

Laboratorio 3

Ejercicio 1

fork() consecutivos = 15 procesos + el padre

```
oscreader@OSC:~/Desktop/OS/Lab3$ ./ejerca
1593
1594
1595
1597
0
1596
1598
1600
1593
0
1599
1601
0
1593
0
1599
0
1596
1598
0
1596
1602
oscreader@OSC:~/Desktop/OS/Lab3$ 0
1596
0
1593
1593
0
0
1594
1604
1595
0
0
1603
1605
1593
0
0
0
0
0
0
1606
0
1603
0
0
0
0
1593
1594
0
1607
1593
1594
0
0
0
0
```

fork() en for = 15 procesos + el padre

```
oscreader@OSC:~/Desktop/OS/Lab3$ ./ejercb
1666
0
1667
1668
0
1669
1670
1671
1672
1673
1674
0
0
1675
0
0
0
1676
1677
oscreader@OSC:~/Desktop/OS/Lab3$ 0
0
0
1678
0
0
1679
0
0
1680
0
0
```

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

En ambos programas se crean 16 procesos. Esto se obtiene al restar el PID del proceso mayor con el del proceso menor y se le suma él mismo (el padre). En ambos programas se obtiene de la resta 15.

- **Explique cómo se crea cada proceso y qué sucede después.**

Para crear los procesos se hace uso de la función `fork()`. Lo que esta función hace es generar un duplicado del proceso actual. El duplicado comparte los valores actuales de todas las variables, ficheros y estructuras de datos. Luego de esto se imprime el identificador de cada proceso (PID), este es el número que se le da a cada proceso cuando inicia. Los procesos durante su ejecución pasan por diferentes estados, entre ellos están:

- S(sleeping): proceso en espera.
- R(running): proceso en ejecución.
- T(stop): proceso parado.
- D: proceso bloqueado a la espera de un recurso.
- Z(zombie): proceso que ha finalizado pero que su proceso padre sigue en ejecución y no se ha “dado cuenta” de la circunstancia de su hijo.

Ejercicio 2

Primer programa

```
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2a
5984.000000
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2a
5492.000000
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2a
6247.000000
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2a
5517.000000
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2a
5428.000000
```

Segundo programa

```
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2b
El resultado es: (68.000000)
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2b
El resultado es: (69.000000)
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2b
El resultado es: (53.000000)
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2b
El resultado es: (55.000000)
osc reader@OSC:~/Desktop/OS/Lab3$ ./ejer2b
El resultado es: (83.000000)
```

Preguntas

- **¿Cuál, en general, toma tiempos más largos?**

El primer programa, ya que en promedio se tarda 5733.6 en ejecutarse, en cambio el segundo se realiza en promedio a los 65.6. Esto indica que es 98.86% más rápido que el primer programa.

- **¿Qué causa la diferencia de tiempo, o por qué se tarda más el que se tarda más?**

La diferencia radica en los procesos concurrentes, es decir, que los intervalos de tiempo se solapan.



Esto aporta a la rapidez en la que se ejecuta el segundo programa.

Ejercicio 3

- **Cambios de contexto voluntarios e involuntarios**

Tipos de contexto:

Cambio "voluntario": Proceso realiza llamada al sistema que implica esperar por evento Transición de en ejecución a bloqueado

Ejemplos: leer del terminal o bajar un semáforo cerrado

Motivo: Eficiencia en el uso del procesador

Cambio "involuntario": S.O. le quita la UCP al proceso.

Transición de en ejecución a listo

Ejemplos: fin de rodaja de ejecución o pasa a listo proceso bloqueado de mayor prioridad Motivo: Reparto del procesador

Cambios en Xorg

11:39:58 AM	0	776	658.00	2.00	Xorg
11:40:01 AM	0	776	571.00	5.00	Xorg
11:40:02 AM	0	776	591.09	13.86	Xorg
Average:	0	776	330.80	18.91	Xorg

Cambios en gnome-terminal

sldksk						
11:52:33 AM	UID	PID	cswch/s	nvcschw/s	Command	
11:52:34 AM	0	3	0.99	0.00	ksoftirqd/0	
11:52:34 AM	0	7	5.94	0.00	rcu_sched	
11:52:34 AM	0	13	1.98	0.00	ksoftirqd/1	
11:52:34 AM	0	98	0.99	0.00	kworker/u8:3	
11:52:34 AM	0	103	113.86	0.00	kworker/1:2	
11:52:34 AM	0	747	11.88	0.00	acpid	
11:52:34 AM	0	776	212.87	0.00	Xorg	
11:52:34 AM	1000	1135	18.81	279.21	gnome-shell	
11:52:34 AM	0	1534	0.99	0.00	kworker/2:0	
11:52:34 AM	0	1649	18.81	0.00	kworker/3:0	
11:52:34 AM	1000	1695	46.53	0.00	gnome-terminal -	
11:52:34 AM	1000	2876	0.99	112.87	pidstat	
ola						
11:52:37 AM	UID	PID	cswch/s	nvcschw/s	Command	
11:52:38 AM	0	7	5.00	0.00	rcu_sched	
11:52:38 AM	0	13	2.00	0.00	ksoftirqd/1	
11:52:38 AM	0	23	1.00	0.00	ksoftirqd/3	
11:52:38 AM	0	103	101.00	0.00	kworker/1:2	
11:52:38 AM	0	747	6.00	0.00	acpid	
11:52:38 AM	0	776	218.00	0.00	Xorg	
11:52:38 AM	1000	1135	21.00	230.00	gnome-shell	
11:52:38 AM	0	1534	1.00	0.00	kworker/2:0	
11:52:38 AM	0	1649	7.00	0.00	kworker/3:0	
11:52:38 AM	1000	1695	31.00	0.00	gnome-terminal -	
11:52:38 AM	0	1761	1.00	0.00	kworker/0:2	
11:52:38 AM	1000	2876	1.00	96.00	pidstat	
Average:	1000	1695	35.41	0.05	gnome-terminal -	

- **¿Qué tipo de cambios de contexto incrementa notablemente en cada caso, y por qué?**
Con Xorg se da un cambio de contexto involuntario ya que se están abriendo y cerrando ventanas, entre otras acciones que son de mayor importancia para el SO. En cambio, con gnome-terminal se realiza un cambio de contexto voluntario ya que se realiza una interrupción a un evento que está en ejecución.
- **Tiempo primer programa:** 15.93 segundos - 11.05 segundos - 10.87 segundos - 13 segundos
- **Tiempo segundo programa:** 18.05 segundos - 11.42 segundos - 11.69 segundos - 10.91 segundos

Programa sin Fork()

```

oscreader@OSC:~/Desktop/OS/Lab3$ pidstat -w 13 1
Linux 3.16.0-4-686-pae (OSC)    02/13/2019    _i686_ (4 CPU)

02:25:48 PM      UID        PID    cswch/s nvcschw/s  Command
02:26:01 PM        0          3      2.00     0.00  ksoftirqd/0
02:26:01 PM        0          6     75.10     0.00  kworker/u8:0
02:26:01 PM        0          7    139.28     0.00  rcu_sched
02:26:01 PM        0         10      0.23     0.00  watchdog/0
02:26:01 PM        0         11      0.23     0.00  watchdog/1
02:26:01 PM        0         13      3.00     0.00  ksoftirqd/1
02:26:01 PM        0         16      0.23     0.00  watchdog/2
02:26:01 PM        0         17      0.08     0.00  migration/2
02:26:01 PM        0         18      5.15     0.00  ksoftirqd/2
02:26:01 PM        0         21      0.23     0.00  watchdog/3
02:26:01 PM        0         22      0.15     0.00  migration/3
02:26:01 PM        0         23      3.00     0.00  ksoftirqd/3
02:26:01 PM        0         29      0.08     0.00  khungtaskd
02:26:01 PM        0        110      0.31     0.00  kworker/0:1H
02:26:01 PM        0        111      0.38     0.00  kworker/3:1H
02:26:01 PM        0        135      0.46     0.00  jbd2/sda1-8
02:26:01 PM        0        171      0.69     0.00  systemd-journal
02:26:01 PM        0        419      0.08     0.00  rpcbind
02:26:01 PM       104        463      0.08     0.00  dbus-daemon
02:26:01 PM        0        747      0.23     0.00  acpid
02:26:01 PM        0        776    534.36     3.54  Xorg
02:26:01 PM        0        905      0.08     0.00  upowerd
02:26:01 PM        0        960      0.15     0.00  packagekitd
02:26:01 PM      1000       1034      0.08     0.00  ssh-agent
02:26:01 PM      1000       1045      0.31     0.00  dbus-daemon
02:26:01 PM      1000       1048      1.61     0.00  at-spi2-registr
02:26:01 PM      1000       1059      0.15     0.00  gnome-settings-

```

02:26:01	PM	1000	1059	0.15	0.00	gnome-settings-
02:26:01	PM	1000	1098	2.31	0.00	xprop
02:26:01	PM	0	1099	0.08	0.00	udisksd
02:26:01	PM	1000	1135	60.72	559.11	gnome-shell
02:26:01	PM	1000	1177	0.08	0.00	evolution-alarm
02:26:01	PM	1000	1200	0.77	0.00	nautilus
02:26:01	PM	1000	1218	0.08	0.00	gconfd-2
02:26:01	PM	0	1649	121431.74	0.00	kworker/3:0
02:26:01	PM	1000	1695	245.96	168.18	gnome-terminal-
02:26:01	PM	1000	1699	0.15	0.00	bash
02:26:01	PM	0	1761	72833.51	0.08	kworker/0:2
02:26:01	PM	0	3305	3.23	0.00	kworker/u8:1
02:26:01	PM	0	3366	65453.27	0.00	kworker/1:3
02:26:01	PM	0	3388	9.07	0.00	kworker/1:0
02:26:01	PM	0	3403	198170.18	0.08	kworker/2:1
02:26:01	PM	1000	3407	0.08	0.00	pidstat

Average:	UID	PID	cswch/s	nvcschw/s	Command
Average:	0	3	2.00	0.00	ksoftirqd/0
Average:	0	6	75.10	0.00	kworker/u8:0
Average:	0	7	139.28	0.00	rcu_sched
Average:	0	10	0.23	0.00	watchdog/0
Average:	0	11	0.23	0.00	watchdog/1
Average:	0	13	3.00	0.00	ksoftirqd/1
Average:	0	16	0.23	0.00	watchdog/2
Average:	0	17	0.08	0.00	migration/2
Average:	0	18	5.15	0.00	ksoftirqd/2
Average:	0	21	0.23	0.00	watchdog/3
Average:	0	22	0.15	0.00	migration/3
Average:	0	23	3.00	0.00	ksoftirqd/3
Average:	0	29	0.08	0.00	khungtaskd
Average:	0	110	0.31	0.00	kworker/0:1H
Average:	0	111	0.38	0.00	kworker/3:1H

Average:	0	111	0.38	0.00	kworker/3:1H
Average:	0	135	0.46	0.00	jbd2/sda1-8
Average:	0	171	0.69	0.00	systemd-journal
Average:	0	419	0.08	0.00	rpcbind
Average:	104	463	0.08	0.00	dbus-daemon
Average:	0	747	0.23	0.00	acpid
Average:	0	776	534.36	3.54	Xorg
Average:	0	905	0.08	0.00	upowerd
Average:	0	960	0.15	0.00	packagekitd
Average:	1000	1034	0.08	0.00	ssh-agent
Average:	1000	1045	0.31	0.00	dbus-daemon
Average:	1000	1048	1.61	0.00	at-spi2-registr
Average:	1000	1059	0.15	0.00	gnome-settings-
Average:	1000	1098	2.31	0.00	xprop
Average:	0	1099	0.08	0.00	udisksd
Average:	1000	1135	60.72	559.11	gnome-shell
Average:	1000	1177	0.08	0.00	evolution-alarm
Average:	1000	1200	0.77	0.00	nautilus
Average:	1000	1218	0.08	0.00	gconfd-2
Average:	0	1649	121431.74	0.00	kworker/3:0
Average:	1000	1695	245.96	168.18	gnome-terminal-
Average:	1000	1699	0.15	0.00	bash
Average:	0	1761	72833.51	0.08	kworker/0:2
Average:	0	3305	3.23	0.00	kworker/u8:1
Average:	0	3366	65453.27	0.00	kworker/1:3
Average:	0	3388	9.07	0.00	kworker/1:0
Average:	0	3403	198170.18	0.08	kworker/2:1
Average:	1000	3407	0.08	0.00	pidstat

Programa con Fork()

```

oscreader@OSC:~/Desktop/OS/Lab3$ pidstat -w 10 1
Linux 3.16.0-4-686-pae (OSC)      02/13/2019      _i686_ (4 CPU)

02:30:00 PM      UID          PID    cswch/s nvcswh/s    Command
02:30:10 PM          0           2       0.10      0.00    kthreadd
02:30:10 PM          0           3       2.80      0.00    ksoftirqd/0
02:30:10 PM          0           6      93.61      0.00    kworker/u8:0
02:30:10 PM          0           7     135.36      0.00    rcu_sched
02:30:10 PM          0          10       0.20      0.00    watchdog/0
02:30:10 PM          0          11       0.20      0.00    watchdog/1
02:30:10 PM          0          12       0.10      0.00    migration/1
02:30:10 PM          0          13       4.60      0.00    ksoftirqd/1
02:30:10 PM          0          16       0.20      0.00    watchdog/2
02:30:10 PM          0          17       0.20      0.00    migration/2
02:30:10 PM          0          18       3.30      0.00    ksoftirqd/2
02:30:10 PM          0          21       0.20      0.00    watchdog/3
02:30:10 PM          0          23       5.89      0.00    ksoftirqd/3
02:30:10 PM          0         111       0.40      0.00    kworker/3:1H
02:30:10 PM          0         112       0.20      0.00    kworker/2:1H
02:30:10 PM          0         135       0.30      0.00    jbd2/sda1-8
02:30:10 PM          0         171       0.30      0.00    systemd-journal
02:30:10 PM          0         446       0.10      0.00    cron
02:30:10 PM          0         747       0.30      0.00    acpid
02:30:10 PM          0         776     379.72     10.89    Xorg

```


02:30:10 PM	0	776	379.72	10.89	Xorg
02:30:10 PM	0	960	0.10	0.00	packagekitd
02:30:10 PM	1000	1034	0.10	0.00	ssh-agent
02:30:10 PM	1000	1045	0.20	0.00	dbus-daemon
02:30:10 PM	1000	1048	0.80	0.00	at-spi2-registr
02:30:10 PM	1000	1059	0.20	0.00	gnome-settings-
02:30:10 PM	1000	1098	1.20	0.00	xprop
02:30:10 PM	1000	1135	98.40	445.95	gnome-shell
02:30:10 PM	1000	1200	0.30	0.00	nautilus
02:30:10 PM	0	1649	58638.66	0.50	kworker/3:0
02:30:10 PM	1000	1695	737.56	567.33	gnome-terminal -
02:30:10 PM	1000	1699	0.10	0.00	bash
02:30:10 PM	0	1761	63983.52	0.80	kworker/0:2
02:30:10 PM	0	3305	53.65	0.10	kworker/u8:1
02:30:10 PM	0	3366	45207.99	0.50	kworker/1:3
02:30:10 PM	0	3388	1.60	0.00	kworker/1:0
02:30:10 PM	0	3403	64581.42	0.90	kworker/2:1
02:30:10 PM	0	3410	5.49	0.00	kworker/1:1
02:30:10 PM	1000	3412	0.10	0.00	pidstat
02:30:10 PM	1000	3413	0.10	0.10	ejer3b
02:30:10 PM	1000	3414	6217.88	79147.85	ejer3b
02:30:10 PM	1000	3415	6533.87	77834.37	ejer3b
02:30:10 PM	1000	3416	6416.48	75487.11	ejer3b
02:30:10 PM	0	3417	0.30	0.00	kworker/1:2
Average:	UID	PID	cswch/s	nvcschw/s	Command
Average:	0	2	0.10	0.00	kthreadd
Average:	0	3	2.80	0.00	ksoftirqd/0
Average:	0	6	93.61	0.00	kworker/u8:0
Average:	0	7	135.36	0.00	rcu_sched
Average:	0	10	0.20	0.00	watchdog/0

Average:	0	11	0.20	0.00	watchdog/1
Average:	0	12	0.10	0.00	migration/1
Average:	0	13	4.60	0.00	ksoftirqd/1
Average:	0	16	0.20	0.00	watchdog/2
Average:	0	17	0.20	0.00	migration/2
Average:	0	18	3.30	0.00	ksoftirqd/2
Average:	0	21	0.20	0.00	watchdog/3
Average:	0	23	5.89	0.00	ksoftirqd/3
Average:	0	111	0.40	0.00	kworker/3:1H
Average:	0	112	0.20	0.00	kworker/2:1H
Average:	0	135	0.30	0.00	jbd2/sda1-8
Average:	0	171	0.30	0.00	systemd-journal
Average:	0	446	0.10	0.00	cron
Average:	0	747	0.30	0.00	acpid
Average:	0	776	379.72	10.89	Xorg
Average:	0	960	0.10	0.00	packagekitd
Average:	1000	1034	0.10	0.00	ssh-agent
Average:	1000	1045	0.20	0.00	dbus-daemon
Average:	1000	1048	0.80	0.00	at-spi2-registr
Average:	1000	1059	0.20	0.00	gnome-settings-
Average:	1000	1098	1.20	0.00	xprop
Average:	1000	1135	98.40	445.95	gnome-shell
Average:	1000	1200	0.30	0.00	nautilus
Average:	0	1649	58638.66	0.50	kworker/3:0
Average:	1000	1695	737.56	567.33	gnome-terminal-
Average:	1000	1699	0.10	0.00	bash
Average:	0	1761	63983.52	0.80	kworker/0:2
Average:	0	3305	53.65	0.10	kworker/u8:1
Average:	0	3366	45207.99	0.50	kworker/1:3
Average:	0	3388	1.60	0.00	kworker/1:0
Average:	0	3403	64581.42	0.90	kworker/2:1
Average:	0	3410	5.49	0.00	kworker/1:1

Average:	0	3305	53.65	0.10	kworker/u8:1
Average:	0	3366	45207.99	0.50	kworker/1:3
Average:	0	3388	1.60	0.00	kworker/1:0
Average:	0	3403	64581.42	0.90	kworker/2:1
Average:	0	3410	5.49	0.00	kworker/1:1
Average:	1000	3412	0.10	0.00	pidstat
Average:	1000	3413	0.10	0.10	ejer3b
Average:	1000	3414	6217.88	79147.85	ejer3b
Average:	1000	3415	6533.87	77834.37	ejer3b
Average:	1000	3416	6416.48	75487.11	ejer3b
Average:	0	3417	0.30	0.00	kworker/1:2

- ¿Qué diferencia hay en el número y tipo de cambios de contexto de entre programas?

En ambos hay una mayor cantidad de cambios voluntarios. Sin embargo, en el programa del fork se crea 4 procesos con un mayor cambio de contextos involuntarios.

- **¿A qué puede atribuir los cambios de contexto voluntarios realizados por sus programas?**

Se le atribuye a las ejecuciones de terminal y llamadas a sistema.

- **¿A qué puede atribuir los cambios de contexto involuntarios realizados por sus programas?**

A la prioridad dada a cada acción que realiza el cambio de contexto.

- **¿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?**

Debido a que el programa tiene 4 procesos concurrentes, que se ejecutan de hijo en hijo. El cambio de contexto nulo se debe a que el padre debe esperar hasta que termine el último proceso termine, para poder terminar su propio proceso.

```
02:55:00 PM 1000 3482 1.00 1.00 ejer3b
02:55:00 PM 1000 3483 6038.00 67605.00 ejer3b
02:55:00 PM 1000 3484 6524.00 78944.00 ejer3b
02:55:00 PM 1000 3485 10029.00 71997.00 ejer3b
```

```
02:55:01 PM 1000 3483 6821.00 100682.00 ejer3b
02:55:01 PM 1000 3484 8380.00 87249.00 ejer3b
02:55:01 PM 1000 3485 8839.00 87013.00 ejer3b
```

```
02:55:09 PM 1000 3483 7387.00 81288.00 ejer3b
02:55:09 PM 1000 3484 5432.00 74091.00 ejer3b
02:55:09 PM 1000 3485 8148.00 108658.00 ejer3b
```

```
02:55:10 PM 1000 3483 3102.00 32579.00 ejer3b
02:55:10 PM 1000 3484 4486.00 62020.00 ejer3b
02:55:10 PM 1000 3485 4464.00 48301.00 ejer3b
```

- **¿Qué efecto percibe sobre el número de cambios de contexto de cada tipo?**

Ambos contextos varían de diferente manera, en algunos casos el último proceso que correspondía al ejercicio mostraba un cambio de contexto voluntario mayor, pero en general, los cambios de contexto involuntarios mostraban una diferencia mayor al compararlos con los voluntarios. Esto es debido a las interrupciones realizadas por nosotros como usuario, lo que contribuye una prioridad mayor en el SO.

Ejercicio 4


```

oscreader@OSC:~/Desktop/OS/Lab3$ ps -ae1
  PID TTY          STAT       TIME COMMAND
    1 ?            Ss          0:02 /sbin/init auto
   758 tty1        Ss+         0:00 /sbin/agetty --noclear tty1 linux
   778 tty7        Ssl+        0:09 /usr/bin/Xorg :0 -novtswitch -background none -norese
  1349 pts/0      Ss          0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscreader/D
  1429 pts/1      Ss          0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscreader/D
  1437 pts/0      R+          0:02 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESSION_ID=
  1438 pts/0      Z+          0:00 [ejer4] <defunct>
  1439 pts/1      R+          0:00 ps -ae1 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESSION_ID=

```

- ¿Qué significa la Z y a qué se debe?

Significa “zombie”. Un proceso zombie o difunto es un proceso que ha completado su ejecución pero aún tiene una entrada en la tabla de procesos, permitiendo al proceso que lo ha creado leer el estado de su salida. Es decir, el proceso hijo ha terminado exitosamente pero el padre no lo ha reconocido. Esto se debe a que el padre necesita tiempo para pedirle al kernel información sobre los recursos que utilizó el hijo.

PROCESS STATE CODES

```

R  running or runnable (on run queue)
D  uninterruptible sleep (usually IO)
S  interruptible sleep (waiting for an event to complete)
Z  defunct/zombie, terminated but not reaped by its parent
T  stopped, either by a job control signal or because
    it is being traced
[...]
```

Ejecutando en terminal: ps -ae1

```

oscreader@OSC:~/Desktop/OS/Lab3$ ps -ae1
  PID TTY          STAT       TIME COMMAND
    1 ?            Ss          0:02 /sbin/init auto
   758 tty1        Ss+         0:00 /sbin/agetty --noclear tty1 linux
   778 tty7        Rsl+        0:15 /usr/bin/Xorg :0 -novtswitch -background no
  1349 pts/0      Ss          0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/o
  1429 pts/1      Ss          0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/o
  1464 pts/0      R+          0:02 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_S
  1465 pts/0      R+          0:01 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_S
  1466 pts/1      R+          0:00 ps -ae1 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_S

```

```

oscreader@OSC:~/Desktop/OS/Lab3$ ps -ael
  PID TTY          STAT       TIME COMMAND
    1  ?   I       Ss          0:02 /sbin/init auto
   758 tty1      Ss+        0:00 /sbin/agetty --noclear tty1 linux
   778 tty7      Ssl+       0:19 /usr/bin/Xorg :0 -novtswitch -background none
  1349 pts/0    Ss         0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscr
  1429 pts/1    Ss         0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscr
  1469 pts/0    R+         0:03 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESS
  1470 pts/0    R+         0:01 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESS
  1471 pts/1    R+         0:00 ps -ael XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESS
oscreader@OSC:~/Desktop/OS/Lab3$ kill -0 1469
oscreader@OSC:~/Desktop/OS/Lab3$ ps -ael
  PID TTY          STAT       TIME COMMAND
    1  ?   I       Ss          0:02 /sbin/init auto
   758 tty1      Ss+        0:00 /sbin/agetty --noclear tty1 linux
   778 tty7      Ssl+       0:20 /usr/bin/Xorg :0 -novtswitch -background none
  1349 pts/0    Ss         0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscr
  1429 pts/1    Ss         0:00 bash LANG=en_US.utf8 XDG_VTNR=7 PWD=/home/oscr
  1469 pts/0    R+         0:20 ./ejer4 XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESS
  1470 pts/0    Z+         0:08 [ejer4] <defunct>
  1473 pts/1    R+         0:00 ps -ael XDG_VTNR=7 SSH_AGENT_PID=1045 XDG_SESS

```

- **¿Qué sucede en la ventana donde ejecutó su programa?**
El programa sigue corriendo sin interrupciones.
- **¿Quién es el padre del proceso que quedó huérfano?**
Cuando el proceso padre muere antes que el proceso hijo los sistemas unix suelen producir una adopción automática: el proceso es "reparented" al proceso init (PID 1).

Ejercicio 5

magic: <https://www.geeksforgeeks.org/ipc-using-message-queues/>

- **¿Qué diferencia hay entre realizar comunicación usando memoria compartida en lugar de usando un archivo común y corriente?**
Utilizar memoria compartida permite la comunicación directa e inmediato acceso a la información construida por otro proceso, en cambio al utilizar un archivo se debe copiar la data nuevamente como mensaje para el proceso que lo necesite. Sin embargo, se corre el riesgo de dañar la memoria del proceso que la creó.
- **¿Por qué no se debe usar el file descriptor de la memoria compartida producido por otra instancia para realizar el mmap?**
Al realizar el mmap se crea un mapeo a la memoria en algún lugar en la máquina virtual. Al especificar el file descriptor permite que la memoria sea intercambiada al disco. No se debe utilizar el mismo fd para diferentes instancias del mmap ya que puede causar un fallo entre las comunicaciones y corromper la información en el espacio que se accede de la memoria compartida.
- **¿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).**
Sí. Al terminar de ejecutar el comando ls, el resultado de ls es el output y se envía como input a less. El trabajo de interpretar el símbolo de pipe como una instrucción

de ejecutar múltiples procesos y canalizar la salida de un proceso en la entrada de otro proceso es responsabilidad del shell.

- **¿Cómo puede asegurarse de que ya se ha abierto un espacio de memoria compartida con un nombre determinado? Investigue y explique errno.**

Al utilizar la función `shmget()`, esta devuelve el identificador del segmento de la memoria compartida asociado con el key enviado como parámetro. Para saber si ya se ha abierto el espacio, dicha memoria compartida devuelve `errno`.

Ahora bien, el comando `errno(1)` se puede utilizar para buscar nombres y números específicos.

Link: <http://man7.org/linux/man-pages/man3/errno.3.html>

- **¿Qué pasa si se ejecuta `shm_unlink` cuando hay procesos que todavía están usando la memoria compartida?**

Para poder explicar lo que sucede, es importante entender qué hace `shm_unlink`.

`shm_open()` crea y abre un nuevo, o abre un objeto de memoria compartida POSIX existente. Un objeto de memoria compartida POSIX es en efecto un identificador que puede ser usado por procesos no relacionados con `mmap()` (2) la misma región de la memoria compartida. La función `shm_unlink()` realiza la operación inversa, eliminando un objeto creado previamente por `shm_open()`.

Al ejecutarlo borra el nombre, pero no borra el espacio de memoria. Espera hasta que terminen de ejecutarse los procesos que utilizan la memoria compartida y hasta que todos terminan, se elimina el espacio de memoria.

- **¿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.**

No es posible mandar punteros entre procesos, pero se puede enviar el offset desde el momento en que se instancia la memoria compartida.

Para esto es necesario instanciar el puntero `void* ptr`, el cual se asigna a la función `mmap()`.

- **Imagine que una ejecución de su programa sufre un error que termina la ejecución prematuramente, dejando el espacio de memoria compartido 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”?**

Para ver la memoria compartida se usa el comando `ipcs`.

Para ser capaces de liberar/eliminar segmentos de la memoria compartida se utiliza `ipcrm`.

- **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.**

Un fork() tarda aproximadamente 0.000048 s
usleep() suspende la ejecución en microsegundos.