



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



SISTEMAS OPERATIVOS

ADMINISTRADOR DE PROCESOS EN LINUX Y WINDOWS

Práctica #3. Administrador de Procesos en Linux y Windows (1)

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

Grupo: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

19 Septiembre, 2021

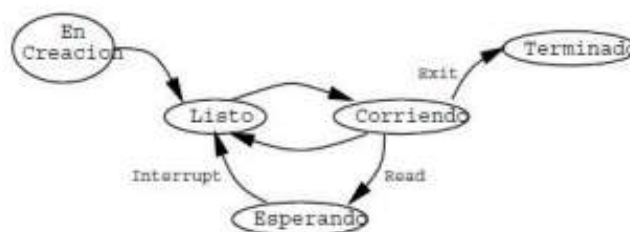
I.INTRODUCCIÓN TEÓRICA

La actividad más importante del núcleo del sistema operativo es implementar los procesos. Cada uno de estos procesos es un procesador virtual en donde se ejecuta una aplicación o una herramienta del sistema operativo. El núcleo debe encargarse entonces de administrar los recursos del hardware del computador para que sean asignados convenientemente a los procesos.

En la mayoría de los sistemas computacionales existe un solo procesador real. Por lo tanto el núcleo debe asignar el procesador por turnos a los numerosos procesos que pueden estar activos. Hay distintas estrategias para asignar estos turnos, dependiendo del objetivo que se persiga. Por ejemplo en un sistema de multiprogramación se busca maximizar el tiempo de uso del procesador, mientras que en un sistema de tiempo compartido se busca atender en forma expedita a muchos usuarios que trabajan interactivamente. La asignación estratégica del procesador a los procesos es lo que se denomina **scheduling de procesos**. Es estratégica porque se intenta lograr algún objetivo particular como alguno de los que se mencionó previamente y para ello se usan estrategias que pueden funcionar muy bien en determinados sistemas, pero muy mal en otros. El componente del núcleo que se encarga de esta labor se denomina **scheduler** del procesador. Además de scheduling de procesos, también se realiza scheduling de los accesos a disco y para ello existe un scheduler asociado a cada disco. Este scheduler ordena estratégicamente los múltiples accesos a disco de varios procesos con el fin de minimizar el tiempo de acceso.

Ahora para entender un poco más el tema principal de esta práctica, definimos lo que es un proceso. Y es que un proceso no es más que la instancia de un programa en ejecución. A menudo también se les puede conocer como tareas. El contexto de un programa que está en ejecución es lo que se llama un proceso. En cuanto a Linux debido a que es un sistema operativo multitarea y multiusuario, múltiples procesos pueden operar simultáneamente sin interferirse unos con los otros. Cada proceso tiene la "ilusión" que es el único proceso en el sistema y que tiene acceso exclusivo a todos los servicios del sistema operativo.

Mientras un proceso se ejecuta puede pasar por distintos estados. Estos estados se aprecian en la siguiente figura:



- **En creación:** el núcleo está obteniendo los recursos que necesita el proceso para poder correr, como por ejemplo memoria o disco.
- **Corriendo:** El proceso está en posesión del procesador, el que ejecuta sus instrucciones.
- **Esperando:** El proceso espera que se lea un sector del disco, que llegue un mensaje de otro proceso, que transcurra un intervalo de tiempo, que termine otro proceso, etc.
- **Listo:** El proceso está activo pero no está en posesión del procesador.
- **Terminado:** El proceso terminó su ejecución, pero sigue existiendo para que otros procesos puedan determinar que terminó.

II.DESARROLLO EXPERIMENTAL

Sección Linux

1. Se introdujeron los siguientes comandos a través de la consola del sistema operativo Linux:

ps:

```
cesar@cesar-HP-Notebook:~$ ps
  PID TTY          TIME CMD
 2504 pts/0        00:00:00 bash
 2575 pts/0        00:00:00 ps
cesar@cesar-HP-Notebook:~$
```

ps -fea:

```
cesar@cesar-HP-Notebook:~$ ps -fea
UID          PID    PPID  C STIME TTY          TIME CMD
root           1        0  2 00:27 ?        00:00:04 /sbin/init splash
root           2        0  0 00:27 ?        00:00:00 [kthreadd]
root           3        2  0 00:27 ?        00:00:00 [rcu_gp]
root           4        2  0 00:27 ?        00:00:00 [rcu_par_gp]
root           5        2  0 00:27 ?        00:00:00 [kworker/0:0-rcu_gp]
root           6        2  0 00:27 ?        00:00:00 [kworker/0:0H-events_highpri]
root           7        2  0 00:27 ?        00:00:00 [kworker/0:1-cgroup_destroy]
root           8        2  0 00:27 ?        00:00:00 [kworker/u8:0-events_unbound]
root           9        2  0 00:27 ?        00:00:00 [mm_percpu_wq]
root          10        2  0 00:27 ?        00:00:00 [rcu_tasks_rude_]
root          11        2  0 00:27 ?        00:00:00 [rcu_tasks_trace]
root          12        2  0 00:27 ?        00:00:00 [ksoftirqd/0]
root          13        2  0 00:27 ?        00:00:00 [rcu_sched]
root          14        2  0 00:27 ?        00:00:00 [migration/0]
root          15        2  0 00:27 ?        00:00:00 [idle_inject/0]
root          16        2  0 00:27 ?        00:00:00 [cpuhp/0]
root          17        2  0 00:27 ?        00:00:00 [cpuhp/1]
root          18        2  0 00:27 ?        00:00:00 [idle_inject/1]
root          19        2  0 00:27 ?        00:00:00 [migration/1]
root          20        2  0 00:27 ?        00:00:00 [ksoftirqd/1]
root          21        2  0 00:27 ?        00:00:00 [kworker/1:0-events]
root          22        2  0 00:27 ?        00:00:00 [kworker/1:0H-events_highpri]
root          23        2  0 00:27 ?        00:00:00 [kdevtmpfs]
root          24        2  0 00:27 ?        00:00:00 [netns]
root          25        2  0 00:27 ?        00:00:00 [inet_frag_wq]
root          26        2  0 00:27 ?        00:00:00 [kauditd]
root          27        2  0 00:27 ?        00:00:00 [khungtaskd]
root          28        2  0 00:27 ?        00:00:00 [oom_reaper]
root          29        2  0 00:27 ?        00:00:00 [writeback]
root          30        2  0 00:27 ?        00:00:00 [kcompactd0]
root          31        2  0 00:27 ?        00:00:00 [ksmd]
root          32        2  0 00:27 ?        00:00:00 [khugepaged]
root          37        2  0 00:27 ?        00:00:00 [kworker/1:1-events]
root          79        2  0 00:27 ?        00:00:00 [kintegrityd]
root          80        2  0 00:27 ?        00:00:00 [kblockd]
root          81        2  0 00:27 ?        00:00:00 [blkcg_punt_bio]
```


