

|  |
| --- |
| MiniTP  Shell y Procesos  2023 |
|  |
|  |



Universidad de General Sarmiento

26 marzo

SOR – Sistemas Operativos y Redes (Com1)

Profesor/es: Lic. Mariano Vargas - TUI. Augusto Liali

Alumno: Guillermo Dominguez

# Sistemas Operativos y Redes

MiniTP – Shell y Procesos

Condiciones para Aprobar

Para la evaluación del presente trabajo, deben realizar los siguientes puntos:

* El trabajo es individual
* Enviar el trabajo en formato digital dentro de los plazos establecidos. Adjuntar el código fuente de su implementación y un informe del trabajo realizado punto por punto, dificultades encontradas y soluciones propuestas.

Puntaje / Calificación:

El presente trabajo se califica con las notas:

* I (insuficiente)
* A- (aprobado menos, no puede tener dos A- en la cursada),
* A (aprobado)
* A+ (aprobado más, redondea para arriba la nota final en caso de promocionar)

Recuperatorio:

En caso de no aprobar tiene un plazo de una semana para entregar el TP con las correcciones indicadas más ejercicios adicionales que se agregaran al enunciado. En recuperatorio no se pone A +.

Los objetivos de este trabajo son:

* Familiarizar al alumno con la línea de comandos.
* Familiarizar al alumno con las nociones de Proceso y Thread.

Ejercicio 1 – Shell y Terminal

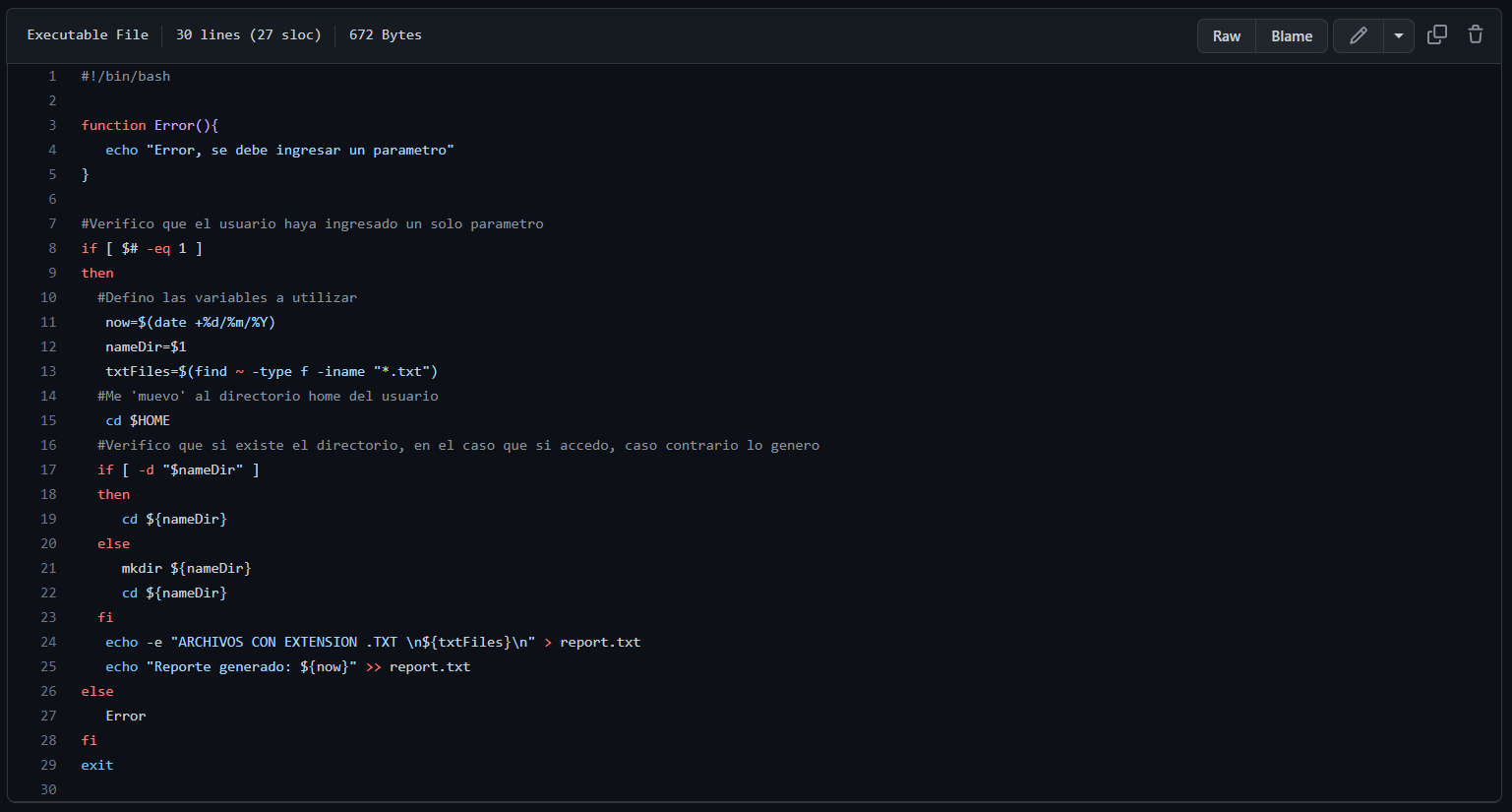
Realice un script de shell tal que realice las siguientes tareas:

* Debe recibir por parámetro una palabra y debe crear un directorio con dicho nombre en tu home de usuario.
* Dentro de ese directorio debe crear un archivo .txt
* Debe agregar al archivo anterior el listado de todos los archivos de la computadora que terminan con la extensión .txt, y además de los nombres de los archivos se tienen que ver los permisos de los mismos.
* Al final del archivo y del listado debe agregar la fecha y la hora del sistema
* Cuando el script termine debe mostrar por pantalla el contenido del archivo.

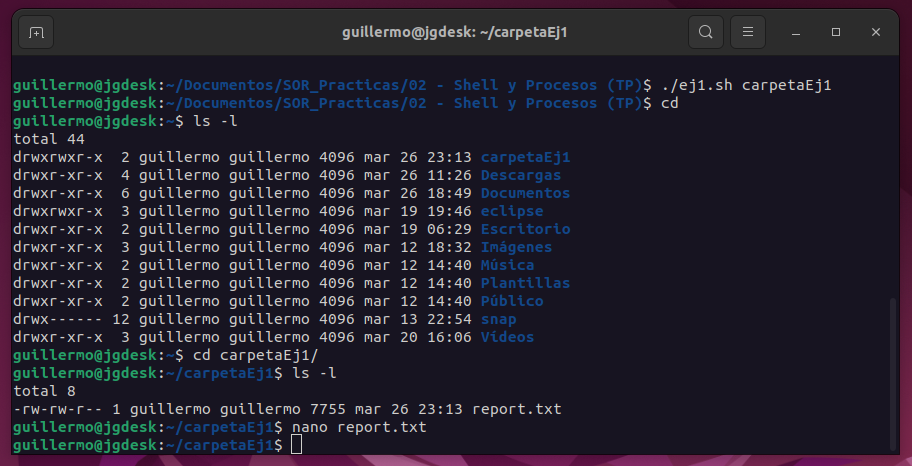
Puede usar los siguientes operadores y comandos:

* pipe | , redirección > , redirección concatenando >>, grep, ls y cat

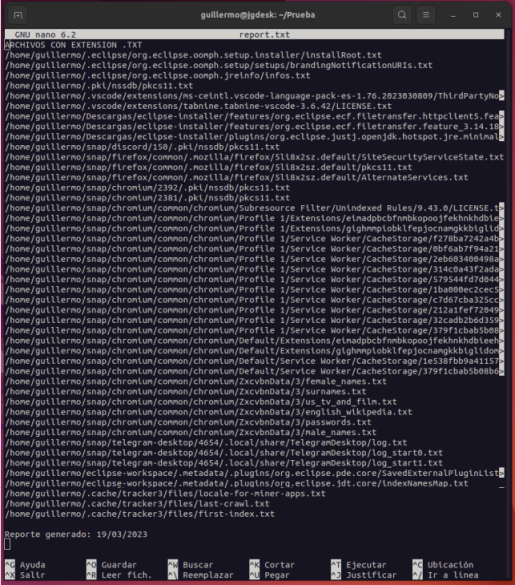
Resolucion:



En esta implementacion se cumplen con los puntos citados en la consigna y adicionalmente se aplican dos verificaciones adicionales. En primer lugar se controla que el usaurio ingrese un solo parametro y en segundo lugar se verifica que el directorio no exista.



