Emanuel Esquivel Lopez 2016133597
Taller 3
Principios de sistemas operativos

Preguntas guia

1. Explique las etapas de creación de un proceso en Windows (CreateProcess).

   Para la creación de un proceso se puede mediante la siguiente funcion.

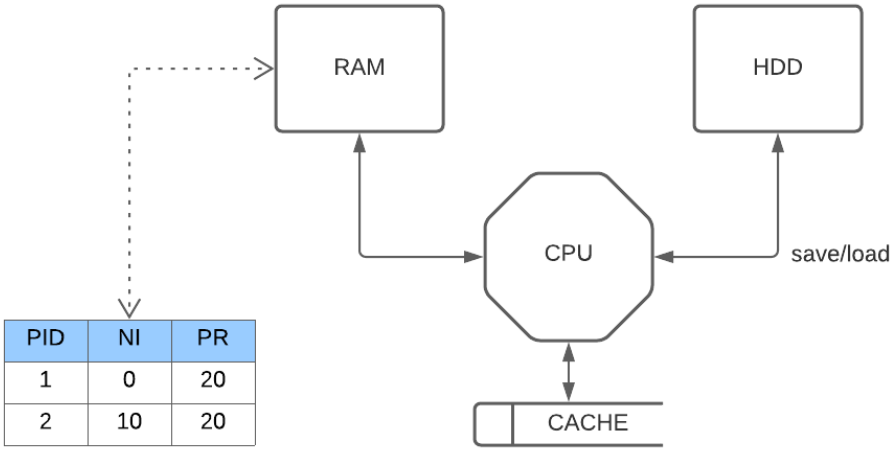
```
BOOL CreateProcessA(
     LPCSTR                lpApplicationName,
     LPSTR                 lpCommandLine,
     LPSECURITY_ATTRIBUTES lpProcessAttributes,
     LPSECURITY_ATTRIBUTES lpThreadAttributes,
     BOOL                  bInheritHandles,
     DWORD                 dwCreationFlags,
     LPVOID                lpEnvironment,
     LPCSTR                lpCurrentDirectory,
     LPSTARTUPINFOA        lpStartupInfo,
     LPPROCESS_INFORMATION lpProcessInformation
);
```

   Esto en el lenguaje c++, se puede ver que tenemos distintos parámetros, como lo son `lpApplicationName` el cual es el nombre del módulo a ejecutar, además de esto `lpCommandLine` es el comando a ejecutar para dicha aplicación, También tenemos atributos los cuales son necesarios para darles prioridades al sistema sobre este dicho proceso, además currentDirectory y Enviroment nos da información sobre en donde se ejecutará, así como StartUpInfo el puntero que guarda la información de inicio, así como también ProccessInfo nos da la información relacionada a este para su identificación, semejante al PID.

2. ¿Cuáles son las variables necesarias que se deben de guardar cuando se quiere implementar un cambio de contexto?

   Se necesita PID tanto del proceso actual como el que se quiere ejecutar.
   Se necesita PID del proceso a ejecutar.
   Prioridades de los procesos.
   Guardado de proceso 1 y cargado del proceso 2.

3. ¿Cómo se podría implementar un cambio de contexto por hardware y no por software? Realice un esquema de arquitectura con su propuesta.



Una posible solución por hardware podría ser la siguiente, tenemos los procesos en RAM, una vez se da un cambio de contexto debe ser notificado al CPU, para que este se encargue de guardar el estado actual del proceso 1, en HDD o en la misma RAM, tambien cargue el proceso 2 y cargue su estado anterior.

