Universidad del Valle de Guatemala
Sistemas Operativos
Docentes: Juan Luis Garcia Zarceño
Semestre I, 2021

Laboratorio #3

Fecha de Entrega: 16 de marzo, 2022.

Descripción: en este laboratorio se empleará *multithreading* por medio de `pthreads` y OpenMP para desarrollar un verificador de soluciones para *sudokus* de nueve por nueve. Los entregables serán todo el código escrito, así como un documento que responda a las preguntas planteadas al final. Se recomienda auxiliar sus respuestas con *screenshots* de la ejecución de su programa.

Materiales: una máquina virtual con Linux o Windows, pero que tenga GCC, y documentación sobre OpenMP.

Contenido:

OpenMP busca paralelizar (no sólo ejecutar de forma concurrente), por lo que su funcionamiento requiere de más de un procesador. Si usará una máquina virtual, comience por asegurarse de que su máquina cuente con cuatro procesadores (en VirtualBox, ventana *Settings*, menú *System*, *tab Processor*). La cantidad se especifica para uniformizar las respuestas a las preguntas que se plantean al final.

Cree un programa en C llamado `SudokuValidator.c`. En él escriba tres funciones que se encarguen de revisar que todos los números del uno al nueve estén:

- En cada columna de un arreglo de nueve por nueve.
- En cada fila de un arreglo de nueve por nueve.
- En un subarreglo de tres por tres dentro de un arreglo de nueve por nueve.

Las funciones para verificación de filas y columnas serán iguales exceptuando un intercambio de índices al recorrer el arreglo. La función de revisión de subarreglos debe recibir una fila y una columna para ubicar la esquina superior izquierda de un cuadrado de tres por tres, donde iniciará la revisión dentro del arreglo de nueve por nueve. Todas estas funciones se deben basar en ciclos `for` obligatoriamente.

Este programa recibirá, en terminal, la ubicación de un archivo (sólo el nombre, si está en el mismo directorio que `SudokuValidator.c`) que contiene una solución a un *sudoku* de nueve por nueve. El formato de las soluciones debe ser un único *string* de ochenta y un dígitos, en la primera línea, comenzando por la celda superior izquierda del *sudoku* y avanzando de izquierda a derecha, por filas. Para este laboratorio se provee una solución de ejemplo en el archivo "*sudoku*".

Lo primero que su `main()` deberá hacer es abrir el archivo usando `open()` y *mappearlo* a su memoria usando `mmap()`. Luego debe ejecutar un `for` en el que se copie cada símbolo del *string* en el archivo de solución a un arreglo bidimensional de nueve por nueve, de modo que le quede una grilla lógica como la que se muestra en la página siguiente.

Se recomienda que su arreglo bidimensional sea global (es decir, que esté declarado fuera del `main()`) para que sea accesible por varios *threads*. Luego de llenar la grilla, escriba un `for` que haga la revisión, con su función, de los subarreglos de tres por tres que conforman el arreglo de nueve por nueve (**nota**: revise los subarreglos de tres por tres cuya primera posición (si comenzamos desde 1) sea $[i, i]$ para $i \in \{1,4,7\}$).

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

Grilla lógica ejemplar

Luego de lo anterior, obtenga el número de proceso (no el de *thread*) y ejecute un `fork()`. En el proceso hijo convierta el número del proceso padre (no el de *thread*) a texto, y ejecute por medio de `execlp()` el siguiente comando:

```
ps -p <#proc> -lLf
```

donde `<#proc>` es el número del proceso padre. Este comando permite ver información relacionada al proceso `<#proc>` que incluye los *lightweight processes* que tenga asociados.

En el proceso padre:

- Cree un `pthread` que haga su revisión de columnas.
- Ejecute `pthread_join()` y luego despliegue el número de *thread* en ejecución. Para lograrlo debe #incluir `<sys/syscall.h>` en su programa y ejecutar `syscall(SYS_gettid)` (el resultado de esta llamada de sistema es el *id* del *thread*).
- Espere a que concluya el hijo que está ejecutando `ps`.
- Realice su revisión de filas.
- Despliegue si la solución al *sudoku* es válida o no.
- Ejecute un nuevo `fork()` y ejecute el comando `ps` en el proceso hijo, tal como se describe en instrucciones anteriores. Esto servirá para comparar el número de LWP's asociados al proceso

padre cuando se está realizando la revisión de columnas y cuando (el padre) está a punto de terminar.

