

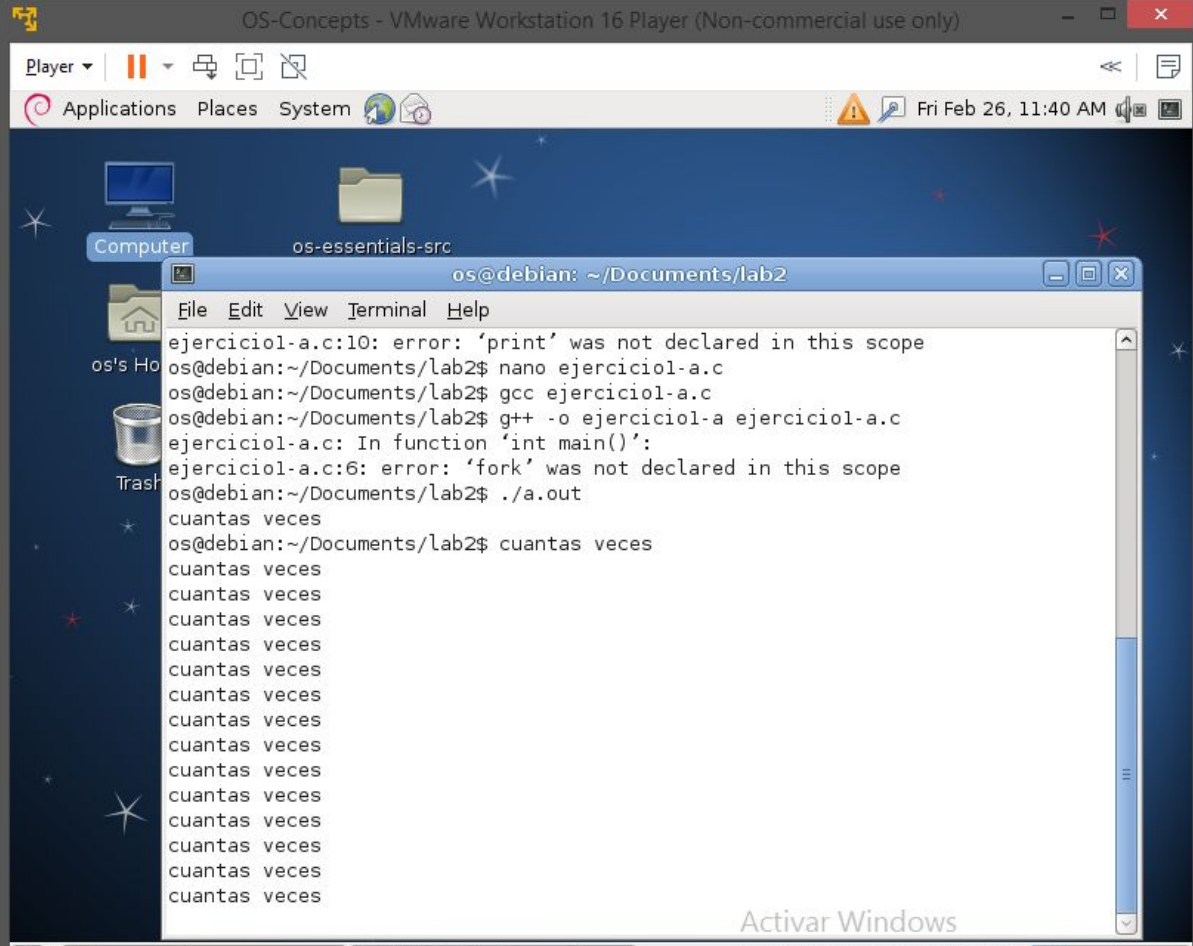
## Lab #2 - Comunicación entre procesos del sistema operativo

### Ejercicio 1 (10 puntos)

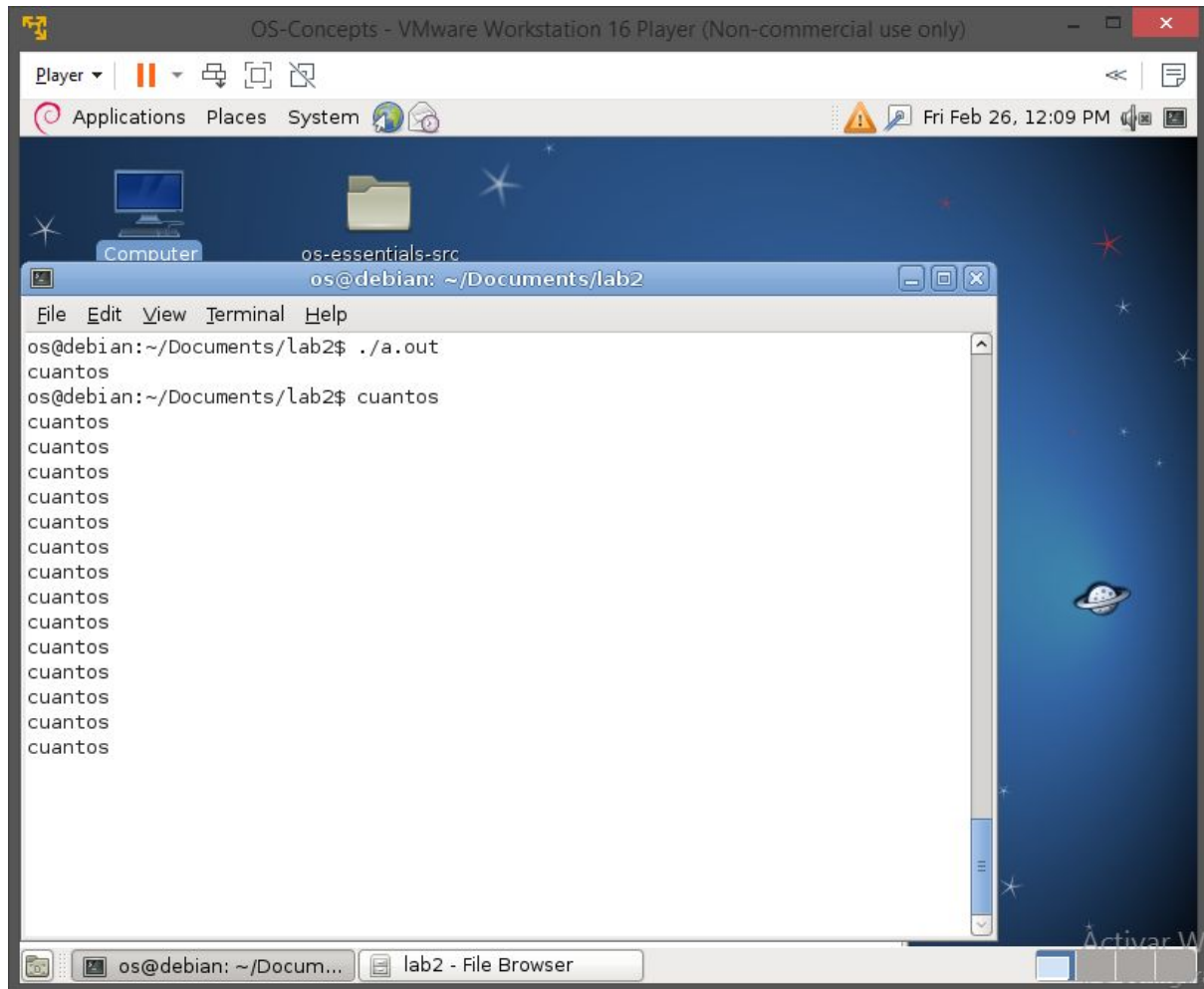
Si no está apagada, apague su máquina virtual y revise que esté usando más de un procesador (recomendable, cuatro) en el menú de configuración de VirtualBox. De no ser así, configúrela para que use más de un procesador (o cámbiese de computadora host a una multinúcleo).

Cree un programa en C que ejecute cuatro fork()s consecutivos. Luego cree otro programa en C que ejecute fork() dentro de un ciclo for de cuatro iteraciones.

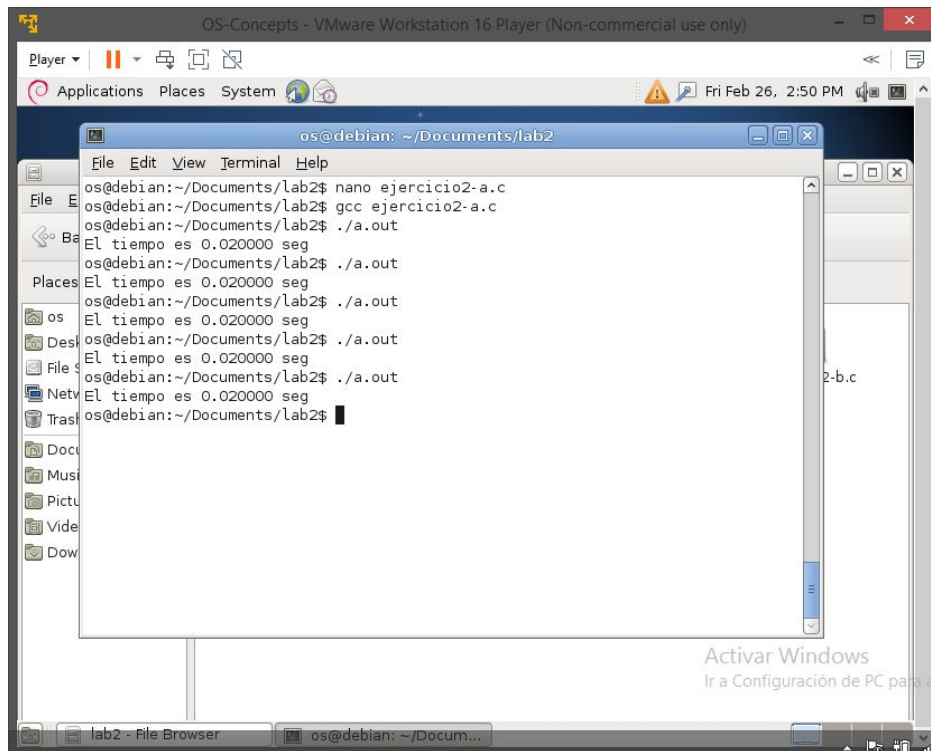
- ¿Cuántos procesos se crean en cada uno de los programas?



```
OS-Concepts - VMware Workstation 16 Player (Non-commercial use only)
Player
Applications Places System
Fri Feb 26, 11:40 AM
Computer os-essentials-src
os@debian: ~/Documents/lab2
File Edit View Terminal Help
ejercicio1-a.c:10: error: 'print' was not declared in this scope
os@debian:~/Documents/lab2$ nano ejercicio1-a.c
os@debian:~/Documents/lab2$ gcc ejercicio1-a.c
os@debian:~/Documents/lab2$ g++ -o ejercicio1-a ejercicio1-a.c
ejercicio1-a.c: In function 'int main()':
ejercicio1-a.c:6: error: 'fork' was not declared in this scope
os@debian:~/Documents/lab2$ ./a.out
cuantas veces
os@debian:~/Documents/lab2$ cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
cuantas veces
Activar Windows
Ir a Configuración de PC para
```

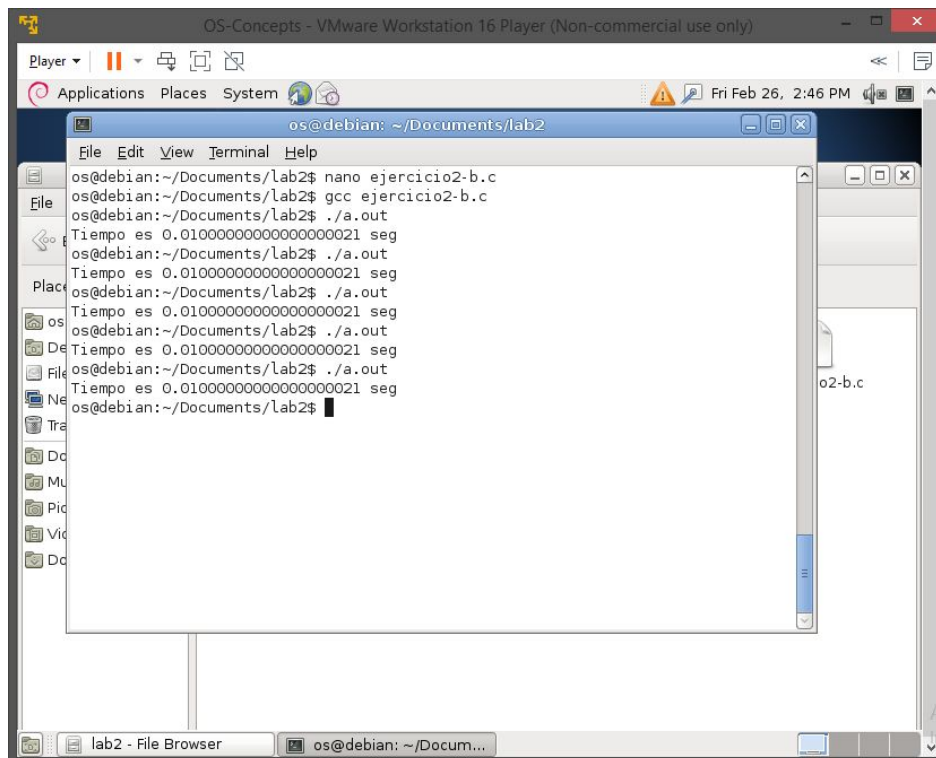


- ¿Por qué hay tantos procesos en ambos programas cuando uno tiene cuatro llamadas `fork()` y el otro sólo tiene una?
  - En el primer programa como tiene 4 forks se añade un proceso hijo por cada una y ellos otros hijos.
  - El otro programa, utilizando el ciclo cada vez que pasa por `fork` entonces esto se va incrementado dos ala  $n$  menos uno, como se ve hay 15 pero con el otro serían 16

Ejercicio 2 (20 puntos)

```
OS-Concepts - VMware Workstation 16 Player (Non-commercial use only)
Player | | | | |
Applications Places System | | | | |
Fri Feb 26, 2:50 PM | | | | |

os@debian: ~/Documents/lab2
File Edit View Terminal Help
os@debian:~/Documents/lab2$ nano ejercicio2-a.c
os@debian:~/Documents/lab2$ gcc ejercicio2-a.c
os@debian:~/Documents/lab2$ ./a.out
El tiempo es 0.020000 seg
os@debian:~/Documents/lab2$ ./a.out
El tiempo es 0.020000 seg
os@debian:~/Documents/lab2$ ./a.out
El tiempo es 0.020000 seg
os@debian:~/Documents/lab2$ ./a.out
El tiempo es 0.020000 seg
os@debian:~/Documents/lab2$ ./a.out
El tiempo es 0.020000 seg
os@debian:~/Documents/lab2$
```



```
OS-Concepts - VMware Workstation 16 Player (Non-commercial use only)
Player | | | | |
Applications Places System | | | | |
Fri Feb 26, 2:46 PM | | | | |

os@debian: ~/Documents/lab2
File Edit View Terminal Help
os@debian:~/Documents/lab2$ nano ejercicio2-b.c
os@debian:~/Documents/lab2$ gcc ejercicio2-b.c
os@debian:~/Documents/lab2$ ./a.out
Tiempo es 0.01000000000000000021 seg
os@debian:~/Documents/lab2$ ./a.out
Tiempo es 0.01000000000000000021 seg
os@debian:~/Documents/lab2$ ./a.out
Tiempo es 0.01000000000000000021 seg
os@debian:~/Documents/lab2$ ./a.out
Tiempo es 0.01000000000000000021 seg
os@debian:~/Documents/lab2$ ./a.out
Tiempo es 0.01000000000000000021 seg
os@debian:~/Documents/lab2$
```

- ¿Cuál, en general, toma tiempos más largos?
  - Primero: 0.020000
  - Segundo: 0.01000000000000000021
 Se tarda más el primero
- ¿Qué causa la diferencia de tiempo, o por qué se tarda más el que se tarda más?
  - La máquina virtual solo puede correr un proceso a la vez como es solo de cuatro núcleos, osea que si se emplea una máquina con más núcleos se ejecutarán más rápido.

### Ejercicio 3 (20 puntos)

```

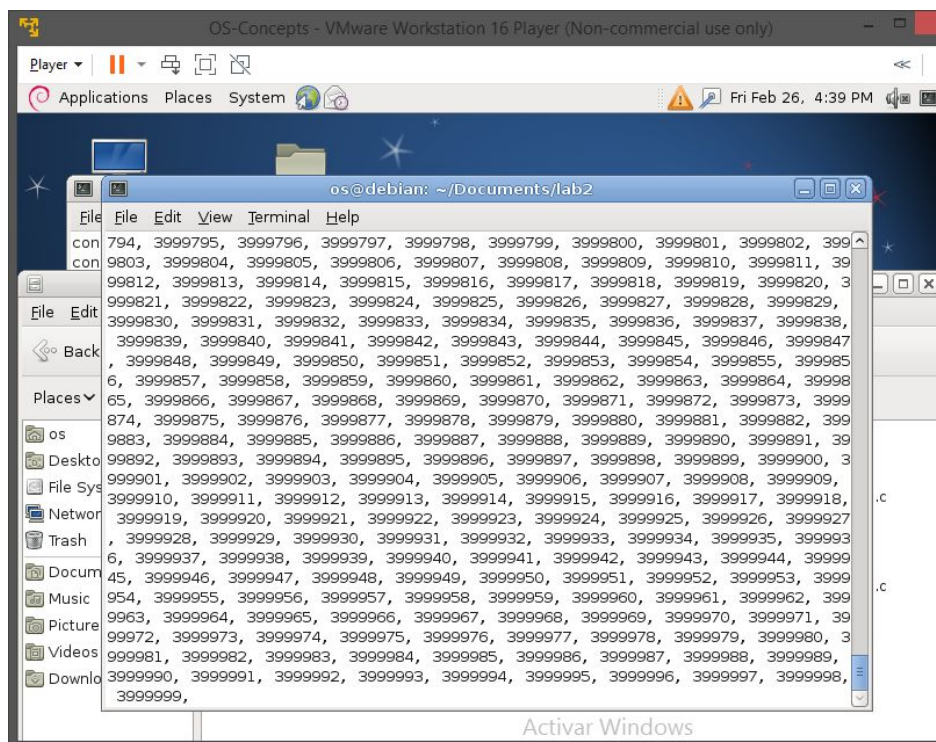
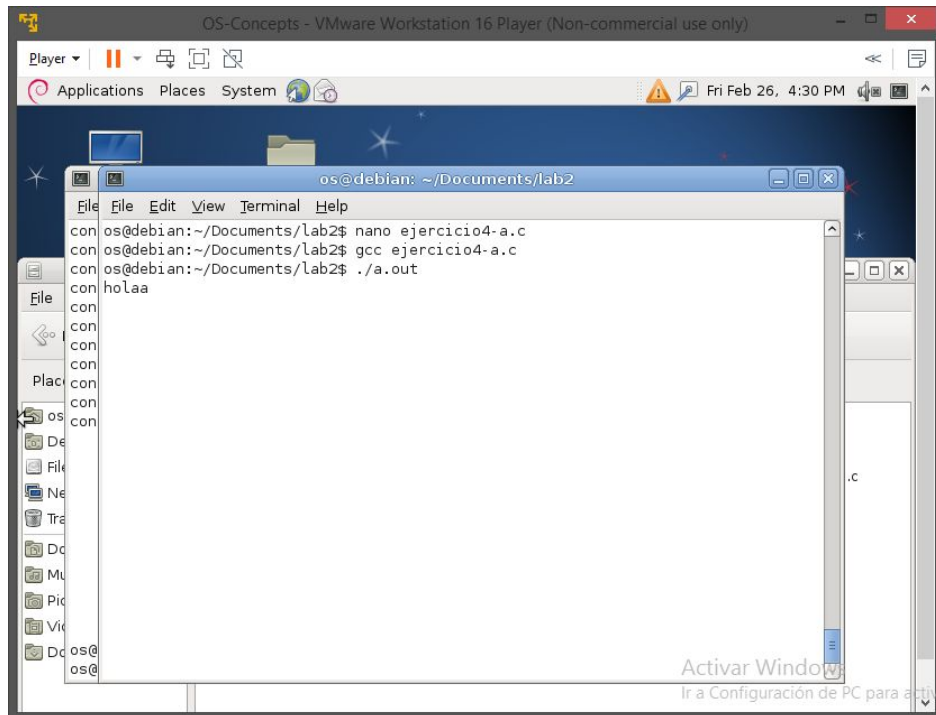
os@debian:~/Documents/lab2
File Edit View Terminal Help
sysstat-11.0.0/man/cif... 9998221 dx 9998231 dx 9998241 dx 9998251 dx 9998261 dx 9998271 dx 9998281 dx 9998291 dx
sysstat-11.0.0/man/sys... 9998301 dx 9998311 dx 9998321 dx 9998331 dx 9998341 dx 9998351 dx 9998361 dx 9998371 dx
sysstat-11.0.0/man/pid... 9998381 dx 9998391 dx 9998401 dx 9998411 dx 9998421 dx 9998431 dx 9998441 dx 9998451 dx
sysstat-11.0.0/man/nfs... 9998461 dx 9998471 dx 9998481 dx 9998491 dx 9998501 dx 9998511 dx 9998521 dx 9998531 dx
sysstat-11.0.0/man/mps... 9998541 dx 9998551 dx 9998561 dx 9998571 dx 9998581 dx 9998591 dx 9998601 dx 9998611 dx
sysstat-11.0.0/man/sad... 9998621 dx 9998631 dx 9998641 dx 9998651 dx 9998661 dx 9998671 dx 9998681 dx 9998691 dx
sysstat-11.0.0/man/sa2... 9998701 dx 9998711 dx 9998721 dx 9998731 dx 9998741 dx 9998751 dx 9998761 dx 9998771 dx
sysstat-11.0.0/sadf_mi... 9998781 dx 9998791 dx 9998801 dx 9998811 dx 9998821 dx 9998831 dx 9998841 dx 9998851 dx
sysstat-11.0.0/loconf... 9998861 dx 9998871 dx 9998881 dx 9998891 dx 9998901 dx 9998911 dx 9998921 dx 9998931 dx
sysstat-11.0.0/pr_stat... 9998941 dx 9998951 dx 9998961 dx 9998971 dx 9998981 dx 9998991 dx 9999001 dx 9999011 dx
sysstat-11.0.0/configu... 9999021 dx 9999031 dx 9999041 dx 9999051 dx 9999061 dx 9999071 dx 9999081 dx 9999091 dx
sysstat-11.0.0/activi... 9999101 dx 9999111 dx 9999121 dx 9999131 dx 9999141 dx 9999151 dx 9999161 dx 9999171 dx
sysstat-11.0.0/sar.c... 9999181 dx 9999191 dx 9999201 dx 9999211 dx 9999221 dx 9999231 dx 9999241 dx 9999251 dx
sysstat-11.0.0/lostat... 9999261 dx 9999271 dx 9999281 dx 9999291 dx 9999301 dx 9999311 dx 9999321 dx 9999331 dx
sysstat-11.0.0/rd_sens... 9999341 dx 9999351 dx 9999361 dx 9999371 dx 9999381 dx 9999391 dx 9999401 dx 9999411 dx
sysstat-11.0.0/preallo... 9999421 dx 9999431 dx 9999441 dx 9999451 dx 9999461 dx 9999471 dx 9999481 dx 9999491 dx
sysstat-11.0.0/sysstat... 9999501 dx 9999511 dx 9999521 dx 9999531 dx 9999541 dx 9999551 dx 9999561 dx 9999571 dx
sysstat-11.0.0/sa2.in... 9999581 dx 9999591 dx 9999601 dx 9999611 dx 9999621 dx 9999631 dx 9999641 dx 9999651 dx
os@debian:~$ cd sysstat-11.0.0
os@debian:~/sysstat-11.0.0$ sysstat -l 1.890000 segn
bash: ./config: No such file or directory
os@debian:~/sysstat-11.0.0$ sysstat -l 1.890000 segn
bash: pidstat: command not found
os@debian:~/sysstat-11.0.0$ sysstat -l 1.890000 segn
os@debian:~/Documents/lab2$
  
```

```

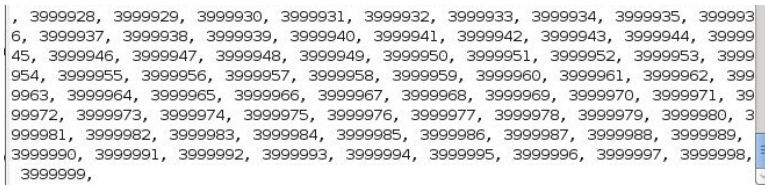
os@debian:~/Documents/lab2
File Edit View Terminal Help
config.status: creating m... 9998221 dx 9998231 dx 9998241 dx 9998251 dx 9998261 dx 9998271 dx 9998281 dx 9998291 dx
config.status: creating m... 9998301 dx 9998311 dx 9998321 dx 9998331 dx 9998341 dx 9998351 dx 9998361 dx 9998371 dx
config.status: creating m... 9998381 dx 9998391 dx 9998401 dx 9998411 dx 9998421 dx 9998431 dx 9998441 dx 9998451 dx
config.status: creating m... 9998461 dx 9998471 dx 9998481 dx 9998491 dx 9998501 dx 9998511 dx 9998521 dx 9998531 dx
config.status: creating m... 9998541 dx 9998551 dx 9998561 dx 9998571 dx 9998581 dx 9998591 dx 9998601 dx 9998611 dx
config.status: creating m... 9998621 dx 9998631 dx 9998641 dx 9998651 dx 9998661 dx 9998671 dx 9998681 dx 9998691 dx
config.status: creating m... 9998701 dx 9998711 dx 9998721 dx 9998731 dx 9998741 dx 9998751 dx 9998761 dx 9998771 dx
config.status: creating m... 9998781 dx 9998791 dx 9998801 dx 9998811 dx 9998821 dx 9998831 dx 9998841 dx 9998851 dx
config.status: creating m... 9998861 dx 9998871 dx 9998881 dx 9998891 dx 9998901 dx 9998911 dx 9998921 dx 9998931 dx
config.status: creating m... 9998941 dx 9998951 dx 9998961 dx 9998971 dx 9998981 dx 9998991 dx 9999001 dx 9999011 dx
config.status: creating m... 9999021 dx 9999031 dx 9999041 dx 9999051 dx 9999061 dx 9999071 dx 9999081 dx 9999091 dx
config.status: creating m... 9999101 dx 9999111 dx 9999121 dx 9999131 dx 9999141 dx 9999151 dx 9999161 dx 9999171 dx
Sysstat version: 9999181 dx 9999191 dx 9999201 dx 9999211 dx 9999221 dx 9999231 dx 9999241 dx 9999251 dx
Installation prefix: 9999261 dx 9999271 dx 9999281 dx 9999291 dx 9999301 dx 9999311 dx 9999321 dx 9999331 dx
rc directory: 9999341 dx 9999351 dx 9999361 dx 9999371 dx 9999381 dx 9999391 dx 9999401 dx 9999411 dx
init directory: 9999421 dx 9999431 dx 9999441 dx 9999451 dx 9999461 dx 9999471 dx 9999481 dx 9999491 dx
Sysstat unit dir: 9999501 dx 9999511 dx 9999521 dx 9999531 dx 9999541 dx 9999551 dx 9999561 dx 9999571 dx
Configuration director... 9999581 dx 9999591 dx 9999601 dx 9999611 dx 9999621 dx 9999631 dx 9999641 dx 9999651 dx
Man pages directory: 9999661 dx 9999671 dx 9999681 dx 9999691 dx 9999701 dx 9999711 dx 9999721 dx 9999731 dx
Compiler: 9999741 dx 9999751 dx 9999761 dx 9999771 dx 9999781 dx 9999791 dx 9999801 dx 9999811 dx
Compiler flags: 9999821 dx 9999831 dx 9999841 dx 9999851 dx 9999861 dx 9999871 dx 9999881 dx 9999891 dx
os@debian:~/sysstat-11.0.0$ sysstat -l 1.920000 segn
os@debian:~/sysstat-11.0.0$ sysstat -l 1.920000 segn
os@debian:~/Documents/lab2$
  
```

- ¿Qué tipo de cambios de contexto incrementa notablemente en cada caso, y por qué?  
El gnome de linux utiliza una herramienta llamada xorg, esto proporciona un entorno grafico:
  - Cambio voluntarios se ven en gnome-terminal
  - Cambios no voluntarios tiene solo la entrada de teclado. esto pasa cuando la gui cambia de manera inesperada
- ¿Qué diferencia hay en el número y tipo de cambios de contexto de entre programas?
  - Primero: 51 cswch/s voluntarios, 2 nvcswhs/s no voluntarios
  - Segundo: 21 cswch/s voluntarios, 1 nvcswhs/s no voluntarios
- ¿A qué puede atribuir los cambios de contexto voluntarios realizados por sus programas?
  - Solamente los voluntarios se muestran en los programas cuando la computadora no está utilizando la cpu osea suspendido.
- ¿A qué puede atribuir los cambios de contexto involuntarios realizados por sus programas?
  - Cuando se deja mucho tiempo el proceso corriendo puede llegar a expirar, esto significa que el sistema operativo decide realizar otro proceso.
- ¿Por qué el reporte de cambios de contexto para su programa con fork()s muestra cuatro procesos, uno de los cuales reporta cero cambios de contexto?
  - Los procesos que se muestran son los hijos de los hijos del padre sucesivamente hasta hacer 4, osea cuando tiene cero cambio de contexto se debe a que el sistema espera, de ahí la utilización en el programa de wait().
- ¿Qué efecto percibe sobre el número de cambios de contexto de cada tipo?
  - Yo veo que cuando se disminuyen los cambios de contexto no voluntarios se incrementan los voluntarios, ya que los recursos que este tiene son insuficientes.



Ejercicio 4 (10 puntos)

- ¿Qué significa la Z y a qué se debe?
  - La z viene de zombie, esto es porque el proceso ha terminado o algo ha salido mal y sus procesos secundarios siguen ejecutándose.
- ¿Qué sucede en la ventana donde ejecutó su programa?



```
, 3999928, 3999929, 3999930, 3999931, 3999932, 3999933, 3999934, 3999935, 3999936,
3999937, 3999938, 3999939, 3999940, 3999941, 3999942, 3999943, 3999944, 3999945,
3999946, 3999947, 3999948, 3999949, 3999950, 3999951, 3999952, 3999953, 3999954,
3999955, 3999956, 3999957, 3999958, 3999959, 3999960, 3999961, 3999962, 3999963,
3999964, 3999965, 3999966, 3999967, 3999968, 3999969, 3999970, 3999971, 3999972,
3999973, 3999974, 3999975, 3999976, 3999977, 3999978, 3999979, 3999980, 3999981,
3999982, 3999983, 3999984, 3999985, 3999986, 3999987, 3999988, 3999989, 3999990,
3999991, 3999992, 3999993, 3999994, 3999995, 3999996, 3999997, 3999998, 3999999,
```

La ventana se va ejecutando hasta que termine su tarea.

- ¿Quién es el padre del proceso que quedó huérfano?
  - El pobrecito padre que queda huérfano es init, y luego sigue ejecutándose hasta que termina.

### Ejercicio 5 (40 puntos)



```
b: Shared memory has: bbbbbbbbbbbbbbbbbbbbbbbbbbb
os@debian:~$
a: Shared memory has: bbbbbbbbbbbbbbbbbbbbbbbbbbaaa
```

- ¿Qué diferencia hay entre realizar comunicación usando memoria compartida en lugar de usando un archivo de texto común y corriente?
  - Tenemos memoria compartida y modelo de transición de mensajes, el primero la comunicación es mas rapida que el otro pero puede llegar a generar algunos errores.
- ¿Por qué no se debe usar el file descriptor de la memoria compartida producido por otra instancia para realizar el mmap?
  - Cuando el proceso pone en el mismo archivo su espacio de memoria virtual, el siguiente proceso puede llegar a tener diferentes protecciones. los que estén marcados como solo lectura pueden cambiar.
- ¿Es posible enviar el output de un programa ejecutado con exec a otro proceso por medio de un pipe? Investigue y explique cómo funciona este mecanismo en la terminal (e.g., la ejecución de ls | less).
  - Si es posible enviar el output de un programa ejecutado con otro proceso por medio de pipe, cuando se programa se puede usar dup() y dup2() para que el proceso hijo hereda los file descriptors, luego hay que conectar un último pipe a la entrada o salida estándar, se cierran los file descriptors con pipe.
- ¿Cómo puede asegurarse de que ya se ha abierto un espacio de memoria compartida con un nombre determinado? Investigue y explique errno.
  - Con <errno.h> cuando se tiene el objeto de memoria compartida, cuando tiera el mensaje se puede llegar e ello con shm\_open(), se establecen las variables con llamadas de sistema y funciones de biblioteca.

- ¿Qué pasa si se ejecuta `shm_unlink` cuando hay procesos que todavía están usando la memoria compartida?
  - El objeto se elimina ya que esta acción desvaría la memoria compartida aunque los procesos estén utilizando algún objeto. Después se cierran las referencias abiertas.
- ¿Cómo puede referirse al contenido de un espacio en memoria al que apunta un puntero? Observe que su programa deberá tener alguna forma de saber hasta dónde ha escrito su otra instancia en la memoria compartida para no escribir sobre ello.
  - Se usa los operadores:
    - Address of (&)
    - Value at Address (\*)Con \* podemos apuntar la variable a un puntero
- Imagine que una ejecución de su programa sufre un error que termina la ejecución prematuramente, dejando el espacio de memoria compartida abierto y provocando que nuevas ejecuciones se queden esperando el file descriptor del espacio de memoria compartida. ¿Cómo puede liberar el espacio de memoria compartida “manualmente”?
  - Si se desea eliminar un segmento de memoria compartida se puede usar debido a la version `shmctl(shm_id, ipc_rmid, null)`:
    - `shm_id`: memoria rápida
    - `ipc_rmid`: operación eliminación.
- Observe que el programa que ejecute dos instancias de `ipc.c` debe cuidar que una instancia no termine mucho antes que la otra para evitar que ambas instancias abran y cierren su propio espacio de memoria compartida. ¿Aproximadamente cuánto tiempo toma la realización de un `fork()`? Investigue y aplique `usleep`.
  - Según la máquina que se este corriendo el programa: osea en el mio serian alrededor de 200ms, pero en otras máquinas sería otro número.