root	82	2	0	00:27	?	00:00:00	[tpm_dev_wq]
root	83	2	0	00:27	?	00:00:00	[ata_sff]
root	84	2	0	00:27	?	00:00:00	[md]
root	85	2	0	00:27	?	00:00:00	[edac-poller]
root	86	2	0	00:27	?	00:00:00	[devfreq_wq]
root	87	2	0	00:27	?	00:00:00	[watchdogd]
root	88	2	0	00:27	?	00:00:00	[kworker/u8:1-events_unbound]
root	89	2	0	00:27	?	00:00:00	[kworker/0:1H-events_highpri]
root	91	2	0	00:27	?	00:00:00	[kswapd0]
root	92	2	0	00:27	?	00:00:00	[ecryptfs-kthrea]
root	94	2	0	00:27	?	00:00:00	[kthrotld]
root	95	2	0	00:27	?	00:00:00	[acpi_thermal_pm]
root	96	2	0	00:27	?	00:00:00	[vfio-irqfd-clea]
root	97	2	0	00:27	?	00:00:00	[kworker/1:2-events]
root	98	2	0	00:27	?	00:00:00	[kworker/1:3-events]
root	99	2	0	00:27	?	00:00:00	[kworker/1:1H-events_highpri]
root	100	2	0	00:27	?	00:00:00	[ipv6_addrconf]
root	109	2	0	00:27	?	00:00:00	[kstrp]
root	112	2	0	00:27	?	00:00:00	[zswap-shrink]
root	113	2	0	00:27	?	00:00:00	[kworker/u9:0-i915_flip]
root	120	2	0	00:27	?	00:00:00	[charger_manager]
root	164	2	0	00:27	?	00:00:00	[kworker/0:2-events]
root	166	2	0	00:27	?	00:00:00	[scsi_eh_0]
root	167	2	0	00:27	?	00:00:00	[scsi_tmf_0]
root	168	2	0	00:27	?	00:00:00	[scsi_eh_1]
root	170	2	0	00:27	?	00:00:00	[scsi_tmf_1]
root	171	2	0	00:27	?	00:00:00	[kworker/u8:2]
root	172	2	0	00:27	?	00:00:00	[kworker/u8:3-events_unbound]
root	194	2	0	00:27	?	00:00:00	[jbd2/sda2-8]
root	195	2	0	00:27	?	00:00:00	[ext4-rsv-conver]
root	234	1	0	00:27	?	00:00:01	/lib/systemd/systemd-journald
root	254	2	0	00:27	?	00:00:00	[kworker/0:3-events]
root	265	2	0	00:27	?	00:00:00	[loop0]
root	269	2	0	00:27	?	00:00:00	[loop1]
root	271	2	0	00:27	?	00:00:00	[loop2]
root	274	2	0	00:27	?	00:00:00	[loop3]
root	281	1	1	00:27	?	00:00:02	/lib/systemd/systemd-udevd
root	282	2	0	00:27	?	00:00:00	[loop4]
root	283	2	0	00:27	?	00:00:00	[loop5]
root	284	2	0	00:27	?	00:00:00	[loop6]
root	285	2	0	00:27	?	00:00:00	[loop7]
root	286	2	0	00:27	?	00:00:00	[loop8]
root	287	2	0	00:27	?	00:00:00	[loop9]
root	288	2	0	00:27	?	00:00:00	[loop10]
root	289	2	0	00:27	?	00:00:00	[loop11]
root	290	2	0	00:27	?	00:00:00	[loop12]
root	291	2	0	00:27	?	00:00:00	[loop13]
root	292	2	0	00:27	?	00:00:00	[loop14]
root	293	2	0	00:27	?	00:00:00	[loop15]
root	318	2	0	00:28	?	00:00:00	[irq/121-ACPI:Ev]
root	319	2	0	00:28	?	00:00:00	[irq/122-proc_th]
root	328	2	0	00:28	?	00:00:00	[irq/123-mei_txe]
root	336	2	0	00:28	?	00:00:00	[cfg80211]
root	346	2	0	00:28	?	00:00:00	[cryptd]
root	350	2	0	00:28	?	00:00:00	[wl_event_handle]
root	364	2	0	00:28	?	00:00:00	[card0-crtc0]
root	365	2	0	00:28	?	00:00:00	[card0-crtc1]
root	366	2	0	00:28	?	00:00:00	[card0-crtc2]
root	381	2	0	00:28	?	00:00:00	[kworker/1:4-events]


```
root      407      2 0 00:28 ?      00:00:00 [kworker/u9:1-hci0]
root      408      2 0 00:28 ?      00:00:00 [kworker/u9:2-hci0]
systemd+  595      1 0 00:28 ?      00:00:00 /lib/systemd/systemd-resolved
systemd+  596      1 0 00:28 ?      00:00:00 /lib/systemd/systemd-timesyncd
root      634      1 0 00:28 ?      00:00:00 /usr/lib/accounts-service/accounts-daemon
root      635      1 0 00:28 ?      00:00:00 /usr/sbin/acpid
root      636      1 0 00:28 ?      00:00:00 /usr/sbin/anacron -d -q -s
avahi     638      1 0 00:28 ?      00:00:00 avahi-daemon: running [cesar-HP-Notebook.local]
root      639      1 0 00:28 ?      00:00:00 /usr/lib/bluetooth/bluetoothd
root      640      1 0 00:28 ?      00:00:00 /usr/sbin/cron -f
root      641      1 0 00:28 ?      00:00:00 /usr/sbin/cupsd -l
```

```
message+  642      1 1 00:28 ?      00:00:02 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
```

```
root      643      1 0 00:28 ?      00:00:01 /usr/sbin/NetworkManager --no-daemon
root      652      1 0 00:28 ?      00:00:00 /usr/sbin/irqbalance --foreground
```

```
root      653      1 0 00:28 ?      00:00:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
root      655      1 2 00:28 ?      00:00:04 /usr/lib/policykit-1/polkitd --no-debug
syslog    657      1 0 00:28 ?      00:00:00 /usr/sbin/rsyslogd -n -iNONE
root      672      1 2 00:28 ?      00:00:03 /usr/lib/napd/napd
root      673      1 0 00:28 ?      00:00:00 /usr/libexec/switcheroo-control
root      675      1 0 00:28 ?      00:00:00 /lib/systemd/systemd-logind
root      676      1 0 00:28 ?      00:00:00 /usr/sbin/thermald --systemd --dbus-enable --adaptive
root      680      1 0 00:28 ?      00:00:00 /usr/lib/udisks2/udisksd
root      681      1 0 00:28 ?      00:00:00 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
avahi     687      638 0 00:28 ?      00:00:00 avahi-daemon: chroot helper
root      759      1 0 00:28 ?      00:00:00 /usr/sbin/cups-browsed
root      766      1 0 00:28 ?      00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-sig
root      799      1 0 00:28 ?      00:00:00 /usr/sbin/ModemManager
root      805      1 0 00:28 ?      00:00:00 /usr/sbin/gdm3
rtkit     893      1 0 00:28 ?      00:00:00 /usr/libexec/rtkit-daemon
root      985      1 0 00:28 ?      00:00:00 /usr/lib/upower/upowerd
whoopsie  1024     1 0 00:28 ?      00:00:00 /usr/bin/whoopsie -f
kernoops  1030     1 0 00:28 ?      00:00:00 /usr/sbin/kerneloops --test
kernoops  1032     1 0 00:28 ?      00:00:00 /usr/sbin/kerneloops
root      1108     1 7 00:28 ?      00:00:09 /usr/lib/packagekit/packagekitd
root      1242     805 0 00:28 ?      00:00:00 gdm-session-worker [pam/gdm-password]
colord    1250     1 0 00:28 ?      00:00:00 /usr/libexec/colord
cesar     1274     1 0 00:28 ?      00:00:01 /lib/systemd/systemd --user
cesar     1275     1274 0 00:28 ?      00:00:00 (sd-pam)
cesar     1302     1274 0 00:28 ?      00:00:00 /usr/bin/pulseaudio --daemonize=no --log-target=journal
cesar     1304     1274 1 00:28 ?      00:00:01 /usr/libexec/tracker-miner-fs
cesar     1307     1274 0 00:28 ?      00:00:01 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation
cesar     1312     1 0 00:29 ?      00:00:00 /usr/bin/gnome-keyring-daemon --daemonize --login
cesar     1329     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfsd
cesar     1335     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f -o big_writes
cesar     1344     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfs-udisks2-volume-monitor
cesar     1353     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfs-gphoto2-volume-monitor
cesar     1359     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfs-goa-volume-monitor
cesar     1363     1274 0 00:29 ?      00:00:00 /usr/libexec/goa-daemon
root      1364     2 0 00:29 ?      00:00:00 [krfcommnd]
cesar     1366     1242 0 00:29 tty2      00:00:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnom
cesar     1368     1366 5 00:29 tty2      00:00:06 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -
cesar     1368     1366 5 00:29 tty2      00:00:06 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -
cesar     1376     1274 0 00:29 ?      00:00:00 /usr/libexec/goa-identity-service
cesar     1381     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfs-afc-volume-monitor
cesar     1387     1274 0 00:29 ?      00:00:00 /usr/libexec/gvfs-mtp-volume-monitor
cesar     1418     1366 0 00:29 tty2      00:00:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu
cesar     1487     1418 0 00:29 ?      00:00:00 /usr/bin/ssh-agent /usr/bin/im-launch env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-s
cesar     1508     1274 0 00:29 ?      00:00:00 /usr/libexec/at-spi-bus-launcher
cesar     1514     1508 0 00:29 ?      00:00:00 /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork
cesar     1524     1274 0 00:29 ?      00:00:00 /usr/libexec/gnome-session-ctl --monitor
cesar     1531     1274 0 00:29 ?      00:00:00 /usr/libexec/gnome-session-binary --systemd-service --session=ubuntu
cesar     1549     1274 16 00:29 ?      00:00:18 /usr/bin/gnome-shell
cesar     1572     1549 0 00:29 ?      00:00:00 ibus-daemon --panel disable --xim
cesar     1576     1572 0 00:29 ?      00:00:00 /usr/libexec/ibus-memconf
cesar     1577     1572 2 00:29 ?      00:00:02 /usr/libexec/ibus-extension-gtk3
cesar     1581     1274 0 00:29 ?      00:00:00 /usr/libexec/ibus-x11 --kill-daemon
cesar     1584     1274 0 00:29 ?      00:00:00 /usr/libexec/ibus-portal
cesar     1594     1274 0 00:29 ?      00:00:00 /usr/libexec/at-spi2-registryd --use-gnome-session
cesar     1598     1274 0 00:29 ?      00:00:00 /usr/libexec/xdg-permission-store
cesar     1603     1274 0 00:29 ?      00:00:00 /usr/libexec/gnome-shell-calendar-server
cesar     1611     1274 0 00:29 ?      00:00:00 /usr/libexec/evolution-source-registry
cesar     1616     1274 0 00:29 ?      00:00:00 /usr/libexec/dconf-service
cesar     1622     1274 0 00:29 ?      00:00:00 /usr/libexec/evolution-calendar-factory
cesar     1635     1274 0 00:29 ?      00:00:00 /usr/libexec/evolution-addressbook-factory
cesar     1637     1274 0 00:29 ?      00:00:00 /usr/bin/gjs /usr/share/gnome-shell/org.gnome.Shell.Notifications
cesar     1649     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-ally-settings
cesar     1651     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-color
cesar     1656     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-datetime
cesar     1658     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-housekeeping
cesar     1659     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-keyboard
cesar     1663     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-media-keys
cesar     1664     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-power
cesar     1670     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-print-notifications
cesar     1671     1531 5 00:29 ?      00:00:06 /System/Applications/komorebi
cesar     1672     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-rfkill
cesar     1674     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-screensaver-proxy
cesar     1676     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-sharing
cesar     1679     1531 0 00:29 ?      00:00:00 /usr/libexec/gsd-disk-utility-notify
cesar     1682     1274 0 00:29 ?      00:00:00 /usr/libexec/gsd-smartcard
```



```

cesar      1682    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-smartcard
cesar      1688    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-sound
cesar      1691    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-usb-protection
cesar      1692    1531    0 00:29 ?        00:00:00 /usr/libexec/evolution-data-server/evolution-alarm-notify
cesar      1699    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-wacom
cesar      1702    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-wwan
cesar      1709    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-xsettings
cesar      1723    1329    0 00:29 ?        00:00:00 /usr/libexec/gvfsd-trash --spawner :1.3 /org/gtk/gvfs/exec_spaw/0
cesar      1757    1274    0 00:29 ?        00:00:00 /usr/libexec/gsd-printer
cesar      1788    1572    0 00:29 ?        00:00:00 /usr/libexec/ibus-engine-simple
cesar      1801    1274    13 00:29 ?       00:00:14 /snap/snap-store/547/usr/bin/snap-store --gapplication-service
cesar      1826    1274    0 00:29 ?        00:00:00 /usr/libexec/xdg-document-portal
cesar      1902    1671    0 00:29 ?        00:00:00 /usr/lib/x86_64-linux-gnu/webkit2gtk-4.0/WebKitNetworkProcess 4 18
cesar      1949    1274    0 00:29 ?        00:00:00 /usr/libexec/xdg-desktop-portal
cesar      1953    1274    0 00:29 ?        00:00:00 /usr/libexec/xdg-desktop-portal-gtk
root       1962      1      1 00:29 ?        00:00:01 /usr/libexec/fwupd/fwupd
cesar      2421    1274    0 00:29 ?        00:00:00 /usr/bin/gnome-calendar --gapplication-service
cesar      2424    1274     3 00:29 ?        00:00:02 /usr/libexec/gnome-terminal-server
cesar      2504    2424    0 00:29 pts/0    00:00:00 bash
cesar      2520    1274    0 00:30 ?        00:00:00 /usr/libexec/gvfsd-metadata
cesar      2523    1531    0 00:30 ?        00:00:00 update-notifier
cesar      2585    1274     4 00:30 ?        00:00:00 /usr/libexec/tracker-store
cesar      2592    1274     6 00:30 ?        00:00:00 /usr/libexec/tracker-extract
cesar      2611    2504    0 00:31 pts/0    00:00:00 ps -fea
cesar@cesar-HP-Notebook:~$ █

```

¿Qué información le proporcionan los comandos anteriores?

El comando **ps** despliega información por consola acerca de la selección de los procesos activos.

El resultado y significado de las columnas:

UID ->El identificador de Usuario

PID ->El ID único del proceso

PPID -> El ID único del proceso padre

TTY ->El tipo de terminal en el que el usuario está logueado.

TIME ->Cantidad del CPU en minutos y segundos que el proceso a estado corriendo

CMD ->Nombre del comando que lanzó el proceso

```

PS(1)                                     User Commands                                     PS(1)

NAME
ps - report a snapshot of the current processes.

SYNOPSIS
ps [options]

DESCRIPTION
ps displays information about a selection of the active processes.  If you want a repetitive update of the selection and the displayed information, use top(1) instead.

This version of ps accepts several kinds of options:

1  UNIX options, which may be grouped and must be preceded by a dash.
2  BSD options, which may be grouped and must not be used with a dash.
3  GNU long options, which are preceded by two dashes.

```

2. A través de la ayuda en línea que proporciona Linux, se investigó para que se utiliza el comando **ps** así como sus opciones que se pueden utilizar con dicho comando:

DESCRIPCIÓN: muestra una instantánea de los procesos actuales. Si quiere una actualización continua.

OPCIONES DE LA LÍNEA DE COMANDOS:

Las opciones de la línea de comandos para esta versión de **ps** proceden de la versión **BSD** de **ps**, no de la versión System V.

Es recomendado que los argumentos de la línea de comandos no estén precedidos por un carácter '-', porque él '-' se utiliza para indicar argumentos del estándar Unix98, mientras que sin '-' indicará el modo actual '**BSD** extendido'

l formato largo

u formato usuario: muestra el usuario y la hora de inicio

j formato trabajo (jobs): pgid sid

s formato señal (signal)

v formato vm

m muestra información de memoria (combínese con p para obtener el número de páginas).

f formato "forest" ("bosque") de familias en forma de árbol

a muestra también los procesos de otros usuarios

x muestra procesos que no están controlados por ninguna terminal

S añade tiempo de CPU y fallos de página de los hijos

c nombre del comando obtenido de task_struct

e muestra ambiente (environment) después del nombre del comando y `+'

w Salida ancha (wide): no se truncan las líneas de comando para que quepan en una línea. Para ser exactos, cada w que se especifica añadirá una posible línea a la salida. Si el espacio no se necesita, no se usa. Puede usar hasta 100 w's.

h sin cabecera (header)

r sólo procesos que se están ejecutando

n salida numérica para USER y WCHAN.

txx sólo procesos controlados por la tty xx; para xx debe usar bien el nombre de un dispositivo bajo "/dev" o bien ese nombre sin las letras tty o cu que lo preceden.

Esta es la operación inversa que ps utiliza para imprimir el nombre abreviado de tty en el campo TT, por ejemplo, ps -tl.

O[+|-]k1[,+|-]k2[,...]