- Espere al hijo y retorne 0.

Observe que la creación de un *thread* que ejecute la revisión de columnas implica la creación de una función que sea asignable a un *thread* en el cual, a su vez, se ejecute su función de revisión de columnas. Es decir, una función que tenga tipo de retorno `void*` y que termine con `pthreads_exit(0)`. En esa función tipo `void*` también despliegue el número de *thread* en ejecución.

```
gabriel@debian:~/Escritorio/Lab3$ gcc -o sdk SudokuValidator.c -pthread
gabriel@debian:~/Escritorio/Lab3$ ./sdk sudoku
Thread de ejecucion de columnas: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En la revision de columnas el thread en ejecucion es: 9162
En el main trabaja el thread numero: 9160
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9160    3497    9160  0    1  80   0 - 19073 -      03:09 pts/0    00:00:00 ./sdk sudoku
Sudoku validado!
Antes de terminar el estado de este proceso y sus threads es:
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9160    3497    9160  0    1  80   0 - 19073 -      03:09 pts/0    00:00:00 ./sdk sudoku
gabriel@debian:~/Escritorio/Lab3$
```

El siguiente es un ejemplo de cómo podría verse el *output* de su programa hasta este momento:

```
os@debian:~/sudoku$ ./SudokuValidator.o sudoku
El thread que ejecuta el metodo para ejecutar el metodo de revision de columnas es: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
En la revision de columnas el siguiente es un thread en ejecucion: 9027
El thread en el que se ejecuta main es: 9025
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S os        9025  1813  9025  0    1  80   0 -  2842 -      06:53 pts/1    00:00:00 ./SudokuValidator.o sudoku
Sudoku resuelto!
Antes de terminar el estado de este proceso y sus threads es:
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S os        9025  1813  9025  0    1  80   0 -  2842 -      06:53 pts/1    00:00:00 ./SudokuValidator.o sudoku
os@debian:~/sudoku$
```

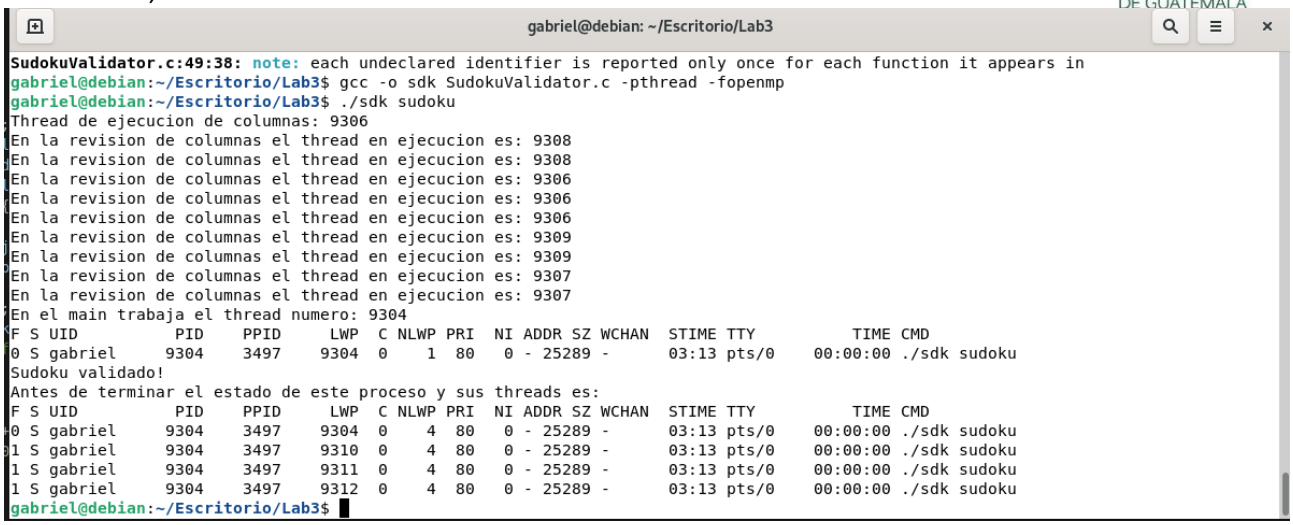
Como siguiente paso deberá paralelizar todos los ciclos `for` que pueda (vea la nota **importante**) usando OpenMP. Para ello simplemente es necesario que la siguiente línea preceda inmediatamente a la del `for` en cada caso:

```
#pragma omp parallel for
```

Importante: evite las *race conditions*. Investigue el uso de la directiva `private` de OpenMP para auxiliarse en este aspecto. No todos los ciclos `for` deberán ser precedidos por la directiva.

Ejecutar su programa ahora deberá resultar en un *output* similar al siguiente:

Universidad del Valle de Guatemala
Sistemas Operativos
Docentes: Juan Luis Garcia Zarceño
Semestre I, 2021



```
os@debian:~/sudoku$ ./SudokuValidator.o sudoku
El thread que ejecuta el metodo para ejecutar el metodo de revision de columnas es: 9059
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S os        9054  1813  9054  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 R os        9054  1813  9055  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 R os        9054  1813  9056  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 R os        9054  1813  9057  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 S os        9054  1813  9059  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 S os        9054  1813  9060  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
En la revision de columnas el siguiente es un thread en ejecucion: 9061
En la revision de columnas el siguiente es un thread en ejecucion: 9061
En la revision de columnas el siguiente es un thread en ejecucion: 9061
1 S os        9054  1813  9061  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
En la revision de columnas el siguiente es un thread en ejecucion: 9059
En la revision de columnas el siguiente es un thread en ejecucion: 9059
En la revision de columnas el siguiente es un thread en ejecucion: 9059
1 S os        9054  1813  9062  0    8  80   0 - 15392 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
En la revision de columnas el siguiente es un thread en ejecucion: 9060
En la revision de columnas el siguiente es un thread en ejecucion: 9060
En la revision de columnas el siguiente es un thread en ejecucion: 9060
El thread en el que se ejecuta main es: 9054
Sudoku resuelto!
Antes de terminar el estado de este proceso y sus threads es:
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S os        9054  1813  9054  0    4  80   0 - 15648 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 S os        9054  1813  9055  0    4  80   0 - 15648 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 S os        9054  1813  9056  0    4  80   0 - 15648 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
1 S os        9054  1813  9057  0    4  80   0 - 15648 -      07:12 pts/1    00:00:00 ./SudokuValidator.o sudoku
os@debian:~/sudoku$
```

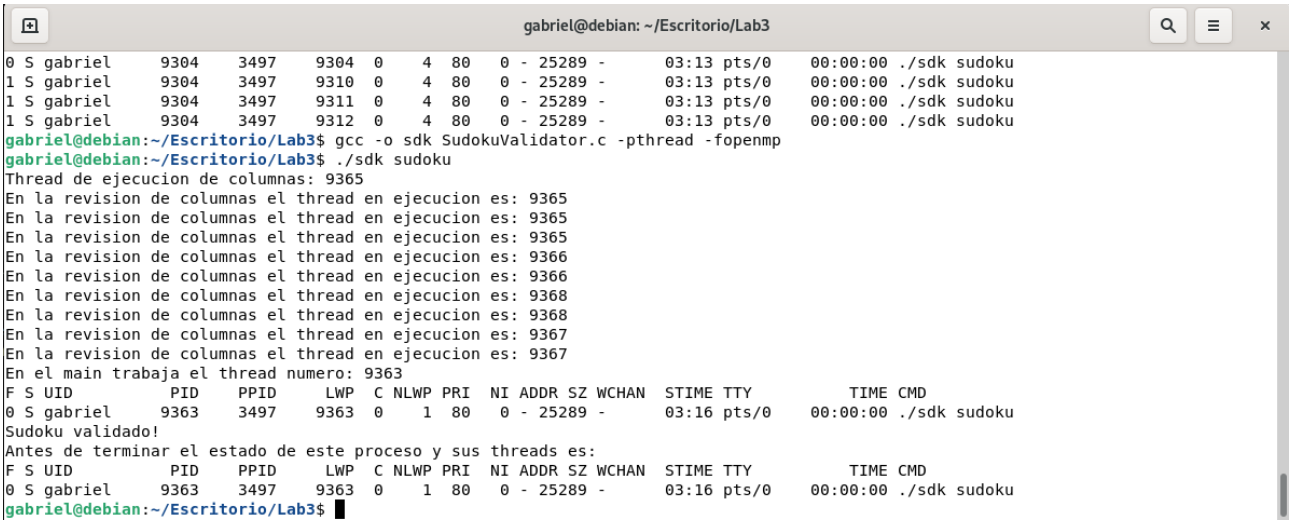
Anote el número de LWP's que se tienen durante la revisión de columnas y antes de terminar el programa.

LWP's: 9304,9310,9311,9312

Agregue la siguiente instrucción al principio de `main()`:

```
omp_set_num_threads(1);
```

Ejecute su programa y note el resultado de las ejecuciones de `ps`. También anote los números de *thread* desplegados durante la revisión de columnas.



Ahora, agregue la siguiente directiva a todas las líneas `#pragma` que incluyó anteriormente:

```
schedule(dynamic)
```

Ejecute su programa varias veces y observe los números de *thread* que se despliegan durante la revisión

de columnas. Compárelos con el resultado de `ps` que se despliega durante la ejecución del `pthread` y anote sus observaciones.

```
gabriel@debian:~/Escritorio/Lab3$ gcc -o sdk SudokuValidator.c -pthread -fopenmp
gabriel@debian:~/Escritorio/Lab3$ ./sdk sudoku
Thread de ejecucion de columnas: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9414
En la revision de columnas el thread en ejecucion es: 9416
En la revision de columnas el thread en ejecucion es: 9415
En la revision de columnas el thread en ejecucion es: 9417
En el main trabaja el thread numero: 9412
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9412    3497    9412  0    1  80   0 - 25289 -      03:17 pts/0    00:00:00 ./sdk sudoku
Sudoku validado!
Antes de terminar el estado de este proceso y sus threads es:
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9412    3497    9412  0    1  80   0 - 25289 -      03:17 pts/0    00:00:00 ./sdk sudoku
gabriel@debian:~/Escritorio/Lab3$
```

Durante la revisión se despliegan cuatro diferentes threads pero me llama la atención que uno solo se utiliza para 5 columnas.

Como siguiente paso, agregue una llamada a `omp_set_num_threads()` al inicio de cada función donde se ejecute un `for` paralelo, determinando el número de *threads* adecuados (*e.g.*, si su función ejecuta un `for` paralelo de nueve iteraciones, posiblemente el número de *threads* deba ser nueve). Ejecute su programa varias veces y anote los efectos sobre los *threads* en los resultados de `ps`. Repita el procedimiento comentando la cláusula `schedule()` en el primer `for` paralelo de su revisión de columnas. Finalmente agregue la siguiente instrucción al principio de cada función que use OpenMP:

```
gabriel@debian:~/Escritorio/Lab3$ gcc -o sdk SudokuValidator.c -pthread -fopenmp
gabriel@debian:~/Escritorio/Lab3$ ./sdk sudoku
Thread de ejecucion de columnas: 9467
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9468
En la revision de columnas el thread en ejecucion es: 9469
En el main trabaja el thread numero: 9465
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9465    3497    9465  0    1  80   0 - 74441 -      03:19 pts/0    00:00:00 ./sdk sudoku
Sudoku validado!
Antes de terminar el estado de este proceso y sus threads es:
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9465    3497    9465  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9476  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9477  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9478  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9479  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9480  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9481  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9482  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9465    3497    9483  0    9  80   0 - 82637 -      03:19 pts/0    00:00:00 ./sdk sudoku
gabriel@debian:~/Escritorio/Lab3$
```

Universidad del Valle de Guatemala
Sistemas Operativos
Docentes: Juan Luis Garcia Zarceño
Semestre I, 2021

```
gabriel@debian:~/Escritorio/Lab3$ gcc -o sdk SudokuValidator.c -pthread -fopenmp
gabriel@debian:~/Escritorio/Lab3$ ./sdk sudoku
Thread de ejecucion de columnas: 9527
En la revision de columnas el thread en ejecucion es: 9531
En la revision de columnas el thread en ejecucion es: 9532
En la revision de columnas el thread en ejecucion es: 9533
En la revision de columnas el thread en ejecucion es: 9534
En la revision de columnas el thread en ejecucion es: 9535
En la revision de columnas el thread en ejecucion es: 9528
En la revision de columnas el thread en ejecucion es: 9529
En la revision de columnas el thread en ejecucion es: 9527
En la revision de columnas el thread en ejecucion es: 9530
En el main trabaja el thread numero: 9525
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9525    3497    9525  0    1  80   0 - 74441 -      03:20 pts/0    00:00:00 ./sdk sudoku
Sudoku validado!
Antes de terminar el estado de este proceso y sus threads es:
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9525    3497    9525  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9536  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9537  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9538  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9539  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9540  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9541  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9542  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9525    3497    9543  0    9  80   0 - 82637 -      03:20 pts/0    00:00:00 ./sdk sudoku
gabriel@debian:~/Escritorio/Lab3$
```

```
omp_set_nested(true);
```

