

## CC3086 – Laboratorio 5

Nelson García Bravatti 22434

### I. Competencias a desarrollar

Comprende y aplica el concepto de paralelismo. Es capaz de determinar la existencia de problemas sincronización de variables en procesos que desarrollan tareas paralelas y solucionarlos mediante la implementación de join, variables mutex o variables de condición.

### II. Instrucciones

Esta actividad se realizará individualmente, debe incluir capturas de pantallas en donde considere necesario.

**1) 27 pts, 3 c/una. Cree una infografía en la que compare los siguientes aspectos de Variables**

**Mutex y Condicionales:**

- a) Objetivo principal
- b) Uso común / aplicación
- c) Función para inicializar
- d) Función de bloqueo/espera
- e) Función para liberar/
- f) Función para destruir
- g) Necesita otro mecanismo complementario
- h) Debilidades/problemas
- i) Nivel de eficiencia

[https://www.canva.com/design/DAFvkQYt75c/I5VNBVEDaK1OrRmrjddVKg/edit?utm\\_content=DAFvkQYt75c&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAFvkQYt75c/I5VNBVEDaK1OrRmrjddVKg/edit?utm_content=DAFvkQYt75c&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

**2) 10 puntos. Ejecute el programa Ejercicio01.cpp:**

- a. 02 pts. Agregue una captura de pantalla que muestre el resultado obtenido:

```

Ejercicio01.cpp
40
41     printf("can't create thread :[%s]", strerror(err)); //impresion de mensaje si el hilo no
42     i++;
43 }
44 pthread_join(tid[0], NULL);
45 pthread_join(tid[1], NULL);
46 pthread_mutex_destroy(&lock); //destruccion de mutex dinamica ya usada
47
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $ gcc Ejercicio01.cpp -o Ejercicio01 -lpthread
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $ ./Ejercicio01
Job 1 started
Job 2 started
Job 2 finished
Job 1 finished
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $

```

b. 04 pts. Explique ¿Qué ocasiona el resultado obtenido de la ejecución?

R// El resultado del hilo 2 está sobrescribiéndose en el hilo 1, lo que hace que se imprima “Job 2 finished” 2 veces.

c. 04 pts. ¿Cómo puede solucionarse este problema?

```

Ejercicio01.cpp
22 pthread_mutex_t lock;
23
24 void* doSomething(void *arg)
25 {
26     pthread_mutex_lock(&lock);
27     unsigned long i = 0;
28     counter += 1;
29     printf("Job %d started\n", counter); //inicialización variable utilizada para retardo
30
31     for(i=0; i<(0xFFFFF);i++);
32     printf("Job %d finished\n", counter);
33
34     pthread_mutex_unlock(&lock);
35
36     return NULL;
37
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $ gcc Ejercicio01.cpp -o Ejercicio01 -lpthread
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $ gcc Ejercicio01.cpp -o Ejercicio01 -lpthread
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $ ./Ejercicio01
Job 1 started
Job 1 finished
Job 2 started
Job 2 finished
nelson_11@DESKTOP-EK34SRS:~/workspaces/uvg/Microprocesadores/lab $

```

R// El problema se soluciona agregando dos líneas, haciendo el bloqueo con mutex dentro de la subrutina doSomething antes de la sección crítica, después de eso antes del return agregando la línea de código que hace el unlock con mutex.

3) 30 pts, 7.5 c/una. Para el programa Ejercicio02.cpp:

a. Realice el diagrama de flujo del código.

Definir NTHREADS como 10

Definir una lista tid de tamaño NTHREADS para almacenar identificadores de hilo

Definir una variable contador

-Procedimiento principal

i = 0

Inicializar contador a 0

While i < NTHREADS hacer

Crear un hilo y almacenar su identificador en tid[i], y pasar i como argumento al procedimiento  
function

Si ocurre un error al crear el hilo, imprimir un mensaje de error

Incrementar i

Fin While

For i de 0 a NTHREADS - 1 hacer

Esperar a que el hilo tid[i] termine, realizando join

Fin for

Return 0

-Fin Procedimiento

-Procedimiento function(arg)

i = convertir arg a int

j = 0

Incrementar contador

Imprimir "---- Job" + contador + " started ----"

Imprimir " Realizado por hilo No. " + i

for j de 0 a 0xFFFFF hacer

// retardo (Este bucle parece que está simplemente para retrasar la ejecución del programa

Fin for

Imprimir "---- Job" + contador + " finished ----"

Exit del hilo

-Fin Procedimiento

- b. Responda: de acuerdo con el diagrama de flujo, ¿Cuál debería ser el orden en que se imprime el mensaje de cada hilo?