Se evidencia la creacion del directorio (carpetaEj1) y generacion de archivo con la informacion solicitada en el enunciado (report.txt.).



Reporte final, en el mismo se envidencia que liste los archivos .txt en el sistema e imprime al final del archivo la fecha en la cual se genero el informe.

Quedo pendiente de este ejercicio, listar los permisos de los archivos *.txt*.

Ejercicio 2 – Estados de un Proceso

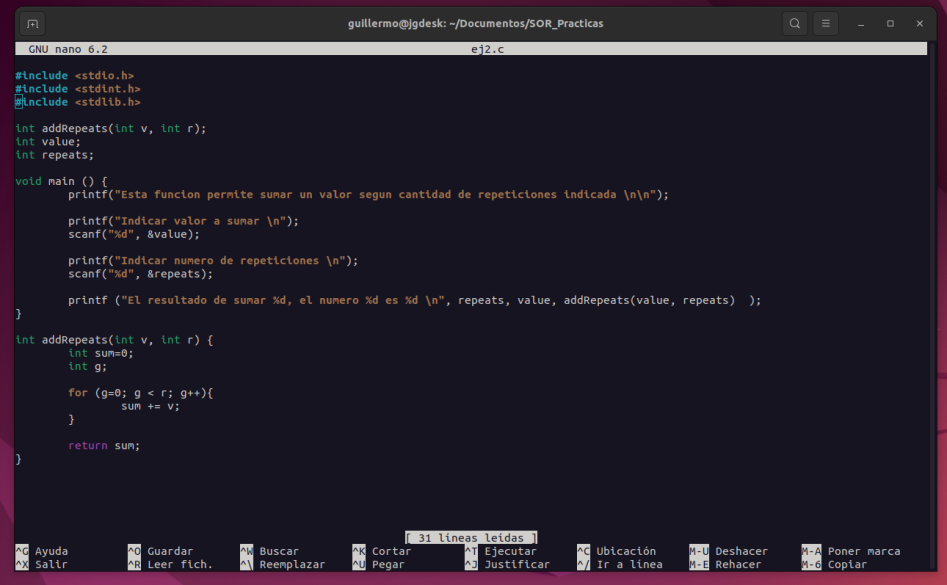
En esta parte vamos a aplicar nuestros conocimientos de procesos y sus estados.

* Realizar un programa en C compuesto de instrucciones que realizan cálculos (operaciones aritméticas) y operaciones de I/O (leer un input del usuario). Compilar, ejecutar su programa y visualizar los estados por los que pasa. Puede usar la herramienta htop.
* Ejecutar su programa y demostrar mediante capturas de pantalla del programa htop que su programa efectivamente cambia de estados.

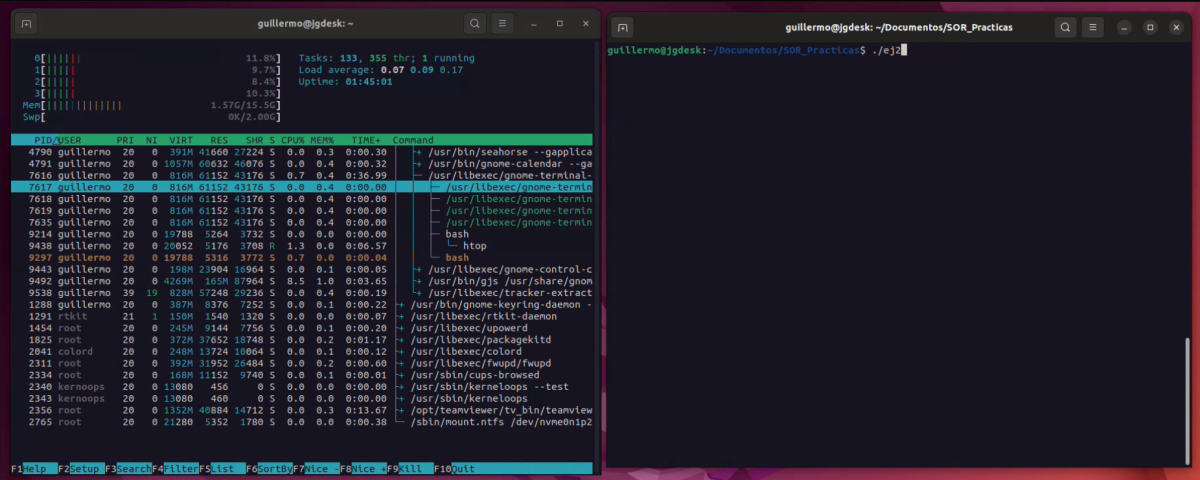
Resolucion:

Se genero un programa en C que solicita al usuario dos valores, con los cuales realiza la suma del primer valor de forma consecutiva segun la cantidad de repeticiones ingresadas como segundo parametro.

