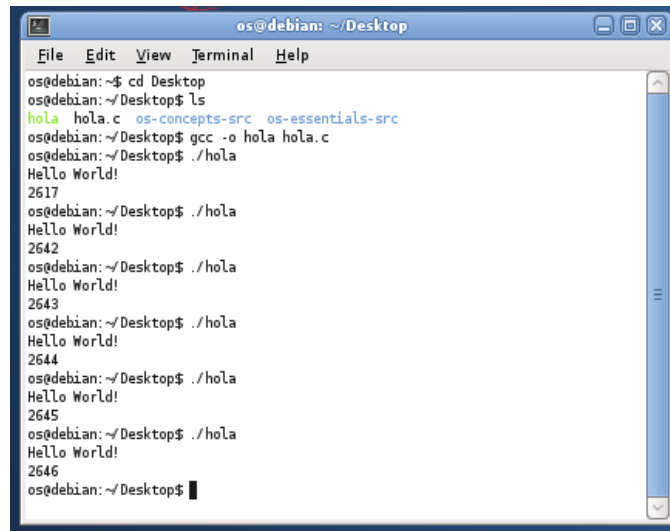


# LABORATORIO 1

## EJERCICIO UNO

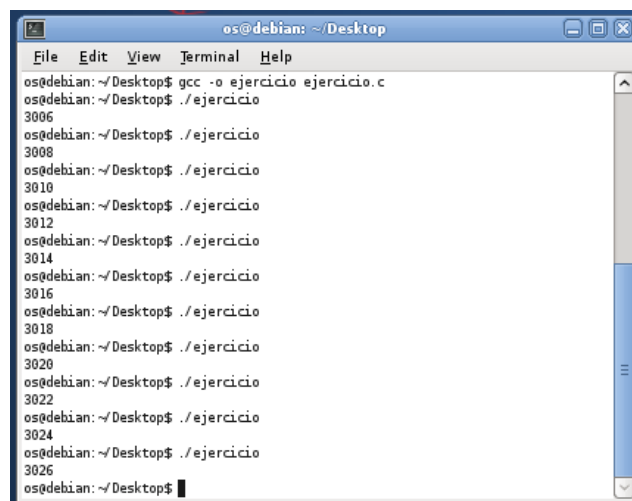
- Compile el primer programa y ejecútelo varias veces. Responda: ¿por qué aparecen números diferentes cada vez?



```
os@debian: ~/Desktop
File Edit View Terminal Help
os@debian:~$ cd Desktop
os@debian:~/Desktop$ ls
hola  hola.c  os-concepts-src  os-essentials-src
os@debian:~/Desktop$ gcc -o hola hola.c
os@debian:~/Desktop$ ./hola
Hello World!
2617
os@debian:~/Desktop$ ./hola
Hello World!
2642
os@debian:~/Desktop$ ./hola
Hello World!
2643
os@debian:~/Desktop$ ./hola
Hello World!
2644
os@debian:~/Desktop$ ./hola
Hello World!
2645
os@debian:~/Desktop$ ./hola
Hello World!
2646
os@debian:~/Desktop$
```

Aparecen números diferentes cada vez porque en cada corrida se le asigna un pid distinto.

- 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?