4. Para qué sirve el comando ps y top en un entorno de Linux.

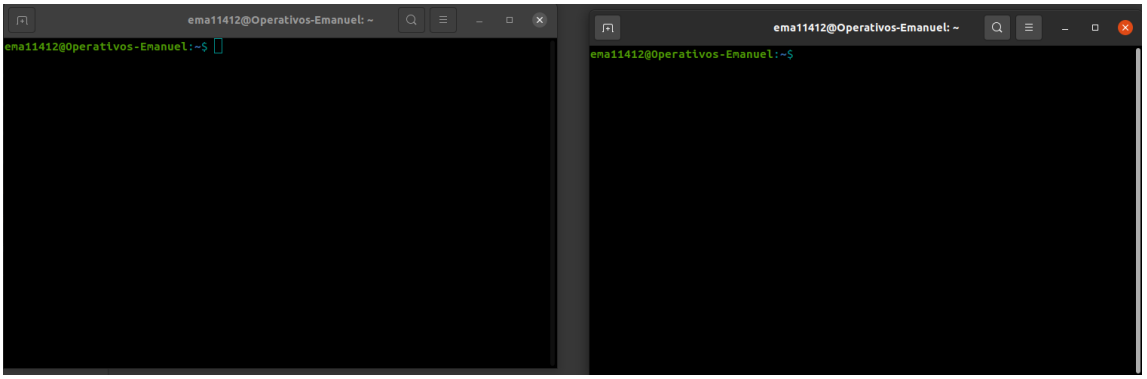
   Estos comandos nos muestran los procesos que están activos en el sistema, el comando top es mas preciso ya que este se actualiza en tiempo real mientras que ps se invoca y retorna en ese instante los valores.

5. Investigue los posibles estados de un proceso en un entorno de Linux y como se representan.

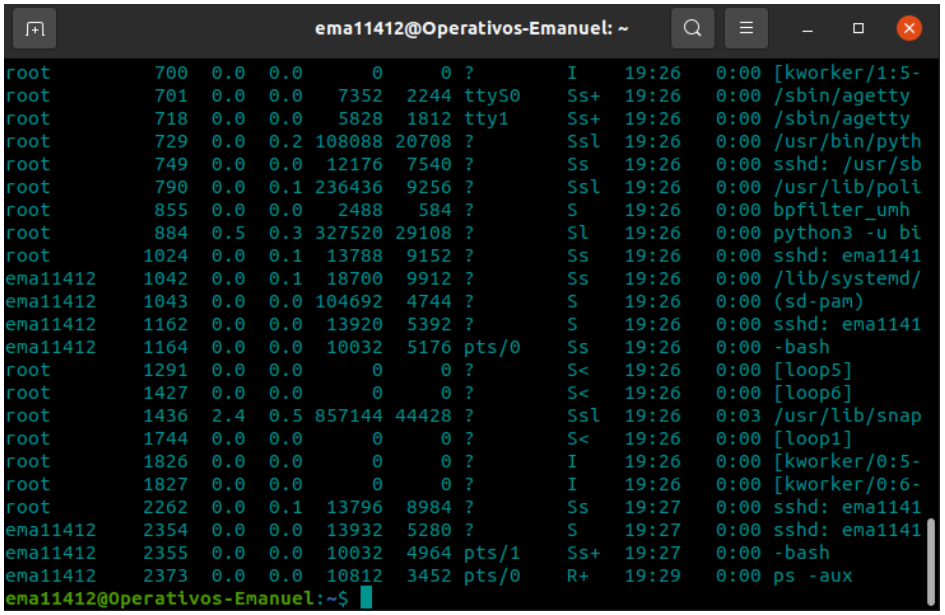
   a. Running: El proceso está en ejecución.
   b. Sleeping: Procesos en reposos esperando a que sean llamados.
   c. Stopped: Procesos detenidos
   d. Zombie: Procesos sin ser ejecutados.