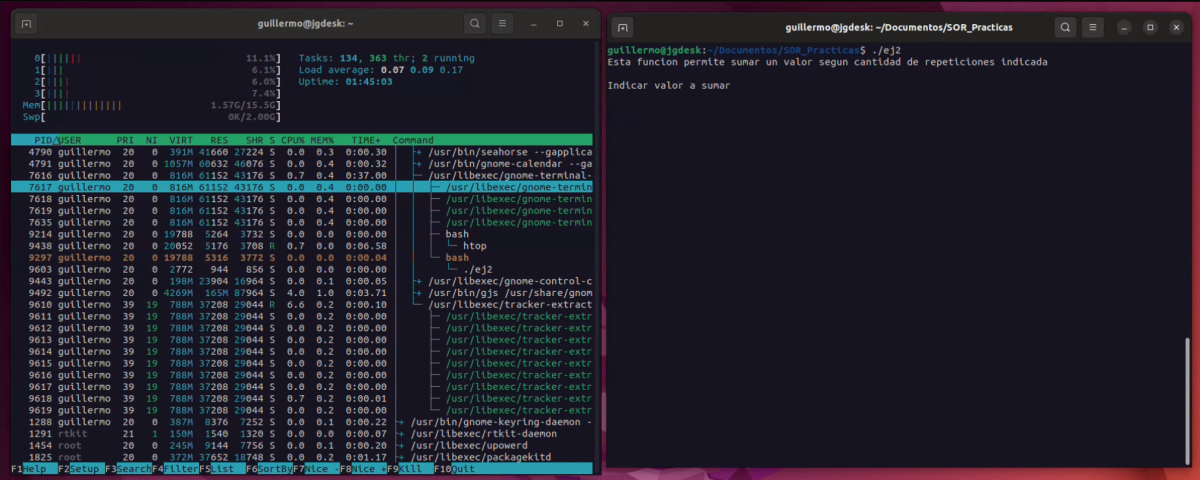
El programa recien mencionado cuenta con dos interrupciones (I/O).



A continuacion se puede apreciar el paso a paso durante la ejecicion del programa (terminal derecha) y el seguimiento de estados del programa *ej2* (resaltado color naranja) mediante el programa *htop* (terminal izquierda).

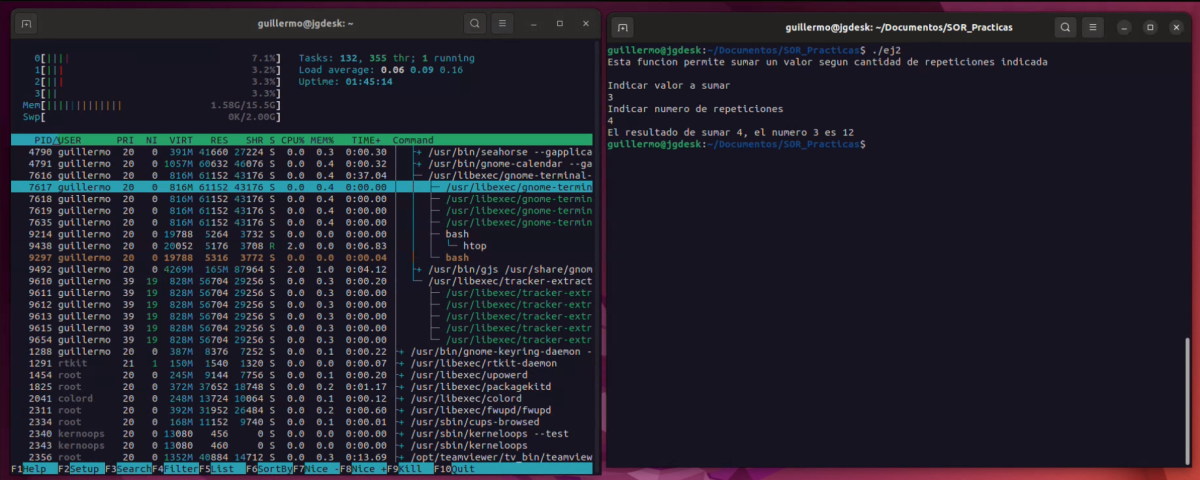


Previo a la ejecucion del programa ej2, nos centramos en el proceso bash (PID 9297).



