



Área Académica de Ingeniería en Computadores

Principios de Sistemas Operativos

Taller 2

Profesor:

Jason Levitón Jiménez

Estudiante:

Jose Ignacio Granados Marín

Grupo 1

Semestre II 2022

Preguntas guía

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

Las etapas de creación de un proceso en Windows son las siguientes [1]:

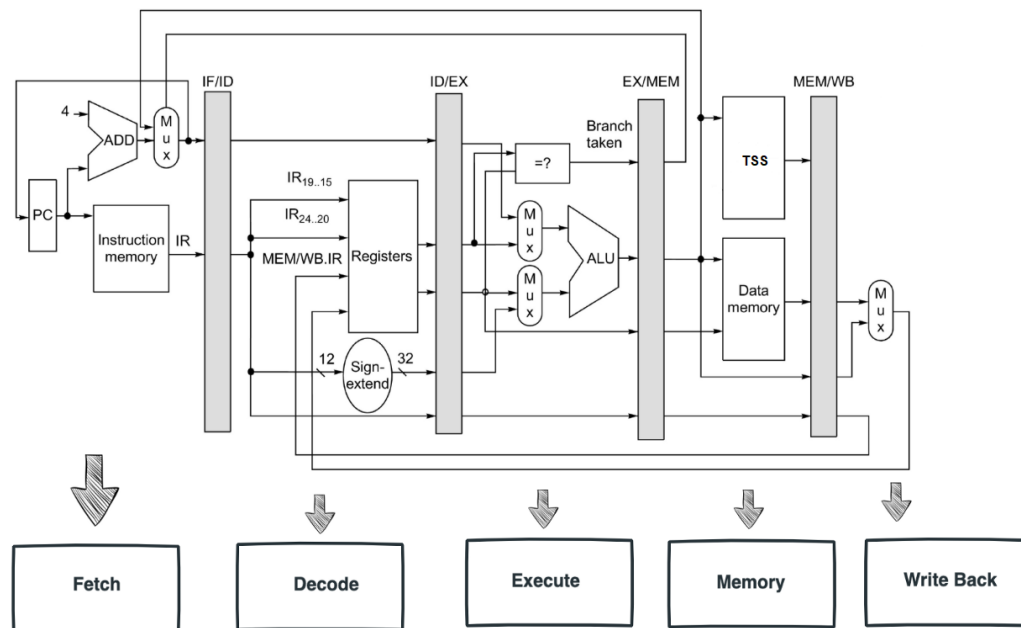
1. Abrir la imagen a ejecutar: esta etapa consiste en encontrar la imagen de Windows adecuada que ejecutará el archivo ejecutable especificado por el autor de la llamada y crear un objeto de sección para luego asignarlo al espacio de direcciones del nuevo proceso.
2. Creación del objeto windows executive process: para este punto se crea un objeto de proceso ejecutivo de Windows para ejecutar la imagen llamando a la función interna del sistema NtCreateProcess, considerando que ya se ha abierto un archivo ejecutable de Windows válido y se ha creado un objeto de sección para asignarlo al nuevo espacio de direcciones de proceso.
3. Creación del subproceso inicial y su pila y contexto: para este punto, el objeto de proceso ejecutivo de Windows está completamente configurado. Sin embargo, todavía no tiene hilo, por lo que aún no puede hacer nada. Antes de que se pueda crear el subproceso, necesita una pila y un contexto en el que ejecutarse, por lo que estos se configuran ahora. El tamaño de pila para el subproceso inicial se toma de la imagen, no hay forma de especificar otro tamaño.
4. Notificación al subsistema de Windows sobre el nuevo proceso: Si las directivas de restricción de software lo dictan, se crea un token restringido para el nuevo proceso. En este punto, se han creado todos los objetos de proceso ejecutivo y subprocesos necesarios. Kernel32.dll siguiente envía un mensaje al subsistema de Windows para que pueda configurarse para el nuevo proceso y subproceso. El mensaje incluye la siguiente información:
 - Manejadores de procesos y roscas
 - Entradas en los indicadores de creación
 - ID del creador del proceso
 - Indicador que indica si el proceso pertenece a una aplicación de Windows (para que Csrss pueda determinar si se muestra o no el cursor de inicio)
5. Inicio de la ejecución del subproceso inicial: el nuevo subproceso comienza su vida ejecutando la rutina de inicio de subprocesos en modo kernel KiThreadStartup. KiThreadStartup reduce el nivel IRQL del subproceso del nivel DPC/dispatch al nivel APC y, a continuación, llama a la rutina de subprocesos inicial del sistema, PspUserThreadStartup. La dirección de inicio de subproceso especificada por el usuario se pasa como parámetro a esta rutina.

Las variables a guardar ante un cambio de contexto son [2]:

- PC.
- Registros del procesador.
- Información de la pila.

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

Un cambio de contexto, implementado por hardware en una arquitectura x86, se puede utilizar para cambiar todo el estado del CPU, excepto el estado FPU/MMX y SSE. Para realizar el cambio en cuestión, es necesario definir el lugar dónde se va a guardar el estado existente y dónde se va a cargar el nuevo estado, dicha información debe ser de conocimiento del CPU. Por lo general, el estado actual del procesador se almacena en una estructura de datos especial llamada TSS (Task State Segment). Al momento de realizar el cambio de contexto se debe hacer ejecutar una instrucción CALL o JMP al registro TR (Registro de Tareas) el cual le indicará al CPU que debe mirar la entrada del GDT (Global Descriptor Table) donde estará el descriptor TSS con la dirección base del nuevo estado del CPU [3].



Para este caso en particular se propone colocar la unidad de TSS en la etapa de memory, en la cual se puede guardar o cargar el estado del CPU directamente en ese módulo y no es memoria directamente.

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

Linux proporciona una utilidad llamada PS que permite ver información relacionada con los procesos en un sistema. PS significa Estado del proceso (Process Status) y se utiliza para enumerar los procesos actualmente en ejecución y sus PID asociados [4].

El comando TOP muestra la información resumida del sistema y la lista de procesos o subprocesos que actualmente son administrados por el kernel de Linux, proporcionando una vista dinámica en tiempo real del sistema en ejecución [5].

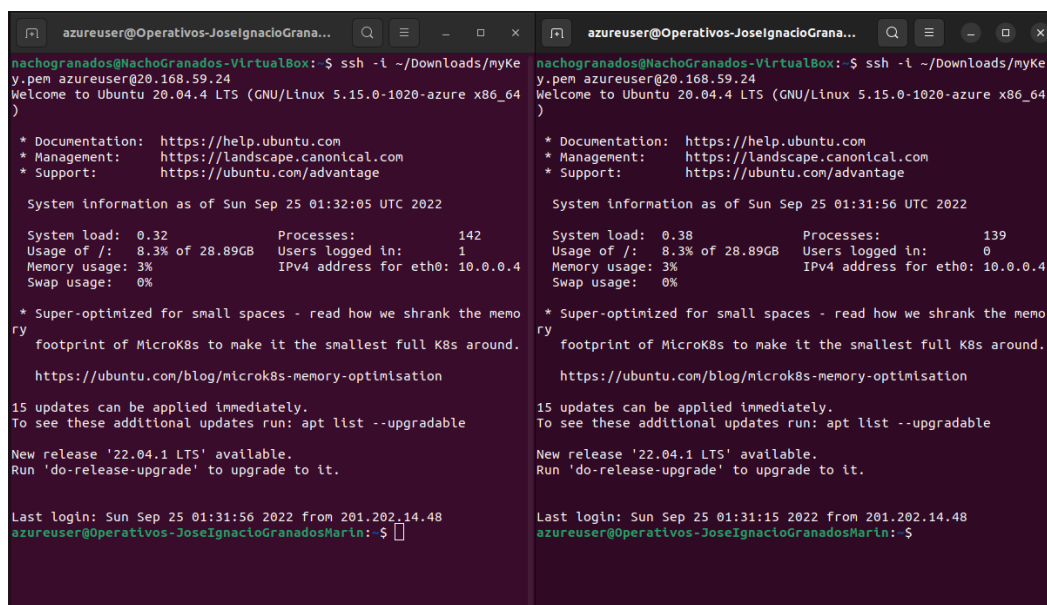
- ❖ Investigue los posibles estados de un proceso en un entorno de Linux y cómo se representan.