Procesos en Linux

1. Conexión a la máquina virtual



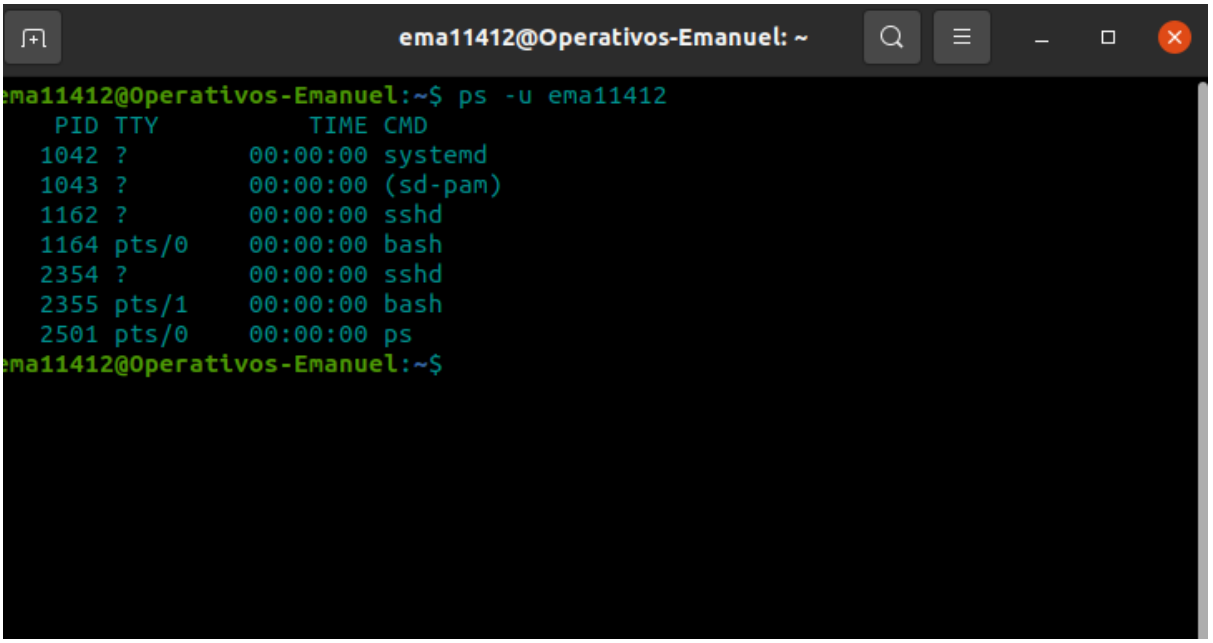
2. Utilizando el comando `ps -aux`

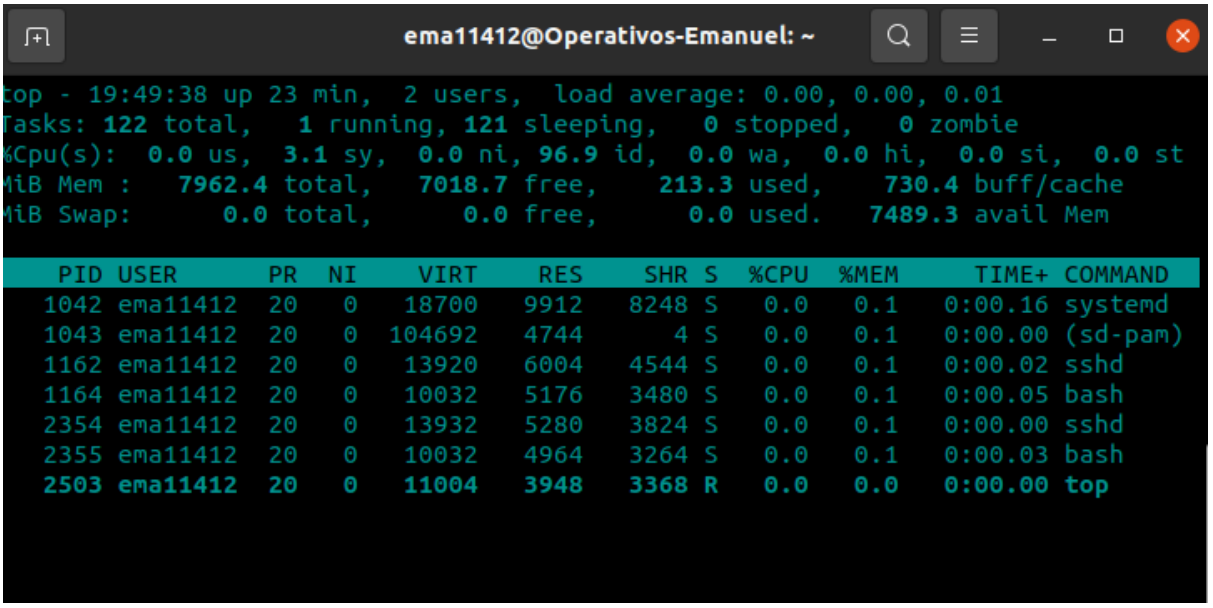


3. El significado se puede ver a continuación:
    a. PID: Es el identificador del proceso.