Al ejecutar el programa ej2, solicita el ingreso de valores por parte del usuario con lo cual el proceso se encuentra en estado S (Sleeping).

Es imperceptible el cambio de estado R (Running) a S, al momento de lanzar el programa.



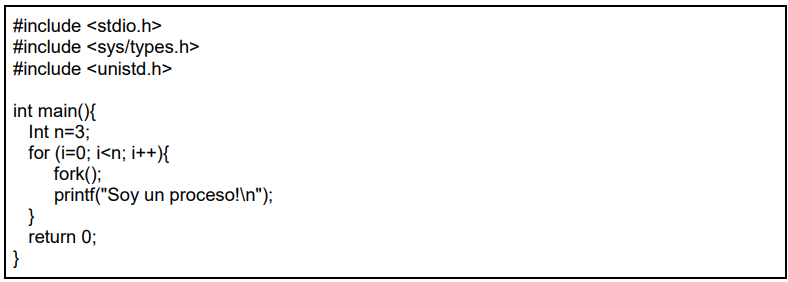
Al finalizar la ejecucion del programa ej2, se elimina el subproceso de Bash.

En resumen, es imperceptible el cambio de estado *R* (Running) a *S* (Sleeping) al momento de lanzar el programa, pero si se observa claramente que al momento de solicitar un imput por parte del usuario se cambia al estado *S*.

Ejercicio 3 – Procesos y Fork

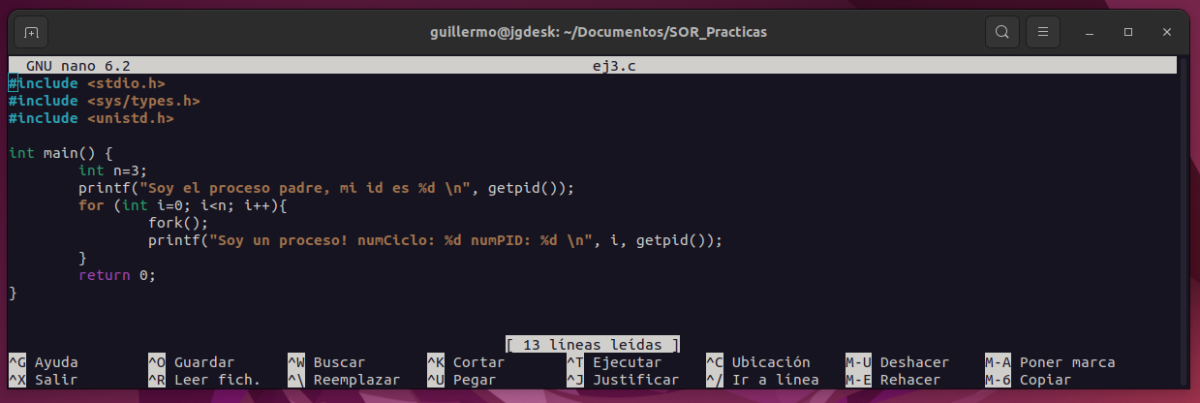
Fork es la llamada al sistema que permite crear nuevos procesos.

* Compilar, ejecutar el siguiente programa. Describir y justificar el output. Dibujar el árbol de proceso padre-hijo que se van generando para n=3.

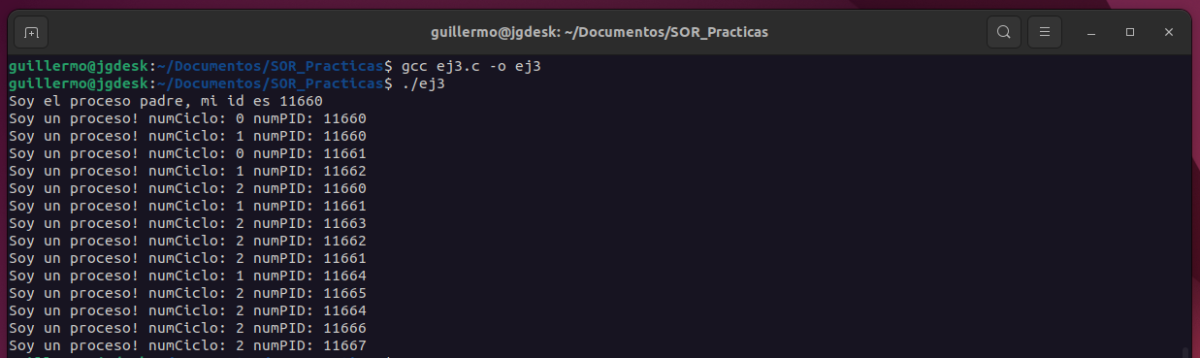


Resolucion:

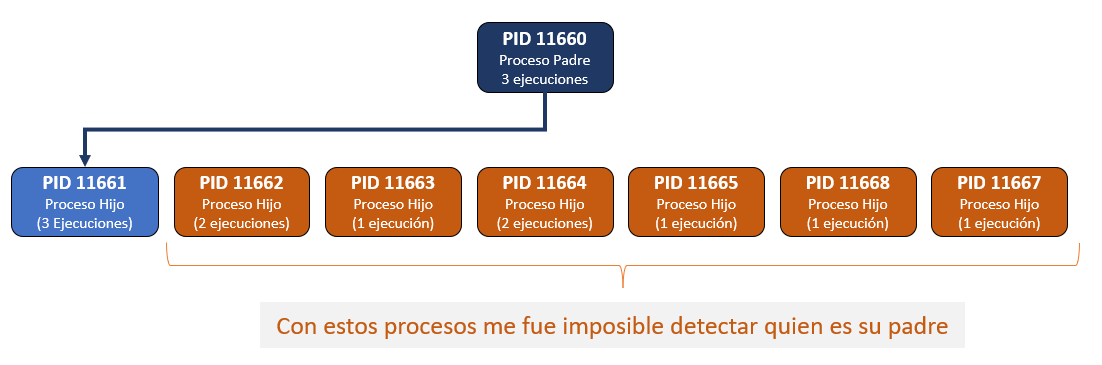
Al momento de ejecutar el programa imprimio en pantalla 12 veces *“Soy un proceso!”*, me fue imposible detectar en que proceso se originaba el mensaje. Teniendo presente el problema recien mencionado, modifique el mensaje para que me indique el ID del proceso padre, numero de ciclo y el proceso del cual se originaba el mensaje, se puede evidencia esto en las siguientes imágenes.



Programa con las modificaciones para identificar el proceso *padre* y el proceso que origina el mensaje.



Salida en pantalla



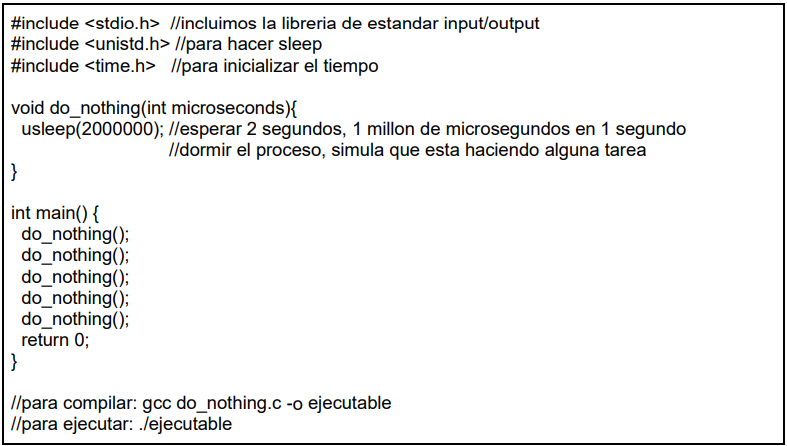
En este ejercicio pude evidenciar varios conceptos dictados durante la cursada:

* El cambio de proceso del scheduler.
* La importancia de aplicar tecnicas para mantener un “orden” en la ejecucion de los procesos.
* Los procesos conservan su estado al momento de ser retirados del CPU, esto se evidencia con el numero de ciclo (0, 1, 0 ,1 ,2 ,1 ,etc.).