En un entorno de Linux se tiene los siguientes 5 estados de un proceso [6]:

1. Sleep (S): Durmiendo. Indica que se está en ejecución, pero en ese momento no se encuentra ejecutándose ningún código dentro del procesador.
2. Sleep (D): Durmiendo. Igual que el anterior, pero no es posible interrumpirlo.
3. Stopped (T): Parado. Indica que se ha detenido su ejecución.
4. Running (R): En ejecución. Corresponde a un proceso que se está ejecutando de forma activa en el CPU.
5. Zombie (Z): Zombie. Corresponde a un proceso que debería de haber sido terminado, pero aún tiene ciertas dependencias que no son posibles de terminar. Solo se puede terminar el proceso hasta que se eliminen dichas dependencias.

Procesos en Linux

1. Conéctese a su máquina virtual por medio de SSH. Acceda mediante dos conexiones, es decir, dos consolas.



The image shows two terminal windows side-by-side, both connected to a virtual machine named 'NachoGranados-VirtualBox' via SSH. The left terminal shows the initial login prompt and system information as of Sun Sep 25 01:32:05 UTC 2022. The right terminal shows the same login prompt and system information as of Sun Sep 25 01:31:56 UTC 2022. Both terminals display the same system information, including system load, memory usage, and available updates.

```
azureuser@Operativos-JoseIgnacioGrana... nachogranados@NachoGranados-VirtualBox:~$ ssh -t ~/Downloads/myKey.pem azureuser@20.168.59.24
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1020-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Sep 25 01:32:05 UTC 2022

System load: 0.32      Processes:            142
Usage of /:  8.3% of 28.89GB   Users logged in:      1
Memory usage: 3%          IPv4 address for eth0: 10.0.0.4
Swap usage:  0%

 * Super-optimized for small spaces - read how we shrank the memory footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

15 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Sep 25 01:31:56 2022 from 201.202.14.48
azureuser@Operativos-JoseIgnacioGranaMarin:~$

azureuser@Operativos-JoseIgnacioGrana... nachogranados@NachoGranados-VirtualBox:~$ ssh -t ~/Downloads/myKey.pem azureuser@20.168.59.24
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1020-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Sep 25 01:31:56 UTC 2022

System load: 0.38      Processes:            139
Usage of /:  8.3% of 28.89GB   Users logged in:      0
Memory usage: 3%          IPv4 address for eth0: 10.0.0.4
Swap usage:  0%

 * Super-optimized for small spaces - read how we shrank the memory footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

15 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Sep 25 01:31:15 2022 from 201.202.14.48
azureuser@Operativos-JoseIgnacioGranaMarin:~$
```

2. Ejecute el comando: `ps -aux` (obtenga la captura de los últimos ítem) explique cuál es el significado de `aux`.

```
root      655  0.0  0.0   0   0 ?        S    01:31   0:00 [jbd2/sdb1-8]
root      656  0.0  0.0   0   0 ?        I<   01:31   0:00 [ext4-rsv-conver]
root      657  0.0  0.1 241056 9268 ?        Ssl  01:31   0:00 /usr/lib/accountsservice/accounts-daemon
root      666  0.0  0.0   8548 2948 ?        Ss   01:31   0:00 /usr/sbin/cron -f
message+  667  0.0  0.0   7576 4696 ?        Ss   01:31   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nop
_chrony   675  0.0  0.0   4828 2636 ?        S    01:31   0:00 /usr/sbin/chronyd -F -1
root      677  0.0  0.0   81832 3740 ?        Ssl  01:31   0:00 /usr/sbin/irqbalance --foreground
root      678  0.0  0.2 298888 18372 ?        Ss   01:31   0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-tri
_chrony   679  0.0  0.0   4696   192 ?        S    01:31   0:00 /usr/sbin/chronyd -F -1
root      680  0.0  0.1 236320 8816 ?        Ssl  01:31   0:00 /usr/lib/policykit-1/polkitd --no-debug
syslog    682  0.0  0.0 224500 6988 ?        Ssl  01:31   0:00 /usr/sbin/rsyslogd -n -iNONE
root      687  2.5  0.5 800964 46324 ?        Ssl  01:31   0:05 /usr/lib/snapd/snapd
root      689  0.0  0.0   17340 7648 ?        Ss   01:31   0:00 /lib/systemd/systemd-logind
root      691  0.0  0.1 395548 13820 ?        Ssl  01:31   0:00 /usr/lib/udisks2/udisksd
root      692  0.0  0.2 28820 21248 ?        Ss   01:31   0:00 /usr/bin/python3 -u /usr/sbin/waagent -daemon
daemon    704  0.0  0.0   3804  2276 ?        Ss   01:31   0:00 /usr/sbin/atd -f
root      715  0.0  0.0   0   0 ?        I    01:31   0:00 [kworker/0:6-events]
root      716  0.0  0.0   0   0 ?        I    01:31   0:00 [kworker/0:7-events]
root      717  0.0  0.0   0   0 ?        I    01:31   0:00 [kworker/0:8-events]
root      726  0.0  0.0   7360 2404 ttyS0    Ss+  01:31   0:00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0 vt
root      732  0.0  0.0   5836 1700 tty1    Ss+  01:31   0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root      734  0.0  0.0  12184 7504 ?        Ss   01:31   0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
root      735  0.0  0.1 318828 15536 ?        Ssl  01:31   0:00 /usr/sbin/ModemManager
root      768  0.3  0.3 401804 27008 ?        Sl   01:31   0:00 python3 -u bin/WALinuxAgent-2.8.0.11-py2.7.egg -run-exhandlers
root      774  0.0  0.2 108136 20844 ?        Ssl  01:31   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgr
azureus+  889  0.0  0.1 19088 9684 ?        Ss   01:31   0:00 /lib/systemd/systemd --user
azureus+  890  0.0  0.0 103892 5440 ?        S    01:31   0:00 (sd-pam)
root      1140  0.0  0.1 13808 8924 ?        Ss   01:31   0:00 sshd: azureuser [priv]
azureus+ 1212  0.0  0.0 13940 5368 ?        S    01:31   0:00 sshd: azureuser@pts/0
azureus+ 1213  0.0  0.0 10048 5132 pts/0    Ss+  01:31   0:00 -bash
root      1222  0.0  0.1 13804 9028 ?        Ss   01:32   0:00 sshd: azureuser [priv]
azureus+ 1294  0.0  0.0 13936 5936 ?        S    01:32   0:00 sshd: azureuser@pts/1
azureus+ 1295  0.0  0.0 10048 5064 pts/1    Ss   01:32   0:00 -bash
azureus+ 1308  0.0  0.0 10816 3576 pts/1    R+   01:34   0:00 ps -aux
azureuser@Operativos-JoseIgnacioGranadosMarín:~$
```

El comando `aux` despliega una lista con procesos de todos los usuarios con la información respectiva de cada uno de ellos.

