Fallatti Franco

MiniTP de Shell y Procesos
Sistemas Operativos y Redes

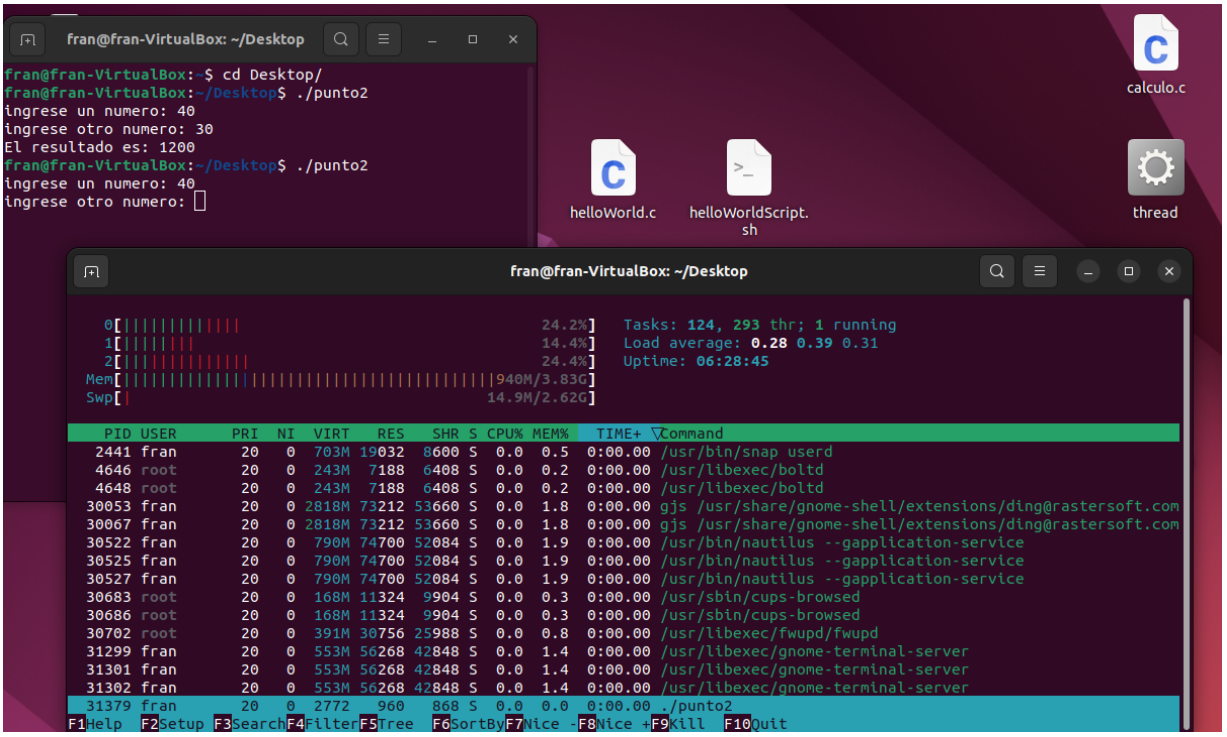
Ejercicio 1: Shell y terminal

En este punto se pide realizar un script de shell que reciba una palabra como parámetro, cree una carpeta en el directorio home con una lista de todos los archivos que terminan con extensión .txt de la computadora, junto a sus permisos y la fecha actual dentro de dicha carpeta. Al finalizar el script se muestra el contendido de la lista por consola

La principal dificultad en este ejercicio fue el uso de los comparadores en los if ya que no encontraba uno se que adecue a mi implementación pude resolverlo implementando `if [ -z "$folder" ];` el operador -z verifica si una cadena está vacía, si lo está devuelve true.
Otra dificultad fue el poder resolver como leer los archivos terminados en extensión .txt de la computadora `ls -lR / | grep '\.txt$' > "text.txt"` se lista todos los archivos y directorios del sistema de archivos a partir del directorio raíz (/). Y se utiliza un | (pipe) para redirigir la salida del comando hacia grep '\.txt$', que filtra por las líneas que contienen nombres de archivos que terminan con la extensión .txt. y haciendo uso de > se escriben en text.txt.

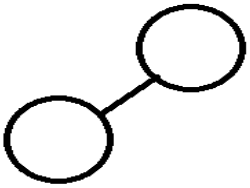
Ejercicio 2: Estados de un Proceso

En este punto se pide realizar un programa en C compuesto de instrucciones que realizan operaciones aritméticas y operaciones de I/O, para visualizar con htop como el programa cambia de estados.
La principal dificultad de este ejercicio fue el uso de NINA con htop, ya que no logré poder ejecutar htop y ver los cambios. Para resolverlo tuve que utilizar VirtualBox con Ubuntu y correr dos terminales a la vez, ejecutando htop mientras el programa esperaba la segunda entrada por teclado.