    b. RSS: Cantidad de memoria asignada al proceso en ese momento, no incluye memoria de intercambio.
    c. VSZ: Es la cantidad de memoria asignada virtual, esta incluye toda la memoria que el proceso puede acceder.

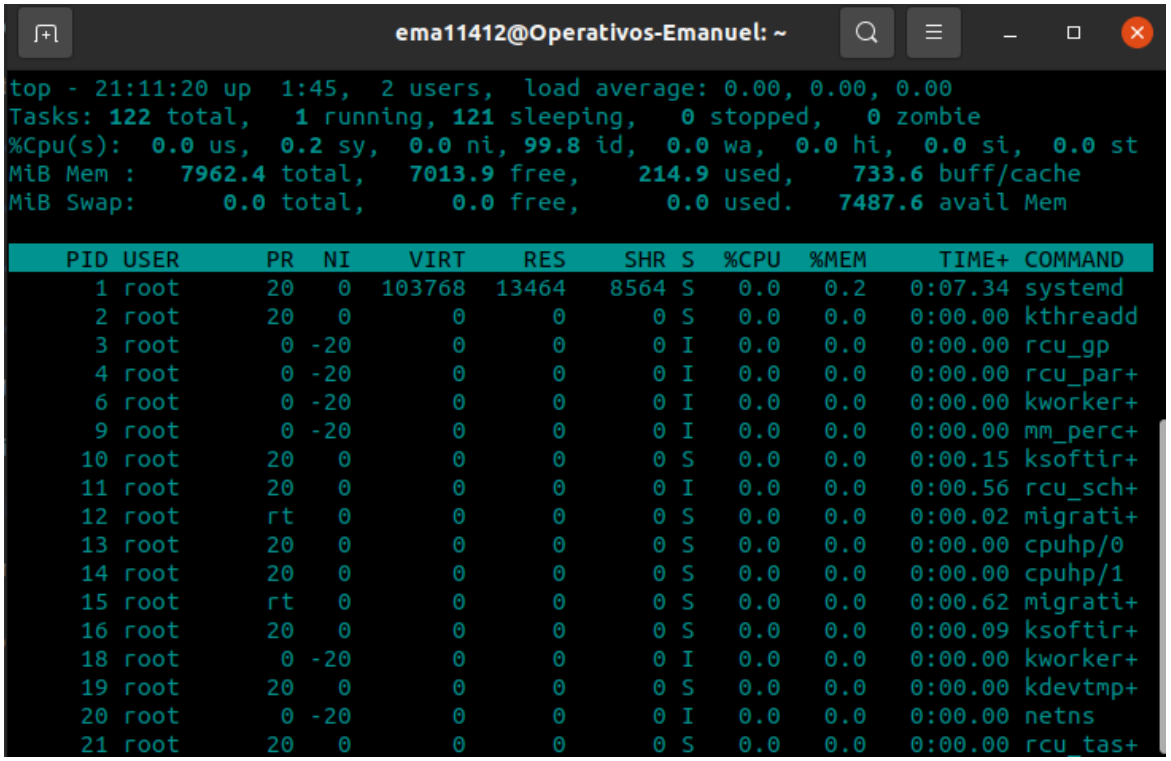
    Los demás se pueden entender con facilidad ya que son nombres explícitos.