3. Investigue cada uno de los datos de los procesos del punto anterior (PID, VSZ ...).

Los procesos del punto anterior son [7]:

- UID: ID de usuario efectivo del propietario del proceso.
- PID: ID de proceso.
- PPID: ID de proceso principal.
- C: uso del procesador para la programación. Este campo no se muestra cuando se utiliza la opción `-c`.
- CLS: clase de programación a la que pertenece el proceso, como tiempo real, sistema o tiempo compartido. Este campo sólo se incluye con la opción `-c`.
- PRI: prioridad de programación del subproceso del núcleo. Los números más altos indican una prioridad superior.
- NI: número de nice del proceso, que contribuye a su prioridad de programación. Aumentar el valor del comando `nice` de un proceso significa reducir su prioridad.
- ADDR: dirección de la estructura `proc`.
- SZ: tamaño de la dirección virtual del proceso.
- WCHAN: dirección de un evento o bloqueo para el que el proceso está inactivo.
- STIME: hora de inicio del proceso en horas, minutos y segundos.
- TTY: terminal desde el cual se inició el proceso o su proceso principal. Un signo de interrogación indica que no existe un terminal de control.

- TIME: cantidad total de tiempo de CPU utilizado por el proceso desde que comenzó.
- CMD: comando que generó el proceso.
- VSZ: cantidad total de memoria a la que puede acceder un proceso.

4. Busque el comando que retorna los procesos propios de un usuario y tome la captura de pantalla del que posee mayor tiempo en el procesador.

```
436 ?      00:00:00 lb-comp-unb-wq
437 ?      00:00:00 lb_mcast
438 ?      00:00:00 lb_nl_sa_wq
439 ?      00:00:00 mkey_cache
443 ?      00:00:00 kaluad
444 ?      00:00:00 kmpath_rdacd
445 ?      00:00:00 kmpathd
446 ?      00:00:00 kmpath_handlerd
447 ?      00:00:00 multipathd
655 ?      00:00:00 jbd2/sdb1-8
656 ?      00:00:00 ext4-rsv-conver
657 ?      00:00:00 accounts-daemon
666 ?      00:00:00 cron
677 ?      00:00:00 irqbalance
678 ?      00:00:00 networkd-dispat
680 ?      00:00:00 polkitd
687 ?      00:00:05 snapd
689 ?      00:00:00 systemd-logind
691 ?      00:00:00 udisksd
692 ?      00:00:00 python3
717 ?      00:00:00 kworker/0:8-cgroup_destroy
726 ttyS0   00:00:00 agetty
732 tty1    00:00:00 agetty
734 ?      00:00:00 sshd
735 ?      00:00:00 ModemManager
768 ?      00:00:01 python3
774 ?      00:00:00 unattended-upgr
1140 ?     00:00:00 sshd
1222 ?     00:00:00 sshd
1327 ?     00:00:00 kworker/1:1-cgroup_destroy
1328 ?     00:00:00 kworker/u4:1-events_unbound
1342 ?     00:00:00 kworker/0:0-events
1343 ?     00:00:00 kworker/u4:0-events_power_efficient
1352 ?     00:00:00 kworker/1:0-mm_percpu_wq
azureuser@Operativos-JoseIgnacioGranadosMarin: $
```

5. Ejecute el comando: top en la primera consola. ¿Para qué sirve?

```
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ top
top - 01:53:12 up 22 min,  2 users,  load average: 0.02, 0.01, 0.00
Tasks: 128 total,  1 running, 127 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.0 us,  0.2 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  7951.3 total,  7264.3 free,   296.1 used,   390.9 buff/cache
MiB Swap:   0.0 total,   0.0 free,   0.0 used.  7408.3 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
    1 root        20   0 102664 13264  8128  S   0.0   0.2   0:01.74 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
    4 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 rcu_par_gp
    5 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 netns
    7 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
    9 root         0 -20     0     0     0  I   0.0   0.0   0:00.05 kworker/0:1H-events_highpri
   10 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 mm_percpu_wq
   11 root        20   0     0     0     0  S   0.0   0.0   0:00.00 rcu_tasks_rude_
   12 root        20   0     0     0     0  S   0.0   0.0   0:00.00 rcu_tasks_trace
   13 root        20   0     0     0     0  S   0.0   0.0   0:00.07 ksoftirqd/0
   14 root        20   0     0     0     0  I   0.0   0.0   0:00.24 rcu_sched
   15 root        rt   0     0     0     0  S   0.0   0.0   0:00.01 migration/0
   17 root        20   0     0     0     0  S   0.0   0.0   0:00.00 cpuhp/0
   18 root        20   0     0     0     0  S   0.0   0.0   0:00.00 cpuhp/1
   19 root        rt   0     0     0     0  S   0.0   0.0   0:00.39 migration/1
   20 root        20   0     0     0     0  S   0.0   0.0   0:00.09 ksoftirqd/1
   22 root         0 -20     0     0     0  I   0.0   0.0   0:00.04 kworker/1:0H-events_highpri
   23 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 kworker/1:1H
   24 root        20   0     0     0     0  S   0.0   0.0   0:00.00 kdevtmpfs
   25 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 inet_frag_wq
   26 root        20   0     0     0     0  S   0.0   0.0   0:00.00 kauditd
   28 root        20   0     0     0     0  S   0.0   0.0   0:00.00 khungtaskd
   29 root        20   0     0     0     0  S   0.0   0.0   0:00.00 oom_reaper
   30 root         0 -20     0     0     0  I   0.0   0.0   0:00.00 writeback
   31 root        20   0     0     0     0  S   0.0   0.0   0:00.05 kcompactd0
   32 root        25   5     0     0     0  S   0.0   0.0   0:00.00 ksnd
```

El comando despliega un listado con el tiempo real de los procesos que se están ejecutando en el sistema.

6. Ejecute el comando (5 veces en la segunda consola): `cat /dev/zero > /dev/null &`

```
root        691  0.0  0.1 395548 13820 ?        Ssl  01:31   0:00 /usr/lib/udisks2/udisksd
root        692  0.0  0.2 28820 21256 ?        Ss   01:31   0:00 /usr/bin/python3 -u /usr/sbin/waagent -daemon
daemon      704  0.0  0.0   3804 2276 ?        Ss   01:31   0:00 /usr/sbin/atd -f
root        726  0.0  0.0   7360 2404 ttyS0    Ss+  01:31   0:00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600
root        732  0.0  0.0   5836 1700 tty1     Ss+  01:31   0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root        734  0.0  0.0   12184 7504 ?        Ss   01:31   0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startup
root        735  0.0  0.1 318828 15536 ?        Ssl  01:31   0:00 /usr/sbin/ModemManager
root        768  0.1  0.3 401804 27288 ?        Sl   01:31   0:01 python3 -u bin/WALinuxAgent-2.8.0.11-py2.7.egg -run-ex
root        774  0.0  0.2 108136 20844 ?        Ssl  01:31   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unatte
azureus+    889  0.0  0.1 19088 9684 ?        Ss   01:31   0:00 /lib/systemd/systemd --user
azureus+    890  0.0  0.0 103892 5440 ?        S    01:31   0:00 (sd-pam)
root        1222  0.0  0.1 13804 9028 ?        Ss   01:32   0:00 sshd: azureuser [priv]
azureus+    1294  0.0  0.0 13936 5936 ?        S    01:32   0:00 sshd: azureuser@pts/1
azureus+    1295  0.0  0.0 10048 5068 pts/1    Ss   01:32   0:00 -bash
root        1328  0.0  0.0   0 0 ?        I    01:36   0:00 [kworker/u4:1-events_power_efficient]
root        1342  0.0  0.0   0 0 ?        I    01:46   0:00 [kworker/0:0-events]
root        1343  0.0  0.0   0 0 ?        I    01:47   0:00 [kworker/u4:0-events_unbound]
root        1352  0.0  0.0   0 0 ?        I    01:49   0:00 [kworker/1:0-events]
azureus+    1361  0.1  0.0 11032 3720 pts/1    S+   01:52   0:00 top
root        1375  0.0  0.0   0 0 ?        I    01:56   0:00 [kworker/1:1]
root        1376  0.0  0.1 13808 8928 ?        Ss   01:56   0:00 sshd: azureuser [priv]
azureus+    1448  0.0  0.0 13940 5884 ?        S    01:56   0:00 sshd: azureuser@pts/0
azureus+    1449  0.1  0.0 10048 5080 pts/0    Ss   01:56   0:00 -bash
azureus+    1458  0.0  0.0 10816 3472 pts/0    R+   01:57   0:00 ps -aux
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ cat /dev/zero > /dev/null &
[1] 1459
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ cat /dev/zero > /dev/null &
[2] 1460
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ cat /dev/zero > /dev/null &
[3] 1461
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ cat /dev/zero > /dev/null &
[4] 1462
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ cat /dev/zero > /dev/null &
[5] 1463
azureuser@Operativos-JoseIgnacioGranadosMarin:~$
```

7. Ejecute el comando `top` nuevamente en la primera consola. ¿Qué observa con respecto al `top` anterior?

```
top - 01:59:24 up 28 min, 2 users, load average: 4.44, 1.89, 0.72
Tasks: 133 total, 6 running, 127 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.0 us, 94.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7951.3 total, 7294.6 free, 265.2 used, 391.5 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 7439.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 1460 azureus+  20   0   7380    1860   1660 R   41.3   0.0   0:48.65 cat
 1459 azureus+  20   0   7380    2096   1912 R   40.3   0.0   0:50.79 cat
 1461 azureus+  20   0   7380    1960   1764 R   40.0   0.0   0:47.95 cat
 1462 azureus+  20   0   7380    2048   1860 R   39.7   0.0   0:46.69 cat
 1463 azureus+  20   0   7380    2048   1860 R   39.0   0.0   0:46.28 cat
   1 root      20   0 102664 13264   8128 S    0.0   0.2   0:01.75 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
   4 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 rcu_par_gp
   5 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 netns
   7 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 kworker/0:0H-events_highpri
   9 root      0 -20   0 0 0 I    0.0   0.0   0:00.06 kworker/0:1H-events_highpri
  10 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 mm_percpu_wq
  11 root      20   0   0 0 0 S    0.0   0.0   0:00.00 rcu_tasks_rude_
  12 root      20   0   0 0 0 S    0.0   0.0   0:00.00 rcu_tasks_trace
  13 root      20   0   0 0 0 S    0.0   0.0   0:00.08 ksoftirqd/0
  14 root      20   0   0 0 0 I    0.0   0.0   0:00.25 rcu_sched
  15 root      rt    0   0 0 0 S    0.0   0.0   0:00.01 migration/0
  17 root      20   0   0 0 0 S    0.0   0.0   0:00.00 cpuhp/0
  18 root      20   0   0 0 0 S    0.0   0.0   0:00.00 cpuhp/1
  19 root      rt    0   0 0 0 S    0.0   0.0   0:00.40 migration/1
  20 root      20   0   0 0 0 S    0.0   0.0   0:00.10 ksoftirqd/1
  22 root      0 -20   0 0 0 I    0.0   0.0   0:00.05 kworker/1:0H-events_highpri
  23 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 kworker/1:1H
  24 root      20   0   0 0 0 S    0.0   0.0   0:00.00 kdevtmpfs
  25 root      0 -20   0 0 0 I    0.0   0.0   0:00.00 inet_frag_wq
  26 root      20   0   0 0 0 S    0.0   0.0   0:00.00 kauditd
  28 root      20   0   0 0 0 S    0.0   0.0   0:00.00 khungtaskd
```

Se observa que los procesos iniciados por la segunda consola son los que están consumiendo mayor tiempo del procesador.

8. ¿Qué significa los valores de cada uno de los parámetros de los procesos creados (PR, NI, VIRT ...)?

- Los parámetros de los procesos creados son [8]:
- PID: Muestra el identificador de proceso único de la tarea.
- PR: La prioridad del proceso. Cuanto menor sea el número, mayor será la prioridad.
- VIRT: Memoria virtual total utilizada por la tarea.
- USUARIO: Nombre de usuario del propietario de la tarea.
- %CPU: Representa el uso de la CPU.
- TIEMPO+: Tiempo de CPU, lo mismo que 'TIEMPO', pero reflejando más granularidad a través de centésimas de segundo.
- SHR: Representa el tamaño de memoria compartida (kb) utilizado por una tarea.
- NI: Representa un valor agradable de la tarea. Un valor agradable negativo implica una mayor prioridad, y el valor positivo de Niza significa menor prioridad.
- %MEM: Muestra el uso de memoria de la tarea.
- RES: Cuánta RAM física está utilizando el proceso, medida en kilobytes.
- MANDAR: Nombre del comando que inició el proceso.

9. Note que todos los procesos creados tienen una prioridad similar ¿Por qué sucede esto?

Este comportamiento sucede principalmente porque, en Linux, existen prioridades por defecto en los procesos. En este caso en particular, se dispone de una prioridad de 20.

10. ¿Por qué el parámetro “Time” aumenta paulatinamente?

Esto se debe a que los procesos que se encuentran en el estado R de ejecución.

11. Obtenga un identificador de alguno de los procesos creados anteriormente.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1460	azureus+	20	0	7380	1860	1660	R	42.2	0.0	5:53.63	cat

12. Aumente la prioridad de dicho proceso con el comando: renice -n 10 PID

```
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ renice -n 10 1460
1460 (process ID) old priority 0, new priority 10
azureuser@Operativos-JoseIgnacioGranadosMarin:~$
```

13. Inicie un proceso con prioridad alta con el siguiente comando: nice -n -10 cat /dev/zero > /dev/null &

```
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ sudo nice -n -10 cat /dev/zero > /dev/null &
[1] 1489
azureuser@Operativos-JoseIgnacioGranadosMarin:~$
```


14. Ejecute el comando top nuevamente.

```

azureuser@Operativos-JoseIgnacioGranadosMarin:~$ top
top - 02:17:29 up 46 min,  2 users,  load average: 5.83, 5.23, 3.81
Tasks: 133 total,  7 running, 126 sleeping,  0 stopped,  0 zombie
%Cpu(s):  5.8 us, 94.0 sy,  0.2 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 7951.3 total, 7290.4 free,  267.3 used,  393.7 buff/cache
MiB Swap:   0.0 total,   0.0 free,   0.0 used. 7437.0 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 1490 root        10  -10   7380    2084   1900 R 100.0   0.0   1:46.65  cat
 1459 azureus+   20   0   7380    2096   1912 R  24.3   0.0   7:59.38  cat
 1461 azureus+  20   0   7380    1960   1764 R  24.3   0.0   7:55.95  cat
 1462 azureus+  20   0   7380    2048   1860 R  24.3   0.0   7:53.27  cat
 1463 azureus+  20   0   7380    2048   1860 R  24.3   0.0   7:54.75  cat
 1460 azureus+  30  10   7380    1860   1660 R   2.7   0.0   6:41.27  cat
 1495 azureus+  20   0 11032    3708   3172 R   0.3   0.0   0:00.10  top
   1 root        20   0 102664   13264   8128 S   0.0   0.2   0:01.75  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
   4 root         0 -20      0      0      0 I   0.0   0.0   0:00.00  rcu_par_gp
   5 root         0 -20      0      0      0 I   0.0   0.0   0:00.00  netns
   7 root         0 -20      0      0      0 I   0.0   0.0   0:00.00  kworker/0:0H-events_highpri
   9 root         0 -20      0      0      0 I   0.0   0.0   0:00.07  kworker/0:1H-events_highpri
  10 root         0 -20      0      0      0 I   0.0   0.0   0:00.00  mm_percpu_wq
  11 root        20   0      0      0      0 S   0.0   0.0   0:00.00  rcu_tasks_rude
  12 root        20   0      0      0      0 S   0.0   0.0   0:00.00  rcu_tasks_trace
  13 root        20   0      0      0      0 S   0.0   0.0   0:00.11  ksoftirqd/0
  14 root        20   0      0      0      0 I   0.0   0.0   0:00.27  rcu_sched
  15 root         rt   0      0      0      0 S   0.0   0.0   0:00.01  migration/0
  17 root        20   0      0      0      0 S   0.0   0.0   0:00.00  cpuhp/0
  18 root        20   0      0      0      0 S   0.0   0.0   0:00.00  cpuhp/1
  19 root         rt   0      0      0      0 S   0.0   0.0   0:00.40  migration/1
  20 root        20   0      0      0      0 S   0.0   0.0   0:00.12  ksoftirqd/1
  22 root         0 -20      0      0      0 I   0.0   0.0   0:00.06  kworker/1:0H-events_highpri
  23 root         0 -20      0      0      0 I   0.0   0.0   0:00.00  kworker/1:1H
  24 root        20   0      0      0      0 S   0.0   0.0   0:00.00  kdevtmpfs

```

Creación de procesos con parámetros establecidos por el usuario

1. Realice un programa recursivo en C y Python que sea capaz calcular el factorial de cualquier número entero.

Python:

```
1 import sys
2 import time
3
4 # set recursion limit
5 sys.setrecursionlimit(10**5)
6
7 # recursive factorial definition
8 def factorial(number):
9
10     print("Factorial")
11
12     # end condition
13     if (number == 0 or number == 1):
14
15         return 1
16
17     else:
18
19         # recursive call
20         return number * factorial(number - 1)
21
22
23 number = 10000
24
25 start = time.time()
26
27 result = factorial(number)
28
29 end = time.time()
30
31 timeConsumed = str( (end - start) )
32
33 print("The factorial of the number:", number, "is:", result)
34
35 print("Time consumed:", timeConsumed, "s")
36
37 # infinite loop
38 while True:
39
40     time.sleep(1)
```

C:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <time.h>
4
5 clock_t start;
6 clock_t end;
7
8 float timeConsumed;
9
10 // recursive factorial definition
11 long factorial(int number) {
12
13     printf("Factorial\n");
14
15     // end condition
16     if (number == 0 || number == 1) {
17
18         return 1;
19
20     } else {
21
22         // recursive call
23         return number * factorial(number - 1);
24
25     }
26 }
27
28
29 int main() {
30
31     int number = 100000;
32
33     long result;
34
35     start = clock();
36
37     result = factorial(number);
38
39     end = clock();
40
41     timeConsumed = (float)(end - start) / CLOCKS_PER_SEC;
42
43     printf("The factorial of the number %d is: %ld \n", number, result);
44
45     printf("Time consumed: %10.6f s\n", timeConsumed);
46
47     // infinite loop
48     while (1) {
49
50         sleep(1);
51
52     }
53
54     return 0;
55
56 }
```

2. Ejecute dichos programas en su máquina virtual y tome el tiempo de ejecución, así como los parámetros de ambos procesos con el comando top (En caso de que sea necesario coloque prints entre cada recursión).

Ejecución en Python:

```
azureuser@Operativos-JoseIgnacioGranadosMarin:~$ top
```

top - 03:30:17 up 1:59, 2 users, load average: 1.05, 1.02, 1.03
Tasks: 130 total, 2 running, 128 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.1 us, 44.9 sy, 0.0 ni, 49.8 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
MiB Mem : 7951.3 total, 7222.9 free, 301.5 used, 426.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 7402.5 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
692	root	20	0	28820	21256	9044	S	0.0	0.3	0:00.24	python3
704	daemon	20	0	3804	2276	2096	S	0.0	0.0	0:00.00	atd
726	root	20	0	7360	2404	2280	S	0.0	0.0	0:00.00	agetty
732	root	20	0	5836	1700	1592	S	0.0	0.0	0:00.00	agetty
734	root	20	0	12184	7504	6572	S	0.0	0.1	0:00.02	sshd
735	root	20	0	318828	15536	11592	S	0.0	0.2	0:00.08	ModemManager
768	root	20	0	401804	27840	10468	S	0.0	0.3	0:06.91	python3
774	root	20	0	108136	20844	13176	S	0.0	0.3	0:00.08	unattended-upgr
889	azureus+	20	0	19088	9684	8092	S	0.0	0.1	0:00.13	systemd
890	azureus+	20	0	103892	5440	20	S	0.0	0.1	0:00.00	(sd-pam)
1352	root	20	0	0	0	0	I	0.0	0.0	0:00.23	kworker/1:0-cgroup_destroy
1489	root	20	0	11180	4680	3964	S	0.0	0.1	0:00.00	sudo
1713	root	20	0	0	0	0	I	0.0	0.0	0:00.11	kworker/0:2-events
1714	root	20	0	0	0	0	I	0.0	0.0	0:00.79	kworker/u4:1-events_power_efficient
1970	root	20	0	13808	9028	7564	S	0.0	0.1	0:00.01	sshd
2042	azureus+	20	0	13940	5976	4512	S	0.0	0.1	0:00.57	sshd
2043	azureus+	20	0	10048	5044	3340	S	0.0	0.1	0:00.04	bash
2076	root	20	0	0	0	0	I	0.0	0.0	0:00.77	kworker/u4:2-events_unbound
2082	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/1:2-events
2088	root	20	0	0	0	0	I	0.0	0.0	0:00.10	kworker/0:1-events
2089	root	20	0	13808	9040	7572	S	0.0	0.1	0:00.02	sshd
2161	azureus+	20	0	13940	6184	4520	S	0.0	0.1	0:00.60	sshd
2162	azureus+	20	0	10180	5320	3528	S	0.0	0.1	0:00.06	bash
2189	root	20	0	0	0	0	I	0.0	0.0	0:00.27	kworker/u4:0-events_unbound
2213	azureus+	20	0	20444	13816	5924	S	0.0	0.2	0:00.15	python3
2214	azureus+	20	0	11032	3716	3184	R	0.0	0.0	0:00.64	top

Ejecución en C:

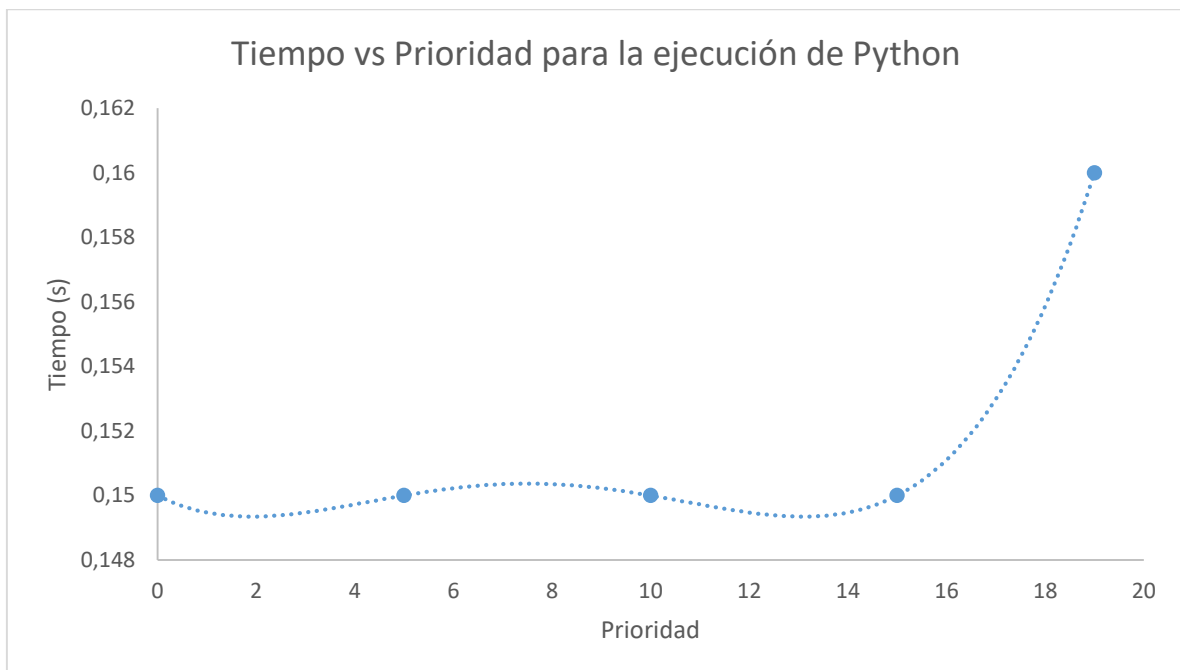
```
top - 03:23:05 up 1:52, 2 users, load average: 1.00, 1.02, 1.06
```

Tasks: 130 total, 2 running, 128 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.8 us, 44.3 sy, 0.0 ni, 49.8 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
MiB Mem : 7951.3 total, 7220.3 free, 304.7 used, 426.3 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 7399.3 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
691	root	20	0	395548	13820	11792	S	0.0	0.2	0:00.07	udisksd
692	root	20	0	28820	21256	9044	S	0.0	0.3	0:00.24	python3
704	daemon	20	0	3804	2276	2096	S	0.0	0.0	0:00.00	atd
726	root	20	0	7360	2404	2280	S	0.0	0.0	0:00.00	agetty
732	root	20	0	5836	1700	1592	S	0.0	0.0	0:00.00	agetty
734	root	20	0	12184	7504	6572	S	0.0	0.1	0:00.01	sshd
735	root	20	0	318828	15536	11592	S	0.0	0.2	0:00.08	ModemManager
768	root	20	0	401804	27840	10468	S	0.0	0.3	0:06.53	python3
774	root	20	0	108136	20844	13176	S	0.0	0.3	0:00.08	unattended-upgr
889	azureus+	20	0	19088	9684	8092	S	0.0	0.1	0:00.13	systemd
890	azureus+	20	0	103892	5440	20	S	0.0	0.1	0:00.00	(sd-pam)
1352	root	20	0	0	0	0	I	0.0	0.0	0:00.23	kworker/1:0-cgroup_destroy
1489	root	20	0	11180	4680	3964	S	0.0	0.1	0:00.00	sudo
1713	root	20	0	0	0	0	I	0.0	0.0	0:00.11	kworker/0:2-events
1714	root	20	0	0	0	0	I	0.0	0.0	0:00.72	kworker/u4:1-events_power_efficient
1970	root	20	0	13808	9028	7564	S	0.0	0.1	0:00.01	sshd
2042	azureus+	20	0	13940	5976	4512	S	0.0	0.1	0:00.34	sshd
2043	azureus+	20	0	10048	5044	3340	S	0.0	0.1	0:00.04	bash
2076	root	20	0	0	0	0	I	0.0	0.0	0:00.75	kworker/u4:2-events_power_efficient
2082	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/1:2-events
2088	root	20	0	0	0	0	I	0.0	0.0	0:00.04	kworker/0:1-events
2089	root	20	0	13808	9040	7572	S	0.0	0.1	0:00.02	sshd
2161	azureus+	20	0	13940	6184	4520	S	0.0	0.1	0:00.51	sshd
2162	azureus+	20	0	10048	5104	3404	S	0.0	0.1	0:00.05	bash
2188	azureus+	20	0	7056	6040	1288	S	0.0	0.1	0:00.50	factorial.out
2189	root	20	0	0	0	0	I	0.0	0.0	0:00.21	kworker/u4:0-events_power_efficient
2190	azureus+	20	0	11032	3748	3212	R	0.0	0.0	0:00.59	top

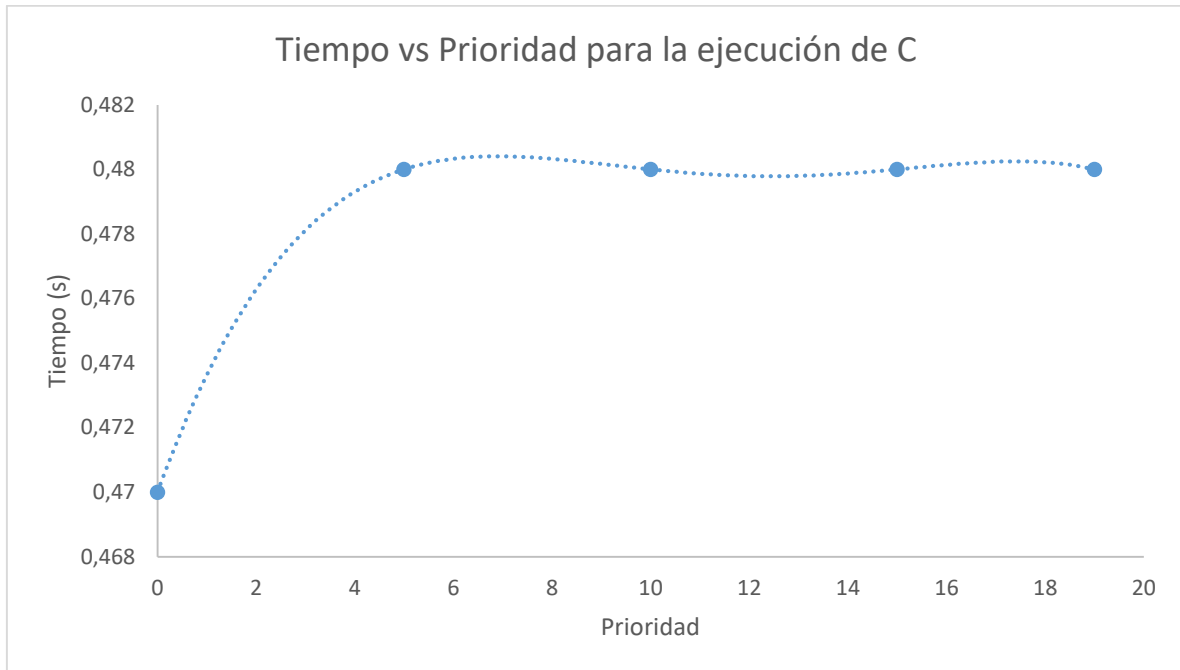
- Ejecute el programa realizado en Python 5 veces con 5 diferentes prioridades de manera ascendente y muestre una gráfica del comportamiento del mismo (Prioridad-Tiempo).