Ordena la lista de procesos de acuerdo con el ordenamiento multi-nivel especificado por la secuencia de claves ordenación de CLAVES DE ORDENACIÓN, k1, k2. Existen especificaciones de ordenación por defecto para cada uno de los formatos de ps.

pids Lista sólo los procesos especificados; están delimitados por comas. La lista se debe dar inmediatamente después de la última opción en un argumento simple, sin intervención de espacios, por ejemplo ps -jl,4,5. Las listas especificadas en los argumentos siguientes son concatenadas, por ejemplo ps -l 1,2 3,4 5 6 listará todos los procesos del 1 al 6 en formato largo. Los pids se listan incluso si contradicen a las opciones 'a' y 'x'

OPCIONES DE LA LÍNEA DE COMANDOS LARGAS:

Estas opciones están precedidas por un doble guión.

--sortX[+|-]key[,+|-]key[,...]

Selecciona una clave de varias letras de la sección CLAVES DE ORDENACIÓN. X puede ser cualquier carácter de separación. GNU prefiere '='. El '+' es realmente opcional, ya que la dirección por defecto es creciente en orden numérico o lexicográfico. Por ejemplo: ps -jax --sort=uid,-ppid,+pid

--help

Muestra un mensaje de ayuda que resume el uso y da una lista de claves de ordenación soportadas. Esa lista puede estar más actualizada que la de esta página del manual.

--versión

Muestra la versión y la procedencia de este programa.

Además se investigó el uso de las llamadas al sistema **fork()**, **execv()**, **getpid()**, **getppid()** y **wait()** en la ayuda en línea.

fork()

El proceso puede ejecutar más de una función al mismo tiempo, lo cual puede crear procesos enteramente separados.

La mayoría de sistemas operativos identifican a los procesos de acuerdo a un identificador único de proceso (pid) el cual es comúnmente un número entero.

En UNIX la llamada al sistema **fork()** es usada para crear un nuevo proceso, el cual se convierte en el proceso hijo de la llamada. Ambos procesos se ejecutan simultáneamente desde la siguiente instrucción que sigue a la llamada al sistema **fork()**. Necesitamos distinguir el padre del hijo. Esto se puede hacer probando el valor de retorno del **fork()** el cual regresa un pid con un valor:

-1 si ocurrió un error y el **fork()** fallo

0 para el proceso hijo

Identificación positiva del proceso hijo al padre

wait()

Nos la podemos arreglar para que el proceso padre espere a que el hijo termine antes de continuar si llamamos a **wait()**

La sintaxis para **wait()** seria:

pid_t wait(int *stat_val);

El valor de retorno es el PID del proceso hijo el cual ha terminado.

El argumento es la información de estado que permite que el proceso padre determine el estado de salida del proceso hijo, esto es, el valor regresado del principal o pasado a la salida. Si **stat_val** (el valor de estado) no es un apuntador a nulo, la información de estado será escrita a la localización a la cual apunta.

getpid()

La llamada al sistema **getpid** devuelve la identificación del proceso que lo invoca. ¿Cómo averigua el kernel qué proceso está invocando la llamada al sistema?

El kernel realiza la programación de trabajos y proporciona llamadas al sistema.

Cuando un proceso se está ejecutando, el kernel programa su tiempo de ejecución, especialmente le asigna un PID, dicha información se almacena dentro del espacio de direcciones del kernel, en estructuras de datos (por ejemplo, dentro de una estructura de tareas).

Por tanto cuando un proceso llama al **getpid()**, el kernel solo tiene que buscar en la estructura de tareas del proceso de llamada (es decir, en ejecución actualmente).

getppid()

Es una llamada al sistema para el control del procesos que retorna el pid del padre del proceso que la invoca.

En forma análoga a **getpid** no recibe ningún argumento, retorna un entero del tipo **pid_t** y requiere de los archivos de cabecera:

sys/types.h
unistd.h

execv()

La llamada al sistema reemplaza el programa que se está ejecutando en el momento con una imagen de programa recién cargada.

La sintaxis para **execv** seria:

execv(const char *program, char **args);

-> El primer argumento a **execv()** debe de ser el path(dirección) al programa

-> El segundo argumento a **execv()** debe de ser un arreglo de apuntadores hacia las cadenas terminadas en nulo que representen la lista de argumentos disponibles para el nuevo programa.

La Familia **exec()**

La Familia **exec()** de funciones reemplaza la imagen del proceso actual con una nueva imagen de proceso. El argumento inicial de estas funciones es el nombre de un documento que es el que se va a ejecutar. Las funciones pueden ser agrupadas basadas en las letras siguientes del prefijo "exec"

l -> execl(), execlp(), execlxe()

La constante char *arg y los elipses subsecuentes pueden ser pensados como arg0, arg1, ... argn. Juntos estos describen una lista de uno o más apuntadores a una cadena terminada en nulo que representa el argumento lista disponible para la ejecución del programa.

v -> execlt(), execltp(), execltpe()

El argumento char * const argv[] es un arreglo de apuntadores a cadenas terminadas en nulo que representen el argumento de lista disponible para el nuevo programa. El primer argumento, por convención, debe de ser un apuntador al nombre del archivo asociado con el archivo que se está ejecutando.

e -> execlxe(), execltpe()

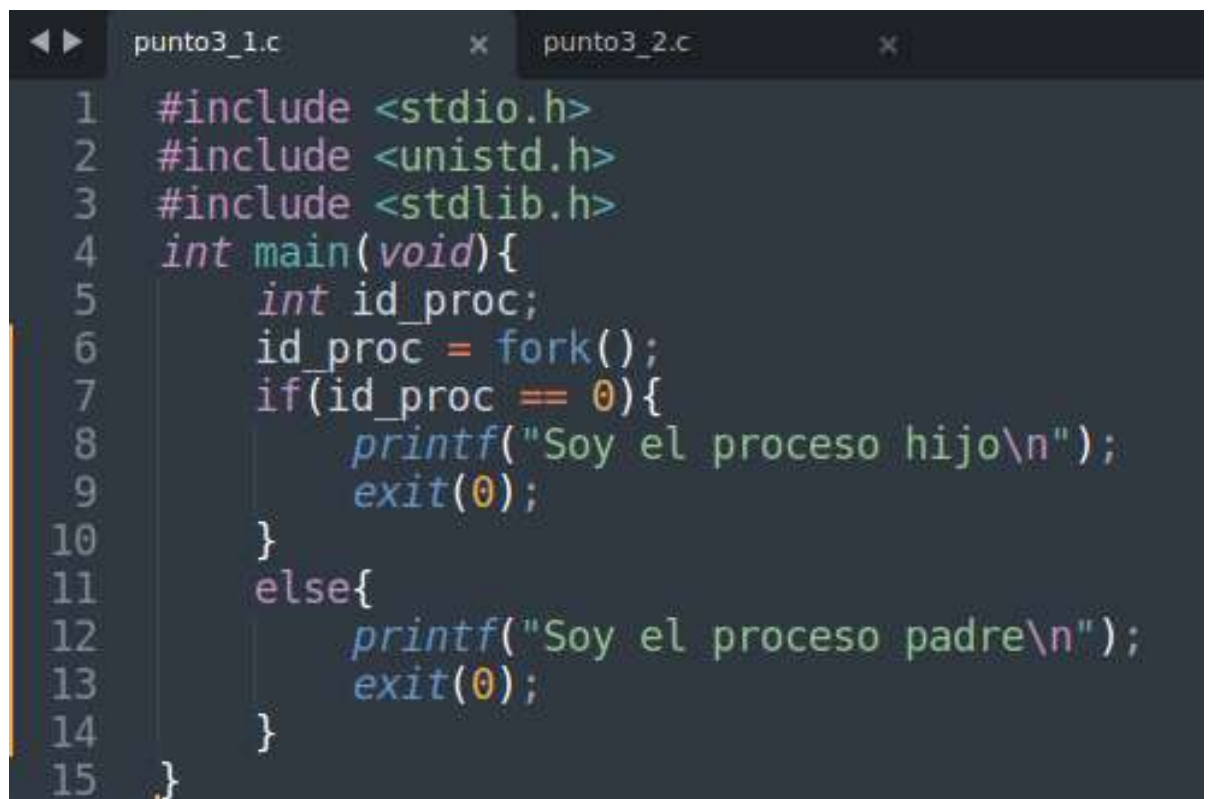
El ambiente del que llama esta especificado via el argumento envp. El argumento envp es un arreglo de apuntadores a cadenas terminadas en null y deben de estar terminadas en un apuntador nulo.

p -> execlxe(), execltp(), execltpe()

Estas funciones duplican las acciones del shell en busca de un archivo ejecutable si el nombres de archivo especificado no contiene un slash(/). El archivo es buscado en la lista separada con dos puntos del directorio de rutas de nombres especificados en el medio PATH.

3. Se capturó, compilo y ejecuto los dos programas de creación de un nuevo proceso por copia exacta de código que a continuación se muestran:

Programa 1:



```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  int main(void){
5      int id_proc;
6      id_proc = fork();
7      if(id_proc == 0){
8          printf("Soy el proceso hijo\n");
9          exit(0);
10     }
11     else{
12         printf("Soy el proceso padre\n");
13         exit(0);
14     }
15 }
```

Programa 2:

```
punto3_1.c  x  punto3_2.c  x
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  int main(void){
5      int id_proc;
6      id_proc = fork();
7      if(id_proc == 0){
8          printf("Soy el proceso hijo\n");
9      }
10     else{
11         printf("Soy el proceso padre\n");
12     }
13     printf("Mensaje en ambos\n");
14     exit(0);
15 }
```

Compilación:

```
cesar@cesar-HP-Notebook: ~/Documentos/SO/Practica3
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ls
punto3_1.c  punto3_2.c
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ gcc punto3_1.c -o punto3_1
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ gcc punto3_2.c -o punto3_2
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ls
punto3_1  punto3_1.c  punto3_2  punto3_2.c
```

Ejecución:

Programa 1:

```
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto3_1
Soy el proceso padre
Soy el proceso hijo
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto3_1
Soy el proceso padre
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ Soy el proceso hijo
./punto3_1
Soy el proceso padre
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ Soy el proceso hijo
./punto3_1
Soy el proceso padre
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ Soy el proceso hijo

cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto3_1
Soy el proceso padre
Soy el proceso hijo
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto3_1
Soy el proceso padre
Soy el proceso hijo
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto3_1
Soy el proceso padre
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ Soy el proceso hijo
./punto3_1
Soy el proceso padre
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ Soy el proceso hijo
```

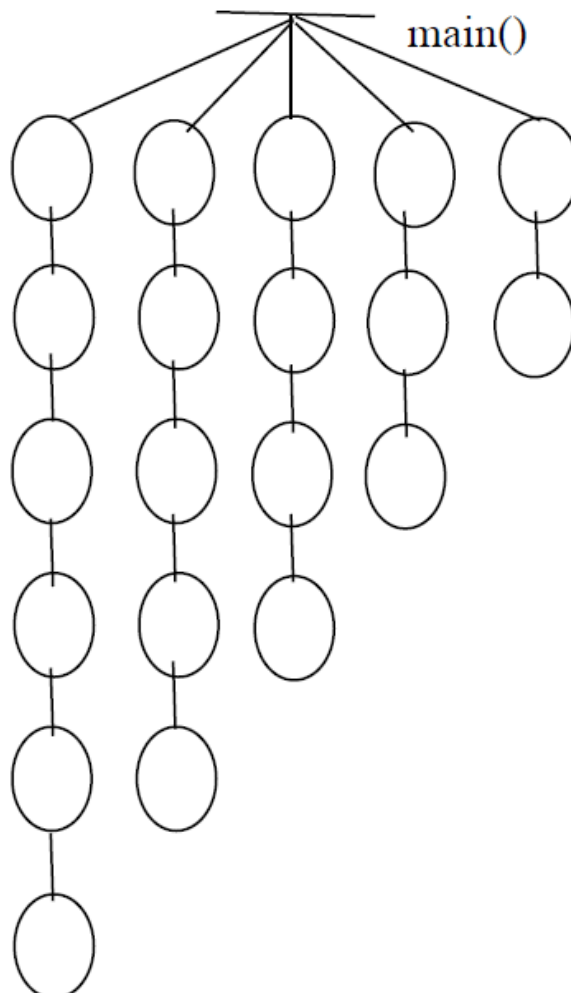

Programa 2:

```
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ ./punto3_2
Soy el proceso padre
Mensaje en ambos
Soy el proceso hijo
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ ./punto3_2
Soy el proceso padre
Mensaje en ambos
Soy el proceso hijo
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ ./punto3_2
Soy el proceso padre
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ Soy el proceso hijo
Mensaje en ambos
./punto3_2
Soy el proceso padre
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ Soy el proceso hijo
Mensaje en ambos

cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ ./punto3_2
Soy el proceso padre
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ Soy el proceso hijo
Mensaje en ambos
./punto3_2
Soy el proceso padre
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ Soy el proceso hijo
Mensaje en ambos

cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ ./punto3_2
Soy el proceso padre
Mensaje en ambos
cesar@cesar-HP-Notebook:~/Documentos/S0/Practica3$ Soy el proceso hijo
Mensaje en ambos
□
```

4. Se programó una aplicación que crea el árbol de procesos siguiente:



Para cada uno de los procesos hijos creados(por copia exacta de código) se imprime en pantalla el pid de su padre, además se imprime en pantalla los pids de los hijos creados de cada proceso padre.

