pthread_create(&thread1,NULL,(void*)print_message,message);

Q3 In pthread_create() call, the 4th parameter is used for passing a pointer to argument of a function. What will we need to do if we want to pass multiple arguments to that function?

Yes we , can pass multiple arguments by using the strucure .

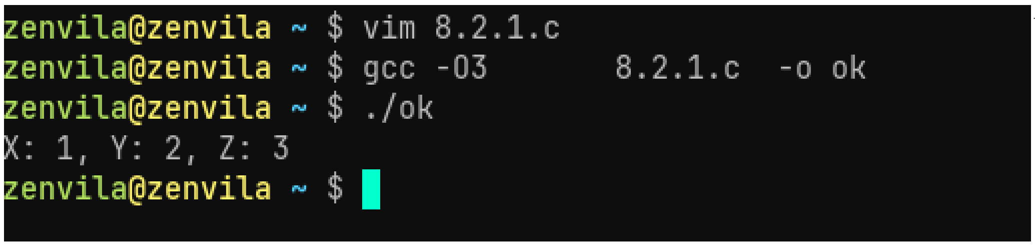
Q4 In the pthread_join() call, what are the 2 paramters?
In pthread_join(thread1,NULL)

Q5 What is the purpose of the return_value1 and return_value2 variables? Find out the contents of both these variables using a printf function. What do they contain?

The purpose of the   return value1 and return value 2   It means thread created successfully according to the given code thr thread 1 and thread 2 return   1 and 2

8.2.1 Passing Multiple Arguments to Thread

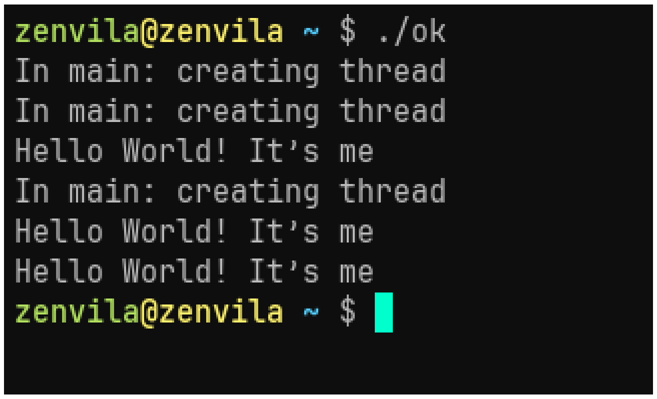
```
#include <stdio.h>
#include <pthread.h>

struct thread_data
{
    int x;
    int y;
    int z;
};

void *print(void *threadArg)
{
    struct thread_data *my_data = (struct thread_data *)threadArg;
    printf("X: %d, Y: %d, Z: %d\n", my_data->x, my_data->y, my_data->z);
    return NULL;
}

int main(void)
{
    pthread_t tid;
    struct thread_data obj;

    obj.x = 1;
    obj.y = 2;
    obj.z = obj.x + obj.y;

    pthread_create(&tid, NULL, print, &obj);
    pthread_join(tid, NULL);

    return 0;
}
```



```
zenvila@zenvila ~ $ vim 8.2.1.c
zenvila@zenvila ~ $ gcc -O3       8.2.1.c  -o ok
zenvila@zenvila ~ $ ./ok
X: 1, Y: 2, Z: 3
zenvila@zenvila ~ $
```

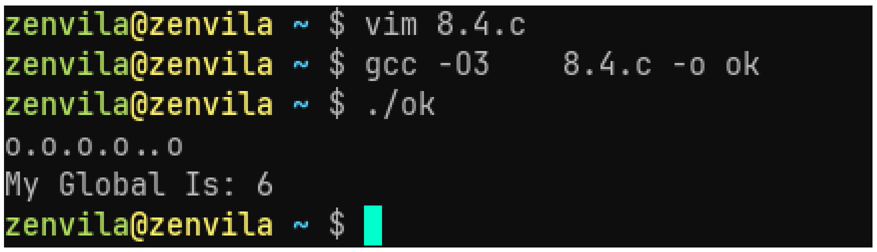
8.3 Thread Termination

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

void *PrintHello()
{
   printf("Hello World! It’s me\n");
   pthread_exit(NULL);
}

int main()
{
   pthread_t threads[3];
   int rc;

   for (int t = 0; t < 3; t++)
   {
      printf("In main: creating thread\n");
      rc = pthread_create(&threads[t], NULL, PrintHello, (void *)&t);
      if (rc)
      {
         printf("ERROR; return code from pthread_create() is %d\n", rc);
         exit(-1);
      }
   }

