Universidad del Valle de Guatemala
Depto. Ingeniería
Sistemas Operativos
Sección 21
Catedrático: Erick Pineda

Nombre	Apellido	Carné
Mario	Perdomo	18029

Laboratorio #3:

Multithreading

Ejecución del programa:

```
mario@mario:~/Desktop/labs/lab3$ ./prueba2 "sudoku.21"
Hijo Columna ID: 5482
Hijo principal es : 5480
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       5480    3186    5480  0    1  80   0 - 19193 hrtime 14:43 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado ...
Hijo Fila ID: 5483
|--- Solucion Valida ---|
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       5480    3186    5480  0    1  80   0 - 19193 hrtime 14:43 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado...
mario@mario:~/Desktop/labs/lab3$
```

Figura 1: Ejecución final del programa con solo un thread

```
En la verificacion de las filas el hilo en proceso es: 6167
En la verificacion de las filas el hilo en proceso es: 6167
En la verificacion de las filas el hilo en proceso es: 6167
En la verificacion de las filas el hilo en proceso es: 6167
En la verificacion de las filas el hilo en proceso es: 6167
En la verificacion de las filas el hilo en proceso es: 6167
Hijo principal es : 6165
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       6165    3186    6165  0    1  80   0 - 19193 hrtime 16:08 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado ...
Hijo Fila ID: 6168
En la verificacion de las filas el hilo en proceso es: 6169
En la verificacion de las filas el hilo en proceso es: 6168
En la verificacion de las filas el hilo en proceso es: 6172
En la verificacion de las filas el hilo en proceso es: 6173
En la verificacion de las filas el hilo en proceso es: 6174
En la verificacion de las filas el hilo en proceso es: 6175
En la verificacion de las filas el hilo en proceso es: 6170
En la verificacion de las filas el hilo en proceso es: 6171
En la verificacion de las filas el hilo en proceso es: 6176
|--- Solucion Valida ---|
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       6165    3186    6165  0    1  80   0 - 74492 hrtime 16:08 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado...
```

Figura 2: Ejecución del programa utilizando OpenMP. El número de LWP's es de durante el proceso: 6176. Despues es de: 6165

```
mario@mario:~/Desktop/labs/lab3$ ./prueba2 "sudoku.21"
Hijo Columna ID: 6699
En la verificacion de las filas el hilo en proceso es: 6701
En la verificacion de las filas el hilo en proceso es: 6701
En la verificacion de las filas el hilo en proceso es: 6699
En la verificacion de las filas el hilo en proceso es: 6700
En la verificacion de las filas el hilo en proceso es: 6700
En la verificacion de las filas el hilo en proceso es: 6699
En la verificacion de las filas el hilo en proceso es: 6699
En la verificacion de las filas el hilo en proceso es: 6702
En la verificacion de las filas el hilo en proceso es: 6702
Hijo principal es : 6697
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       6697    3186    6697  0    1  80   0 - 25340 hrtime 17:03 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado ...
Hijo Fila ID: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
En la verificacion de las filas el hilo en proceso es: 6703
|--- Solucion Valida ---|
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S mario       6697    3186    6697  0    1  80   0 - 25340 hrtime 17:03 pts/1    00:00:00 ./prueba2 sudoku.21
Hijo terminado...
```

Figura 3: Ejecución del programa con omp_set_num_threads(1). El número de LWP's en el proceso es de: 6702. 6697 es después.

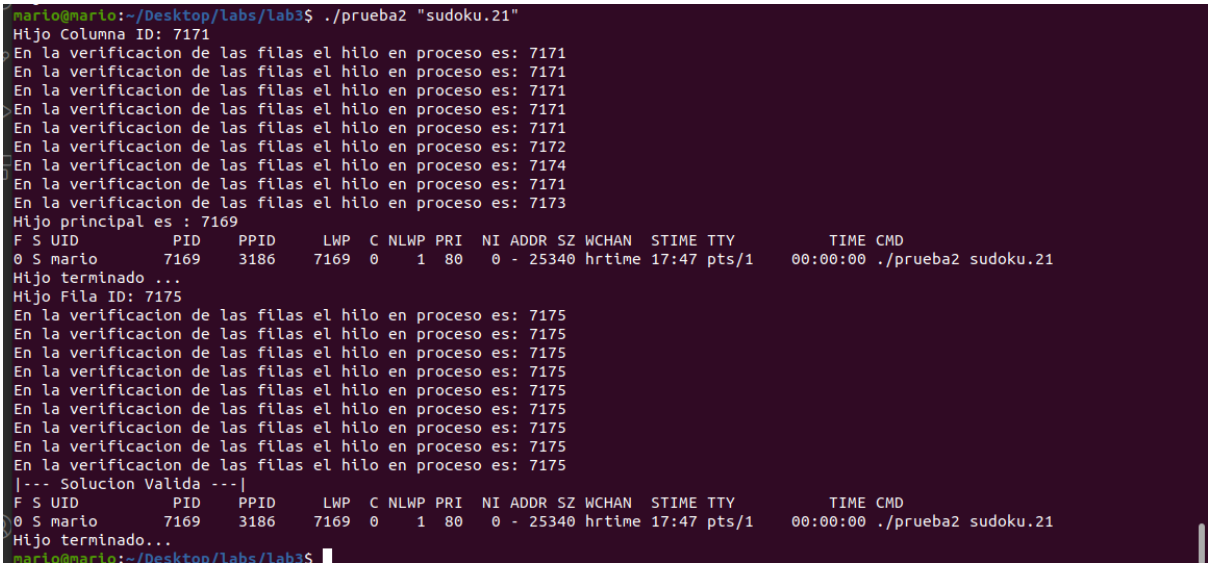

Figura 4: Ejecución del programa con schedule(dynamic). Vemos que algunos procesos de las verificaciones son iniciadas simultáneamente.

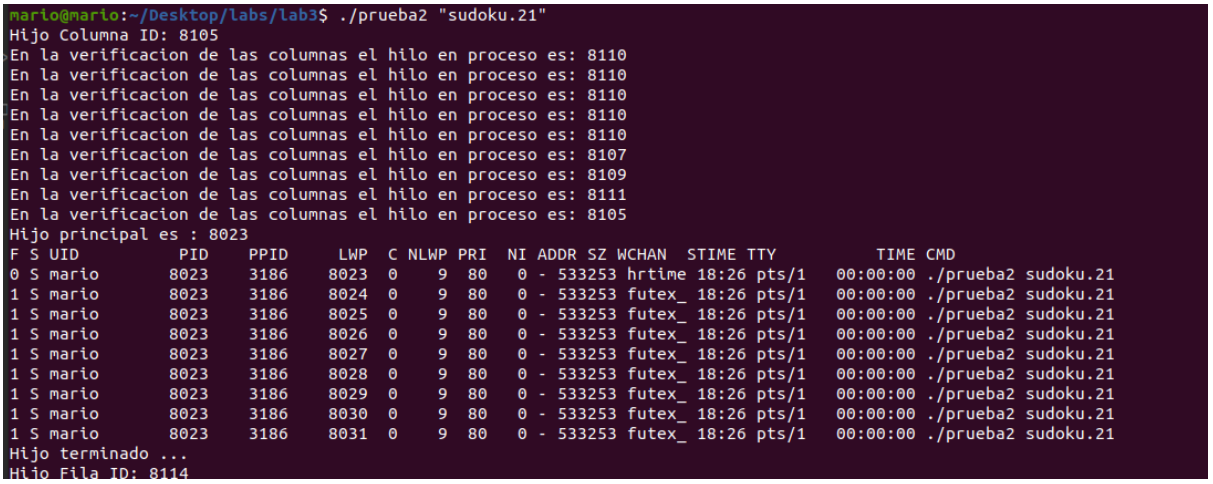

Figura 5: Ejecución del programa con omp_set_nested(). Aquí vemos la paralelización de los ciclos anidados, haciendo sus propios hijos/threads.

Preguntas:

1.¿Qué es una race condition y por qué hay que evitarlas?

Estas condiciones pasan cuando muchos procesos manipulan los datos al mismo tiempo, causando una dependencia de orden de quienes tienen accesos a la memoria. Se convierte en un error cuando uno o más de los posibles comportamientos no son deseables, lo cual es demasiado importante evitarlos. (PD: tuve estos problemas durante mis pruebas).

2.¿Cuál es la relación, en Linux,entre pthreads y clone()? ¿Hay diferencia al crear threads con uno o con otro?¿Qué es más recomendable?

Cuando se realiza el llamado a pthread_create(), se usa clone() para poder crear un nuevo hilo dentro de un proceso. Pthread se utiliza para subprocesos múltiples. Clone() es

una llamada al sistema de bajo nivel específica de Linux y se puede usar para crear procesos e hilos., sin embargo, muchos no lo recomiendan, ya que es un poco difícil de usar en la práctica porque no configura muchas de las sutilezas de los subprocesos POSIX más tradicionales como lo hace pthread_create.

3. ¿Dónde,en su programa, hay paralelización de tareas, y dónde de datos?

Hay paralelización de tareas en la creación de Pthreads, si no estoy mal.

Para la paralelización de datos son los OpenMPs, ya que ejecutan el mismo proceso pero con diferentes datos en distintos hilos.

4 .Al agregar los #pragmas a los ciclos for, ¿cuántos LWP's hay abiertos antes de terminar el main() y cuántos durante la revisión de columnas?¿Cuántos user threads deben haber abierto en cada caso, entonces? Hint:recuerde el modelo de multithreading que usan Linux y Windows.

Sabemos que el modelo de multithreading que usan los sistemas operativos, Linux y Windows, es de 1 a 1, sabemos que deberían de haber 4 LWP, siendo 4 threads de usuarios abiertos pero 5 durante la evaluación, y antes que termine el programa es de 1.

5.Al limitar el número de threadsen main()a uno, ¿cuántos LWP's hay abiertos durante la revisión de columnas? Compare esto con el número de LWP's abiertos antes de limitar el número de threads en main(). ¿Cuántos threads(en general) crea OpenMP por defecto?

Limitando el número de threads dentro del main,el cambio de los 4 threads de usuario fue de a 1. Por ahora, lo que tengo entendido OpenMp, debe crear la misma cantidad de threads por la cantidad de procesadores libres de los métodos donde hacemos estos llamados.

6.Observe cuáles LWP's están abiertos durante la revisión de columnas según ps. ¿Qué significa la primera columna de resultados de este comando? ¿Cuál es el LWP que está inactivo y por qué está inactivo?Hint: consulte las páginas del manual sobre ps.

La columna que representa la "S" es el estatus de un proceso. La columna "F" es la revisión de banderas. Ahora, respecto a que muestra la columna "S", en el programa muestra s los demás procesos, ya que son estados inactivos. Podemos darnos cuenta que en el programa, son inactivados, por el hecho de que estos procesos son del main(), esperando a que los demás hilos terminen las verificaciones de las columnas.

7. Comparelos resultados de psen la pregunta anterior con los que son desplegados por la función de revisión de columnas per se.¿Qué es un thread team en OpenMP y cuál es el master thread en este caso?¿Por qué parece haber un

thread“corriendo”, pero que no está haciendo nada? ¿Qué significa el término busy-wait?¿Cómo maneja OpenMP su thread pool?

OpenMP llama al pool de threads que pone a nuestra disposición a través de un thread team. La región paralela de OpenMP crea ese team, donde se llama a través del main() y es este thread donde inicia las instrucciones de paralelización. OpenMP provee un pool de threads pero no garantiza que se usarán.

El busy-wait es una técnica en la que un proceso verifica repetidamente si una condición es verdadera, como si la entrada del teclado o un bloqueo está disponible, en este caso, anda verificando si un thread anda disponible en lo que se le asigna un proceso.

Tengo entendido de que OpenMP abre un thread team, implementa una “política de espera” que determina cuánto tiempo permanecerán así los threads antes de ser suspendidos.

8.Luego de agregar por primera vez la cláusula schedule(dynamic) y ejecutar su programa repetidas veces, ¿cuál es el máximo número de threads trabajando según la función de revisión de columnas? Al comparar este número con la cantidad de LWP’s que se creaban antes de agregar schedule(), ¿qué deduce sobre la distribución de trabajo que OpenMP hace por defecto?

Máximo 4 diferentes números de threads, por la cantidad de procesadores. Puede ser que OpenMP implemente o tenga una política de “Balance” sobre cargas equitativas por defecto para cada thread.

9.Luego de agregar las llamadas omp_set_num_threads()a cada función donde se usa OpenMP y probar su programa, antes de agregar omp_set_nested(true), ¿hay más o menos concurrencia en su programa? ¿Es esto sinónimo de un mejor desempeño? Explique.

Hay más concurrencia en el programa durante la ejecución, sin embargo, aquí la clave es que que haya más concurrencia, no quiere decir mejor rendimiento, ya que esto incluye el overhead, y aumentaría un tiempo más largo para la organización de los threads.

10.¿Cuál es el efecto de agregar omp_set_nested(true)? Explique.

El efecto es la paralelización , con este tipo de paralelización, habilita o deshabilita las regiones paralelas anidadas, es decir, si los miembros del equipo pueden crear nuevos equipos. Esto aplica para los ciclos anidados al poder crear sus propios threads ocasionando así que se creen muchos más threads y, muchos más LWP’s.