Código:

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/types.h>
5
6  int main()
7  {
8      /*
9       pid = 0  <-- Proceso Hijo
10      pid < 0  <-- Proceso Padre
11      pid = -1 <-- Error
12      */
13      int a,b;
14      pid_t pid;
15
16      for(a = 1; a <= 5;a++){
17          if(fork() == 0){
18              if(a == 1){
19                  for(b = 0; b < 1;b++){
20                      if(fork() > 0){
21                          break;
22                      }
23                  }
24                  printf("[%d_%d]Soy el proceso (%d), hijo de(%d).\n",a,b,getpid(),getppid());
25              }
26              if(a == 2){
27                  for(b = 0; b < 2;b++){
28                      if(fork() > 0){
29                          break;
30                      }
31                  }
32                  printf("[%d_%d]Soy el proceso (%d), hijo de(%d).\n",a,b,getpid(),getppid());
33              }
34              if(a == 3){
35                  for(b = 0; b < 3 ;b++){
36                      if(fork() > 0){
37                          break;
38                      }
39                  }
40                  printf("[%d_%d]Soy el proceso (%d), hijo de(%d).\n",a,b,getpid(),getppid());
41              }
42              if(a == 4){
43                  for(b = 0; b < 4;b++){
44                      if(fork() > 0){
45                          break;
46                      }
47                  }
48                  printf("[%d_%d]Soy el proceso (%d), hijo de(%d).\n",a,b,getpid(),getppid());
49              }
50              if(a == 5){
51                  for(b = 0; b < 5;b++){
52                      if(fork() > 0){
53                          break;
54                      }
55                  }
56                  printf("[%d_%d]Soy el proceso (%d), hijo de(%d).\n",a,b,getpid(),getppid());
57              }
58              break;
59          }
60      }
61      while(1);
62      return 0;
63  }
```

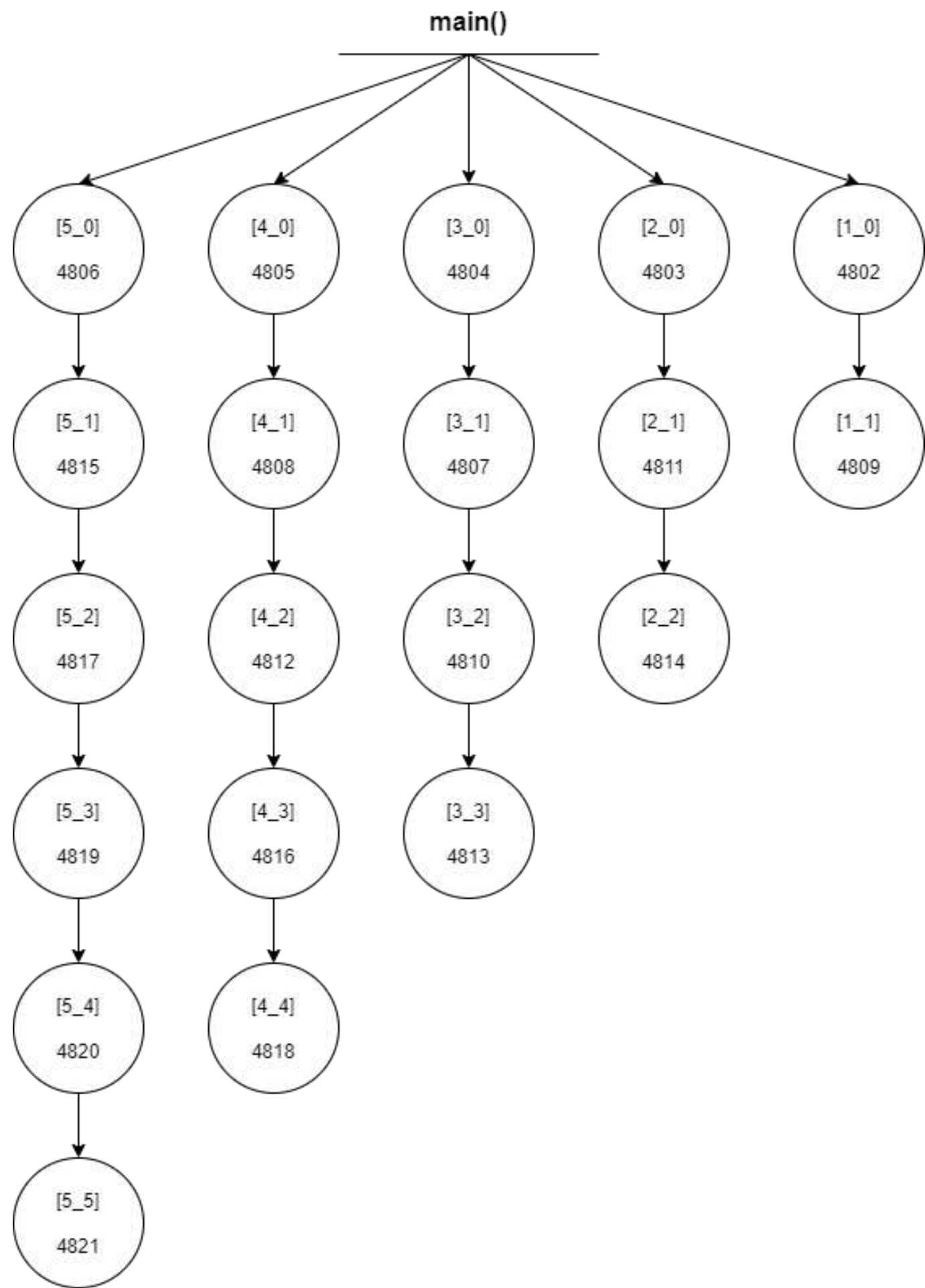

Compilación:

```
cesar@cesar-HP-Notebook: ~/Documentos/SO/Practi...
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ gcc punto4.c -o punto4
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$ ./punto4
[3_0]Soy el proceso (4804), hijo de(4801).
[4_0]Soy el proceso (4805), hijo de(4801).
[1_0]Soy el proceso (4802), hijo de(4801).
[3_1]Soy el proceso (4807), hijo de(4804).
[2_0]Soy el proceso (4803), hijo de(4801).
[4_1]Soy el proceso (4808), hijo de(4805).
[3_2]Soy el proceso (4810), hijo de(4807).
[1_1]Soy el proceso (4809), hijo de(4802).
[2_1]Soy el proceso (4811), hijo de(4803).
[5_0]Soy el proceso (4806), hijo de(4801).
[4_2]Soy el proceso (4812), hijo de(4808).
[3_3]Soy el proceso (4813), hijo de(4810).
[5_1]Soy el proceso (4815), hijo de(4806).
[2_2]Soy el proceso (4814), hijo de(4811).
[4_3]Soy el proceso (4816), hijo de(4812).
[5_2]Soy el proceso (4817), hijo de(4815).
[4_4]Soy el proceso (4818), hijo de(4816).
[5_3]Soy el proceso (4819), hijo de(4817).
[5_4]Soy el proceso (4820), hijo de(4819).
[5_5]Soy el proceso (4821), hijo de(4820).
```

Verificación de los procesos con el comando ps -fe:

```
cesar@cesar-HP-Notebook: ~/Docu...
cesar      4801    2874   7 20:31 pts/0    00:00:00 ./punto4
cesar      4802    4801   6 20:31 pts/0    00:00:00 ./punto4
cesar      4803    4801   8 20:31 pts/0    00:00:00 ./punto4
cesar      4804    4801   7 20:31 pts/0    00:00:00 ./punto4
cesar      4805    4801   8 20:31 pts/0    00:00:00 ./punto4
cesar      4806    4801   6 20:31 pts/0    00:00:00 ./punto4
cesar      4807    4804   6 20:31 pts/0    00:00:00 ./punto4
cesar      4808    4805   6 20:31 pts/0    00:00:00 ./punto4
cesar      4809    4802   7 20:31 pts/0    00:00:00 ./punto4
cesar      4810    4807   7 20:31 pts/0    00:00:00 ./punto4
cesar      4811    4803   7 20:31 pts/0    00:00:00 ./punto4
cesar      4812    4808   7 20:31 pts/0    00:00:00 ./punto4
cesar      4813    4810   8 20:31 pts/0    00:00:00 ./punto4
cesar      4814    4811   7 20:31 pts/0    00:00:00 ./punto4
cesar      4815    4806   7 20:31 pts/0    00:00:00 ./punto4
cesar      4816    4812   6 20:31 pts/0    00:00:00 ./punto4
cesar      4817    4815   7 20:31 pts/0    00:00:00 ./punto4
cesar      4818    4816   6 20:31 pts/0    00:00:00 ./punto4
cesar      4819    4817   7 20:31 pts/0    00:00:00 ./punto4
cesar      4820    4819   6 20:31 pts/0    00:00:00 ./punto4
cesar      4821    4820   7 20:31 pts/0    00:00:00 ./punto4
cesar      4823    2951   0 20:31 pts/1    00:00:00 ps -fe
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica3$
```

Diagrama gráfico de los procesos creados:



5. Se programó una aplicación que crea 5 procesos (por copia exacta de código). Los cuales manipulan una matriz de 7x7 realizando suma, resta, multiplicación, la transpuesta y mostrar los resultados en pantalla, respectivamente. Posterior a eso, es codificar la misma aplicación pero sin la creación de procesos.

Código con fork()

```
Ejer5.c x Ejer5-Sec.c x creadorDeProc.c x hola.c x
1 #include <unistd.h>
2 #include <stdio.h>
3 #include <sys/wait.h>
4 #include <stdlib.h>
5
6 int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
7 int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
8 int mr[7][7];
9 int i = 7, j = 7;
10 int main(){
11     int i;
12     int pid;
13     pid = fork();
14
15     if(pid == 0){
16         printf("Soy el Primer Proceso Hijo que es la Suma \n");//Primer Proceso
17         printf("Suma\n");
18         for(i=0;i<7;i++){
19             for(j=0;j<7;j++){
20                 mr[i][j] = m1[i][j] + m2[i][j];
21                 printf("%d ",mr[i][j]);
22             }
23             printf("\n");
24         }
25         printf("\n");
26         exit(1);
27     }
28     else{
29         printf("Soy el Proceso Padre de todos\n");
30         pid = fork();
31         if(pid == 0){
32             printf("Soy el Segundo Proceso Hijo que es la resta\n");//Segundo Proceso
33             printf("Resta\n");
34             for(i=0;i<7;i++){
35                 for(j=0;j<7;j++){
36                     mr[i][j] = m1[i][j] - m2[i][j];
37                     printf("%d ",mr[i][j]);
38                 }
39                 printf("\n");
40             }
41             printf("\n");
42             exit(2);
43         }
44         else{
45             pid = fork();
46             if(pid == 0){
47                 printf("Soy el Tercer Proceso Hijo que es la multiplicacion\n");//Tercer Proceso
48                 printf("Multiplicacion\n");
49                 for(i=0;i<7;i++){
50                     for(j=0;j<7;j++){
51                         mr[i][j] = m1[i][j] * m2[i][j];
52                         printf("%d ",mr[i][j]);
53                     }
54                     printf("\n");
55                 }
56                 printf("\n");
57                 exit(3);
58             }
59         }
60     }
61 }
```

