Práctica 2.3. Procesos

Objetivos

En esta práctica se revisan las funciones del sistema básicas para la gestión de procesos: políticas de planificación, creación de procesos, grupos de procesos, sesiones, recursos de un proceso y gestión de señales.

Contenidos

Preparación del entorno para la práctica
Políticas de planificación
Grupos de procesos y sesiones
Ejecución de programas
Señales

Preparación del entorno para la práctica

Algunos de los ejercicios de esta práctica requieren permisos de superusuario para poder fijar algunos atributos de un proceso, ej. políticas de tiempo real. Por este motivo, es recomendable realizarla en una **máquina virtual** en lugar de las máquinas físicas del laboratorio.

Políticas de planificación

En esta sección estudiaremos los parámetros del planificador de Linux que permiten variar y consultar la prioridad de un proceso. Veremos tanto la interfaz del sistema como algunos comandos importantes.

Ejercicio 1. La política de planificación y la prioridad de un proceso puede consultarse y modificarse con el comando `chrt`. Adicionalmente, los comandos `nice` y `renice` permiten ajustar el valor de *nice* de un proceso. Consultar la página de manual de ambos comandos y comprobar su funcionamiento cambiando el valor de *nice* de la *shell* a -10 y después cambiando su política de planificación a `SCHED_FIFO` con prioridad 12.

```
[cursoredes@localhost ~]$ sudo renice -n -10 $$
2333 (process ID) old priority 0, new priority -10
[cursoredes@localhost ~]$ sudo chrt -f -p 12 $$
[cursoredes@localhost ~]$ chrt -p $$
pid 2333's current scheduling policy: SCHED_FIFO
pid 2333's current scheduling priority: 12
```

Ejercicio 2. Escribir un programa que muestre la política de planificación (como cadena) y la prioridad del proceso actual, además de mostrar los valores máximo y mínimo de la prioridad para la política de planificación.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>

int main(){
   int sched_policy = sched_getscheduler(0);
   if(sched_policy == SCHED_FIFO){
      printf("FIFO\n");
   }
   if(sched_policy == SCHED_RR){
      printf("ROUND ROBIN\n");
   }
   if(sched_policy == SCHED_OTHER){
      printf("OTHER\n");
   }
   int max = sched_get_priority_max(sched_policy);
   int min = sched_get_priority_min(sched_policy);
   struct sched_param param;
   int error = sched_getparam(0, &param);
   if (error != 0){
      printf("Error");
      return -1;
   }
   int curr_priority = param.sched_priority;

   printf("La prioridad es: %i\n", curr_priority);
   printf("EL máximo y el mínimo es: %i,%i\n", max,min);
   return 1;

}
```

Ejercicio 3. Ejecutar el programa anterior en una *shell* con prioridad 12 y política de planificación `SCHED_FIFO` como la del ejercicio 1. ¿Cuál es la prioridad en este caso del programa? ¿Se heredan los atributos de planificación?

TERMINAL:

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej2
[cursoredes@localhost ~]$ ./ej2
FIFO
La prioridad es: 12
EL máximo y el mínimo es: 99,1
```

Cuando se hace fork, se heredan los atributos de planificación.

Grupos de procesos y sesiones

Los grupos de procesos y sesiones simplifican la gestión que realiza la *shell*, ya que permite enviar de forma efectiva señales a un grupo de procesos (suspender, reanudar, terminar…). En esta sección veremos esta relación y estudiaremos el interfaz del sistema para controlarla.

Ejercicio 4. El comando ps es de especial importancia para ver los procesos del sistema y su estado. Estudiar la página de manual y:

- Mostrar todos los procesos del usuario actual en formato extendido.
- Mostrar los procesos del sistema, incluyendo el identificador del proceso, el identificador del grupo de procesos, el identificador de sesión, el estado y la línea de comandos.
- Observar el identificador de proceso, grupo de procesos y sesión de los procesos. ¿Qué identificadores comparten la *shell* y los programas que se ejecutan en ella? ¿Cuál es el identificador de grupo de procesos cuando se crea un nuevo proceso?

1º:
[cursoredes@localhost ~]$ ps -ef

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 11:00 ?        00:00:01 /usr/lib/systemd/systemd --switched-root --system
--deserialize 22
root         2     0  0 11:00 ?        00:00:00 [kthreadd]
root         3     2  0 11:00 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 11:00 ?        00:00:00 [kworker/0:0H]
root         7     2  0 11:00 ?        00:00:00 [migration/0]
root         8     2  0 11:00 ?        00:00:00 [rcu_bh]
root         9     2  0 11:00 ?        00:00:00 [rcu_sched]
root        10     2  0 11:00 ?        00:00:00 [lru-add-drain]
root        11     2  0 11:00 ?        00:00:00 [watchdog/0]
root        13     2  0 11:00 ?        00:00:00 [kdevtmpfs]
root        14     2  0 11:00 ?        00:00:00 [netns]
root        15     2  0 11:00 ?        00:00:00 [khungtaskd]
root        16     2  0 11:00 ?        00:00:00 [writeback]
root        17     2  0 11:00 ?        00:00:00 [kintegrityd]
root        18     2  0 11:00 ?        00:00:00 [bioset]
root        19     2  0 11:00 ?        00:00:00 [bioset]
root        20     2  0 11:00 ?        00:00:00 [bioset]
root        21     2  0 11:00 ?        00:00:00 [kblockd]
root        22     2  0 11:00 ?        00:00:00 [md]
root        23     2  0 11:00 ?        00:00:00 [edac-poller]
root        29     2  0 11:00 ?        00:00:00 [kswapd0]
root        30     2  0 11:00 ?        00:00:00 [ksmd]
root        31     2  0 11:00 ?        00:00:00 [khugepaged]
root        32     2  0 11:00 ?        00:00:00 [crypto]
root        40     2  0 11:00 ?        00:00:00 [kthrotld]
root        42     2  0 11:00 ?        00:00:00 [kmpath_rdacd]
root        43     2  0 11:00 ?        00:00:00 [kaluad]
root        44     2  0 11:00 ?        00:00:00 [kpsmoused]
root        45     2  0 11:00 ?        00:00:00 [ipv6_addrconf]
root        59     2  0 11:00 ?        00:00:00 [deferwq]
root        90     2  0 11:00 ?        00:00:00 [kauditd]
root       267     2  0 11:00 ?        00:00:00 [ata_sff]
```

```
root     273    2  0 11:00 ?      00:00:00 [scsi_eh_0]
root     275    2  0 11:00 ?      00:00:00 [scsi_tmf_0]
root     276    2  0 11:00 ?      00:00:00 [scsi_eh_1]
root     277    2  0 11:00 ?      00:00:00 [scsi_tmf_1]
root     278    2  0 11:00 ?      00:00:00 [scsi_eh_2]
root     280    2  0 11:00 ?      00:00:00 [kworker/u2:3]
root     281    2  0 11:00 ?      00:00:00 [scsi_tmf_2]
root     311    2  0 11:00 ?      00:00:00 [kworker/0:1H]
root     352    2  0 11:00 ?      00:00:00 [kdmflush]
root     353    2  0 11:00 ?      00:00:00 [bioset]
root     363    2  0 11:00 ?      00:00:00 [kdmflush]
root     364    2  0 11:00 ?      00:00:00 [bioset]
root     376    2  0 11:00 ?      00:00:00 [bioset]
root     377    2  0 11:00 ?      00:00:00 [xfsalloc]
root     378    2  0 11:00 ?      00:00:00 [xfs_mru_cache]
root     379    2  0 11:00 ?      00:00:00 [xfs-buf/dm-0]
root     380    2  0 11:00 ?      00:00:00 [xfs-data/dm-0]
root     381    2  0 11:00 ?      00:00:00 [xfs-conv/dm-0]
root     382    2  0 11:00 ?      00:00:00 [xfs-cil/dm-0]
root     383    2  0 11:00 ?      00:00:00 [xfs-reclaim/dm-]
root     384    2  0 11:00 ?      00:00:00 [xfs-log/dm-0]
root     385    2  0 11:00 ?      00:00:00 [xfs-eofblocks/d]
root     386    2  0 11:00 ?      00:00:00 [xfsaild/dm-0]
root     457    1  0 11:00 ?      00:00:00 /usr/lib/systemd/systemd-journald
root     473    1  0 11:00 ?      00:00:00 /usr/sbin/lvmetad -f
root     480    1  0 11:00 ?      00:00:00 /usr/lib/systemd/systemd-udevd
root     593    2  0 11:00 ?      00:00:00 [iprt-VBoxWQueue]
root     660    2  0 11:00 ?      00:00:00 [xfs-buf/sda1]
root     662    2  0 11:00 ?      00:00:00 [xfs-data/sda1]
root     664    2  0 11:00 ?      00:00:00 [xfs-conv/sda1]
root     671    2  0 11:00 ?      00:00:00 [xfs-cil/sda1]
root     673    2  0 11:00 ?      00:00:00 [xfs-reclaim/sda]
root     675    2  0 11:00 ?      00:00:00 [xfs-log/sda1]
root     677    2  0 11:00 ?      00:00:00 [xfs-eofblocks/s]
root     679    2  0 11:00 ?      00:00:00 [xfsaild/sda1]
root     709    2  0 11:00 ?      00:00:00 [ttm_swap]
root     741    1  0 11:00 ?      00:00:00 /sbin/auditd
root     743   741  0 11:00 ?      00:00:00 /sbin/audispd
root     745   743  0 11:00 ?      00:00:00 /usr/sbin/sedispatch
root     748    2  0 11:00 ?      00:00:00 [rpciod]
root     749    2  0 11:00 ?      00:00:00 [xprtiod]
polkitd   771    1  0 11:00 ?      00:00:00 /usr/lib/polkit-1/polkitd --no-debug
libstor+  772    1  0 11:00 ?      00:00:00 /usr/bin/lsmd -d
root     773    1  0 11:00 ?      00:00:00 /sbin/rngd -f
root     775    1  0 11:00 ?      00:00:00 /usr/sbin/alsactl -s -n 19 -c -E
ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib
rpc      783    1  0 11:00 ?      00:00:00 /sbin/rpcbind -w
root     791    1  0 11:00 ?      00:00:00 /usr/libexec/udisks2/udisksd
rtkit    792    1  0 11:00 ?      00:00:00 /usr/libexec/rtkit-daemon
root     801    1  0 11:00 ?      00:00:00 /usr/sbin/smartd -n -q never
dbus     804    1  0 11:00 ?      00:00:00 /usr/bin/dbus-daemon --system --address=systemd:
--nofork --nopidfile --systemd-activati
chrony   807    1  0 11:00 ?      00:00:00 /usr/sbin/chronyd
root     808    1  0 11:00 ?      00:00:00 /usr/sbin/gssproxy -D
root     824    1  0 11:00 ?      00:00:00 /usr/sbin/ModemManager
root     830    1  0 11:00 ?      00:00:00 /usr/lib/systemd/systemd-logind
root     832    1  0 11:00 ?      00:00:00 /usr/sbin/abrtd -d -s
```

```
root      836    1  0 11:00 ?       00:00:00 /usr/bin/abrt-watch-log -F BUG: WARNING: at WARNING:
CPU: INFO: possible recursive locki
root      843    1  0 11:00 ?       00:00:00 /usr/bin/abrt-watch-log -F Backtrace /var/log/Xorg.0.log --
/usr/bin/abrt-dump-xorg -xD
root      848    1  0 11:00 ?       00:00:00 /usr/libexec/accounts-daemon
root      864    1  0 11:00 ?       00:00:00 /usr/sbin/mcelog --ignorenodev --daemon --syslog
root      889    1  0 11:00 ?       00:00:00 /bin/bash /usr/sbin/ksmtuned
root     1025    1  0 11:00 ?       00:00:00 /usr/sbin/cupsd -f
root     1026    1  0 11:00 ?       00:00:00 /usr/bin/python -Es /usr/sbin/tuned -l -P
root     1027    1  0 11:00 ?       00:00:00 /usr/sbin/sshd -D
root     1030    1  0 11:00 ?       00:00:00 /usr/sbin/rsyslogd -n
root     1043    1  0 11:00 ?       00:00:00 /usr/sbin/crond -n
root     1045    1  0 11:00 ?       00:00:00 /usr/sbin/atd -f
root     1308    1  0 11:00 ?       00:00:00 /usr/libexec/postfix/master -w
postfix  1312  1308  0 11:00 ?       00:00:00 pickup -l -t unix -u
postfix  1313  1308  0 11:00 ?       00:00:00 qmgr -l -t unix -u
root     1331    1  0 11:00 ?       00:00:00 /usr/sbin/gdm
root     1348    1  0 11:00 ?       00:00:01 /usr/sbin/VBoxService --pidfile
/var/run/vboxadd-service.sh
root     1358  1331  0 11:00 tty1    00:00:14 /usr/bin/X :0 -background none -noreset -audit 4
-verbose -auth /run/gdm/auth-for-gdm-ea
root     1369  1331  0 11:00 ?       00:00:00 gdm-session-worker [pam/gdm-autologin]
cursore+ 1374  1369  0 11:00 ?       00:00:00 /usr/bin/openbox --startup
/usr/libexec/openbox-autostart OPENBOX
cursore+ 1383    1  0 11:00 ?       00:00:00 dbus-launch --sh-syntax --exit-with-session
cursore+ 1384    1  0 11:00 ?       00:00:00 /usr/bin/dbus-daemon --fork --print-pid 5 --print-address
7 --session
cursore+ 1452    1  0 11:00 ?       00:00:00 /usr/libexec/imsettings-daemon
cursore+ 1456    1  0 11:00 ?       00:00:00 /usr/libexec/gvfsd
cursore+ 1461    1  0 11:00 ?       00:00:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f -o
big_writes
cursore+ 1548    1  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --clipboard
cursore+ 1550  1548  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --clipboard
cursore+ 1558    1  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --display
cursore+ 1560  1558  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --display
cursore+ 1566    1  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --seamless
cursore+ 1567  1566  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --seamless
cursore+ 1572    1  0 11:00 ?       00:00:00 /usr/bin/VBoxClient --draganddrop
cursore+ 1576  1572  0 11:00 ?       00:00:10 /usr/bin/VBoxClient --draganddrop
cursore+ 1582  1374  0 11:00 ?       00:00:00 /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c
"/usr/bin/openbox-session"
cursore+ 1605    1  0 11:00 ?       00:00:00 tint2
cursore+ 1607    1  0 11:00 ?       00:00:00 /usr/bin/python
/usr/share/system-config-printer/applet.py
cursore+ 1608    1  0 11:00 ?       00:00:00 nm-applet
cursore+ 1610    1  0 11:00 ?       00:00:00 abrt-applet
cursore+ 1642    1  0 11:00 ?       00:00:00 /usr/libexec/at-spi-bus-launcher
cursore+ 1648    1  0 11:00 ?       00:00:00 /usr/bin/pulseaudio --start --log-target=syslog
cursore+ 1662  1642  0 11:00 ?       00:00:00 /bin/dbus-daemon
--config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork -
cursore+ 1677    1  0 11:00 ?       00:00:00 /usr/libexec/at-spi2-registryd --use-gnome-session
cursore+ 1721    1  0 11:00 ?       00:00:00 /usr/libexec/dconf-service
cursore+ 2052  1605  0 11:07 ?       00:00:26 /usr/share/code/code --unity-launch
cursore+ 2054  2052  0 11:07 ?       00:00:00 /usr/share/code/code --type=zygote --no-sandbox
cursore+ 2098  2054  1 11:07 ?       00:00:51 /usr/share/code/code --type=renderer
--js-flags=--nolazy --disable-mojo-local-storage --
cursore+ 2141  2098  0 11:07 ?       00:00:02 /usr/share/code/code
```

```
/usr/share/code/resources/app/out/bootstrap --type=extensionHost
cursore+  2153  2054  0 11:07 ?       00:00:01 /usr/share/code/code --type=renderer
--js-flags=--nolazy --disable-mojo-local-storage --
cursore+  2299     1  0 11:19 ?       00:00:05 /usr/libexec/gnome-terminal-server
cursore+  2304     1  0 11:19 ?       00:00:00 /usr/libexec/xdg-desktop-portal
cursore+  2309     1  0 11:19 ?       00:00:00 /usr/libexec/xdg-document-portal
cursore+  2313     1  0 11:19 ?       00:00:00 /usr/libexec/xdg-permission-store
cursore+  2325     1  0 11:19 ?       00:00:00 /usr/libexec/xdg-desktop-portal-gtk
cursore+  2332  2299  0 11:19 ?       00:00:00 gnome-pty-helper
cursore+  2333  2299  0 11:19 pts/0    00:00:00 bash
cursore+  2458  2299  0 11:21 pts/1    00:00:00 bash
cursore+  2637     1  0 11:28 ?       00:00:00 /usr/libexec/gvfs-udisks2-volume-monitor
cursore+  2642     1  0 11:28 ?       00:00:00 /usr/libexec/gvfs-afc-volume-monitor
cursore+  2648     1  0 11:28 ?       00:00:00 /usr/libexec/gvfs-gphoto2-volume-monitor
cursore+  2653     1  0 11:28 ?       00:00:00 /usr/libexec/gvfs-mtp-volume-monitor
cursore+  2658     1  0 11:28 ?       00:00:00 /usr/libexec/gvfs-goa-volume-monitor
cursore+  2662     1  0 11:28 ?       00:00:00 /usr/libexec/goa-daemon
cursore+  2671     1  0 11:28 ?       00:00:00 /usr/libexec/goa-identity-service
cursore+  2678     1  0 11:28 ?       00:00:00 /usr/libexec/mission-control-5
cursore+  2680     1  0 11:28 ?       00:00:00 /usr/libexec/gvfsd-trash --spawner :1.3
/org/gtk/gvfs/exec_spaw/0
cursore+  2701     1  0 11:28 ?       00:00:00 /usr/libexec/gvfsd-network --spawner :1.3
/org/gtk/gvfs/exec_spaw/1
cursore+  2718     1  0 11:28 ?       00:00:00 /usr/libexec/gvfsd-dnssd --spawner :1.3
/org/gtk/gvfs/exec_spaw/3
root      2765     2  0 11:29 ?       00:00:00 [kworker/u2:0]
root      2857     2  0 11:35 ?       00:00:00 [kworker/0:0]
root      2906     2  0 11:40 ?       00:00:00 [kworker/0:1]
root      3037     2  0 11:47 ?       00:00:00 [kworker/0:2]
cursore+  3101  2458  0 11:53 pts/1    00:00:00 man ps
cursore+  3112  3101  0 11:53 pts/1    00:00:00 less -s
root      3179   889  0 11:57 ?       00:00:00 sleep 60
cursore+  3180  2333  0 11:57 pts/0    00:00:00 ps -ef
```

2º:

```
[cursoredes@localhost ~]$ ps -eo pid,gid,sid,s,command
  PID   GID   SID S COMMAND
    1     0     1 S /usr/lib/systemd/systemd --switched-root --system --deserialize 22
    2     0     0 S [kthreadd]
    3     0     0 S [ksoftirqd/0]
    5     0     0 S [kworker/0:0H]
    7     0     0 S [migration/0]
    8     0     0 S [rcu_bh]
    9     0     0 R [rcu_sched]
   10     0     0 S [lru-add-drain]
   11     0     0 S [watchdog/0]
   13     0     0 S [kdevtmpfs]
   14     0     0 S [netns]
   15     0     0 S [khungtaskd]
   16     0     0 S [writeback]
   17     0     0 S [kintegrityd]
   18     0     0 S [bioset]
   19     0     0 S [bioset]
   20     0     0 S [bioset]
   21     0     0 S [kblockd]
   22     0     0 S [md]
   23     0     0 S [edac-poller]
```

```
 29    0    0 S [kswapd0]
 30    0    0 S [ksmd]
 31    0    0 S [khugepaged]
 32    0    0 S [crypto]
 40    0    0 S [kthrotld]
 42    0    0 S [kmpath_rdacd]
 43    0    0 S [kaluad]
 44    0    0 S [kpsmoused]
 45    0    0 S [ipv6_addrconf]
 59    0    0 S [deferwq]
 90    0    0 S [kauditd]
267    0    0 S [ata_sff]
273    0    0 S [scsi_eh_0]
275    0    0 S [scsi_tmf_0]
276    0    0 S [scsi_eh_1]
277    0    0 S [scsi_tmf_1]
278    0    0 S [scsi_eh_2]
280    0    0 S [kworker/u2:3]
281    0    0 S [scsi_tmf_2]
311    0    0 S [kworker/0:1H]
352    0    0 S [kdmflush]
353    0    0 S [bioset]
363    0    0 S [kdmflush]
364    0    0 S [bioset]
376    0    0 S [bioset]
377    0    0 S [xfsalloc]
378    0    0 S [xfs_mru_cache]
379    0    0 S [xfs-buf/dm-0]
380    0    0 S [xfs-data/dm-0]
381    0    0 S [xfs-conv/dm-0]
382    0    0 S [xfs-cil/dm-0]
383    0    0 S [xfs-reclaim/dm-]
384    0    0 S [xfs-log/dm-0]
385    0    0 S [xfs-eofblocks/d]
386    0    0 S [xfsaild/dm-0]
457    0  457 S /usr/lib/systemd/systemd-journald
473    0  473 S /usr/sbin/lvmetad -f
480    0  480 S /usr/lib/systemd/systemd-udevd
593    0    0 S [iprt-VBoxWQueue]
660    0    0 S [xfs-buf/sda1]
662    0    0 S [xfs-data/sda1]
664    0    0 S [xfs-conv/sda1]
671    0    0 S [xfs-cil/sda1]
673    0    0 S [xfs-reclaim/sda]
675    0    0 S [xfs-log/sda1]
677    0    0 S [xfs-eofblocks/s]
679    0    0 S [xfsaild/sda1]
709    0    0 S [ttm_swap]
741    0  741 S /sbin/auditd
743    0  743 S /sbin/audispd
745    0  743 S /usr/sbin/sedispatch
748    0    0 S [rpciod]
749    0    0 S [xprtiod]
771  998  771 S /usr/lib/polkit-1/polkitd --no-debug
772  993  772 S /usr/bin/lsmd -d
773    0  773 S /sbin/rngd -f
775    0  775 S /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf
```

```
--initfile=/lib/alsa/init/00main rdaemon
  783    32   783 S /sbin/rpcbind -w
  791     0   791 S /usr/libexec/udisks2/udisksd
  792   172   792 S /usr/libexec/rtkit-daemon
  801     0   801 S /usr/sbin/smartd -n -q never
  804    81   804 S /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile
--systemd-activation
  807   991   803 S /usr/sbin/chronyd
  808     0   808 S /usr/sbin/gssproxy -D
  824     0   824 S /usr/sbin/ModemManager
  830     0   830 S /usr/lib/systemd/systemd-logind
  832     0   832 S /usr/sbin/abrtd -d -s
  836     0   836 S /usr/bin/abrt-watch-log -F BUG: WARNING: at WARNING: CPU: INFO: possible
recursive locking detected ernel BUG at lis
  843     0   843 S /usr/bin/abrt-watch-log -F Backtrace /var/log/Xorg.0.log --
/usr/bin/abrt-dump-xorg -xD
  848     0   848 S /usr/libexec/accounts-daemon
  864     0   864 S /usr/sbin/mcelog --ignorenodev --daemon --syslog
  889     0   874 S /bin/bash /usr/sbin/ksmtuned
 1025     0  1025 S /usr/sbin/cupsd -f
 1026     0  1026 S /usr/bin/python -Es /usr/sbin/tuned -l -P
 1027     0  1027 S /usr/sbin/sshd -D
 1030     0  1030 S /usr/sbin/rsyslogd -n
 1043     0  1043 S /usr/sbin/crond -n
 1045     0  1045 S /usr/sbin/atd -f
 1308     0  1308 S /usr/libexec/postfix/master -w
 1312    89  1308 S pickup -l -t unix -u
 1313    89  1308 S qmgr -l -t unix -u
 1331     0  1331 S /usr/sbin/gdm
 1348     0  1346 S /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
 1358     0  1358 R /usr/bin/X :0 -background none -noreset -audit 4 -verbose -auth
/run/gdm/auth-for-gdm-eaINl5/database -seat seat0 -n
 1369  1000  1331 S gdm-session-worker [pam/gdm-autologin]
 1374  1000  1374 S /usr/bin/openbox --startup /usr/libexec/openbox-autostart OPENBOX
 1383  1000  1374 S dbus-launch --sh-syntax --exit-with-session
 1384  1000  1384 S /usr/bin/dbus-daemon --fork --print-pid 5 --print-address 7 --session
 1452  1000  1384 S /usr/libexec/imsettings-daemon
 1456  1000  1384 S /usr/libexec/gvfsd
 1461  1000  1384 S /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f -o big_writes
 1548  1000  1546 S /usr/bin/VBoxClient --clipboard
 1550  1000  1546 S /usr/bin/VBoxClient --clipboard
 1558  1000  1556 S /usr/bin/VBoxClient --display
 1560  1000  1556 S /usr/bin/VBoxClient --display
 1566  1000  1561 S /usr/bin/VBoxClient --seamless
 1567  1000  1561 S /usr/bin/VBoxClient --seamless
 1572  1000  1568 S /usr/bin/VBoxClient --draganddrop
 1576  1000  1568 S /usr/bin/VBoxClient --draganddrop
 1582  1000  1582 S /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c "/usr/bin/openbox-session"
 1605  1000  1374 S tint2
 1607  1000  1374 S /usr/bin/python /usr/share/system-config-printer/applet.py
 1608  1000  1374 S nm-applet
 1610  1000  1374 S abrt-applet
 1642  1000  1384 S /usr/libexec/at-spi-bus-launcher
 1648  1000  1643 S /usr/bin/pulseaudio --start --log-target=syslog
 1662  1000  1384 S /bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf
--nofork --print-address 3
 1677  1000  1384 S /usr/libexec/at-spi2-registryd --use-gnome-session
```

```
 1721  1000  1384 S /usr/libexec/dconf-service
 2052  1000  2052 S /usr/share/code/code --unity-launch
 2054  1000  2052 S /usr/share/code/code --type=zygote --no-sandbox
 2098  1000  2052 S /usr/share/code/code --type=renderer --js-flags=--nolazy
--disable-mojo-local-storage --no-sandbox --disable-feature
 2141  1000  2052 S /usr/share/code/code /usr/share/code/resources/app/out/bootstrap
--type=extensionHost
 2153  1000  2052 S /usr/share/code/code --type=renderer --js-flags=--nolazy
--disable-mojo-local-storage --no-sandbox --disable-feature
 2299  1000  1384 S /usr/libexec/gnome-terminal-server
 2304  1000  1384 S /usr/libexec/xdg-desktop-portal
 2309  1000  1384 S /usr/libexec/xdg-document-portal
 2313  1000  1384 S /usr/libexec/xdg-permission-store
 2325  1000  1384 S /usr/libexec/xdg-desktop-portal-gtk
 2332    22  1384 S gnome-pty-helper
 2333  1000  2333 S bash
 2458  1000  2458 S bash
 2637  1000  1384 S /usr/libexec/gvfs-udisks2-volume-monitor
 2642  1000  1384 S /usr/libexec/gvfs-afc-volume-monitor
 2648  1000  1384 S /usr/libexec/gvfs-gphoto2-volume-monitor
 2653  1000  1384 S /usr/libexec/gvfs-mtp-volume-monitor
 2658  1000  1384 S /usr/libexec/gvfs-goa-volume-monitor
 2662  1000  1384 S /usr/libexec/goa-daemon
 2671  1000  1384 S /usr/libexec/goa-identity-service
 2678  1000  1384 S /usr/libexec/mission-control-5
 2680  1000  1384 S /usr/libexec/gvfsd-trash --spawner :1.3 /org/gtk/gvfs/exec_spaw/0
 2701  1000  1384 S /usr/libexec/gvfsd-network --spawner :1.3 /org/gtk/gvfs/exec_spaw/1
 2718  1000  1384 S /usr/libexec/gvfsd-dnssd --spawner :1.3 /org/gtk/gvfs/exec_spaw/3
 2765     0     0 S [kworker/u2:0]
 2857     0     0 R [kworker/0:0]
 2906     0     0 S [kworker/0:1]
 3037     0     0 S [kworker/0:2]
 3101  1000  2458 S man ps
 3112  1000  2458 S less -s
 3227     0   874 S sleep 60
 3246  1000  2333 R ps -eo pid,gid,sid,s,command
```

3:
Comparten el identificador de grupo (gid).
Es el identificador del proceso padre.

Ejercicio 5. Escribir un programa que muestre los identificadores del proceso: identificador de proceso, de proceso padre, de grupo de procesos y de sesión. Mostrar además el número máximo de ficheros que puede abrir el proceso y el directorio de trabajo actual.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>

int main(){
   int pid = getpid();
   printf("pid: %i\n", pid);
   printf("pid del padre: %i\n",getppid());
   printf("pid del grupo: %i\n",getpgid(pid));
   printf("pid de la sesión: %i\n",getsid(pid));
   struct rlimit param;
   getrlimit(RLIMIT_NOFILE, &param);
   printf("Numero maximo de ficheros: %i\n", param.rlim_max);
   //para directorio de trabajo mejor getcwd
   char *path = (char *)malloc(sizeof(char) * 4097);
   char *rpath = getcwd(path, 4097);
   printf("La ruta absoluta del directorio de trabajo actual es: %s\n", path);
   return 1;

}
```

TERMINAL:

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej2
[cursoredes@localhost ~]$ ./ej2
pid: 3542
pid del padre: 2333
pid del grupo: 3542
pid de la sesión: 2333
Numero maximo de ficheros: 4096
La ruta absoluta del directorio de trabajo actual es: /home/cursoredes
```

Ejercicio 6. Un demonio es un proceso que se ejecuta en segundo plano para proporcionar un servicio. Normalmente, un demonio está en su propia sesión y grupo. Para garantizar que es posible crear la sesión y el grupo, el demonio crea un nuevo proceso para ejecutar la lógica del servicio y crear la nueva sesión. Escribir una plantilla de demonio (creación del nuevo proceso y de la sesión) en el que únicamente se muestren los atributos del proceso (como en el ejercicio anterior). Además, fijar el directorio de trabajo del demonio a `/tmp`.

¿Qué sucede si el proceso padre termina antes que el hijo (observar el PPID del proceso hijo)? ¿Y si el proceso que termina antes es el hijo (observar el estado del proceso hijo con `ps`)?

Nota: Usar `sleep(3)` o `pause(3)` para forzar el orden de finalización deseado.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>

void main_ejercicio5(){
   int pid = getpid();
   printf("pid: %i\n",pid);
   printf("pid del padre, del grupo y de la sesion: %i, %i, %i\n", getppid(), getpgid(pid),
getsid(pid));
   struct rlimit param;
   getrlimit(RLIMIT_NOFILE, &param);
   printf("Numero maximo de ficheros: %i\n", param.rlim_max);
   char *path = (char *)malloc(sizeof(char) * 4097);
   char *rpath = getcwd(path, 4097);
   printf("La ruta absoluta del directorio de trabajo actual es: %s\n", path);

}
int main(){
   pid_t pid = fork();
   switch (pid) {
      case -1:
         perror("fork");
         exit(-1);
         break;
      case 0:
         setsid(); //Creamos una nueva sesión
         chdir("/tmp"); //Cambiamos el directorio

         printf("Estoy en el proceso hijo \n");
         main_ejercicio5();
         sleep(3);
         break;
      default:
         printf("Estoy en el proceso padre\n");
         //sleep(3);
         break;
   }
   return 0;

}
```

TERMINAL:
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej6
[cursoredes@localhost ~]$ ./ej6
Estoy en el proceso hijo
pid: 3765
pid del padre, del grupo y de la sesion: 3764, 3765, 3765
Numero maximo de ficheros: 4096
La ruta absoluta del directorio de trabajo actual es: /tmp
Estoy en el proceso padre
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej6
[cursoredes@localhost ~]$ ./ej6
Estoy en el proceso padre
Estoy en el proceso hijo
pid: 3792
pid del padre, del grupo y de la sesion: 1, 3792, 3792
Numero maximo de ficheros: 4096
La ruta absoluta del directorio de trabajo actual es: /tmp

Si acaba antes el padre, el proceso init se convierte en el padre del proceso hijo.

Ejecución de programas

Ejercicio 7. Escribir dos versiones, una con `system(3)` y otra con `execvp(3)`, de un programa que ejecute otro programa que se pasará como argumento por línea de comandos. En cada caso, se debe imprimir la cadena "`El comando terminó de ejecutarse`" después de la ejecución. ¿En qué casos se imprime la cadena? ¿Por qué?

Los argumentos se pasan con comillas en el caso de system para utilizar argv[1]. En el caso de execvp utilizamos sin comillas para poder usar el array de argv.

VERSION SYSTEM:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>

int main(int argc, char ** argv){
   if (argc < 2){
      printf("Error. Debe pasar el programa a ejecutar\n");
      return -1;
   }
   int ret = system(argv[1]);
   if(ret == -1){
```

```
      printf("Error al ejecutar el comando\n");
      return -1;
   }
   printf("El comando terminó de ejecutarse\n");
   return 0;
}
```

TERMINAL:

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej7
[cursoredes@localhost ~]$ ./ej7 "ps -el"
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 31444 ep_pol ?       00:00:01 systemd
1 S     0     2     0  0  80   0 -     0 kthrea ?       00:00:00 kthreadd
....
```

EL comando terminó de ejecutarse

VERSIÓN EXECVP:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>

int main(int argc, char ** argv){
   if (argc < 2){
      printf("Error. Debe pasar el programa a ejecutar\n");
      return -1;
   }

   int ret = execvp(argv[1],argv + 1);
   if(ret == -1){
      printf("Error al ejecutar el programa \n");
      return -1;
   }
   printf("El comando terminó de ejecutarse\n");
   return 0;

}
```

TERMINAL:

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej7
[cursoredes@localhost ~]$ ./ej7 ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 31444 ep_pol ?       00:00:01 systemd
1 S     0     2     0  0  80   0 -     0 kthrea ?       00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 smpboo ?       00:00:00 ksoftirqd/0
```

```
1 S    0    5    2  0  60 -20 -    0 worker ?       00:00:00 kworker/0:0H
...
```

La frase se muestra en system porque este retorna de la llamada. En el caso de execvp, el proceso actual se sustituye por el nuevo que vamos a ejecutar.

Nota: Considerar cómo deben pasarse los argumentos en cada caso para que sea sencilla la implementación. Por ejemplo: ¿qué diferencia hay entre `./ej7 ps -el` y `./ej7 "ps -el"`?

Ejercicio 8. Usando la versión con `execvp(3)` del ejercicio 7 y la plantilla de demonio del ejercicio 6, escribir un programa que ejecute cualquier programa como si fuera un demonio. Además, redirigir los flujos estándar asociados al terminal usando `dup2(2)`:

- La salida estándar al fichero `/tmp/daemon.out`.
- La salida de error estándar al fichero `/tmp/daemon.err`.
- La entrada estándar a `/dev/null`.

Comprobar que el proceso sigue en ejecución tras cerrar la *shell*.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>

int main(int argc, char ** argv){
   if (argc < 2){
      printf("Error. Debe pasar el programa a ejecutar\n");
      return -1;
   }
   pid_t pid;
   pid = fork();
   switch (pid) {
      case -1:
         perror("fork");
         exit(1);
      case 0:
         setsid(); //Creamos una nueva sesión
         chdir("/tmp"); //Cambiamos el directorio
         int fd1 = open("/tmp/daemon.out.",O_CREAT|O_RDWR,0777);
         int fd2 = open("/tmp/daemon.err.",O_CREAT|O_RDWR,0777);
         int fd3 = open("/dev/null",O_CREAT|O_RDWR,0777);
         int fd4 = dup2(fd1,1); //stdout -> 1
         int fd5 = dup2(fd2,2); //stderr -> 2
```

```
        int fd6 = dup2(fd3,0); //stdin -> 0
        int ret = execvp(argv[1],argv + 1);
        if(ret == -1){
           printf("Error al ejecutar el programa \n");
           return -1;
        }
        break;
   }
   return 0;

}
```

TERMINAL:

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej8
[cursoredes@localhost ~]$ ./ej8 ps -el
[cursoredes@localhost ~]$ cat /tmp/daemon.out.
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 31444 ep_pol ?       00:00:01 systemd
1 S     0     2     0  0  80   0 -     0 kthrea ?       00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 smpboo ?       00:00:00 ksoftirqd/0
1 S     0     5     2  0  60 -20 -     0 worker ?       00:00:00 kworker/0:0H
….
```

Señales

Ejercicio 9. El comando `kill(1)` permite enviar señales a un proceso o grupo de procesos por su identificador (`pkill` permite hacerlo por nombre de proceso). Estudiar la página de manual del comando y las señales que se pueden enviar a un proceso.

Ejercicio 10. En un terminal, arrancar un proceso de larga duración (ej. `sleep 600`). En otra terminal, enviar diferentes señales al proceso, comprobar el comportamiento. Observar el código de salida del proceso. ¿Qué relación hay con la señal enviada?

SEÑALES:

1. SIGKILL (9):

```
[cursoredes@localhost ~]$ sleep 600
Killed
```

2. SIGINT (2):

```
[cursoredes@localhost ~]$ sleep 600
```

3. SIGSTOP (19):

```
[cursoredes@localhost ~]$ sleep 600

[1]+  Stopped                 sleep 600
```

4. SIGHUP (1):

```
[cursoredes@localhost ~]$ sleep 600
Hangup
```

Ejercicio 11. Escribir un programa que bloquee las señales `SIGINT` y `SIGTSTP`. Después de bloquearlas el programa debe suspender su ejecución con `sleep(3)` un número de segundos que se obtendrán de la variable de entorno `SLEEP_SECS`. Al despertar, el proceso debe informar de si recibió la señal `SIGINT` y/o `SIGTSTP`. En este último caso, debe desbloquearla con lo que el proceso se detendrá y podrá ser reanudado en la *shell* (imprimir una cadena antes de finalizar el programa para comprobar este comportamiento).

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <signal.h>

int main(){
   sigset_t blk_set;

   sigemptyset(&blk_set);
   sigaddset(&blk_set, SIGINT);
   sigaddset(&blk_set, SIGTSTP);
   sigprocmask(SIG_BLOCK, &blk_set, NULL);

   /*
   char *s = getenv("SLEEP_SECS");
   int secs = atoi(s);
   */
   printf("Durmiendo...\n");
   sleep(10);

   sigset_t pnd_set;
   sigpending(&pnd_set);

   if(sigismember(&pnd_set, SIGINT) == 1)
      printf("Llegó señal SIGINT\n");

   else printf("No llegó señal SIGINT\n");
```

```
   if(sigismember(&pnd_set, SIGTSTP) == 1)
      printf("Llegó señal SIGTSTP\n");

   else printf("No llegó señal SIGTSTP\n");

   sigprocmask(SIG_UNBLOCK, &blk_set, NULL);
   return 0;
}
```

TERMINAL:

```
(null)[cursoredes@localhost ~]$ g++ practica3.cpp -o ej11
[cursoredes@localhost ~]$ ./ej11
Durmiendo...
No llegó señal SIGINT
No llegó señal SIGTSTP
```

Ejercicio 12. Escribir un programa que instale un manejador para las señales `SIGINT` y `SIGTSTP`. El manejador debe contar las veces que ha recibido cada señal. El programa principal permanecerá en un bucle que se detendrá cuando se hayan recibido 10 señales. El número de señales de cada tipo se mostrará al finalizar el programa.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <signal.h>

volatile int cont1 = 0;
volatile int cont2 = 0;

void handler(int signal){
   if (signal == SIGINT)
      cont1++;
   if (signal == SIGTSTP) cont2++;
 }
```

```
int main(){

   struct sigaction act;
   act.sa_handler = handler;
   sigaction(SIGINT,&act,NULL);
   sigaction(SIGTSTP,&act,NULL);
   sigset_t set;
   sigemptyset(&set);
   while(cont1 + cont2 < 10){
      sigsuspend(&set); //Espera a recibir una señal que no esté en el conjunto set
   }
   printf("Ha recibido %i señales de tipo SIGINT\n", cont1);
   printf("Ha recibido %i señales de tipo SIGTSTP\n", cont2);
   return 0;
}

TERMINAL:
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej12
[cursoredes@localhost ~]$ ./ej12
Ha recibido 7 señales de tipo SIGINT
Ha recibido 3 señales de tipo SIGTSTP
```

Ejercicio 13. Escribir un programa que realice el borrado programado del propio ejecutable. El programa tendrá como argumento el número de segundos que esperará antes de borrar el fichero. El borrado del fichero se podrá detener si se recibe la señal `SIGUSR1`.

FICHERO:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <sched.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <signal.h>

volatile int cont1 = 0;

void handler(int signal){
   if (signal == SIGUSR1)
      cont1 = 1;
```

```
}

int main(int argc, char ** argv){
   if (argc < 2){
      printf("Error. Debe pasar el tiempo de espera\n");
      return -1;
   }

   sigset_t set;
   sigemptyset(&set);
            sigaddset(&set, SIGUSR1);
   sigprocmask(SIG_UNBLOCK, &set, NULL);

   struct sigaction act;
   act.sa_handler = handler;
   sigaction(SIGUSR1,&act,NULL);

   char * s = argv[1];
   int secs = atoi(s);
   sleep(secs);
   if(cont1 == 0){
      //Borrar el fichero
      unlink(argv[0]);
      printf("Se borró el fichero\n");
   }
   else{
      printf("No se borró el fichero\n");
   }
   return 0;
}
```

TERMINAL:

1. Enviando señal SIGUSR1

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej13
[cursoredes@localhost ~]$ ./ej13 20
No se borró el fichero
```

2. Sin enviar señal
3.

```
[cursoredes@localhost ~]$ g++ practica3.cpp -o ej13
[cursoredes@localhost ~]$ ./ej13 20
Se borró el fichero
```

Nota: Usar `sigsuspend(2)` para suspender el proceso y la llamada al sistema apropiada para borrar el fichero.