R//

Forma general:

Job n Started

Realizado por hilo n - 1

Job n finished

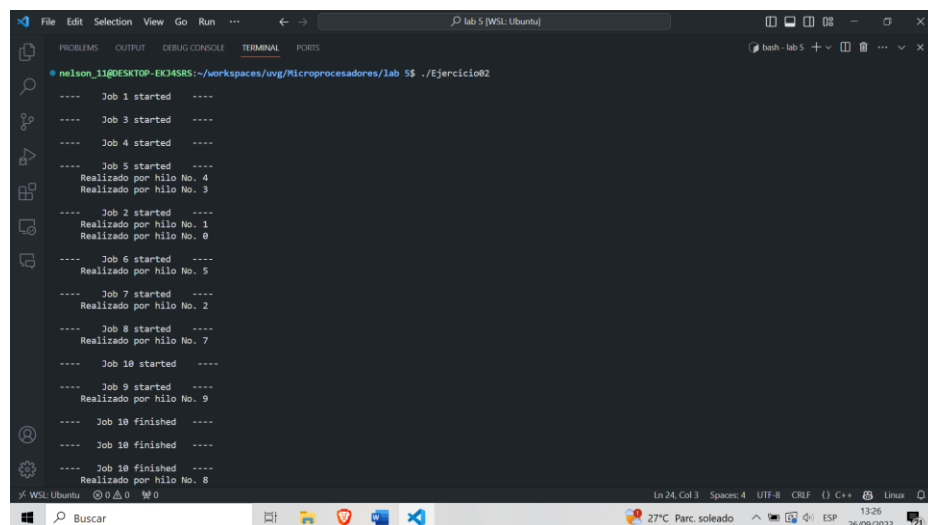
Ejemplo:

Job 1 Started

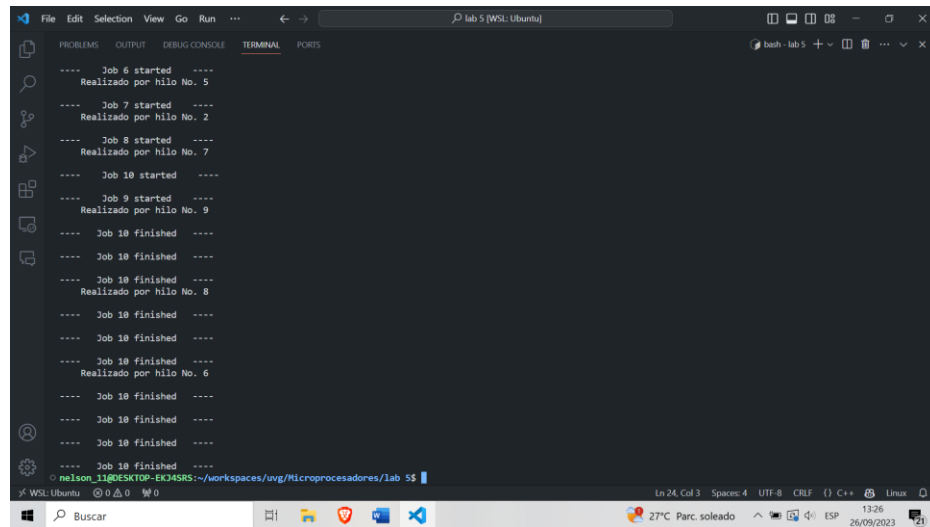
Realizado por hilo 0

Job 1 finished

- c. Corrija los errores y warnings. Ejecute el programa y coloque la captura de pantalla del resultado obtenido:



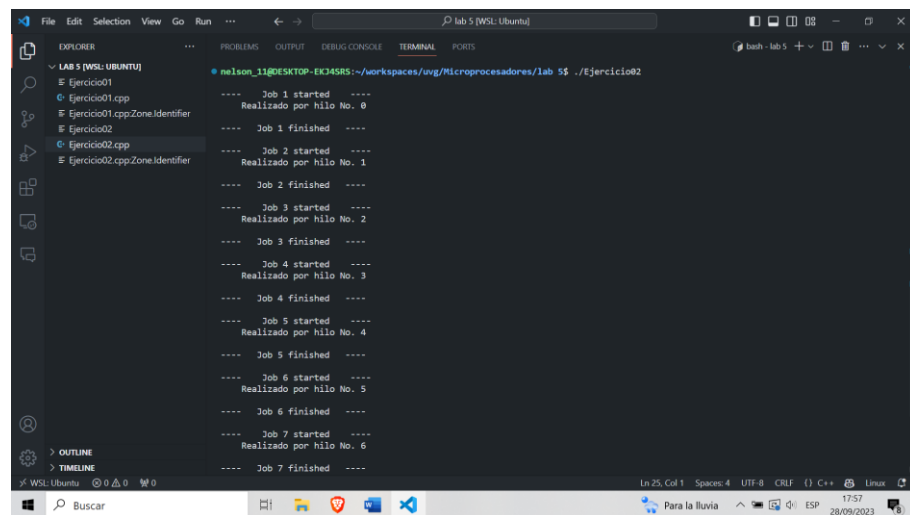
```
nelson_11@DESKTOP-EK45RS:~/workspaces/uvg/Microprocesadores/lab 5$ ./Ejercicio02
---- Job 1 started ----
---- Job 3 started ----
---- Job 4 started ----
---- Job 5 started ----
Realizado por hilo No. 4
Realizado por hilo No. 3
---- Job 2 started ----
Realizado por hilo No. 1
Realizado por hilo No. 0
---- Job 6 started ----
Realizado por hilo No. 5
---- Job 7 started ----
Realizado por hilo No. 2
---- Job 8 started ----
Realizado por hilo No. 7
---- Job 10 started ----
---- Job 9 started ----
Realizado por hilo No. 9
---- Job 10 finished ----
---- Job 10 finished ----
---- Job 10 finished ----
Realizado por hilo No. 8
```



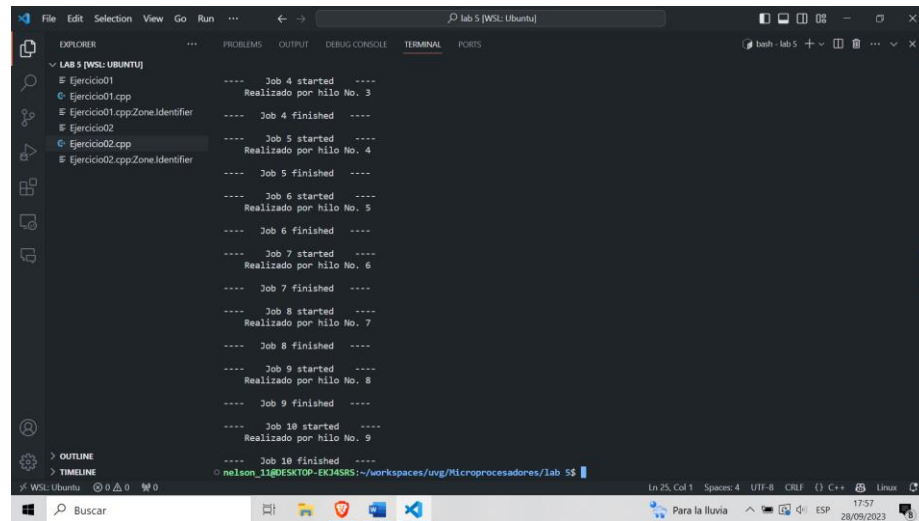
```
lab 5 [WSL: Ubuntu]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
---- Job 6 started ----
Realizado por hilo No. 5
---- Job 7 started ----
Realizado por hilo No. 2
---- Job 8 started ----
Realizado por hilo No. 7
---- Job 10 started ----
---- Job 9 started ----
Realizado por hilo No. 9
---- Job 10 finished ----
---- Job 10 finished ----
---- Job 10 finished ----
Realizado por hilo No. 8
---- Job 10 finished ----
---- Job 10 finished ----
---- Job 10 finished ----
Realizado por hilo No. 6
---- Job 10 finished ----
---- Job 10 finished ----
---- Job 10 finished ----
---- Job 10 finished ----
neilson_11@DESKTOP-EK34SR5:~/workspaces/uvg/Microprocesadores/lab 5$
```