4. Se puede ver mediante el comando `top -u user`, o bien `ps -u user`

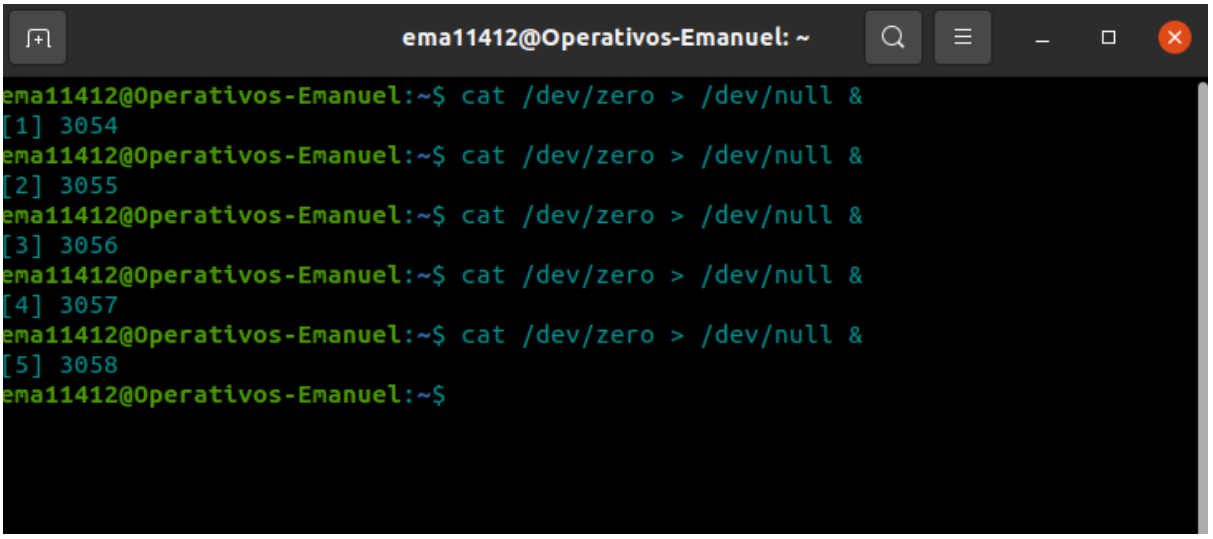




5. El comando `top` nos da el número de procesos activos y resumen del sistema.

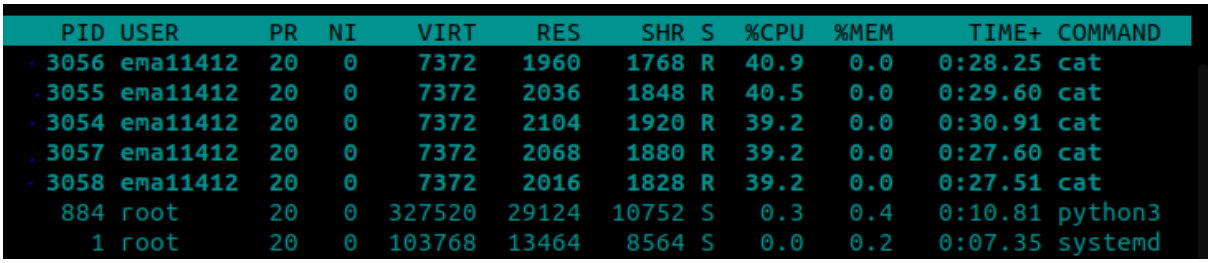


6. Al ejecutar el comando tenemos `cat /dev/zero > /dev/null &`



Consola 2.

7. Al ejecutar el comando `top` nuevamente tenemos los procesos ejecutados en la consola 2 presentes en la primera ya que estas están actuando sobre la misma máquina y se puede llevar un control síncrono.



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3056	ema11412	20	0	7372	1960	1768	R	40.9	0.0	0:28.25	cat
3055	ema11412	20	0	7372	2036	1848	R	40.5	0.0	0:29.60	cat
3054	ema11412	20	0	7372	2104	1920	R	39.2	0.0	0:30.91	cat
3057	ema11412	20	0	7372	2068	1880	R	39.2	0.0	0:27.60	cat
3058	ema11412	20	0	7372	2016	1828	R	39.2	0.0	0:27.51	cat
884	root	20	0	327520	29124	10752	S	0.3	0.4	0:10.81	python3
1	root	20	0	103768	13464	8564	S	0.0	0.2	0:07.35	systemd

Consola 1.