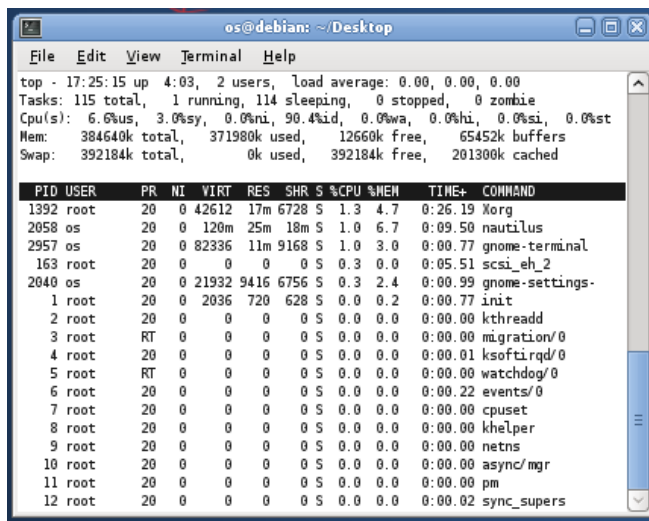
```
os@debian: ~/Desktop
File Edit View Terminal Help
os@debian:~/Desktop$ gcc -o ejercicio ejercicio.c
os@debian:~/Desktop$ ./ejercicio
3006
os@debian:~/Desktop$ ./ejercicio
3008
os@debian:~/Desktop$ ./ejercicio
3010
os@debian:~/Desktop$ ./ejercicio
3012
os@debian:~/Desktop$ ./ejercicio
3014
os@debian:~/Desktop$ ./ejercicio
3016
os@debian:~/Desktop$ ./ejercicio
3018
os@debian:~/Desktop$ ./ejercicio
3020
os@debian:~/Desktop$ ./ejercicio
3022
os@debian:~/Desktop$ ./ejercicio
3024
os@debian:~/Desktop$ ./ejercicio
3026
os@debian:~/Desktop$
```

Aparecen dos números distintos porque tienen un pid distinto.

- ¿Por qué el primer y el segundo número son iguales?

Son iguales porque se refieren al numero de proceso padre.

- 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?



```
os@debian: ~/Desktop
File Edit View Terminal Help
top - 17:25:15 up 4:03, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 115 total, 1 running, 114 sleeping, 0 stopped, 0 zombie
Cpu(s): 6.6%us, 3.0%sy, 0.0%ni, 90.4%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 384640k total, 371980k used, 12660k free, 65452k buffers
Swap: 392184k total, 0k used, 392184k free, 201300k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 1392 root        20   0  42612 17m  6728  S  1.3   4.7   0:26.19 Xorg
2058 os          20   0  120m  25m  18m  S  1.0   6.7   0:09.50 nautilus
2957 os          20   0  82336 11m  9168  S  1.0   3.0   0:00.77 gnome-terminal
163  root        20   0    0    0    0  S  0.3   0.0   0:05.51 scsi_eh_2
2040 os          20   0 21932 9416  6756  S  0.3   2.4   0:00.99 gnome-settings-
 1  root        20   0  2036   720   628  S  0.0   0.2   0:00.77 init
 2  root        20   0    0    0    0  S  0.0   0.0   0:00.00 kthreadd
 3  root        RT   0    0    0    0  S  0.0   0.0   0:00.00 migration/0
 4  root        20   0    0    0    0  S  0.0   0.0   0:00.01 ksoftirqd/0
 5  root        RT   0    0    0    0  S  0.0   0.0   0:00.00 watchdog/0
 6  root        20   0    0    0    0  S  0.0   0.0   0:00.22 events/0
 7  root        20   0    0    0    0  S  0.0   0.0   0:00.00 cpuset
 8  root        20   0    0    0    0  S  0.0   0.0   0:00.00 khelper
 9  root        20   0    0    0    0  S  0.0   0.0   0:00.00 netns
10  root        20   0    0    0    0  S  0.0   0.0   0:00.00 async/mgr
11  root        20   0    0    0    0  S  0.0   0.0   0:00.00 pm
12  root        20   0    0    0    0  S  0.0   0.0   0:00.02 sync_supers
```

Es el primer proceso ejecutado por el sistema.

## EJERCICIO DOS

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

**Open():** abre el archivo que se le especifica, si este no existe, tiene la posibilidad de crearlo.

**Close():** cierra un descriptor de archivo, con el fin de que el archivo no pueda ser reutilizado.

**Read():** lee la información de un archivo especificado, con el objetivo de almacenarlo en distintas variables, que se guardan en un buffer.

**Write():** escribe los bytes del buffer dentro de un archivo asociado con el descriptor de archivos.



1. `read(3 ...)` lee los datos de un archivo previamente abierto.
2. `close`, es una llamada al sistema para cerrar el archivo.
3. `open("/etc/ld.so.cache", O_RDONLY) = 3`, abre el archivo `ld.so.cache` en modo lectura.