```

59 ▾ else{
60     pid = fork();
61 ▾     if(pid == 0){
62         printf("Soy el Cuarto Proceso Hijo que son las matrices transpuestas\n");//Cuarto Proceso
63         printf("Matrices Transpuestas\n");
64         int mr2[7][7];
65 ▾         for(i=0;i<7;i++){
66 ▾             for(j=0;j<7;j++){
67                 mr[i][j] = m1[j][i];
68                 mr2[i][j] = m2[j][i];
69                 printf("%d ",mr[i][j]);
70                 printf("%d ",mr2[i][j]);
71             }
72             printf("\n");
73         }
74         printf("\n");
75         exit(4);
76     }
77 ▾     else{
78         pid = fork();
79 ▾         if(pid == 0){
80             printf("Soy el Quinto Proceso Hijo \n");//Quinto Proceso
81             exit(5);
82         }
83     }
84 }
85 }
86 }
87 return 0;
88 }

```

Soy el Primer Proceso Hijo que es la Suma

Suma

```

8 6 5 10 8 7 10
7 5 7 12 5 3 8
8 6 5 10 8 7 10
12 7 7 10 6 4 5
4 7 4 8 5 12 8
4 7 4 8 5 12 8
9 10 10 10 5 3 5

```

Soy el Segundo Proceso Hijo que es la resta

```

1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4

```

Resta

```

-6 -2 1 -2 2 5 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
-3 3 3 0 -1 -1 0
4 6 6 6 6 4 6 4 5 3 3 5 4 6
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
-2 -1 0 2 3 2 4
2 1 0 -2 -3 -2 -4
5 2 0 -2 1 1 -3

```

Matrices Transpuestas

```

-6 -2 1 -2 2 5 4
1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
-3 3 3 0 -1 -1 0
4 6 6 6 6 4 6 4 5 3 3 5 4 6
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
5 3 2 3 3 5 3 3 4 1 1 4 3 2
-2 -1 0 2 3 2 4
6 1 1 2 1 6 2 2 7 5 5 7 2 1
7 3 4 4 3 7 4 1 6 2 2 6 1 4
2 1 0 -2 -3 -2 -4

5 2 0 -2 1 1 -3

```

Código secuencial

```
Ejer5.c x Ejer5-Sec.c x creadorDeProc.c x hola.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void suma(int m1[7][7],int m2[7][7],int mr[7][7]);
5 void resta(int m1[7][7],int m2[7][7],int mr[7][7]);
6 void multi(int m1[7][7],int m2[7][7],int mr[7][7]);
7 void transpuesta(int m1[7][7],int m2[7][7],int mr[7][7]);
8 void mostrar(int m1[7][7],int m2[7][7],int mr[7][7]);
9 int i,j;
10 int main(){
11     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
12     int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
13     int mr[7][7];
14     int opcion;
15     printf("Selecciona un proceso : ");
16     scanf("%d",&opcion);
17
18     if(opcion == 1){
19         suma(m1,m2,mr);
20     }else if(opcion == 2){
21         resta(m1,m2,mr);
22     }
23     else if(opcion == 3){
24         multi(m1,m2,mr);
25     }
26     else if(opcion == 4){
27         transpuesta(m1,m2,mr);
28     }else if(opcion == 5){
29         mostrar(m1,m2,mr);
30     }else{
31         printf("Opcion incorrecta\n");
32     }
33 }
```

```
Ejer5.c x Ejer5-Sec.c x creadorDeProc.c x Ejer5.c x Ejer5-Sec.c x creadorDeProc.c x ho
37 void suma(int m1[7][7],int m2[7][7],int mr[7][7]){
38     printf("Suma\n");
39     for(i=0;i<7;i++){
40         for(j=0;j<7;j++){
41             mr[i][j] = m1[i][j] + m2[i][j];
42             printf("%d ",mr[i][j]);
43         }
44         printf("\n");
45     }
46     printf("\n");
47 }
48 void resta(int m1[7][7],int m2[7][7],int mr[7][7]){
49     printf("Resta\n");
50     for(i=0;i<7;i++){
51         for(j=0;j<7;j++){
52             mr[i][j] = m1[i][j] - m2[i][j];
53             printf("%d ",mr[i][j]);
54         }
55         printf("\n");
56     }
57     printf("\n");
58 }
59 void multi(int m1[7][7],int m2[7][7],int mr[7][7]){
60     printf("Multiplicacion\n");
61     for(i=0;i<7;i++){
62         for(j=0;j<7;j++){
63             mr[i][j] = m1[i][j] * m2[i][j];
64             printf("%d ",mr[i][j]);
65         }
66         printf("\n");
67     }
68     printf("\n");
69 }
58 }
59 void multi(int m1[7][7],int m2[7][7],int mr[7][7]){
60     printf("Multiplicacion\n");
61     for(i=0;i<7;i++){
62         for(j=0;j<7;j++){
63             mr[i][j] = m1[i][j] * m2[i][j];
64             printf("%d ",mr[i][j]);
65         }
66         printf("\n");
67     }
68     printf("\n");
69 }
70 void transpuesta(int m1[7][7], int m2[7][7], int mr[7][7]){
71     printf("Matrices Transpuestas\n");
72     int mr2[7][7];
73     for(i=0;i<7;i++){
74         for(j=0;j<7;j++){
75             mr[i][j] = m1[j][i];
76             mr2[i][j] = m2[j][i];
77             printf("%d ",mr[i][j]);
78             printf("%d ",mr2[i][j]);
79         }
80         printf("\n");
81     }
82     printf("\n");
83 }
84 void mostrar(int m1[7][7],int m2[7][7],int mr[7][7]){
85     suma(m1,m2,mr);
86     resta(m1,m2,mr);
87     multi(m1,m2,mr);
88     transpuesta(m1,m2,mr);
89 }
```

Lín. 23, col. 23 Sel. 0 (1) 2058 caracteres, 89 líneas


```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer5-Sec.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Selecciona un proceso : 1
Suma
8 6 5 10 8 7 10
7 5 7 12 5 3 8
8 6 5 10 8 7 10
12 7 7 10 6 4 5
4 7 4 8 5 12 8
4 7 4 8 5 12 8
9 10 10 10 5 3 5
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer5-Sec.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Selecciona un proceso : 2
Resta
-6 -2 1 -2 2 5 4
-3 3 3 0 -1 -1 0
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
-2 -1 0 2 3 2 4
2 1 0 -2 -3 -2 -4
5 2 0 -2 1 1 -3
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer5-Sec.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Selecciona un proceso : 3
Multiplicacion
7 8 6 24 15 6 21
10 4 10 36 6 2 16
7 8 6 24 15 6 21
35 6 10 24 9 4 4
3 12 4 15 4 35 12
3 12 4 15 4 35 12
14 24 25 24 6 2 4
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer5-Sec.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Selecciona un proceso : 4
Matrices Transpuestas
1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
4 6 6 6 6 4 6 4 5 3 3 5 4 6
5 3 2 3 3 5 3 3 4 1 1 4 3 2
6 1 1 2 1 6 2 2 7 5 5 7 2 1
7 3 4 4 3 7 4 1 6 2 2 6 1 4
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer5-Sec.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Selecciona un proceso : 5
Suma
8 6 5 10 8 7 10
7 5 7 12 5 3 8
8 6 5 10 8 7 10
12 7 7 10 6 4 5
4 7 4 8 5 12 8
4 7 4 8 5 12 8
9 10 10 10 5 3 5

Resta
-6 -2 1 -2 2 5 4
-3 3 3 0 -1 -1 0
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
-2 -1 0 2 3 2 4
2 1 0 -2 -3 -2 -4
5 2 0 -2 1 1 -3

Multiplicacion
7 8 6 24 15 6 21
10 4 10 36 6 2 16
7 8 6 24 15 6 21
35 6 10 24 9 4 4
3 12 4 15 4 35 12
3 12 4 15 4 35 12
14 24 25 24 6 2 4

Matrices Transpuestas
1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
4 6 6 6 6 4 6 4 5 3 3 5 4 6
5 3 2 3 3 5 3 3 4 1 1 4 3 2
6 1 1 2 1 6 2 2 7 5 5 7 2 1
7 3 4 4 3 7 4 1 6 2 2 6 1 4
```

Pudimos observar y analizar que con la creación de procesos, es más rápido en cuanto a su tiempo de ejecución comparado con una programación secuencial.

6. Capturamos el programa propuesto en la práctica para observar y analizar su funcionamiento

```
1 #include <stdio.h> /* Programa creador de un nuevo proceso */
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <stdlib.h>
5 #include <sys/wait.h>
6 int main() {
7     pid_t pid;
8     char *argv[3];
9     argv[0]="/home/joel/Documentos/MisProgramas/hola"; /*cambiar por ruta propia*/
10    argv[1]="Desde el Hijo";
11    argv[2]=NULL;
12    if((pid=fork())==-1)
13        printf("Error al crear el proceso hijo\n");
14    if(pid==0){
15        printf("Soy el hijo ejecutando: %s\n", argv[0]);
16        execv(argv[0],argv);
17    }
18    else{
19        wait(0);
20        printf("Soy el Padre\n");
21        exit(0);
22    }
23 }
```

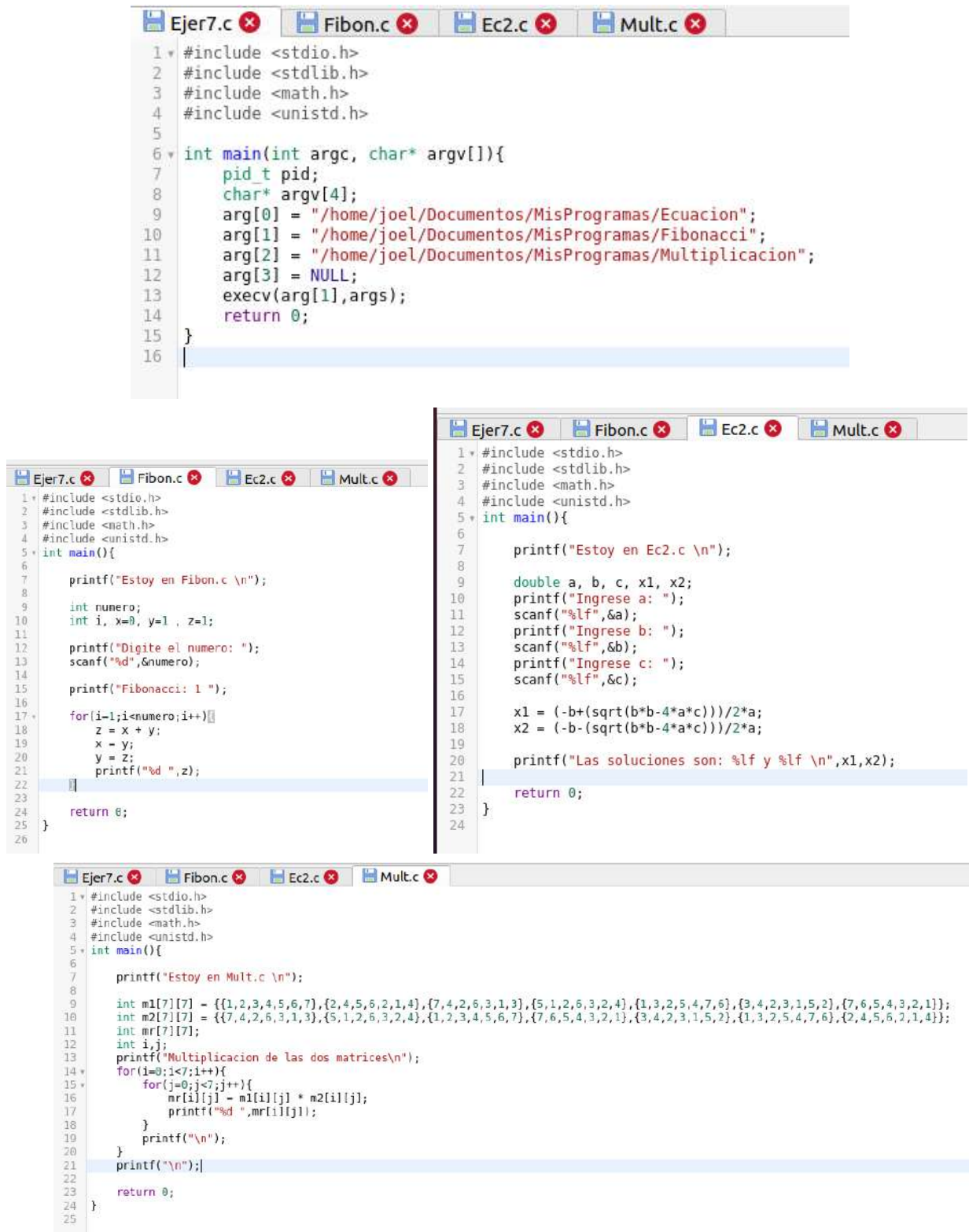
```
1 /* hola.c Programa que será invocado */
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5 int main(int argc, char *argv[])
6 {
7     char mensaje[100];
8     strcpy(mensaje,"Hola Mundo ");
9     strcat(mensaje,argv[1]);
10    printf("%s\n",mensaje);
11    exit(0);
12    return 0;
13 }
```