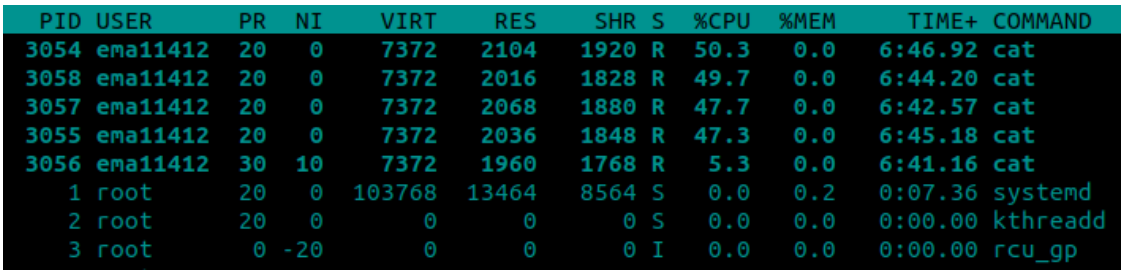
8. Tenemos lo siguiente:
    a. NI: Tiempo agradable del usuario, se asigna prioridad.
    b. PR: Prioridad del proceso.
    c. VIRT: cantidad de memoria virtual utilizada por el proceso.
    d. RES: cantidad de memoria RAM física que utiliza el proceso.
    e. SHR: memoria compartida.

9. Todos tienen prioridad similar ya que fueron ejecutados por el mismo usuario de manera consecutiva, además de que fueron el mismo comando.

10. Porque es el tiempo total que ha usado el CPU este proceso desde su ejecución.

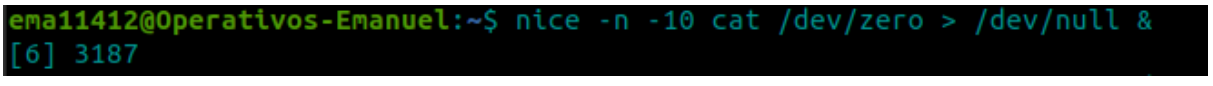
11. Puede ser `3056`.

12. Se aumenta la prioridad y se obtiene



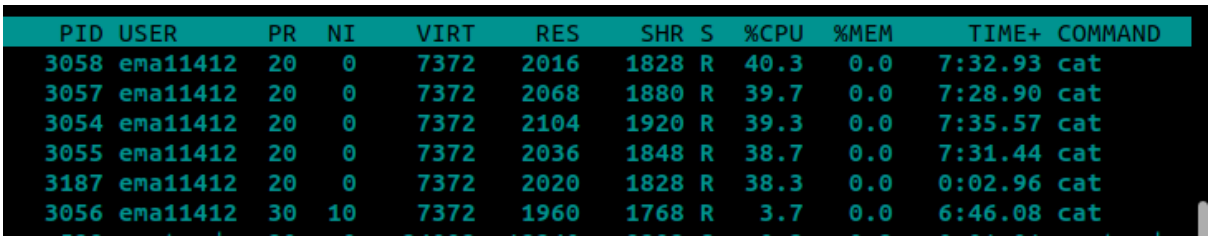
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3054	ema11412	20	0	7372	2104	1920	R	50.3	0.0	6:46.92	cat
3058	ema11412	20	0	7372	2016	1828	R	49.7	0.0	6:44.20	cat
3057	ema11412	20	0	7372	2068	1880	R	47.7	0.0	6:42.57	cat
3055	ema11412	20	0	7372	2036	1848	R	47.3	0.0	6:45.18	cat
3056	ema11412	30	10	7372	1960	1768	R	5.3	0.0	6:41.16	cat
1	root	20	0	103768	13464	8564	S	0.0	0.2	0:07.36	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp

13. Se realiza la ejecución del comando
`nice -n -10 cat /dev/zero > /dev/null &`



```
ema11412@Operativos-Emanuel:~$ nice -n -10 cat /dev/zero > /dev/null &
[6] 3187
```