- d. Modifique el código del programa y agregue las variables mutex necesarias para que todos los hilos puedan ejecutarse concurrentemente y acceder a la variable contador de manera ordenada, como se muestra en la siguiente figura:

```
----- Job 1 started -----  
Realizad por hilo No. 0  
  
----- Job 1 finished -----  
  
----- Job 2 started -----  
Realizad por hilo No. 1  
  
----- Job 2 finished -----  
  
----- Job 3 started -----  
Realizad por hilo No. 2  
  
----- Job 3 finished -----  
  
----- Job 4 started -----  
Realizad por hilo No. 3  
  
----- Job 4 finished -----  
  
----- Job 5 started -----  
Realizad por hilo No. 4  
  
----- Job 5 finished -----  
  
----- Job 6 started -----  
Realizad por hilo No. 5  
  
----- Job 6 finished -----  
  
----- Job 7 started -----  
Realizad por hilo No. 6  
  
----- Job 7 finished -----  
  
----- Job 8 started -----  
Realizad por hilo No. 7  
  
----- Job 8 finished -----  
  
----- Job 9 started -----  
Realizad por hilo No. 8  
  
----- Job 9 finished -----  
  
----- Job 10 started -----  
Realizad por hilo No. 9  
  
----- Job 10 finished -----
```



```
File Edit Selection View Go Run ... lab 5 [WSL: Ubuntu]  
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
LAB 5 [WSL: UBUNTU]  
Ejercicio01  
Ejercicio01.cpp  
Ejercicio01.cppZone.Identifier  
Ejercicio02  
Ejercicio02.cpp  
Ejercicio02.cppZone.Identifier  
nelson_11@DESKTOP-EK34SR5:~/workspaces/ug/Microprocesadores/lab 5$ ./Ejercicio02  
----- Job 1 started -----  
Realizad por hilo No. 0  
----- Job 1 finished -----  
----- Job 2 started -----  
Realizad por hilo No. 1  
----- Job 2 finished -----  
----- Job 3 started -----  
Realizad por hilo No. 2  
----- Job 3 finished -----  
----- Job 4 started -----  
Realizad por hilo No. 3  
----- Job 4 finished -----  
----- Job 5 started -----  
Realizad por hilo No. 4  
----- Job 5 finished -----  
----- Job 6 started -----  
Realizad por hilo No. 5  
----- Job 6 finished -----  
----- Job 7 started -----  
Realizad por hilo No. 6  
----- Job 7 finished -----  
----- Job 8 started -----  
Realizad por hilo No. 7  
----- Job 8 finished -----  
----- Job 9 started -----  
Realizad por hilo No. 8  
----- Job 9 finished -----  
----- Job 10 started -----  
Realizad por hilo No. 9  
----- Job 10 finished -----  
Ln 25, Col 1 Spaces: 4 UTF-8 CRLF C++ Linux  
17:57  
Para la lluvia 28/09/2023
```



```
lab 5 [WSL: Ubuntu]
EXPLORER
LAB 5 [WSL: UBUNTU]
  Ejercicio01
  Ejercicio01.cpp
  Ejercicio01.cpp.Zone.Identifier
  Ejercicio02
  Ejercicio02.cpp
  Ejercicio02.cpp.Zone.Identifier
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash - lab 5
---- Job 4 started ----
Realizado por hilo No. 3
---- Job 4 finished ----
---- Job 5 started ----
Realizado por hilo No. 4
---- Job 5 finished ----
---- Job 6 started ----
Realizado por hilo No. 5
---- Job 6 finished ----
---- Job 7 started ----
Realizado por hilo No. 6
---- Job 7 finished ----
---- Job 8 started ----
Realizado por hilo No. 7
---- Job 8 finished ----
---- Job 9 started ----
Realizado por hilo No. 8
---- Job 9 finished ----
---- Job 10 started ----
Realizado por hilo No. 9
---- Job 10 finished ----
nelson_11@DESKTOP-EK34SR5:~/workspaces/uvg/Microprocesadores/lab 5$
```

- 4) 33 pts. Realice un programa en C++ que calcule el producto punto de dos vectores A y B, de 1000 elementos c/uno. Siguiendo los siguientes lineamientos:
- 05 pts. El programa permite crear 1000 hilos y repartir los elementos del vector que serán operados en cada hilo.
  - 20 pts. Cada hilo debe ejecutar una subrutina (y enviar los datos a operar desde el main hacia la subrutina)
  - El resultado de la operación de multiplicación individual de cada índice de los vectores A y B deberá retornarse desde la subrutina ejecutada por cada hilo, hacia la rutina principal.
  - 08 pts. En la rutina principal debe: colocar cada valor de retorno (resultado de la multiplicación en la subrutina a) en un vector C. Luego, sumar de los valores guardados en el vector C y la impresión del resultado.