1. Ejercicio 1: Hello World, Programa2

Graphical user interface, text

Description automatically generated

A picture containing logo

Description automatically generated

Programa 2

Text

Description automatically generated

Text

Description automatically generated

**Compile el primer programa y ejecútelo varias veces. Responda: ¿Por qué aparecen números diferentes cada vez?**

Text

Description automatically generated

Al compilar el primer programa y ejecutarlo varias veces aparecen números diferentes porque la función getpid() regresa el ID de el proceso actual. Como se ejecuta varias veces son distintos procesos por lo tanto el ID cambia.

**Proceda a compilar el segundo programa y ejecútelo una vez. ¿Por qué aparecen dos números distintos a pesar de que estamos ejecutando un único programa?**

Text

Description automatically generated

Al compilar el segundo programa aparecen dos números distintos ya que dentro de este programa hay un proceso hijo (child) y un proceso padre (parent).

**¿Por qué el primer y el segundo números son iguales?**

El primer y segundo numero son iguales porque se encuentran dentro del mismo proceso, en este caso el child process, esto hace que tengan el mismo getpid(), 1981. Para el proceso parent los dos números son iguales porque están dentro del mismo proceso, 1982.

**En la terminal, ejecute el comando top (que despliega el top de procesos en cuanto a consumo de CPU) y note cuál es el primer proceso en la lista (con identificador 1). ¿Para qué sirve este proceso?**

**Table

Description automatically generated**

El comando top sirve para saber los procesos que están sucediendo en el sistema a tiempo real. Esta muestra una lista de los procesos o threads que están siendo manejados por el kernel además de un pequeño resumen de la información del sistema. El primer proceso es por el usuario root y con el comando init.

2. Ejercicio 2

1. **Investigue acerca de las llamadas a sistema open (), close (), read () y write ().**

open (): Esta llamada a el sistema abre el archivo especificado por el nombre del path. Si el archivo no existe puede ser que sea creado por la llamada a el sistema de open ().

close (): Esta llamada a el sistema cierra un descriptor de archivo. Con esta llamada al sistema ya no se hacer referencia a ningún archivo.

read (): La llamada a el sistema read () lee los datos que fueron escritos en un archivo.

write (): Esta llamada sirve para escribir una llamada a el sistema que se utiliza para escribir datos en un búfer.

1. **Escriba un programa en C que reciba dos argumentos desde terminal (usando int argc, char\* argv[]como parámetros en main). Su programa debe abrir el archivo ubicado en el directorio provisto como primer parámetro, y debe copiar todo su contenido a otro archivo ubicado en el directorio provisto como segundo parámetro.**

**Text

Description automatically generated**

1. **Instale la herramienta strace usando el siguiente comando en la terminal:**

Text, letter

Description automatically generated

1. Compile su programa y pruébelo (no olvide crear un archivo qué copiar). Luego, ejecute el siguiente comando en terminal (desde la ubicación donde esté el código objeto resultante de la compilación):
2. ..