14. Tenemos en `top`



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3058	ema11412	20	0	7372	2016	1828	R	40.3	0.0	7:32.93	cat
3057	ema11412	20	0	7372	2068	1880	R	39.7	0.0	7:28.90	cat
3054	ema11412	20	0	7372	2104	1920	R	39.3	0.0	7:35.57	cat
3055	ema11412	20	0	7372	2036	1848	R	38.7	0.0	7:31.44	cat
3187	ema11412	20	0	7372	2020	1828	R	38.3	0.0	0:02.96	cat
3056	ema11412	30	10	7372	1960	1768	R	3.7	0.0	6:46.08	cat

Creación de procesos con parámetros establecidos por el usuario

Tabla y grafica para C, para n = 999.

C	
NI	Tiempo (s)
20	0,515
10	0,112
0	0,078
-10	0,067
-20	0,058

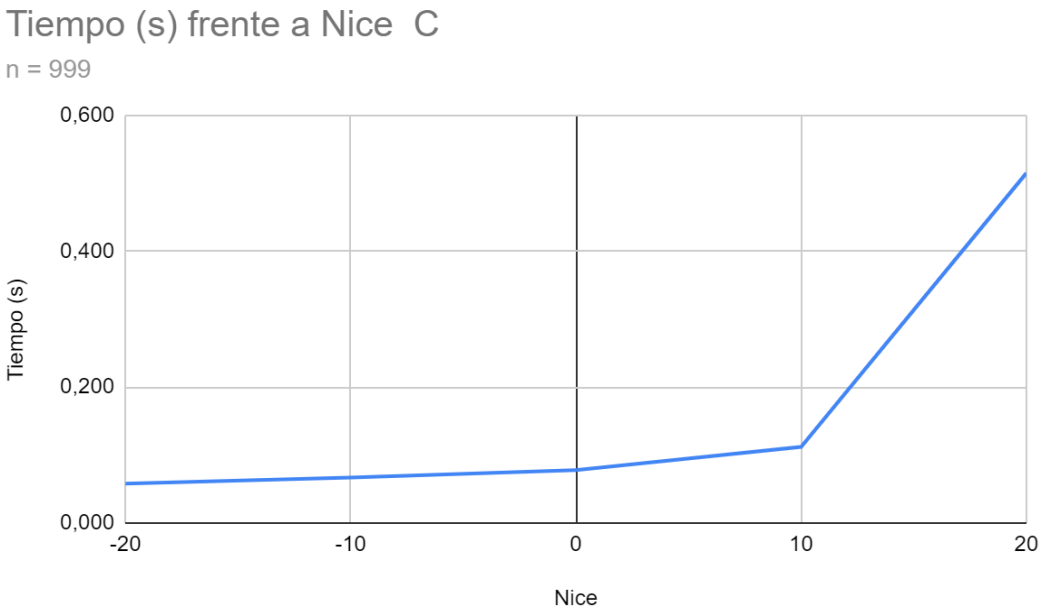
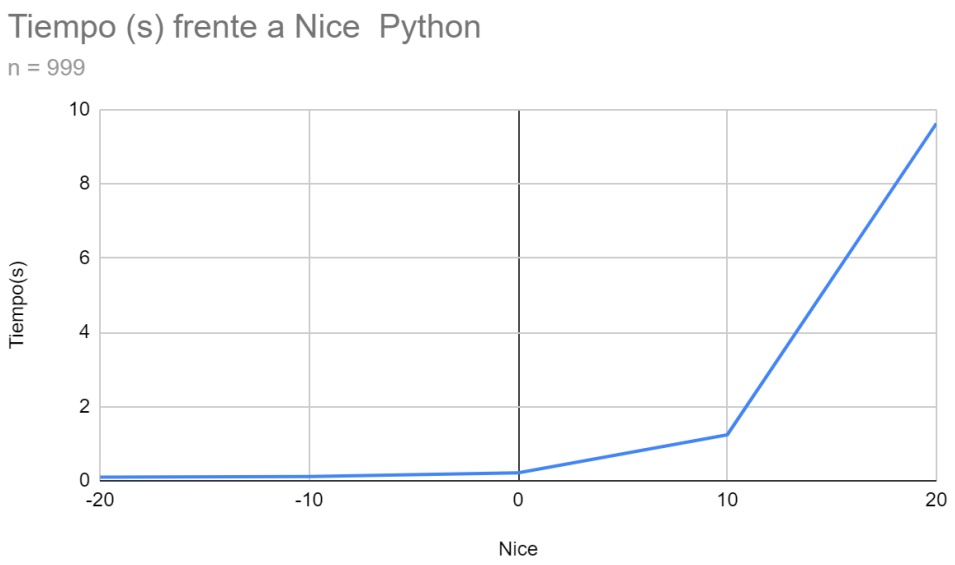