```
joel@joel-VirtualBox: ~/Documentos/MisProgramas
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc hola.c -o hola
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ll
total 60
drwxrwxr-x 3 joel joel 4096 sep 17 14:59 ./
drwxr-xr-x 4 joel joel 4096 sep 4 22:08 ../
-rwxrwxr-x 1 joel joel 16368 sep 17 14:58 a.out*
-rwxrwxrwx 1 joel joel 576 sep 17 14:58 creadorDeProc.c*
-rw-rw-r-- 1 joel joel 2019 sep 17 13:55 Ejer5.c
-rw-rw-r-- 1 joel joel 2058 sep 17 14:26 Ejer5-Sec.c
-rwxrwxr-x 1 joel joel 16184 sep 17 14:59 hola*
-rwxrwxrwx 1 joel joel 282 sep 17 14:45 hola.c*
drwxrwxr-x 2 joel joel 4096 sep 17 14:40 Prac2/
joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc creadorDeProc.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./a.out
Soy el hijo ejecutando: hola
Hola Mundo Desde el Hijo
Soy el Padre
joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

7. Se programó una aplicación la cuál creará un proceso hijo a partir de un proceso padre y ese hijo a su vez creará 3 procesos hijos más. Ahora cada uno de estos 3 procesos hijos, ejecutará 3 programas diferentes por sustitución de código. El primero resuelve una ecuación algebraica de segundo grado mediante la fórmula general. El segundo mostrará la serie Fibonacci de un número N y el tercero obtendrá la multiplicación de dos matrices de 7x7 elementos.

Códigos



The image displays three screenshots of a C code editor, each showing a different child process created by the main program. The editor windows are titled 'Ejer7.c', 'Fibon.c', 'Ec2.c', and 'Mult.c'.

Ejer7.c (Main Program):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <unistd.h>
5
6 int main(int argc, char* argv){
7     pid_t pid;
8     char* argv[4];
9     arg[0] = "/home/joel/Documentos/MisProgramas/Ecuacion";
10    arg[1] = "/home/joel/Documentos/MisProgramas/Fibonacci";
11    arg[2] = "/home/joel/Documentos/MisProgramas/Multiplicacion";
12    arg[3] = NULL;
13    execv(arg[1],args);
14    return 0;
15 }
16
```

Fibon.c (Fibonacci Series):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <unistd.h>
5 int main(){
6
7     printf("Estoy en Fibon.c \n");
8
9     int numero;
10    int i, x=0, y=1, z=1;
11
12    printf("Digite el numero: ");
13    scanf("%d",&numero);
14
15    printf("Fibonacci: 1 ");
16
17    for(i=1;i<numero;i++){
18        z = x + y;
19        x = y;
20        y = z;
21        printf("%d ",z);
22    }
23
24    return 0;
25 }
26
```

Ec2.c (Quadratic Equation Solver):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <unistd.h>
5 int main(){
6
7     printf("Estoy en Ec2.c \n");
8
9     double a, b, c, x1, x2;
10    printf("Ingreso a: ");
11    scanf("%lf",&a);
12    printf("Ingreso b: ");
13    scanf("%lf",&b);
14    printf("Ingreso c: ");
15    scanf("%lf",&c);
16
17    x1 = (-b+(sqrt(b*b-4*a*c)))/2*a;
18    x2 = (-b-(sqrt(b*b-4*a*c)))/2*a;
19
20    printf("Las soluciones son: %lf y %lf \n",x1,x2);
21
22    return 0;
23 }
24
```

Mult.c (Matrix Multiplication):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <unistd.h>
5 int main(){
6
7     printf("Estoy en Mult.c \n");
8
9     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
10    int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
11    int nr[7][7];
12    int i,j;
13    printf("Multiplicacion de las dos matrices\n");
14    for(i=0;i<7;i++){
15        for(j=0;j<7;j++){
16            nr[i][j] = m1[i][j] * m2[i][j];
17            printf("%d ",nr[i][j]);
18        }
19        printf("\n");
20    }
21    printf("\n");
22
23    return 0;
24 }
25
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ll
total 108
drwxrwxr-x 3 joel joel 4096 sep 18 00:03 ./
drwxr-xr-x 4 joel joel 4096 sep 4 22:08 ../
-rwxrwxrwx 1 joel joel 541 sep 17 20:23 creadorDeProc.c*
-rw-rw-r-- 1 joel joel 448 sep 17 23:49 Ec2.c
-rwxrwxr-x 1 joel joel 16240 sep 18 00:03 Ecuacion*
-rw-rw-r-- 1 joel joel 1937 sep 17 19:26 Ejer5.c
-rw-rw-r-- 1 joel joel 2058 sep 17 14:26 Ejer5-Sec.c
-rw-rw-r-- 1 joel joel 104 sep 17 20:25 Ejer7.c
-rwxrwxr-x 1 joel joel 16240 sep 17 23:06 Fibonacci*
-rw-rw-r-- 1 joel joel 369 sep 17 23:41 Fibon.c
-rwxrwxr-x 1 joel joel 16184 sep 17 14:59 hola*
-rwxrwxrwx 1 joel joel 282 sep 17 14:45 hola.c*
-rw-rw-r-- 1 joel joel 647 sep 17 23:42 Mult.c
-rwxrwxr-x 1 joel joel 16240 sep 17 23:08 Multiplicacion*
drwxrwxr-x 2 joel joel 4096 sep 17 14:40 Prac2/
joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer7.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./Fibonacci
Estoy en Fibon.c
Digite el numero: 5
Fibonacci: 1 1 2 3 5 joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

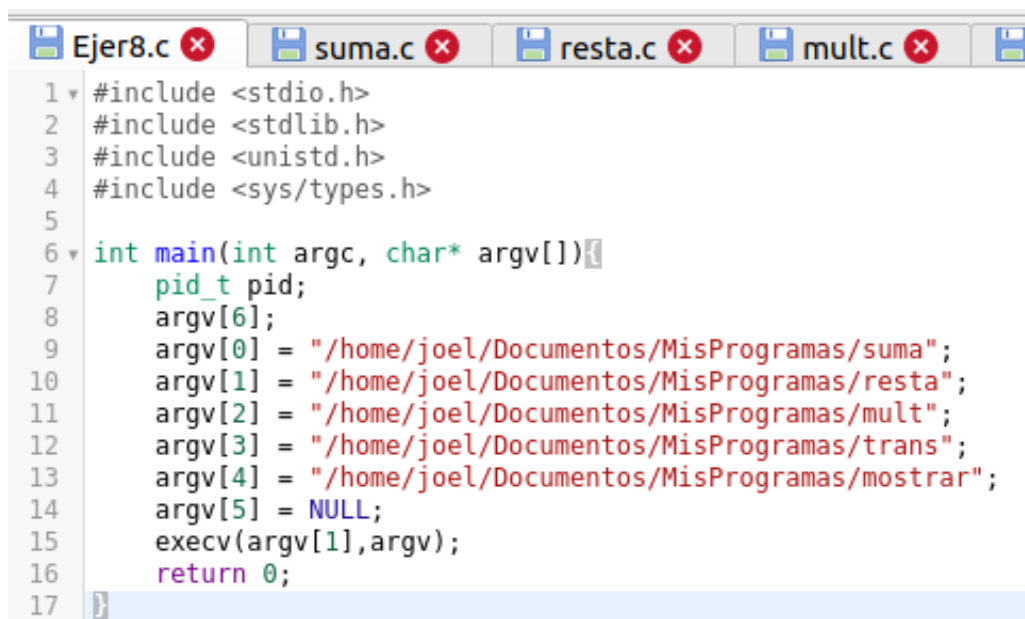
```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer7.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./Multiplicacion
Estoy en Mult.c
Multiplicacion
7 8 6 24 15 6 21
10 4 10 36 6 2 16
7 8 6 24 15 6 21
35 6 10 24 9 4 4
3 12 4 15 4 35 12
3 12 4 15 4 35 12
14 24 25 24 6 2 4

joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

¿Es posible un funcionamiento 100% concurrente de su aplicación?

Después de programar y analizar el programa, consideramos que no es posible el funcionamiento 100% concurrente, ya que si bien es más rápido en cuanto a tiempo de ejecución, es parte de conocimientos más avanzados y necesita más recursos

8. Se programó la misma aplicación del punto 5, solo que en esta ocasión mediante sustitución de código.



```
Ejer8.c suma.c resta.c mult.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5
6 int main(int argc, char* argv[]) {
7     pid_t pid;
8     argv[6];
9     argv[0] = "/home/joel/Documentos/MisProgramas/suma";
10    argv[1] = "/home/joel/Documentos/MisProgramas/resta";
11    argv[2] = "/home/joel/Documentos/MisProgramas/mult";
12    argv[3] = "/home/joel/Documentos/MisProgramas/trans";
13    argv[4] = "/home/joel/Documentos/MisProgramas/mostrar";
14    argv[5] = NULL;
15    execv(argv[1], argv);
16    return 0;
17 }
```



```

Ejer8.c x suma.c x resta.c x mult.c x trans.c x Ejer5-Sec.c x mostrar.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3 void suma(int m1[7][7],int m2[7][7],int mr[7][7]);
4 int main(){
5     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
6     int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
7     int mr[7][7];
8
9     suma(m1,m2,mr);
10
11 }
12 void suma(int m1[7][7],int m2[7][7],int mr[7][7]){
13     int i,j;
14     printf("Suma\n");
15     for(i=0;i<7;i++){
16         for(j=0;j<7;j++){
17             mr[i][j] = m1[i][j] + m2[i][j];
18             printf("%d ",mr[i][j]);
19         }
20         printf("\n");
21     }
22     printf("\n");
23 }

```

```

Ejer8.c x suma.c x resta.c x mult.c x trans.c x Ejer5-Sec.c x mostrar.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3 void resta(int m1[7][7],int m2[7][7],int mr[7][7]);
4 int main(){
5     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
6     int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
7     int mr[7][7];
8
9     resta(m1,m2,mr);
10
11 }
12 void resta(int m1[7][7],int m2[7][7],int mr[7][7]){
13     int i,j;
14     printf("Resta\n");
15     for(i=0;i<7;i++){
16         for(j=0;j<7;j++){
17             mr[i][j] = m1[i][j] - m2[i][j];
18             printf("%d ",mr[i][j]);
19         }
20         printf("\n");
21     }
22     printf("\n");
23 }