   pthread_exit(NULL);
}
```



```
zenvila@zenvila ~ $ ./ok
In main: creating thread
In main: creating thread
Hello World! It’s me
In main: creating thread
Hello World! It’s me
Hello World! It’s me
zenvila@zenvila ~ $
```

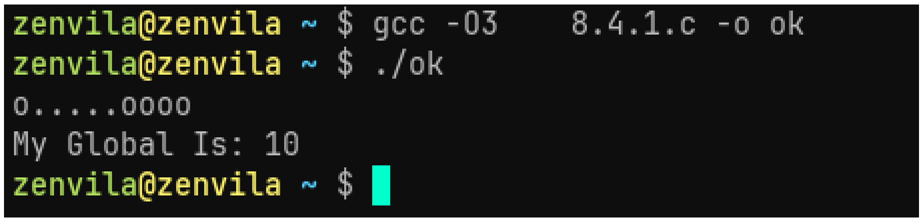
```
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
int myGlobal = 0;
void *threadFunction()
{
int i, j;
for (i = 0; i<5; i++)
{
j = myGlobal;
j = j+1;
printf(".");
fflush(stdout);
sleep(1);
myGlobal = j;
}
}
int main()
{
pthread_t myThread;
int i;
pthread_create(&myThread, NULL, threadFunction, NULL);
for (i = 0; i < 5; i++)
{
myGlobal = myGlobal + 1;
printf("o");
fflush(stdout);
sleep(1);
}
pthread_join(myThread, NULL);
printf("\nMy Global Is: %d\n", myGlobal);
exit(0);
}
```



```
zenvila@zenvila ~ $ vim 8.4.c
zenvila@zenvila ~ $ gcc -O3    8.4.c -o ok
zenvila@zenvila ~ $ ./ok
o.o.o.o..o
My Global Is: 6
zenvila@zenvila ~ $
```

8.4.1 Synchronization through Mutex

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

int myGlobal = 0;
pthread_mutex_t myMutex;

void *threadFunction()
{
    int i, j;
    for (i = 0; i < 5; i++)
    {
        pthread_mutex_lock(&myMutex);
        j = myGlobal;
        j = j + 1;
        printf(".");
        fflush(stdout);
        sleep(1);
        myGlobal = j;
        pthread_mutex_unlock(&myMutex);
    }
    return NULL;
}

int main()
{
    pthread_t myThread;
    int i;

    pthread_create(&myThread, NULL, threadFunction, NULL);

    for (i = 0; i < 5; i++)
    {
        pthread_mutex_lock(&myMutex);
        myGlobal = myGlobal + 1;
        pthread_mutex_unlock(&myMutex);
        printf("o");
        fflush(stdout);
        sleep(1);
    }

    pthread_join(myThread, NULL);
    printf("\nMy Global Is: %d\n", myGlobal);
    return 0;
}
```

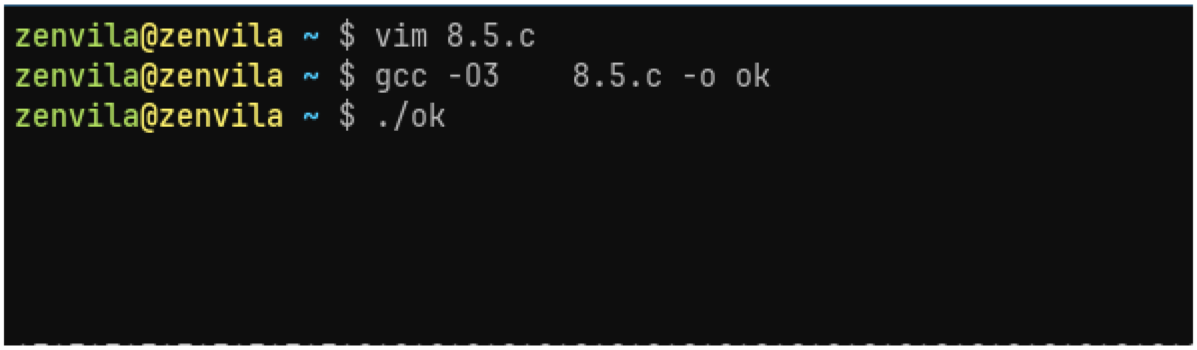


```
zenvila@zenvila ~ $ gcc -O3    8.4.1.c -o ok
zenvila@zenvila ~ $ ./ok
o.....oooo
My Global Is: 10
zenvila@zenvila ~ $
```

8.5 Exercise

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

struct fib_arg {
   int n;
   long result;
};

void *fib_thread(void *arg_ptr)
{
   struct fib_arg *arg = (struct fib_arg *)arg_ptr;

   if (arg->n <= 0)
      arg->result = 0;
   else if (arg->n == 1)
      arg->result = 1;
   else {
      pthread_t t1, t2;
      struct fib_arg a1 = { .n = arg->n - 1 };
      struct fib_arg a2 = { .n = arg->n - 2 };

      pthread_create(&t1, NULL, fib_thread, &a1);
      pthread_create(&t2, NULL, fib_thread, &a2);

      pthread_join(t1, NULL);
      pthread_join(t2, NULL);

      arg->result = a1.result + a2.result;
   }
   return NULL;
}

int main(void)
{
   int find = 40;
   pthread_t top;
```

```
    struct fib_arg top_arg = { .n = find };

    pthread_create(&top, NULL, fib_thread, &top_arg);
    pthread_join(top, NULL);

    printf("Element No. %d in series is: %ld\n", find, top_arg.result);
    return 0;
}
```



```
zenvila@zenvila ~ $ vim 8.5.c
zenvila@zenvila ~ $ gcc -O3    8.5.c -o ok
zenvila@zenvila ~ $ ./ok
```