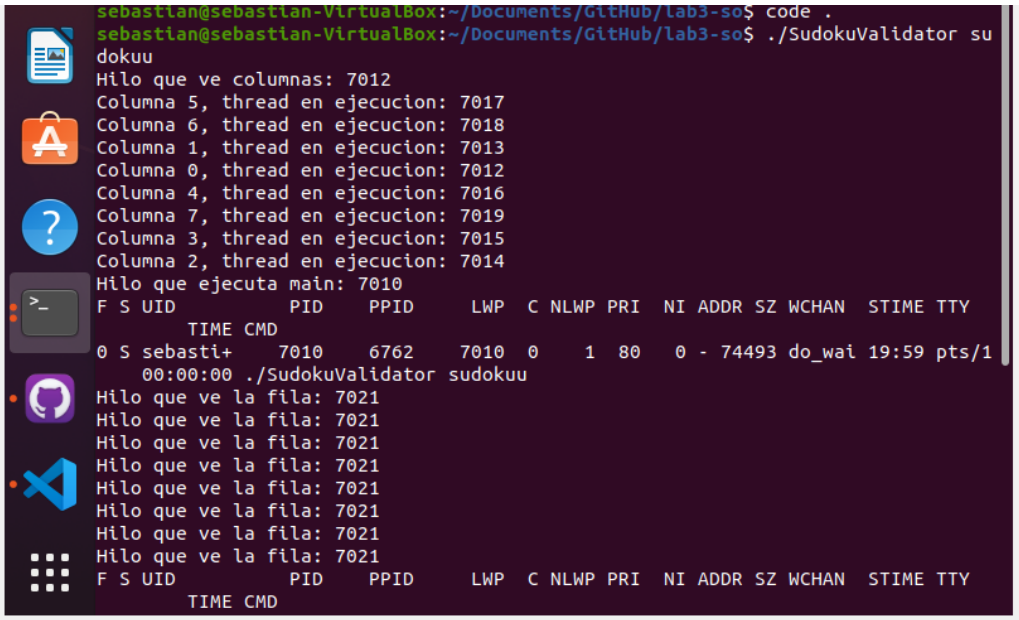
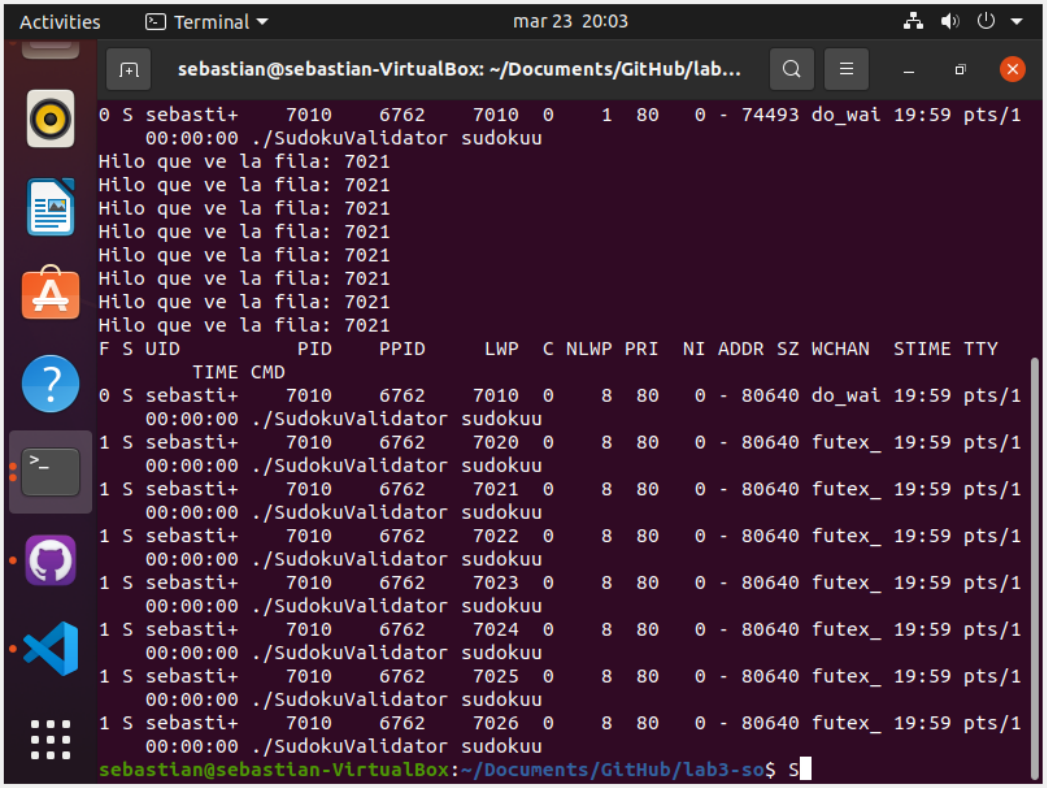
Donaldo Sebastian Garcia Jimenez 19683

Laboratorio 3- SO





1. ¿Qué es una race condition y por qué hay que evitarlas?
    a. Es que se estén ejecutando varios threads al mismo tiempo y que pasen por una función común entre ellos, se pierde el orden. Hay que evitarlos para que los datos dentro del scope siendo modificados den los valores esperados.
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?

   a. La fortaleza y la debilidad de la bifurcación (y compañía) es que crean un nuevo proceso que es un clon del proceso existente. Esta es una debilidad porque, como señaló, la creación de un nuevo proceso tiene una gran cantidad de gastos generales. También significa que la comunicación entre los procesos debe realizarse a través de algún canal "aprobado" (tuberías, sockets, archivos, región de memoria compartida, etc.)
3. ¿Dónde, en su programa, hay paralelización de tareas, y dónde de datos?
   a. Pasa cuando en paralelo se van checando las columnas y filas por un proceso distinto.
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.
   a. Se crea un proceso para cada columna, subcuadrado y fila por lo que habrían 9 para cada proceso.
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?
   a. Cuando esto sucede openmp va a alocar los procesos necesarios para satisfacer la cantidad de procesadores que posee la máquina virtual.
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.
   a. La primera columna es el nivel de jerarquía. Asimismo, la que se queda inactiva es la del padre dado a que se colocó un wait para esperar al proceso hijo.
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?
   a. Openmp nos permite crear grupos de procesos los cuales se ejecutarán paralelamente. Busy-wait se usa para asegurar que otro thread haya terminado su trabajo antes de que este realice alguna otra acción.
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?
   a. Sería 9 el número máximo de procesos trabajando por ser el número de columnas. El Schedule dynamic hace que cada proceso ejecute un bloque de información y cuando se acabe solicitará más, haciendo esto hasta que ya no quede memoria.
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

a. Al agregarlo se pudo ver una forma más agrupada de ejecutar los procesos y no se vio condición de carrera como al principio que había mayor concurrencia en los datos.

10. ¿Cuál es el efecto de agregar omp_set_nested(true)? Explique.
    a. Agrupa los procesos para que lleven un orden, cada uno cuenta con su memoria privada.