Ejercicio 4 – Threads

El siguiente programa ejecuta la función do\_nothing() cinco veces. Esta función sólo espera 2

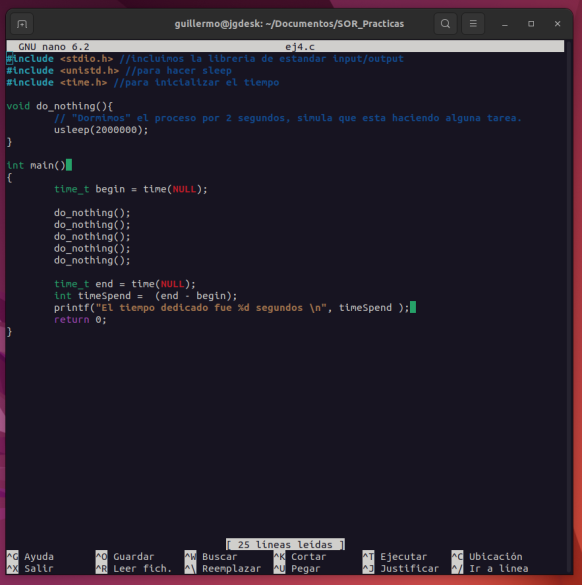
segundos y continúa:



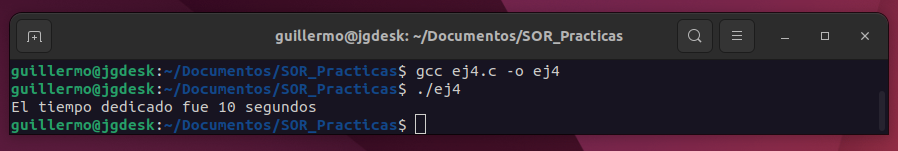
* Con la función *time*, medir el tiempo que tarda el programa anterior.
* Modificar el programa anterior para que cada una de las 5 llamadas a la función *do\_nothing()* se ejecute por un thread.
* Medir el tiempo que tarda su nuevo programa. Qué diferencias observa? Porque?

Resolucion:

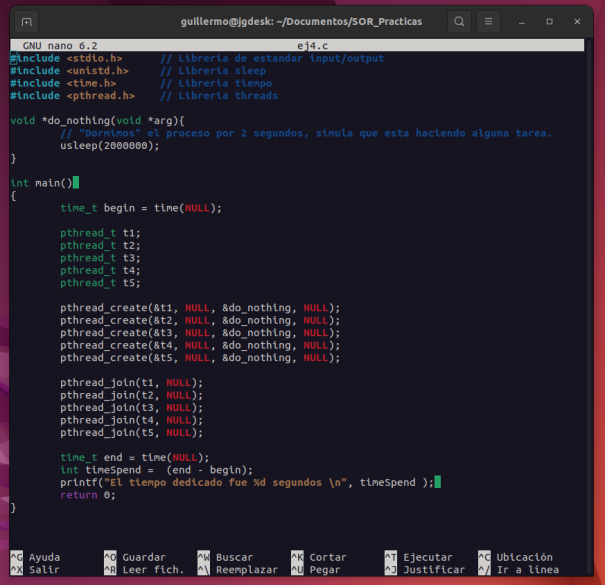
En primer lugar, se implemento la funcion *time* para medir el tiempo dedicado al ejecutar el programa.



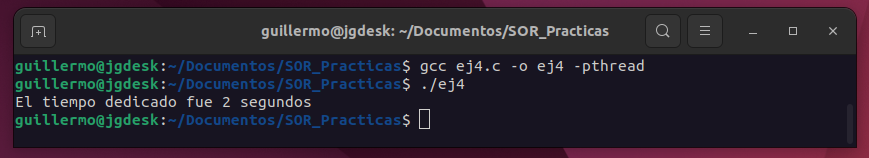
Implementacion de funcion *time* e impresión en pantalla de los segundos dedicados en la ejecucion.



Resultado de la ejecucion del programa ej4, con esta implementacion se dedicaron 10 segundos para ejecutar el programa completo.



Implementacion de *threads* para ejecutar la funcion *do\_nothing()*.



Como resultado de la ejecucion del programa *ej4* con la implementacion de *threads*, se logro ejecutar por completo en 2 segundos.

En este ejercicio se logra observar claramente la genialidad de “trabajar” con hilos!!

Se logro ejecutar el programa reduciendo el tiempo al minimo (2 segundos) dado que cada hilo adelantaba las instrucciones de dicho proceso en forma paralela y posteriormente el CPU solo dedicaba rafagas para ejecutar el resultado de cada hilo, con todo “pre-cocinado”.