```

```

Ejer8.c x suma.c x resta.c x mult.c x trans.c x Ejer5-Sec.c x mostrar.c x
Ejer8.c x suma.c x resta.c x mult.c x trans.c x Ejer5-Sec.c x mostrar.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3 void transpuesta(int m1[7][7],int m2[7][7],int mr[7][7]);
4 int main(){
5     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
6     int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
7     int mr[7][7];
8
9     transpuesta(m1,m2,mr);
10
11 }
12 void transpuesta(int m1[7][7], int m2[7][7], int mr[7][7]){
13     int i,j;
14     printf("Matrices Transpuestas\n");
15     int mr2[7][7];
16     for(i=0;i<7;i++){
17         for(j=0;j<7;j++){
18             mr[i][j] = m1[j][i];
19             mr2[i][j] = m2[j][i];
20             printf("%d ",mr[i][j]);
21             printf("%d ",mr2[i][j]);
22         }
23         printf("\n");
24     }
25     printf("\n");
26 }

```

```
Ejer8.c suma.c resta.c mult.c trans.c mostrar.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void suma(int m1[7][7],int m2[7][7],int mr[7][7]);
5 void resta(int m1[7][7],int m2[7][7],int mr[7][7]);
6 void mult(int m1[7][7],int m2[7][7],int mr[7][7]);
7 void transpuesta(int m1[7][7],int m2[7][7],int mr[7][7]);
8 void mostrar(int m1[7][7],int m2[7][7],int mr[7][7]);
9 int i,j;
10 int main(){
11     int m1[7][7] = {{1,2,3,4,5,6,7},{2,4,5,6,2,1,4},{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,3,2,5,4,7,6},{3,4,2,3,1,5,2},{7,6,5,4,3,2,1}};
12     int m2[7][7] = {{7,4,2,6,3,1,3},{5,1,2,6,3,2,4},{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,4,2,3,1,5,2},{1,3,2,5,4,7,6},{2,4,5,6,2,1,4}};
13     int mr[7][7];
14
15     mostrar(m1,m2,mr);
16
17
18     return 0;
19 }
20
21 void suma(int m1[7][7],int m2[7][7],int mr[7][7]){
22     printf("Suma\n");
23     for(i=0;i<7;i++){
24         for(j=0;j<7;j++){
25             mr[i][j] = m1[i][j] + m2[i][j];
26             printf("%d ",mr[i][j]);
27         }
28         printf("\n");
29     }
30     printf("\n");
31 }
32 void resta(int m1[7][7],int m2[7][7],int mr[7][7]){
33     printf("Resta\n");
34 }
```

Lin. 16, col. 20 Sel. 0 (1) 1743 caracteres, 73 líneas

C Formato UNI

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer8.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./suma
Suma
8 6 5 10 8 7 10
7 5 7 12 5 3 8
8 6 5 10 8 7 10
12 7 7 10 6 4 5
4 7 4 8 5 12 8
4 7 4 8 5 12 8
9 10 10 10 5 3 5

joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer8.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./resta
Resta
-6 -2 1 -2 2 5 4
-3 3 0 -1 -1 0
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
-2 -1 0 2 3 2 4
2 1 0 -2 -3 -2 -4
5 2 0 -2 1 1 -3

joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer8.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./mult
Multiplicacion
7 8 6 24 15 6 21
10 4 10 36 6 2 16
7 8 6 24 15 6 21
35 6 10 24 9 4 4
3 12 4 15 4 35 12
3 12 4 15 4 35 12
14 24 25 24 6 2 4

joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer8.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./trans
Matrices Transpuestas
1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
4 6 6 6 6 4 6 4 5 3 3 5 4 6
5 3 2 3 3 5 3 3 4 1 1 4 3 2
6 1 1 2 1 6 2 2 7 5 5 7 2 1
7 3 4 4 3 7 4 1 6 2 2 6 1 4

joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

```
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Ejer8.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./mostrar
Suma
8 6 5 10 8 7 10
7 5 7 12 5 3 8
8 6 5 10 8 7 10
12 7 7 10 6 4 5
4 7 4 8 5 12 8
4 7 4 8 5 12 8
9 10 10 10 5 3 5

Resta
-6 -2 1 -2 2 5 4
-3 3 0 -1 -1 0
6 2 -1 2 -2 -5 -4
-2 -5 -3 2 0 0 3
-2 -1 0 2 3 2 4
2 1 0 -2 -3 -2 -4
5 2 0 -2 1 1 -3

Multiplicacion
7 8 6 24 15 6 21
10 4 10 36 6 2 16
7 8 6 24 15 6 21
35 6 10 24 9 4 4
3 12 4 15 4 35 12
3 12 4 15 4 35 12
14 24 25 24 6 2 4

Matrices Transpuestas
1 7 2 5 7 1 5 7 1 3 3 1 7 2
2 4 4 1 4 2 1 6 3 4 4 3 6 4
3 2 5 2 2 3 2 5 2 2 2 2 5 5
4 6 6 6 6 4 6 4 5 3 3 5 4 6
5 3 2 3 3 5 3 3 4 1 1 4 3 2
6 1 1 2 1 6 2 2 7 5 5 7 2 1
7 3 4 4 3 7 4 1 6 2 2 6 1 4

joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

Pudimos observar que fue más sencillo, aplicar la creación de procesos por sustitución de código, sin embargo, esto implica el crear otros archivos para su invocación.

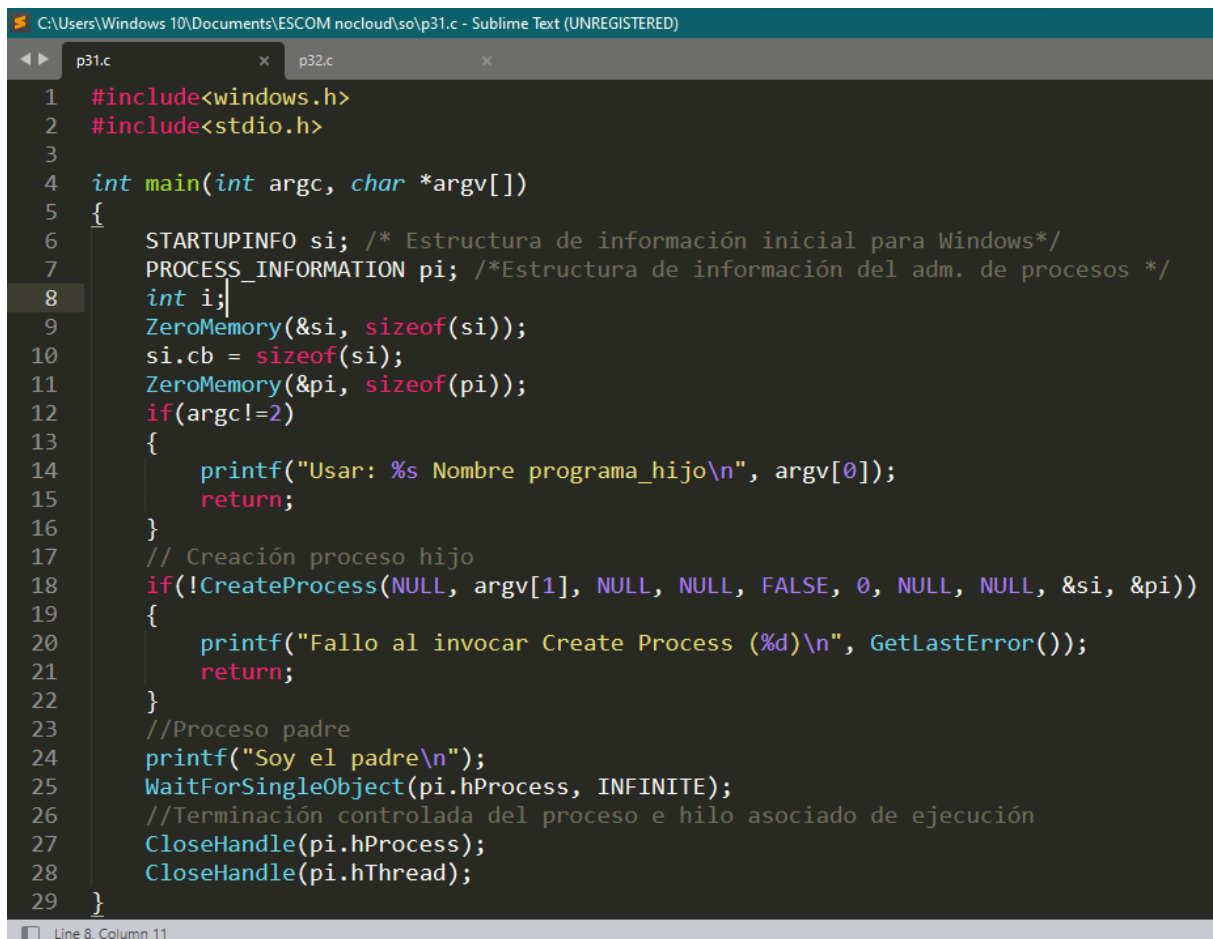
Sección Windows

1. Inicie sesión en Windows

```
C:\Users\Windows 10>ver  
  
Microsoft Windows [Versión 10.0.22000.194]
```

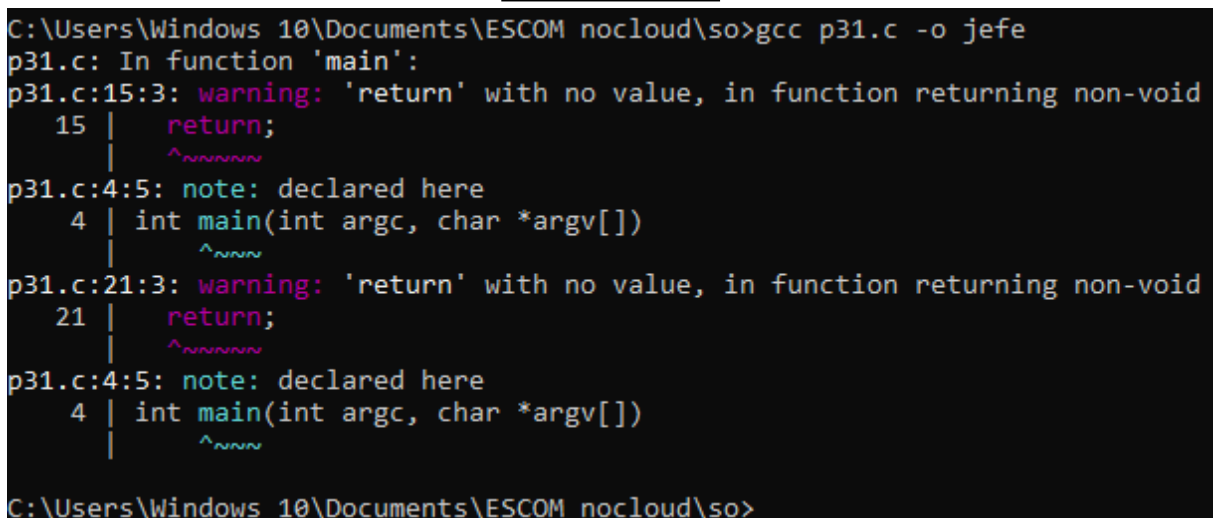
2. Debido a que no se cuenta con el IDE Dev C/C++, se utilizó Sublime Text 3 como procesador de texto, y se compiló desde la terminal.
3. Capture y compile el programa de creación de un nuevo proceso

CÓDIGO



```
C:\Users\Windows 10\Documents\ESCOM nocloud\so\p31.c - Sublime Text (UNREGISTERED)  
p31.c  
1  #include<windows.h>  
2  #include<stdio.h>  
3  
4  int main(int argc, char *argv[])  
5  {  
6      STARTUPINFO si; /* Estructura de información inicial para Windows*/  
7      PROCESS_INFORMATION pi; /*Estructura de información del adm. de procesos */  
8      int i;  
9      ZeroMemory(&si, sizeof(si));  
10     si.cb = sizeof(si);  
11     ZeroMemory(&pi, sizeof(pi));  
12     if(argc!=2)  
13     {  
14         printf("Usar: %s Nombre programa_hijo\n", argv[0]);  
15         return;  
16     }  
17     // Creación proceso hijo  
18     if(!CreateProcess(NULL, argv[1], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))  
19     {  
20         printf("Fallo al invocar Create Process (%d)\n", GetLastError());  
21         return;  
22     }  
23     //Proceso padre  
24     printf("Soy el padre\n");  
25     WaitForSingleObject(pi.hProcess, INFINITE);  
26     //Terminación controlada del proceso e hilo asociado de ejecución  
27     CloseHandle(pi.hProcess);  
28     CloseHandle(pi.hThread);  
29 }
```

COMPILACIÓN



