#pragma omp parallel for

```
jack@Mai-san:~/U/VII/SO/Lab3$ gcc -o lab3 SudokuValidator.c -fopenmp
jack@Mai-san:~/U/VII/SO/Lab3$ ./lab3
PID: 3410
ID del PAPA: 3410
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S jack        3410     506    3410  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3411  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3412  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3413  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 S jack        3410     506    3414  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3415  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 S jack        3410     506    3416  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3417  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 S jack        3410     506    3418  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3420  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3421  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 S jack        3410     506    3422  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3423  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3424  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3425  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3426  0   18  80   0 - 51985 -      17:53 pts/4    00:00:00 ./lab3
Columna 6 revisanda por el Thread 3426
Columna 2 revisanda por el Thread 3422
Columna 5 revisanda por el Thread 3425
1 S jack        3410     506    3427  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
Columna 4 revisanda por el Thread 3424
1 S jack        3410     506    3428  0   18  80   0 - 51985 futex_ 17:53 pts/4    00:00:00 ./lab3
Columna 3 revisanda por el Thread 3423
Columna 0 revisanda por el Thread 3420
Columna 8 revisanda por el Thread 3428
Columna 7 revisanda por el Thread 3427
Columna 1 revisanda por el Thread 3421
Sudoku aceptado
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 jack        3410     506    3410  0    9  80   0 - 41772 do_wai 17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3411  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3412  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3413  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3414  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3415  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3416  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3417  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
1 R jack        3410     506    3418  0    9  80   0 - 41772 -      17:53 pts/4    00:00:00 ./lab3
jack@Mai-san:~/U/VII/SO/Lab3$
```

Omp_set_num_threads(1):

```
jack@Mai-san:~/U/VII/SO/Lab3$ gcc -o lab3 SudokuValidator.c -fopenmp
jack@Mai-san:~/U/VII/SO/Lab3$ ./lab3
PID: 3536
ID del PAPA: 3536
Columna 0 revisanda por el Thread 3538
Columna 1 revisanda por el Thread 3538
Columna 2 revisanda por el Thread 3538
Columna 3 revisanda por el Thread 3538
Columna 4 revisanda por el Thread 3538
Columna 5 revisanda por el Thread 3538
Columna 6 revisanda por el Thread 3538
Columna 7 revisanda por el Thread 3538
Columna 8 revisanda por el Thread 3538
F S UID          PID    PPID     LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S jack        3536     506    3536  0    2  80   0 - 19233 futex_ 18:13 pts/4    00:00:00 ./lab3
1 R jack        3536     506    3538  0    2  80   0 - 19233 -      18:13 pts/4    00:00:00 ./lab3
Sudoku aceptado
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 jack        3536     506    3536  0    1  80   0 - 19233 do_wai 18:13 pts/4    00:00:00 ./lab3
jack@Mai-san:~/U/VII/SO/Lab3$
```

Schedule(dynamic)

```
jack@Mai-san:~/U/VII/SO/Lab3$ ./lab3 sudoku
PID: 3813
ID del PAPA: 3813
Columna 3 revisanda por el Thread 3818
Columna 8 revisanda por el Thread 3821
Columna 0 revisanda por el Thread 3824
Columna 6 revisanda por el Thread 3817
Columna 4 revisanda por el Thread 3820
Columna 1 revisanda por el Thread 3822
Columna 5 revisanda por el Thread 3819
Columna 7 revisanda por el Thread 3825
Columna 2 revisanda por el Thread 3823
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S jack      3813   506  3813  0    3  80   0 - 29478 do_wai 18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3814  0    3  80   0 - 29478 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3815  0    3  80   0 - 29478 -      18:33 pts/4    00:00:00 ./lab3 sudoku
Sudoku aceptado
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 jack      3813   506  3813  0    9  80   0 - 33576 do_wai 18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3814  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3815  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3826  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3827  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3828  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3829  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3830  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3813   506  3831  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
jack@Mai-san:~/U/VII/SO/Lab3$ ./lab3 sudoku
PID: 3847
ID del PAPA: 3847
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S jack      3847   506  3847  0    8  80   0 - 33544 futex_ 18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3848  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3849  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3851  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3852  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3853  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3854  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3855  0    8  80   0 - 33544 -      18:33 pts/4    00:00:00 ./lab3 sudoku
Columna 4 revisanda por el Thread 3854
Columna 3 revisanda por el Thread 3855
Columna 1 revisanda por el Thread 3859
Columna 5 revisanda por el Thread 3856
Columna 6 revisanda por el Thread 3851
Columna 8 revisanda por el Thread 3858
Columna 7 revisanda por el Thread 3852
Columna 0 revisanda por el Thread 3853
Columna 2 revisanda por el Thread 3857
Sudoku aceptado
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 jack      3847   506  3847  0    9  80   0 - 33576 do_wai 18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3848  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3849  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3860  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3861  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3862  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3863  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3864  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3847   506  3865  0    9  80   0 - 33576 -      18:33 pts/4    00:00:00 ./lab3 sudoku
jack@Mai-san:~/U/VII/SO/Lab3$
```