```
2318 azureus+ 20 0 20436 13816 5928 S 0.0 0.2 0:00.15 python3
2318 azureus+ 25 5 20436 13816 5928 S 0.0 0.2 0:00.15 python3
2318 azureus+ 30 10 20436 13816 5928 S 0.0 0.2 0:00.15 python3
2318 azureus+ 35 15 20436 13816 5928 S 0.0 0.2 0:00.15 python3
2318 azureus+ 39 19 20436 13816 5928 S 0.0 0.2 0:00.16 python3
```



- Ejecute el programa realizado en C 5 veces con 5 diferentes prioridades de manera descendente y muestre una gráfica del comportamiento de mismo (Prioridad-Tiempo).

```
2353 azureus+ 20 0 7056 5960 1208 S 0.0 0.1 0:00.47 factorial.out
2353 azureus+ 25 5 7056 5960 1208 S 0.0 0.1 0:00.48 factorial.out
2353 azureus+ 30 10 7056 5960 1208 S 0.0 0.1 0:00.48 factorial.out
2353 azureus+ 35 15 7056 5960 1208 S 0.0 0.1 0:00.48 factorial.out
2353 azureus+ 39 19 7056 5960 1208 S 0.0 0.1 0:00.48 factorial.out
```



- Investigue el comando para eliminar un proceso. Posteriormente ejecute el programa en C y elimínelo antes de que termine su ejecución.

```

azureuser@Operativos-JoseIgnacioGrana...  azureuser@Operativos-JoseIgnacioGrana...
2089 root      20    0   13808   9040   7572 S    0.0   0.1   0:   Factorial
00.02 sshd
2161 azureus+  20    0   13940   6184   4520 S    0.0   0.1   0:   Factorial
01.25 sshd
2162 azureus+  20    0   10180   5324   3528 S    0.0   0.1   0:   Factorial
00.06 bash
2330 root      20    0          0      0      0 I    0.0   0.0   0:   Factorial
00.05 kworker/0:0-mm_percpu_wq
2331 root      20    0          0      0      0 I    0.0   0.0   0:   Factorial
00.39 kworker/u4:2-events_power_efficient
2347 root      20    0          0      0      0 I    0.0   0.0   0:   Factorial
00.36 kworker/u4:0-events_power_efficient
2353 azureus+  39   19    7056   5960   1208 S    0.0   0.1   0:   Factorial
00.49 factorial.out
2354 root      20    0          0      0      0 I    0.0   0.0   0:   Factorial
00.00 kworker/1:1
2368 root      20    0          0      0      0 I    0.0   0.0   0:   Factorial
00.03 kworker/u4:1-events_power_efficient

The factorial of the number 100000 is: 0
Time consumed:  0.485648 s
Terminated
azureuser@Operativos-JoseIgnacioGrana...  azureuser@Operativos-JoseIgnacioGrana...
$ kill 2353
$
~/Taller2$
  
```

6. Discuta el comportamiento de la gráfica. ¿Es el comportamiento que esperaba?

La teoría indica que, a menor prioridad, menor será el tiempo de ejecución que le tome a un proceso ejecutarse por completo. Sin embargo, las gráficas anteriores no demuestran en su totalidad dicho concepto debido a que en las mismas se pueden observar variaciones de tan solo 0.1 s.

En la ejecución de Python, se puede apreciar que la última muestra posee el mayor tiempo de ejecución mientras que para la ejecución de C, la primera muestra posee el menor tiempo de ejecución y en ambos casos, se esperaba que el comportamiento fuera lineal y ascendente conforme incrementaba el valor de la prioridad.

Hilos en Linux

1. Realice un programa en C que tome un archivo de texto (.txt) y cuente la cantidad de apariciones de una determinada palabra. Tome el tiempo de ejecución del mismo.

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <time.h>
4
5 #define MAX_LENGTH 1024
6
7 clock_t start;
8 clock_t end;
9
10 float timeConsumed;
11
12 char word[10];
13
14 // function that look for the word in a single line
15 int wordCounter(char *line) {
16
17     int lineLength = 0;
18     int wordLength = 0;
19     int counter = 0;
20     int flag = 0;
21
22     int i = 0;
23     int j = 0;
24
25     lineLength = strlen(line);
26     wordLength = strlen(word);
27
28     for (i = 0; i <= lineLength - wordLength; i++) {
29
30         flag = 1;
31
32         for (j = 0; j < wordLength; j++) {
33
34             if (line[i + j] != word[j]) {
35
36                 flag = 0;
37
38                 break;
39
40             }
41
42         }
43
44         // word found
45         if (flag == 1) {
46
47             counter++;
48
49         }
50     }
51
52     return counter;
53 }
54
55 // function that open the file and read line by line
56 int readFile() {
57
58     char *filename = "bigText.txt";
59
60     FILE *fp = fopen(filename, "r");
61
62     int result = 0;
63
64     char buffer[MAX_LENGTH];
65
66     // loop to read line by line
67     while (fgets(buffer, MAX_LENGTH, fp)) {
68
69         result += wordCounter(buffer);
70
71     }
72
73     fclose(fp);
74
75     return result;
76 }
77
78 }
79
80 int main() {
81
82     int result = 0;
83
84     strcpy(word, "for");
85
86     start = clock();
87
88     result = readFile();
89
90     end = clock();
91
92     timeConsumed = (float)(end - start) / CLOCKS_PER_SEC;
93
94     printf("The word '%s' appears %d times in the text\n", word, result);
95     printf("Time consumed: %10.6f s\n", timeConsumed);
96
97     return 0;
98 }
99 }
```

2. Implemente dicho programa con 2,3,4,5 hilos y grafique el comportamiento (Cantidad de hilos- Tiempo) puede utilizar un software como excel o su equivalente.

Código:

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <time.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 #define MAX_LENGTH 1024
8
9 clock_t start;
10 clock_t end;
11
12 float timeConsumed;
13
14 pthread_t thread1;
15 pthread_t thread2;
16 pthread_t thread3;
17 pthread_t thread4;
18 pthread_t thread5;
19
20 char word[10];
21
22 char filename[10];
23
24 FILE *fp = NULL;
25
26 int finalResult = 0;
27
28 // function that look for the word in a single line
29 int wordCounter(char *line) {
30
31     int lineLength = 0;
32     int wordLength = 0;
33     int counter = 0;
34     int flag = 0;
35
36     int i = 0;
37     int j = 0;
38
39     lineLength = strlen(line);
40     wordLength = strlen(word);
41
42     for (i = 0; i <= lineLength - wordLength; i++) {
43
44         flag = 1;
45
46         for (j = 0; j < wordLength; j++) {
47
48             if (line[i + j] != word[j]) {
49
50                 flag = 0;
51
52                 break;
53
54             }
55
56         }
57
58         // word found
59         if (flag == 1) {
60
61             counter++;
62
63         }
64
65     }
66
67     return counter;
68
69 }
70
71 // function that open the file and read line by line
72 void *readFile(void *counter) {
73
74     int result = *((int *) counter);
75
76     char buffer[MAX_LENGTH];
77
78     // loop to read line by line
79     while (fgets(buffer, MAX_LENGTH, fp)) {
80
81         result += wordCounter(buffer);
82
83     }
84
85     printf("%i \n", result);
86
87     finalResult += result;
88
89     free(counter);
90
91 }
92
93 int main() {
94
95     strcpy(filename, "bigText.txt");
96
97     fp = fopen(filename, "r");
98
99     strcpy(word, "for");
100
101     start = clock();
102
103     int *counter1 = malloc(sizeof(*counter1));
104     int *counter2 = malloc(sizeof(*counter2));
105     int *counter3 = malloc(sizeof(*counter3));
106     int *counter4 = malloc(sizeof(*counter4));
107     int *counter5 = malloc(sizeof(*counter5));
108
109     *counter1 = 0;
110     *counter2 = 0;
111     *counter3 = 0;
112     *counter4 = 0;
113     *counter5 = 0;
114
115     // create threads
116     pthread_create(&thread1, NULL, readFile, (void *)counter1);
117     pthread_create(&thread2, NULL, readFile, (void *)counter2);
118     pthread_create(&thread3, NULL, readFile, (void *)counter3);
119     pthread_create(&thread4, NULL, readFile, (void *)counter4);
120     pthread_create(&thread5, NULL, readFile, (void *)counter5);
121
122     // wait until threads are done
123     pthread_join(thread1, NULL);
124     pthread_join(thread2, NULL);
125     pthread_join(thread3, NULL);
126     pthread_join(thread4, NULL);
127     pthread_join(thread5, NULL);
128
129     fclose(fp);
130
131     end = clock();
132
133     timeConsumed = (float)(end - start) / CLOCKS_PER_SEC;
134
135     printf("The word '%s' appears %d times in the text\n", word, finalResult);
136     printf("Time consumed: %18.6f s\n", timeConsumed);
137
138     return 0;
139
140 }
```

Ejecución con 2 hilos:

```
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ gcc -o wordCounterTh
read wordCounterThread.c
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ ./wordCounterThread
6132
7604
The word 'for' appears 13736 times in the text
Time consumed: 0.039929 s
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$
```

Ejecución con 3 hilos:

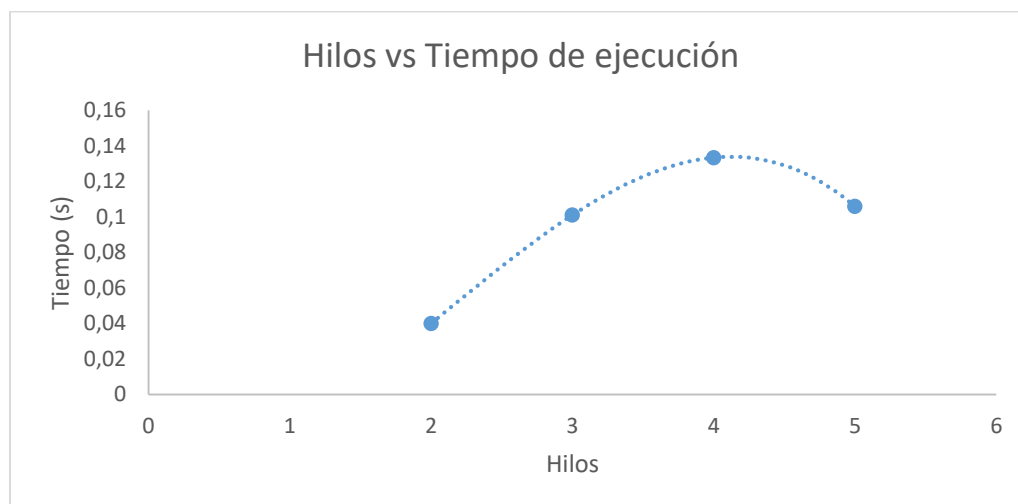
```
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ gcc -o wordCounterTh
read wordCounterThread.c
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ ./wordCounterThread
4193
4809
4734
The word 'for' appears 13736 times in the text
Time consumed: 0.101000 s
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$
```

Ejecución con 4 hilos:

```
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ gcc -o wordCounterTh
read wordCounterThread.c
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ ./wordCounterThread
5039
3014
3003
2680
The word 'for' appears 13736 times in the text
Time consumed: 0.133361 s
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$
```

Ejecución con 5 hilos:

```
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ gcc -o wordCounterTh
read wordCounterThread.c
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$ ./wordCounterThread
3326
2588
4359
806
2657
The word 'for' appears 13736 times in the text
Time consumed: 0.105917 s
nachogranados@NachoGranados-VirtualBox:~/Documents/GitHub/OperatingSystemsPrinciples.Classwork2$
```



3. ¿La mejora es lineal? Justifique dicha monotonía.

Tal y como se puede observar en la gráfica anterior, el comportamiento solo es lineal para los casos de 2, 3 y 4 hilos, lo que implica que no hay una mejoría en el rendimiento y tiempo de ejecución del programa porque a menor cantidad de hilos, menor es el tiempo de ejecución. Lo anterior es contradictorio con la teoría puesto que, debería presentarse el comportamiento inverso.

Si se compara la ejecución de 4 hilos con hilos, se observa que sí hay una disminución en el tiempo de ejecución, por lo que, en este caso en particular, la teoría sí se ve evidenciada. Sin embargo, el uso de hilos está altamente ligado con los recursos de hardware, los cuales son limitados y esto se ve reflejado en el mayor número de hilos que se puedan ejecutar.

Referencias

- [1] *Flow of CreateProcess / Microsoft Windows Internals (4th Edition): Microsoft Windows Server 2003, Windows XP, and Windows 2000*. (2017). Recuperado 22 de septiembre de 2022, de <https://flylib.com/books/en/4.491.1.52/1/>
- [2] González, P. (2015). *Control de un proceso*. <https://lsi.vc.ehu.eus/pablogn/>
- [3] *Context Switching - OSDev Wiki*. (2020). Recuperado 24 de septiembre de 2022, de https://wiki.osdev.org/Context_Switching
- [4] GeeksforGeeks. (2022, 10 enero). *ps command in Linux with Examples*. Recuperado 24 de septiembre de 2022, de <https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/>
- [5] GeeksforGeeks. (2022b, mayo 19). *top command in Linux with Examples*. Recuperado 24 de septiembre de 2022, de <https://www.geeksforgeeks.org/top-command-in-linux-with-examples/>
- [6] Solutions, S. A. L. (2010, 27 noviembre). *Procesos en GNU/Linux*. Altenwald Blog. Recuperado 24 de septiembre de 2022, de <https://altenwald.org/2010/11/27/procesos-en-gnulinux/>
- [7] *Comandos para gestionar procesos del sistema - Guía de administración del sistema: administración avanzada*. (2011, 1 enero). Recuperado 24 de septiembre de 2022, de https://docs.oracle.com/cd/E24842_01/html/E23086/spprocess-1.html
- [8] GeeksforGeeks. (2022c, mayo 19). *top command in Linux with Examples*. Recuperado 24 de septiembre de 2022, de <https://www.geeksforgeeks.org/top-command-in-linux-with-examples/>
- [9] Córdoba, D. (2017, 4 julio). *Procesos en Linux, estados y prioridades*. Junco TIC. Recuperado 24 de septiembre de 2022, de <https://juncotic.com/procesos-en-linux-estados-y-prioridades/>