Ejecute su programa y anote los efectos sobre el resultado.

```
gabriel@debian:~/Escritorio/Lab3$ gcc -o sdk SudokuValidator.c -pthread -fopenmp
gabriel@debian:~/Escritorio/Lab3$ ./sdk sudoku
Thread de ejecucion de columnas: 9714
En la revision de columnas el thread en ejecucion es: 9715
En la revision de columnas el thread en ejecucion es: 9716
En la revision de columnas el thread en ejecucion es: 9714
En la revision de columnas el thread en ejecucion es: 9717
En la revision de columnas el thread en ejecucion es: 9721
En la revision de columnas el thread en ejecucion es: 9720
En la revision de columnas el thread en ejecucion es: 9718
En la revision de columnas el thread en ejecucion es: 9722
En la revision de columnas el thread en ejecucion es: 9719
En el main trabaja el thread numero: 9712
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9712    3497    9712  0    1  80   0 - 74441 -      03:24 pts/0    00:00:00 ./sdk sudoku
Sudoku validado!
Antes de terminar el estado de este proceso y sus threads es:
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S gabriel     9712    3497    9712  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9723  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9724  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9725  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9726  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9727  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9728  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9729  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
1 S gabriel     9712    3497    9730  0    9  80   0 - 82637 -      03:24 pts/0    00:00:00 ./sdk sudoku
gabriel@debian:~/Escritorio/Lab3$
```

Al añadir omp_set_nested(true) se generaron mas hilos y prácticamente cada columna contaba con su propio hilo para revisarla.

Universidad del Valle de Guatemala
Sistemas Operativos
Docentes: Juan Luis Garcia Zarceño
Semestre I, 2021

Responda las siguientes preguntas:

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

   Una race condition es cuando dos o mas procesos acceden a un espacio o recurso compartido sin control. Hay que evitarlas ya que conlleva muchos problemas para el sistema operativo, desde procesos mucho mas lentos hasta muerte del sistema operativo.
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?

   Pthreads se utiliza para crear multithreading y clone es una llamada a sistema de Linux que permite crear procesos y threads. Si hay diferencia ya que con clone mas bien se estaría creando multiprocesos y con Pthreads multithreading, dependiendo de la necesidad es mejor uno u otro.
3. ¿Dónde, en su programa, hay paralelización de tareas, y dónde de datos?
   Hay paralelización de tareas en los hilos que revisan las columnas y paralelización de datos en el main donde se ejecuta la función de revisión de filas y de 3x3.
4. Al agregar los `#pragma`s 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 abiertos en cada caso, entonces? ***Hint*****:** recuerde el modelo de *multithreading* que usan Linux y Windows.
   Durante la revisión de columnas hay 1 y al terminar el main 4. Deberían haber 9 en la revisión de columnas y 1 en el main.
5. Al limitar el número de *threads* en `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?
   OpenMp crea 3 threads por defecto y al limitar el numero de threads en el main hay 5 durante la revisión de columnas.
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 primera columna muestra la bandera del estado del proceso y el que esta inactivo el 9712 debido a que esta esperando.
7. Compare los resultados de `ps` en 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*?
   Un thread team en OpenMP es una construcción combinada de distintos threads y en este caso el master thread es el 9712 pues es el único que ejecuta algo diferente, los secundarios o hijos ejecutan lo mismo. Parece haber un thread corriendo debido a que esta en espera pues debe verificar que otras condiciones se cumplan. El termino busy-wait significa que un proceso repetidamente verifica una condición. OpenMP maneja el thread pool siempre dejando un thread principal y utilizando el busy-wait para cuando los threads ya finalizaron sus tareas en una zona especifica esperando a ser activados nuevamente.
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?
   Luego de agregar Schedule el numero de threads trabajando en la revisión de columnas fue 4, al compararlo con el numero de LWP'S de antes se puede deducir que la distribución que hace OMP se basa en la dificultad o costo de finalizar un proceso.

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.
10. Hay mayor concurrencia pues hay más procesos ejecutándose al mismo tiempo y el desempeño depende de para que se utilice pero diría que en su mayoría va a garantizar un mejor desempeño.
11. ¿Cuál es el efecto de agregar `omp_set_nested(true)` ? Explique.
    Provoca que cada columna sea revisada por un hilo distinto, ya que esta activa que se use el mayor numero de hilos posible para realizar paralelismo.