Tabla y gráfica para Python, para n = 999

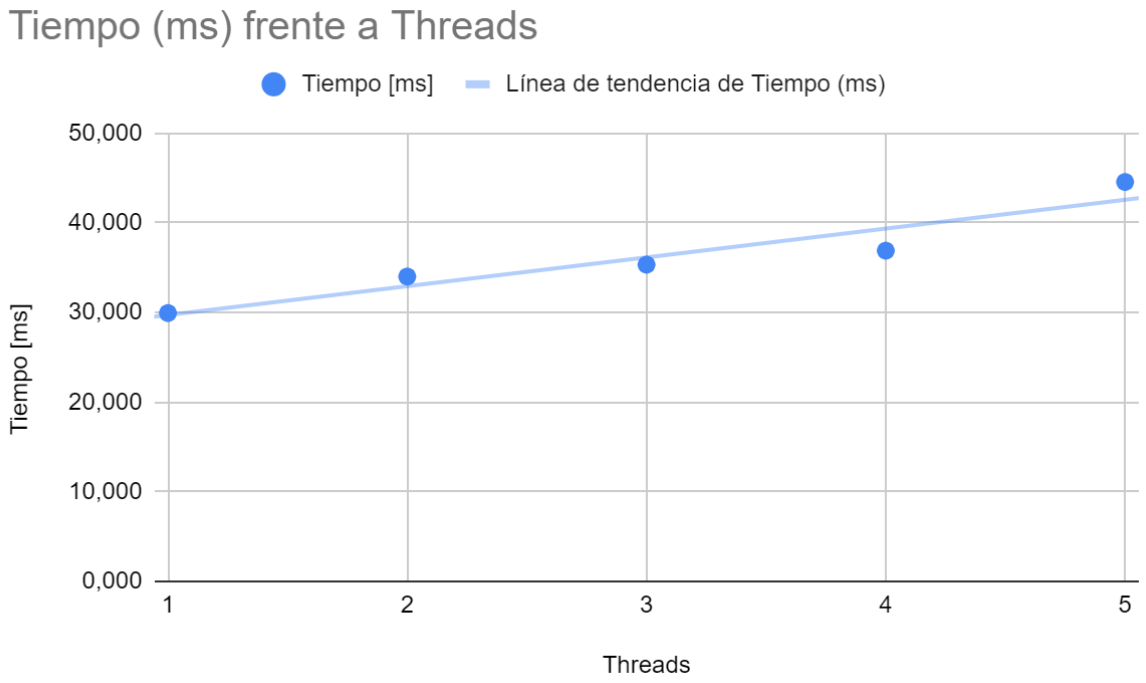
Python	
Ni	Tiempo (s)
20	9,63
10	1,240
0	0,215
-10	0,113
-20	0,1



Hilos en Linux

Tabla asociada

Threads	Tiempo [ms]
1	29,935
2	34,010
3	35,335
4	36,896
5	44,56



Como se puede ver la mejora es aproximadamente lineal, ya que al aumentar de manera lineal los hilos esto implica una carga equilibrada lineal en cada hilo por lo que esta mejora ocurre de manera lineal.

Referencias

K. (2018, 5 diciembre). *CreateProcessA function (processthreadsapi.h) - Win32 apps*. Microsoft Docs.

https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa