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

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 de 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.

```
Nice (consultamos el valor de nice)
0

ps (consultamos el pid de la shell)
3274 pts/0    00:00:00 bash
3357 pts/0    00:00:00 ps

Cambiamos el valor de nice
renice -n -10 -p 3274

nice (volvemos a consultar)
-10

chrt -f -p 12 3274

chrt -p 3274
política actual de planificación del pid 3274: SCHED_FIFO
política actual de planificación del pid 3274: 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.

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char** argv){
	int sch = sched_getscheduler(0);
	if(sch == -1){
		perror("sched_getscheduler");
		exit(1);
	}
	if(sch == SCHED_OTHER){
		printf("Politica estándar: SCHED_OTHER\n");
	}
	else if (sch == SCHED_FIFO){
		printf("Politica fifo: SCHED_FIFO\n");
	}
	else if (sch == SCHED_RR){
		printf("Politica roud robin: SCHED_RR\n");
	}
	struct sched_param param;
	int err = sched_getparam(0, &param);
	if(err == -1){
		perror("sched_getparam");
		exit(1);
	}
	printf("Prioridad: %i\n", param.sched_priority);
	printf("Valor maximo: %i, Valor mínimo: %i\n", sched_get_priority_min(sch),
sched_get_priority_max(sch));
	return 0;
}

Salida:
Politica estándar: SCHED_OTHER
Prioridad: 0
Valor maximo: 0, Valor mínimo: 0
```

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?

```
Sí, se heredan los atributos. Se obtiene:
Politica fifo: SCHED_FIFO
Prioridad: 12
Valor maximo: 99, Valor mínimo: 1
```

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?

```
ps -u usuarioso -f

Salida:
ID       PID  PPID  C STIME TTY          TIME CMD
usuario+ 1498    1  0 20:43 ?      00:00:00 /lib/systemd/systemd --user
usuario+ 1499  1498  0 20:43 ?        00:00:00 (sd-pam)
usuario+ 1515    1  0 20:43 ?      00:00:00 /usr/bin/gnome-keyring-daemon --
usuario+ 1525  1362  0 20:43 ?        00:00:00 mate-session
usuario+ 1588    1  0 20:43 ?      00:00:02 /usr/bin/ibus-daemon --daemonize
usuario+ 1592  1498  0 20:43 ?        00:00:00 /usr/bin/dbus-daemon --session -
usuario+ 1594  1498  0 20:43 ?        00:00:00 /usr/lib/gvfs/gvfsd
usuario+ 1599  1498  0 20:43 ?        00:00:00 /usr/lib/gvfs/gvfsd-fuse /run/us
usuario+ 1608  1588  0 20:43 ?        00:00:00 /usr/lib/ibus/ibus-dconf
usuario+ 1609  1588  0 20:43 ?        00:00:01 /usr/lib/ibus/ibus-ui-gtk3
usuario+ 1613    1  0 20:43 ?      00:00:00 /usr/lib/ibus/ibus-x11 --kill-da
usuario+ 1616  1498  0 20:43 ?        00:00:00 /usr/lib/ibus/ibus-portal
usuario+ 1626  1498  0 20:43 ?        00:00:00 /usr/lib/at-spi2-core/at-spi-bus
usuario+ 1631  1626  0 20:43 ?        00:00:00 /usr/bin/dbus-daemon --config-fi
usuario+ 1636  1498  0 20:43 ?        00:00:00 /usr/lib/at-spi2-core/at-spi2-re
usuario+ 1642  1588  0 20:44 ?        00:00:00 /usr/lib/ibus/ibus-engine-simple
usuario+ 1693    1  0 20:44 ?      00:00:00 /usr/bin/VBoxClient --clipboard
usuario+ 1694  1693  0 20:44 ?        00:00:00 /usr/bin/VBoxClient --clipboard
usuario+ 1703    1  0 20:44 ?      00:00:00 /usr/bin/VBoxClient --display
usuario+ 1704  1703  0 20:44 ?        00:00:00 /usr/bin/VBoxClient --display
usuario+ 1708    1  0 20:44 ?      00:00:00 /usr/bin/VBoxClient --seamless
usuario+ 1709  1708  0 20:44 ?        00:00:00 /usr/bin/VBoxClient --seamless
usuario+ 1713    1  0 20:44 ?      00:00:00 /usr/bin/VBoxClient --draganddro
usuario+ 1714  1713  0 20:44 ?        00:00:04 /usr/bin/VBoxClient --draganddro
usuario+ 1725  1525  0 20:44 ?        00:00:00 /usr/bin/ssh-agent /usr/bin/im-l
usuario+ 1731  1498  0 20:44 ?        00:00:00 /usr/lib/dconf/dconf-service
usuario+ 2319  1525  0 20:44 ?        00:00:01 /usr/bin/mate-settings-daemon
usuario+ 2406  1525  0 20:44 ?        00:00:04 marco
usuario+ 2412  1525  0 20:44 ?        00:00:00 mate-panel
usuario+ 2419    1  0 20:44 ?      00:00:02 /usr/bin/pulseaudio --start --lo
usuario+ 2423  1525  0 20:44 ?        00:00:04 caja
usuario+ 2430  1498  0 20:44 ?        00:00:01 /usr/lib/mate-panel/wnck-applet
usuario+ 2432  1498  0 20:44 ?        00:00:00 /usr/lib/mate-applets/trashapple
usuario+ 2434  1498  0 20:44 ?        00:00:00 /usr/lib/x86_64-linux-gnu/brisk-
```

```
usuario+  2444  1525  0 20:44 ?      00:00:01 mate-maximus
usuario+  2455  1525  0 20:44 ?      00:00:00 mate-power-manager
usuario+  2462  1498  0 20:44 ?      00:00:00 /usr/lib/mate-panel/clock-applet
usuario+  2463  1498  0 20:44 ?      00:00:00 /usr/lib/mate-panel/notification
usuario+  2464  1498  0 20:44 ?      00:00:00 /usr/lib/mate-indicator-applet/m
usuario+  2467  1594  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfsd-trash --spaw
usuario+  2479  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/indica
usuario+  2484  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/indica
usuario+  2503  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfsd-metadata
usuario+  2519  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/indica
usuario+  2524  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfs-udisks2-volum
usuario+  2539  1525  0 20:44 ?      00:00:00 mate-screensaver
usuario+  2547  1525  0 20:44 ?      00:00:00 /usr/lib/deja-dup/deja-dup-monit
usuario+  2555  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfs-mtp-volume-mo
usuario+  2559  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/indica
usuario+  2561  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfs-gphoto2-volum
usuario+  2577  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/indica
usuario+  2578  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfs-goa-volume-mo
usuario+  2579  1525  0 20:44 ?      00:00:00 /usr/bin/python3 /usr/share/syst
usuario+  2583  1498  0 20:44 ?      00:00:00 /usr/lib/gnome-online-accounts/g
usuario+  2584  1525  0 20:44 ?      00:00:00 nm-applet
usuario+  2598  1525  0 20:44 ?      00:00:00 update-notifier
usuario+  2629  1525  0 20:44 ?      00:00:00 /usr/lib/x86_64-linux-gnu/polkit
usuario+  2639  1525  0 20:44 ?      00:00:00 /usr/bin/python3 /usr/bin/bluema
usuario+  2684  1498  0 20:44 ?      00:00:00 /usr/lib/gnome-online-accounts/g
usuario+  2690  1498  0 20:44 ?      00:00:00 /usr/lib/gvfs/gvfs-afc-volume-mo
usuario+  2761  1498  0 20:45 ?      00:00:00 /usr/lib/bluetooth/obexd
usuario+  2843     1  0 20:46 ?      00:00:13 /opt/sublime_text/sublime_text /
usuario+  2858  2843  0 20:46 ?      00:00:01 /opt/sublime_text/plugin_host 28
usuario+  2862  2412  0 20:46 ?      00:00:13 mate-terminal
usuario+  3724  2862  0 21:33 pts/1    00:00:00 bash
usuario+  3821  3724  0 21:40 pts/1    00:00:00 man ps
usuario+  3831  3821  0 21:40 pts/1    00:00:00 pager
usuario+  3851  2862  0 21:44 pts/0    00:00:00 bash
usuario+  3859  3851  0 21:45 pts/0    00:00:00 ps -u usuarioso -f

ps -eo pid,gid,sid,s,command
  PID   GID   SID S COMMAND
    1     0     1 S /sbin/init splash
    2     0     0 S [kthreadd]
    4     0     0 I [kworker/0:0H]
    6     0     0 I [mm_percpu_wq]
    7     0     0 S [ksoftirqd/0]
    8     0     0 I [rcu_sched]
    9     0     0 I [rcu_bh]
   10     0     0 S [migration/0]
   11     0     0 S [watchdog/0]
   12     0     0 S [cpuhp/0]
   13     0     0 S [kdevtmpfs]
   14     0     0 I [netns]
   15     0     0 S [rcu_tasks_kthre]
   16     0     0 S [kauditd]
   17     0     0 S [khungtaskd]
   18     0     0 S [oom_reaper]
   19     0     0 I [writeback]
   20     0     0 S [kcompactd0]
   21     0     0 S [ksmd]
```

```
 22    0    0 S [khugepaged]
 23    0    0 I [crypto]
 24    0    0 I [kintegrityd]
 25    0    0 I [kblockd]
 26    0    0 I [ata_sff]
 27    0    0 I [md]
 28    0    0 I [edac-poller]
 29    0    0 I [devfreq_wq]
 30    0    0 I [watchdogd]
 32    0    0 I [kworker/0:1]
 34    0    0 S [kswapd0]
 35    0    0 S [ecryptfs-kthrea]
 77    0    0 I [kthrotld]
 78    0    0 I [acpi_thermal_pm]
 79    0    0 S [scsi_eh_0]
 80    0    0 I [scsi_tmf_0]
 81    0    0 S [scsi_eh_1]
 82    0    0 I [scsi_tmf_1]
 88    0    0 I [ipv6_addrconf]
 97    0    0 I [kstrp]
114    0    0 I [charger_manager]
152    0    0 I [kworker/0:2]
154    0    0 S [scsi_eh_2]
155    0    0 I [scsi_tmf_2]
158    0    0 I [kworker/0:1H]
232    0    0 S [jbd2/sda1-8]
233    0    0 I [ext4-rsv-conver]
274    0  274 S /lib/systemd/systemd-journald
285    0  285 S /lib/systemd/systemd-udevd
295    0    0 S [loop0]
297    0    0 S [loop1]
299    0    0 S [loop2]
303    0    0 S [loop3]
308    0    0 S [loop4]
310    0    0 S [loop5]
312    0    0 S [loop6]
314    0    0 S [loop7]
316    0    0 S [loop8]
318    0    0 S [loop9]
322    0    0 S [loop10]
323    0    0 S [loop11]
327    0    0 S [loop12]
330    0    0 S [loop13]
332    0    0 S [loop14]
334    0    0 S [loop15]
345    0    0 S [loop16]
348    0    0 S [loop17]
350    0    0 S [loop18]
352    0    0 S [loop19]
355    0    0 S [loop20]
356    0    0 S [loop21]
404  103  404 S /lib/systemd/systemd-resolved
421    0    0 I [iprt-VBoxWQueue]
426    0    0 I [ttm_swap]
646    0  646 S /usr/sbin/cron -f
651  107  651 S /usr/bin/dbus-daemon --system --address=systemd: --nofork --
876    0  876 S /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-
```

```
 877    0   877 S /usr/sbin/ModemManager
 878    0   878 S /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
 880  122   880 S avahi-daemon: running [ssoo.local]
 881    0   881 S /usr/sbin/NetworkManager --no-daemon
 886  122   880 S avahi-daemon: chroot helper
 888  106   888 S /usr/sbin/rsyslogd -n
 890    0   890 S /usr/lib/accountsservice/accounts-daemon
 891    0   891 S /usr/sbin/acpid
 893    0   893 S /lib/systemd/systemd-logind
 897    0   897 S /usr/lib/udisks2/udisksd
 901    0   901 S /usr/lib/snapd/snapd
 959    0   959 S /usr/lib/policykit-1/polkitd --no-debug
1055    0  1055 S /usr/sbin/lightdm
1060    0  1058 S /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
1068    0   881 S /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-hel
1077    0  1077 S /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/light
1079    0  1079 S /sbin/agetty -o -p -- \u --noclear tty1 linux
1100  117  1100 S /usr/bin/whoopsie -f
1114    4  1114 S /usr/sbin/kerneloops --test
1116    4  1116 S /usr/sbin/kerneloops
1352    0  1352 S /usr/lib/upower/upowerd
1362    0  1055 S lightdm --session-child 13 20
1498 1001  1498 S /lib/systemd/systemd --user
1499 1001  1498 S (sd-pam)
1515 1001  1514 S /usr/bin/gnome-keyring-daemon --daemonize --login
1525 1001  1525 S mate-session
1536    0  1536 S /usr/sbin/cupsd -l
1537    0  1537 S /usr/sbin/cups-browsed
1588 1001  1588 S /usr/bin/ibus-daemon --daemonize --xim --address unix:tmpdir
1592 1001  1592 S /usr/bin/dbus-daemon --session --address=systemd: --nofork -
1594 1001  1594 S /usr/lib/gvfs/gvfsd
1599 1001  1594 S /usr/lib/gvfs/gvfsd-fuse /run/user/1001/gvfs -f -o big_write
1608 1001  1588 S /usr/lib/ibus/ibus-dconf
1609 1001  1588 S /usr/lib/ibus/ibus-ui-gtk3
1613 1001  1588 S /usr/lib/ibus/ibus-x11 --kill-daemon
1616 1001  1592 S /usr/lib/ibus/ibus-portal
1626 1001  1626 S /usr/lib/at-spi2-core/at-spi-bus-launcher
1631 1001  1626 S /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-sp
1636 1001  1626 S /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-session
1642 1001  1588 S /usr/lib/ibus/ibus-engine-simple
1693 1001  1691 S /usr/bin/VBoxClient --clipboard
1694 1001  1691 S /usr/bin/VBoxClient --clipboard
1703 1001  1701 S /usr/bin/VBoxClient --display
1704 1001  1701 S /usr/bin/VBoxClient --display
1708 1001  1706 S /usr/bin/VBoxClient --seamless
1709 1001  1706 S /usr/bin/VBoxClient --seamless
1713 1001  1711 S /usr/bin/VBoxClient --draganddrop
1714 1001  1711 S /usr/bin/VBoxClient --draganddrop
1725 1001  1725 S /usr/bin/ssh-agent /usr/bin/im-launch mate-session
1731 1001  1592 S /usr/lib/dconf/dconf-service
2319 1001  1525 S /usr/bin/mate-settings-daemon
2406 1001  1525 S marco
2412 1001  1525 S mate-panel
2419 1001  2418 S /usr/bin/pulseaudio --start --log-target=syslog
2420  114  2420 S /usr/lib/rtkit/rtkit-daemon
2423 1001  1525 S caja
2430 1001  1592 S /usr/lib/mate-panel/wnck-applet
```

```
2432  1001  1592 S /usr/lib/mate-applets/trashapplet
2434  1001  1592 S /usr/lib/x86_64-linux-gnu/brisk-menu//brisk-menu
2444  1001  1525 S mate-maximus
2455  1001  1525 S mate-power-manager
2462  1001  1592 S /usr/lib/mate-panel/clock-applet
2463  1001  1592 S /usr/lib/mate-panel/notification-area-applet
2464  1001  1592 S /usr/lib/mate-indicator-applet/mate-indicator-applet-complet
2467  1001  1594 S /usr/lib/gvfs/gvfsd-trash --spawner :1.2 /org/gtk/gvfs/exec_
2479  1001  1525 S /usr/lib/x86_64-linux-gnu/indicator-sound/indicator-sound-se
2484  1001  1525 S /usr/lib/x86_64-linux-gnu/indicator-session/indicator-sessio
2503  1001  2503 S /usr/lib/gvfs/gvfsd-metadata
2519  1001  1525 S /usr/lib/x86_64-linux-gnu/indicator-messages/indicator-messa
2524  1001  2524 S /usr/lib/gvfs/gvfs-udisks2-volume-monitor
2539  1001  1525 S mate-screensaver
2547  1001  1525 S /usr/lib/deja-dup/deja-dup-monitor
2555  1001  2555 S /usr/lib/gvfs/gvfs-mtp-volume-monitor
2559  1001  1525 S /usr/lib/x86_64-linux-gnu/indicator-application/indicator-ap
2561  1001  2561 S /usr/lib/gvfs/gvfs-gphoto2-volume-monitor
2577  1001  1525 S /usr/lib/x86_64-linux-gnu/indicator-power/indicator-power-se
2578  1001  2578 S /usr/lib/gvfs/gvfs-goa-volume-monitor
2579  1001  1525 S /usr/bin/python3 /usr/share/system-config-printer/applet.py
2583  1001  1592 S /usr/lib/gnome-online-accounts/goa-daemon
2584  1001  1525 S nm-applet
2598  1001  1525 S update-notifier
2629  1001  1525 S /usr/lib/x86_64-linux-gnu/polkit-mate/polkit-mate-authentica
2639  1001  1525 S /usr/bin/python3 /usr/bin/blueman-applet
2684  1001  1592 S /usr/lib/gnome-online-accounts/goa-identity-service
2690  1001  2690 S /usr/lib/gvfs/gvfs-afc-volume-monitor
2761  1001  2761 S /usr/lib/bluetooth/obexd
2843  1001  2843 S /opt/sublime_text/sublime_text /home/usuarioso/pr3.c
2858  1001  2843 S /opt/sublime_text/plugin_host 2843 --auto-shell-env
2862  1001  1525 S mate-terminal
3712     0     0 I [kworker/u2:1]
3724  1001  3724 S bash
3807     0     0 I [kworker/u2:0]
3821  1001  3724 S man ps
3831  1001  3724 S pager
3839     0     0 I [kworker/u2:2]
3851  1001  3851 S bash
3860  1001  3851 R ps -eo pid,gid,sid,s,command
```

Respuestas:
Comparten el gid y el sid (1001 y 3851).
Es gid es 1001 cuando se crea un nuevo proceso.

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 archivos que puede abrir el proceso y el directorio de trabajo actual.

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
```

```
#include <sys/resource.h>
#include <linux/limits.h>
int main(int argc, char** argv){
	pid_t pid = getpid();
	int sid = getsid(pid);
	if(sid == -1){
		perror("getsid");
		exit(1);
	}
	printf("PID: %i, PPID: %i, GPID: %i, SID: %i\n", pid, getppid(), getpgrp(), sid);
	struct rlimit rlim;
	int err = getrlimit(RLIMIT_NOFILE, &rlim);
	char* buf = malloc(PATH_MAX);
	if(getcwd(buf, PATH_MAX) == NULL){
		perror("getcwd");
		exit(1);
	}
	printf("Directorio de trabajo: %s\n", buf);
	free(buf);
	if(err == 1){
		perror("getrlimit");
		exit(1);
	}
	printf("Maximo archivos. Soft: %li Hard: %li\n", rlim.rlim_cur, rlim.rlim_max);

	return 0;
}
```

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.

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>

void muestraInfo(){
	pid_t pid = getpid();
```

```
	int sid = getsid(pid);
	if(sid == -1){
		perror("getsid");
		exit(1);
	}
	printf("PID: %i, PPID: %i, GPID: %i, SID: %i\n", pid, getppid(), getpgrp(), sid);
	struct rlimit rlim;
	int err = getrlimit(RLIMIT_NOFILE, &rlim);
	char* buf = malloc(PATH_MAX);
	if(getcwd(buf, PATH_MAX) == NULL){
		perror("getcwd");
		exit(1);
	}
	printf("Directorio de trabajo: %s\n", buf);
	free(buf);
	if(err == 1){
		perror("getrlimit");
		exit(1);
	}
	printf("Maximo archivos. Soft: %li Hard: %li\n", rlim.rlim_cur, rlim.rlim_max);
}


int main(int argc, char** argv){
	int pid = fork();
	if(pid == -1){
		perror("fork");
		exit(1);
	}
	else if(pid == 0){
		pid_t sesion = setsid();
		if(sesion == -1){
			perror("setsid");
			exit(1);
		}
		int err = chdir("/tmp");
		if(err == -1){
			perror("chdir");
			exit(1);
		}
		muestraInfo();
		exit(0);
	}
	else{
		sleep(3);
	}



	return 0;
}
Salida:
./pr3
PID: 4458, PPID: 4457, GPID: 4458, SID: 4458
Directorio de trabajo: /tmp
Maximo archivos. Soft: 1024 Hard: 1048576
```

Ocurre que el hijo se queda huérfano y el ppid es el de INIT (1).

EL ppi es el del proceso padre.

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?

Version 1:

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>


int main(int argc, char** argv){
	if(argc < 2){
		printf("Falta argumentos\n");
		exit(1);
	}
	int err = system(argv[1]);
	if(err == -1){
		printf("No se pudo crear el hijo\n");
		exit(1);
	}
	printf("El comando terminó de ejecutarse\n");
	return 0;
}
```

Salida 1:

```
./pr3 "ls -l"
total 308
-rw-rw-r-- 1 usuarioso usuarioso     64 dic 20 21:36 compilar
drwxr-xr-x 2 usuarioso usuarioso   4096 jul 10  2018 Descargas
drwxr-xr-x 2 usuarioso usuarioso   4096 jul 10  2018 Documentos
drwxr-xr-x 3 usuarioso usuarioso   4096 jul 13  2018 eclipse
-rwxrwxr-x 1 usuarioso usuarioso 205024 dic 20 21:34 ejecutable.exe
-rw-r--r-- 3 usuarioso usuarioso      0 dic 13 19:52 ejemploEjer11
-rw-r--r-- 3 usuarioso usuarioso      0 dic 13 19:52 ejemploEjer11.hard
lrwxrwxrwx 1 usuarioso usuarioso     13 dic 13 20:02 ejemploEjer11.sym -> ejemploEjer11
drwxr-xr-x 2 usuarioso usuarioso   4096 jul 10  2018 Escritorio
-rw-r--r-- 1 usuarioso usuarioso   8980 jul 10  2018 examples.desktop
-rw-r--r-x 2 usuarioso usuarioso      0 dic 13 18:04 fichero
-rw-r----- 1 usuarioso usuarioso      0 dic 13 18:32 fichero2
```

```
lrwxrwxrwx 1 usuarioso usuarioso     7 dic 13 19:31 ficherolink -> fichero
-rw-r----- 1 usuarioso usuarioso     0 dic 13 18:37 ficheroPruebaEjer7
lrwxrwxrwx 1 usuarioso usuarioso    22 dic 13 19:50 ficheroPruebaEjer7.sym ->
ficheroPruebaEjer7.sym
-rw-r--r-x 2 usuarioso usuarioso     0 dic 13 18:04 ficherorigido
-rw-rw-r-- 1 usuarioso usuarioso     0 dic 26 20:50 ficheroSalida10000000NuevaRep.txt
-rw-rw-r-- 1 usuarioso usuarioso  9293 dic 20 21:26 generadorNAND.cpp
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Imágenes
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Música
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Plantillas
-rwxrwxr-x 1 usuarioso usuarioso 13024 dic 14 13:51 pr2
-rwxrwxr-x 1 usuarioso usuarioso  8376 dic 26 23:10 pr3
-rw-rw-r-- 1 usuarioso usuarioso   451 dic 26 23:10 pr3.c
drwxrwxr-x 2 usuarioso usuarioso  4096 nov 29 13:46 Practica1ASOR
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Público
-rw-r--r-- 1 usuarioso usuarioso    44 dic 13 20:25 redireccion
drwxr-xr-x 2 usuarioso usuarioso  4096 jul 10  2018 Vídeos
El comando terminó de ejecutarse
```

En esta versión sí se vuelve de la llamada y se muestra la frase de después. Hay que pasarle entre comillas el comando, si no lo toma como argumentos diferentes.

Versión 2:

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>


int main(int argc, char** argv){
        if(argc < 2){
                printf("Falta argumentos\n");
                exit(1);
        }
        int err = execl("/bin/sh", "sh", "-c", argv[1], (char *) 0);
        if(err == -1){
                perror("execl");
                exit(1);
        }
        printf("El comando terminó de ejecutarse\n");
        return 0;
}
```

Misma salida pero sin enseñar la frase final. Se cambia de proceso al hacer un execl y nunca se regresa a este.

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

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argc, char** argv){
	if(argc < 2){
		printf("Falta argumentos\n");
		exit(1);
	}
	int pid = fork();
	if(pid == -1){
		perror("fork");
		exit(1);
	}
	else if(pid == 0){
		pid_t sesion = setsid();
		if(sesion == -1){
			perror("setsid");
			exit(1);
		}
		int fd1 = open("/tmp/daemon.out", O_CREAT | O_RDWR, 00777);
		if(fd1 == -1){
			perror("open");
			exit(1);
		}
		int fd2 = open("/tmp/daemon.err", O_CREAT | O_RDWR, 00777);
		if(fd2 == -1){
			perror("open");
			exit(1);
		}
		int fd3 = open("/dev/null", O_CREAT | O_RDWR, 00777);
		if(fd3 == -1){
			perror("open");
			exit(1);
		}
		int d1 = dup2(fd1,1);
		if(d1 == -1){
```

```
                        perror("dup2");
                        exit(1);
                }
                int d2 = dup2(fd2,2);
                if(d2 == -1){
                        perror("dup2");
                        exit(1);
                }
                int d3 = dup2(fd3,0);
                if(d3 == -1){
                        perror("dup2");
                        exit(1);
                }

                int val = execvp(argv[1], argv + 1);
                if(val == -1){
                        perror("execvp");
                        exit(1);
                }
                exit(0);
        }
        else{
                sleep(3);
        }

        return 0;
}
```

Ejecutamos:
./pr3 ./pr2 .

Y comprobamos la salida en el fichero redireccionado
cat /tmp/daemon.out
fichero*
Descargas/
.bashrc
.ssh/
../
.vboxclient-display.pid
redireccion
ejemploEjer11
pr3*
Imágenes/
.vboxclient-draganddrop.pid
ficheroSalida10000000NuevaRep.txt
.eclipse/
.p2/
ficheroPruebaEjer7.sym -> ficheroPruebaEjer7.sym
Escritorio/
generadorNAND.cpp
.bash_history
.gnupg/
.config/
./
ficheroPruebaEjer7
.swt/
.cache/

```
Plantillas/
ficherorigido*
.profile
examples.desktop
.bash_logout
.sudo_as_admin_successful
eclipse/
.Xauthority
.dmrc
Vídeos/
Practica1ASOR/
pr3.c
fichero2
.xsession-errors
pr2*
Documentos/
.ICEauthority
ejemploEjer11.hard
.vboxclient-clipboard.pid
.vboxclient-seamless.pid
Público/
.pam_environment
ejecutable.exe*
.xsession-errors.old
ejemploEjer11.sym -> ejemploEjer11Ejer7.sym
ficherolink -> fichero
.lesshst
compilar
.local/
Música/
.viminfo
```

Señales

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

```
Vemos las señales que hay
kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7
42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13
52) SIGRTMAX-12
```

53) SIGRTMAX-11	54) SIGRTMAX-10	55) SIGRTMAX-9	56) SIGRTMAX-8
57) SIGRTMAX-7			
58) SIGRTMAX-6	59) SIGRTMAX-5	60) SIGRTMAX-4	61) SIGRTMAX-3
62) SIGRTMAX-2			
63) SIGRTMAX-1	64) SIGRTMAX		

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?

Por ejemplo, al enviar la señal SIGHUP:
Terminal 1:
kill -s 1 5171

Terminal 2:
sleep 600
Colgar (hangup)

O parar:

Terminal 1:
kill -s 19 5191

Terminal 2:
sleep 600

[1]+ Detenido sleep 600

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`.

Después de despertar de `sleep(3)`, 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).

export SLEEP_SECS="15" en el terminal antes de empezar

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>

int main(int argc, char** argv){
	sigset_t set;
	sigemptyset(&set);
	sigaddset(&set, SIGINT);
	sigaddset(&set, SIGTSTP);
	char* duerme = getenv("SLEEP_SECS");
	if(duerme == NULL){
		printf("Error en getenv\n");
		exit(1);
	}
	sigprocmask(SIG_BLOCK, &set, NULL);
	sleep(atoi(duerme));
	sigset_t pendientes;
	int err = sigpending(&pendientes);
	if(err == -1){
		perror("sigpending");
		exit(1);
	}
	if(sigismember(&pendientes, SIGINT)){
		printf("SIGINT recibida\n");
	}
	if(sigismember(&pendientes, SIGTSTP)){
		printf("SIGTSTP recibida\n");
		sigset_t unloblock_set;
		sigemptyset(&unloblock_set);
		sigaddset(&unloblock_set, SIGTSTP);
		sigprocmask(SIG_UNBLOCK, &unloblock_set, NULL);
	}
	printf("Llega el final\n");
	return 0;
}

Probamos:
En otro terminal:
kill -s 2 5456
kill -s 18 5456

Salida:
./pr3
SIGINT recibida
SIGTSTP recibida

[1]+  Detenido              ./pr3
usuarioso@ssoo:~$ Llega el final
```

Ejercicio 12. Escribir un programa que instale un manejador sencillo 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.

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>

volatile int numSignalsINT = 0;
volatile int numSignalsSTP = 0;

void handlerINT(int valor){
	printf("Señal SIGINT\n");
	numSignalsINT++;
}

void handlerSTP(int valor){
	printf("Señal SIGSTP\n");
	numSignalsSTP++;
}


int main(int argc, char** argv){
	struct sigaction actINT;
	actINT.sa_handler = handlerINT;
	struct sigaction actSTP;
	actSTP.sa_handler = handlerSTP;
	sigaction(2, &actINT, NULL);
	sigaction(20, &actSTP, NULL);
	while(numSignalsINT + numSignalsSTP < 10){}
	printf("Señales SIGNIT: %i, señales SIGSTP: %i\n", numSignalsINT, numSignalsSTP);
	return 0;
}



./pr3
Señal SIGINT
Señal SIGINT
Señal SIGINT
Señal SIGINT
Señal SIGINT
Señal SIGINT
```

```
Señal SIGSTP
Señal SIGSTP
Señal SIGSTP
Señal SIGSTP
Señales SIGNIT: 6, señales SIGSTP: 4
```

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`.

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

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <linux/limits.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>

volatile int suicidio = 1;

void handler(int valor){
	suicidio = 0;
}


int main(int argc, char** argv){
	if(argc < 2){
		printf("Falta argumentos\n");
		exit(1);
	}
	struct sigaction act;
	act.sa_handler = handler;
	sigaction(10, &act, NULL);
	sleep(atoi(argv[1]));
	if (suicidio == 1) unlink(argv[0]);
	else printf("Salvado!\n");
	return 0;
}

./pr3 50
Salida: (Mandando la señal 10 a tiempo)

Salvado!
```