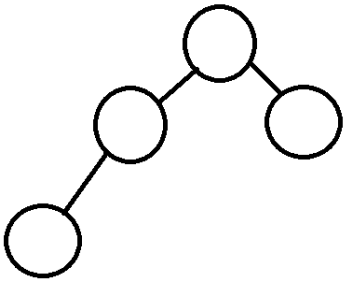
Ejercicio 3: Procesos y Fork

El programa crea un proceso principal (padre) y luego entra en un bucle que se ejecuta tres veces por n=3. En cada iteración, se llama a fork(), lo que da como resultado la creación de un nuevo proceso hijo. La llamada a fork() duplica el proceso en dos procesos independientes y cada uno de ellos continúa la ejecución desde el punto en que se llamó a fork().



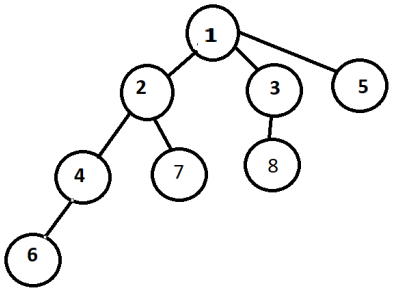
Iteración 1:

El proceso padre crea un nuevo proceso hijo mediante fork() y ambos procesos imprimen “Soy un proceso!”.



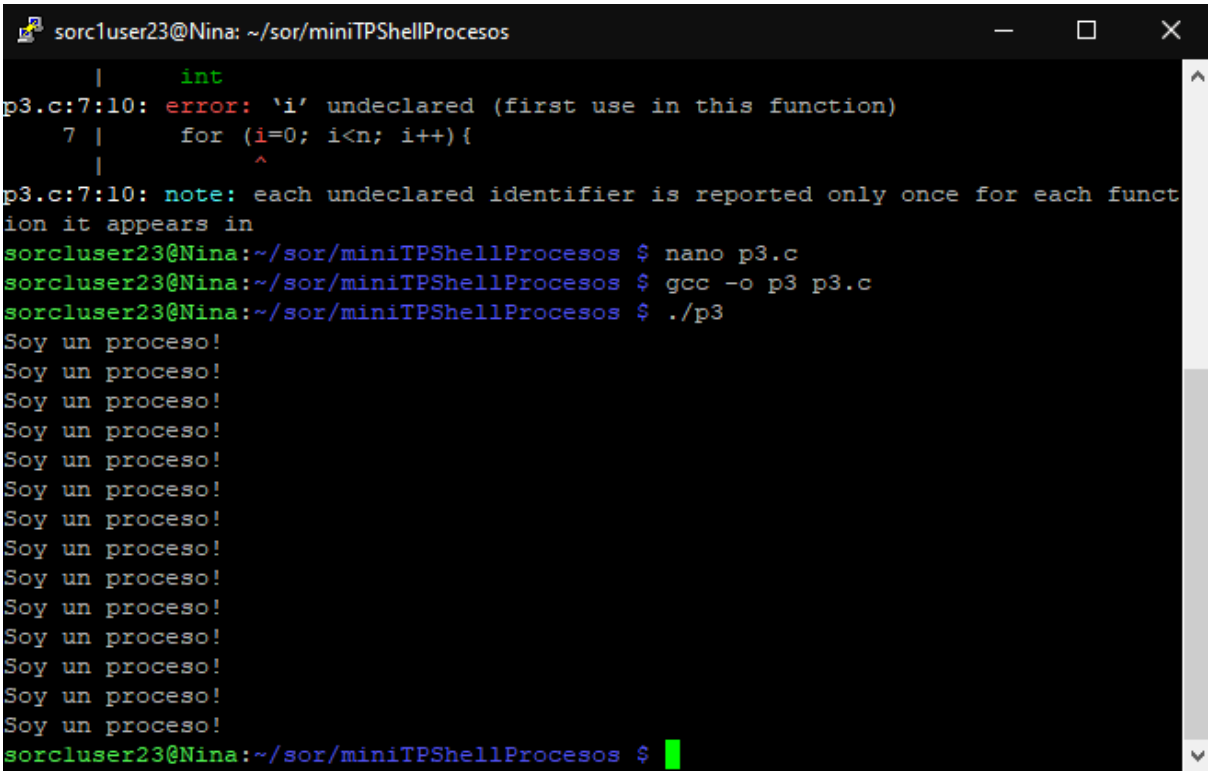
Iteración 2:

Cada uno de los procesos crea otro proceso hijo
y los cuatro procesos imprimen "Soy un proceso!".



Iteración 3:

Cada uno de los cuatro procesos crea otro proceso hijo. Los ocho procesos imprimen "Soy un proceso!".



Ejercicio 4: Threads

En este punto se pide realizar la comparación de tiempo en la ejecución de una función 5 veces, ejecutándola una atrás de otra en un programa y mediante hilos en otro. Se observa una gran diferencia en el tiempo de ejecución de ambos programas. Ya que, cuando se utilizan hilos en un programa, estos pueden ejecutarse en paralelo o en concurrencia, dependiendo del hardware. Esto significa que varios hilos pueden ejecutar sus rutinas al mismo tiempo o de manera intercalada, reduciendo el tiempo de ejecución significativamente.