

1. HelloPT.c

```
rigovil@rodrigo:~/Escritorio/UCR/2020 - II S  
Hello from thread 1 of 4  
Otro metodo  
Hello from thread 0 of 4  
Otro metodo  
Hello from thread 3 of 4  
Otro metodo  
Hello from the main thread  
Hello from thread 2 of 4  
Otro metodo  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II S  
Hello from thread 0 of 8  
Otro metodo  
Hello from thread 1 of 8  
Otro metodo  
Hello from thread 2 of 8  
Otro metodo  
Hello from thread 3 of 8  
Otro metodo  
Hello from thread 4 of 8  
Otro metodo  
Hello from thread 5 of 8  
Otro metodo  
Hello from thread 6 of 8  
Otro metodo  
Hello from the main thread  
Hello from thread 7 of 8
```

Programa corriendo con 4 y 8 hilos.

2.1. sumaUno.c

```
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
4$ gcc sumaUno.c  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
4$ ./a.out  
Serial version:      Valor acumulado es 100000 con 100 procesos  
econds  
Fork, Race Cond.:   Valor acumulado es 97652 con 100 procesos  
econds  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea
```

El programa no realiza el cálculo correctamente pues, a pesar de utilizar memoria compartida, existe race condition pues no se limita el acceso a la variable que almacena el resultado a un solo hilo a la vez.

2.2. sumaUnoNoRaceCondition.c

```
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
4$ gcc sumaUnoNoRaceCondition.c  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
4$ ./a.out  
Serial version:      Valor acumulado es 100000 con 100 procesos  
econds  
Fork, Race Cond.:   Valor acumulado es 100000 con 100 procesos  
econds  
Fork, No Race Cond.: Valor acumulado es 100000 con 100 procesos  
econds  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea
```

En este caso el programa sí realiza el cálculo de forma correcta pues utiliza semáforos para que cada hilo sume su parte a la variable global que almacena el resultado.

2.3. sumaUnoThread.c

```
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tar
4$ g++ sumaUnoThread.cc Thread.cc Mutex.cc ThreadBase.cc -lpth
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tar
4$ ./a.out
Serial version:      Valor acumulado es 100000 con 100 hilos
nds
PThr, Race Cond.:   Valor acumulado es 100000 con 100 hilos
nds
PThr, NO Race Cond.: Valor acumulado es 100000 con 100 hilos
onds
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tar
```

Este programa logra realizar el cálculo de forma correcta utilizando los mutex de la librería `pthread`. Primero, cada hilo calcula su parte y luego se suma a la variable global, pero ese acceso a la variable se controla mediante el mutex. Los hilos son creados y utilizados a partir de una clase que se deriva de `ThreadBase`.

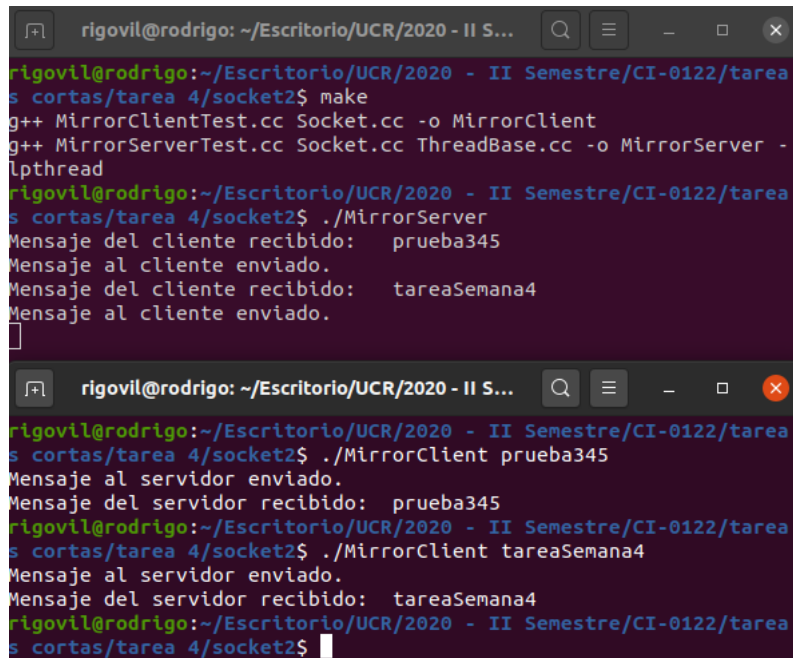
3. MirrorServerTest.cc con pthread

```
rigovil@rodrigo: ~/Escritorio/UCR/2020 - II Semestre/C...
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/t...
/tarea 4/socket$ make
g++ MirrorClientTest.cc Socket.cc -o MirrorClient
g++ MirrorServerTest.cc Socket.cc -o MirrorServer -lpthread
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/t...
/tarea 4/socket$ ./MirrorServer
Mensaje del cliente recibido: prueba123
Mensaje al cliente enviado.
Mensaje del cliente recibido: ci-0122
Mensaje al cliente enviado.
█

rigovil@rodrigo: ~/Escritorio/UCR/2020 - II Semestre/CI...
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/t...
/tarea 4/socket$ ./MirrorClient prueba123
Mensaje al servidor enviado.
Mensaje del servidor recibido: prueba123
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/t...
/tarea 4/socket$ ./MirrorClient ci-0122
Mensaje al servidor enviado.
Mensaje del servidor recibido: ci-0122
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/t...
/tarea 4/socket$
```

En este caso, se modificó únicamente el archivo `MirrorServerTest.cc` para que funcionara implementado `pthreads` en lugar de `fork`. Para compilar el programa utilizar el comando `make`, para ejecutarlo, abrir una terminal y correr el server con `./MirrorServer`, luego en otra terminal correr el client con `./MirrorClient MSJ`. El segundo parámetro MSJ es el mensaje que se envía al servidor.

4. MirrorServerTest.cc con la clase ThreadBase



```
rigovil@rodrigo: ~/Escritorio/UCR/2020 - II S...  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
s cortas/tarea 4/socket2$ make  
g++ MirrorClientTest.cc Socket.cc -o MirrorClient  
g++ MirrorServerTest.cc Socket.cc ThreadBase.cc -o MirrorServer -  
lpthread  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
s cortas/tarea 4/socket2$ ./MirrorServer  
Mensaje del cliente recibido: prueba345  
Mensaje al cliente enviado.  
Mensaje del cliente recibido: tareaSemana4  
Mensaje al cliente enviado.  
[  
rigovil@rodrigo: ~/Escritorio/UCR/2020 - II S...  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
s cortas/tarea 4/socket2$ ./MirrorClient prueba345  
Mensaje al servidor enviado.  
Mensaje del servidor recibido: prueba345  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
s cortas/tarea 4/socket2$ ./MirrorClient tareaSemana4  
Mensaje al servidor enviado.  
Mensaje del servidor recibido: tareaSemana4  
rigovil@rodrigo:~/Escritorio/UCR/2020 - II Semestre/CI-0122/tarea  
s cortas/tarea 4/socket2$
```

En este caso, el programa hace lo mismo que el anterior pero utilizando la clase `ThreadBase`. Esta la utiliza una subclase llamada `Hilos`, que sobrescribe los métodos del constructor y de `startRoutine` para poder funcionar de acuerdo con el ejemplo, en este caso el servidor. Para compilar el programa utilizar el comando `make`, para ejecutarlo, abrir una terminal y correr el server con `./MirrorServer`, luego en otra terminal correr el client con `./MirrorClient MSJ`. El segundo parámetro MSJ es el mensaje que se envía al servidor.