Omp_set_nested(true):

```
jack@Mai-san:~/U/VII/SO/Lab3$ gcc -o lab3 SudokuValidator.c -fopenmp
jack@Mai-san:~/U/VII/SO/Lab3$ ./lab3 sudoku
PID: 3759
ID del PAPA: 3759
Columna 1 revisanda por el Thread 3768
Columna 0 revisanda por el Thread 3765
Columna 8 revisanda por el Thread 3771
Columna 3 revisanda por el Thread 3764
Columna 4 revisanda por el Thread 3766
Columna 2 revisanda por el Thread 3770
Columna 6 revisanda por el Thread 3769
Columna 5 revisanda por el Thread 3767
Columna 7 revisanda por el Thread 3763
F S UID        PID  PPID   LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S jack      3759   506  3759  0   12  80   0 - 39729 futex_ 18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3760  0    8  80   0 - 39723 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3761  0    4  80   0 - 31527 -      18:17 pts/4    00:00:00 ./lab3 sudoku
Sudoku aceptado
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 jack      3759   506  3759  0    9  80   0 - 33576 do_wai 18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3760  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3761  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3772  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3773  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3774  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3775  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3776  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
1 R jack      3759   506  3777  0    9  80   0 - 33576 -      18:17 pts/4    00:00:00 ./lab3 sudoku
jack@Mai-san:~/U/VII/SO/Lab3$
```

Responda las siguientes preguntas:

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

Las race conditions se refieren a problemas de concurrencia donde dos o mas hilos o procesos acceden a un recurso compartido como una variable y lo tratan de modificar al mismo tiempo lo que puede causar resultados inesperados por lo que se busca evitarlas y eliminarlas con métodos de sincronización.

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?

En Linux, pthreads y clone() son dos formas diferentes de crear hilos de ejecución. pthreads es una interfaz de programación de aplicaciones (API) que proporciona una serie de funciones para crear y administrar hilos de ejecución en un programa. Por otro lado, clone() es una llamada al sistema que permite crear un nuevo proceso o hilo de ejecución a partir de un proceso existente. En general, pthreads es más fácil de usar y proporciona una abstracción de nivel más alto para trabajar con hilos, mientras que clone() proporciona un mayor control y flexibilidad, pero también es más complicado de usar.

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

El programa realiza las tareas de forma paralela donde se realiza el fork() de las tareas para el padre y el hijp. Para la paralelización de datos tenemos todas las funciones que buscan un error dentro del sudoku ya sea en filas, columnas o en los subconjuntos ya que al usar el parallel for cada thread ira buscando su propio set de datos paralelamente que los otros.

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 abiertos en cada caso, entonces? Hint: recuerde el modelo de multithreading que usan Linux y Windows.

Como podemos observar en la primera imagen hay 18 LWPs abiertos durante la revisión de las columnas y hay la misma cantidad de threads ya que en Linux cada hilo ejecuta su propio LWP y por eso podemos observar que cada columnda revisada por un hilo diferente muestra el LWP del hilo en cuestión. Al terminar el programa hay 9 LWPs abiertos lo cual es justo la mitad lo que nso podría indicar que cerraron todos los hilos donde se realizaron las verficaciones de las columnas

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?

Al limitar el numero de hilos a 1 podemos observar como la revison de columnas es dada por un mismo hilo al igual que solo se crean 2 threads uno para el proceso main y otro de la verificación de las columnas. Antes de terminar el programa se puede ver que solo queda el proceso padre esperando y se ha eliminado el thread de verificaciond ela columna.

El número de hilos que crea OpenMP por defecto depende de la implementación y de la configuración del sistema en el que se ejecuta el programa. En general, la especificación de OpenMP no establece un número predeterminado de hilos, por lo que el número de hilos por defecto puede variar según la implementación. En algunos sistemas, el número de hilos por defecto puede ser el número de núcleos de CPU disponibles en el sistema o el número de hilos de ejecución disponibles en el momento de la ejecución.

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. Las primeras columnas son F y S, la primera nos indica la process flags esta nos indica si esta en 1 que un proceso esta forked pero no ejecutado, la segunda nos habla sobre el process state codes, los dos codes que vemos al correr el programa son S y R que significa sleeping y running respectivamente. Podemos observar en la segunda imagen que el LWP es el padre del fork ya que esta esperando que el hijo termine su ejecución.

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?

En la salida de la función de revisión de columnas, podemos ver que hay varios threads corriendo simultáneamente, y que el thread con identificador 0 es el que se encarga de imprimir el resultado final.

En OpenMP, un thread team es un conjunto de threads que trabajan juntos para completar una tarea. El master thread es el thread que crea el thread team y controla la ejecución del programa. En este caso, el master thread es el thread principal que ejecuta la función main().

Hay un thread "corriendo" pero que no está haciendo nada debido a que está esperando a que los otros threads finalicen su trabajo para luego imprimir el resultado final.

El término busy-wait se refiere a un tipo de espera activa donde un proceso o thread está esperando a que ocurra un evento y verifica periódicamente si ha sucedido. Esto puede ser ineficiente, ya que consume recursos mientras espera, y puede afectar negativamente el rendimiento. OpenMP maneja su thread pool de manera dinámica, asignando tareas a los threads disponibles en el equipo en función de la carga de trabajo y de los recursos disponibles. OpenMP también permite al programador especificar el número de threads a utilizar en una sección específica del código mediante la función omp_set_num_threads().

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?

Después de agregar la cláusula schedule(dynamic) y ejecutar el programa varias veces, se puede observar que el número máximo de threads trabajando en la función de revisión de columnas varía entre diferentes ejecuciones del programa. Esto se debe a que la cláusula schedule(dynamic) asigna dinámicamente bloques de trabajo a los threads disponibles, lo que significa que el número de threads que se utilizan puede variar según la carga de trabajo.

En comparación con la cantidad de LWP's que se creaban antes de agregar la cláusula schedule(), se puede deducir que OpenMP distribuye el trabajo entre los threads disponibles de manera más eficiente al utilizar la cláusula schedule(dynamic), ya que se pueden utilizar menos threads para realizar la misma cantidad de trabajo.

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.

Al agregar la llamada a omp_set_num_threads() en cada función que utiliza OpenMP, es posible que se tenga más concurrencia en el programa, lo que significa que se ejecutarán más hilos de manera simultánea. Sin embargo, no necesariamente esto se traduce en un mejor desempeño. El rendimiento de un programa depende de muchos factores, como la cantidad de trabajo que realiza cada hilo, la eficiencia del algoritmo utilizado, la cantidad de recursos disponibles en el sistema, entre otros.

En general, si se aumenta el número de hilos ejecutándose simultáneamente, se incrementará el grado de paralelismo en el programa, lo que en principio podría mejorar el desempeño. Pero también puede ocurrir que al aumentar el número de hilos se produzca una sobrecarga en el sistema, lo que llevaría a un peor desempeño. Por esta razón, es importante encontrar el número óptimo de hilos que maximice el rendimiento del programa que en este caso fue 9 ya que es la cantidad de arrays que el sudoku tendría que revisar en filas y columnas.

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

Al agregar la llamada a omp_set_nested(true), se habilita el anidamiento de regiones paralelas en el programa. Esto significa que si dentro de una región paralela se encuentra otra región paralela, las tareas de ambas regiones podrán ser asignadas a diferentes hilos de manera simultánea. En otras palabras, se podrán crear hilos dentro de hilos.