```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>gcc p31.c -o jefe  
p31.c: In function 'main':  
p31.c:15:3: warning: 'return' with no value, in function returning non-void  
15 |     return;  
    |     ^~~~~~  
p31.c:4:5: note: declared here  
4  | int main(int argc, char *argv[])  
    |     ^~~~~~  
p31.c:21:3: warning: 'return' with no value, in function returning non-void  
21 |     return;  
    |     ^~~~~~  
p31.c:4:5: note: declared here  
4  | int main(int argc, char *argv[])  
    |     ^~~~~~  
C:\Users\Windows 10\Documents\ESCOM nocloud\so>
```


4. Capture y compile el programa que contendrá el proceso hijo.

CÓDIGO

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so\p32.c - Sublime Text (UNREGISTERED)
p31.c x p32.c x
1  #include<windows.h>
2  #include<stdio.h>
3
4  int main(void)
5  {
6      printf("Soy el hijo\n");
7      exit(0);
8  }
```

COMPILACIÓN

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>gcc p32.c -o mijo
C:\Users\Windows 10\Documents\ESCOM nocloud\so>
```

5. Ejecute el primer código pasando como argumento el nombre del archivo ejecutable del segundo código capturado. Observe el funcionamiento del programa, reporte sus observaciones y experimente con el código.

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>jefe
Usar: jefe Nombre programa_hijo
```

Primero, ejecuté el programa “padre” solo, la ejecución resulta en que se imprime en pantalla como se debe usar el programa, nos indica que hay que escribir en terminal el nombre del programa padre seguido del nombre del programa hijo.

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>jefe mijo
Soy el padre
Soy el hijo
```

Después, ejecuté el programa “padre” pasando como argumento el nombre del archivo ejecutable del programa “hijo”. Al hacerlo, ya no se cae en los ifs del código del programa padre, por lo que se imprime “Soy el padre” y después “Soy el hijo”. La primera línea pertenece al programa “padre”, y la segunda al “hijo”

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>mijo
Soy el hijo
```

Si ejecutamos solo el programa “hijo”, naturalmente solo se imprimirá “soy el hijo”.

```
C:\Users\Windows 10\Documents\ESCOM nocloud\so>jefe jefe
Soy el padre
Usar: jefe Nombre programa_hijo
```

Si ejecutamos el programa “padre” y pasamos como argumento el mismo programa “padre”, en el primer proceso padre no se cae en los ifs, por lo que se imprime la línea “Soy el padre”, pero al invocar otra vez al “padre” como si fuera un proceso hijo, sin otro argumento para él, se caerá en el primer if del código, y nos indicará la forma correcta de usar el programa.

6. Compare y reporte tanto las diferencias como similitudes que encuentra con respecto a la creación de procesos por sustitución de código en Linux.

Creación de Procesos Linux

Para crear procesos en Linux, hacemos uso de la función fork de POSIX. Es muy sencillo de usar. Lo único que requiere saber es que, los procesos en Linux, y en general, los procesos en Unix, se identifican por un número entero llamado PID. En los sistemas Linux, hay un proceso especial con PID igual a 1. Está reservado al proceso llamado init, que es el proceso de arranque del núcleo. Un proceso puede crear varios procesos hijos (que también tendrán su PID) y la particularidad de estos procesos, es que tienen un proceso padre, del cual los procesos hijos lo conocen por su PPID (PID del padre).

Creación de Procesos Windows

La funcionalidad es similar en los sistemas operativos Windows, pero aquí una llamada al sistema siempre se convierte primero internamente: una función de biblioteca de la API de Windows (abreviatura WinAPI) se convierte automáticamente en una llamada al sistema que el sistema operativo puede leer, incluyendo un número único que hace referencia a la función deseada en el modo núcleo.

Tipo de system call	Función	Linux	Windows
Control del proceso	Crear proceso	fork()	CreateProcess()
Control del proceso	Terminar proceso	exit()	ExitProcess()
Gestión de archivos	Crear/abrir archivo	open()	CreateFile()
Gestión de archivos	Leer archivo	read()	ReadFile()
Gestión de archivos	Editar archivo	write()	WriteFile()
Gestión de archivos	Cerrar archivo	close()	CloseHandle()
Gestión de dispositivos	Abrir dispositivo	read()	ReadConsole()
Gestión de dispositivos	Cerrar dispositivo	close()	CloseConsole()
Gestión de la información	Definición de un intervalo de tiempo específico	alarm()	SetTimer()
Gestión de la información	Pausa (por ejemplo, de un proceso)	sleep()	Sleep()
Comunicación	Crear Pipe (memoria intermedia para el flujo de datos entre dos procesos)	pipe()	CreatePipe()
Comunicación	Creación de una memoria compartida (Shared memory)	shmget()	CreateFileMapping()

7. Programe una aplicación que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará 5 procesos hijos más. A su vez cada uno de los cinco procesos creará 3 procesos más. Cada uno de los procesos creados imprimirá en pantalla su identificador. CONSEJO: INVESTIGUE LA FUNCIÓN `GetCurrentProcessId()` DEL API WIN32.

GetCurrentProcessId():

Recupera el identificador de proceso del proceso de llamada.

Sintaxis:

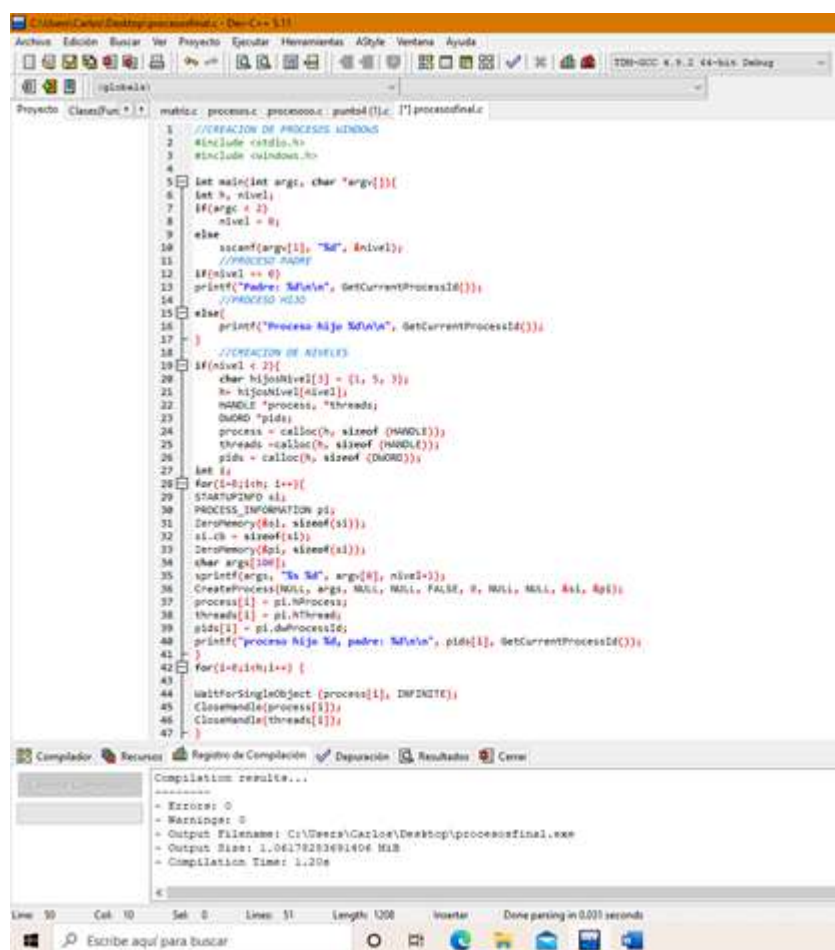
DWORD `GetCurrentProcessId()`;

Valor devuelto

El valor devuelto es el identificador de proceso del proceso de llamada.

Observaciones

Hasta que finalice el proceso, el identificador del proceso identifica de forma única el proceso en todo el sistema.



```
1 //CREACION DE PROCESOS LINDOS
2 #include <stdio.h>
3 #include <windows.h>
4
5 int main(int argc, char *argv[]) {
6     int n, nivel;
7     if (argc < 2)
8         nivel = 0;
9     else
10         sscanf(argv[1], "%d", &nivel);
11     //PROCESO PADRE
12     if (nivel == 0)
13         printf("Padre: %d\n", GetCurrentProcessId());
14     //PROCESO HIJO
15     else {
16         printf("Proceso Hijo %d\n", GetCurrentProcessId());
17     }
18     //CREACION DE HIJOS
19     if (nivel < 2) {
20         char hijosNivel[3] = {1, 5, 3};
21         n = hijosNivel[nivel];
22         HANDLE *process, *threads;
23         DWORD *pids;
24         process = calloc(n, sizeof(HANDLE));
25         threads = calloc(n, sizeof(HANDLE));
26         pids = calloc(n, sizeof(DWORD));
27         int i;
28         for (i = 0; i < n; i++) {
29             STARTUPINFO si;
30             PROCESS_INFORMATION pi;
31             ZeroMemory(&si, sizeof(si));
32             si.cb = sizeof(si);
33             ZeroMemory(&pi, sizeof(pi));
34             char args[100];
35             sprintf(args, "%s %d", argv[0], nivel+1);
36             CreateProcess(NULL, args, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi);
37             process[i] = pi.hProcess;
38             threads[i] = pi.hThread;
39             pids[i] = pi.dwProcessId;
40             printf("Proceso Hijo %d, padre: %d\n", pids[i], GetCurrentProcessId());
41         }
42         for (i = 0; i < n; i++) {
43             WaitForSingleObject(process[i], INFINITE);
44             CloseHandle(process[i]);
45             CloseHandle(threads[i]);
46         }
47     }
48 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Carlos\Desktop\procesosfinal.exe
- Output Size: 1,06178283691406 KiB
- Compilation Time: 1.20s

Line: 50 Col: 10 Sel: 0 Line: 51 Length: 1208 Insert Done parsing in 0.033 seconds

C:\Users\Carlos\Desktop\matriz.c - Dev-C++ 5.11

Archivo Edición Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda

(globals)

Proyecto Clases(Fun) matriz.c procesos.c procesos.o punto4 (1).c [*] procesosfinal.c

```

89 printf("\n\n\t\t\t\t\t SUMA");
90 printf("\n");
91 for(i=0;i<af;i++){
92     printf("\n\t\t\t\t\t");
93     for(j=0;j<bc;j++){
94         printf(" %6d ", SUM[i][j]);
95     }
96 }
97
98 printf("\n\n\t\t\t\t\t RESTA");
99 printf("\n");
100 for(i=0;i<af;i++){
101     printf("\n\t\t\t\t\t");
102     for(j=0;j<bc;j++){
103         printf(" %6d ", RES[i][j]);
104     }
105 }
106
107 printf("\n\n\t\t\t\t\t MULTIPLICACION");
108 printf("\n");
109 for(i=0;i<af;i++){
110     printf("\n\t\t\t\t\t");
111     for(j=0;j<bc;j++){
112         printf(" %6d ", MUL[i][j]);
113     }
114 }
115
116 printf("\n\n\t\t\t\t\t Transpuesta Matriz A");
117 printf("\n");
118 for(i=0;i<ac;i++){
119     printf("\n\t\t\t\t\t");
120     for(j=0;j<af;j++){
121         printf(" %6d ", A[j][i]);
122     }
123 }
124 printf("\n\n\t\t\t\t\t Transpuesta Matriz B");
125 printf("\n");
126 for(i=0;i<bc;i++){
127     printf("\n\t\t\t\t\t");
128     for(j=0;j<bf;j++){
129         printf(" %6d ", B[j][i]);
130     }
131 }
132
133 printf("\n");
134 }

```

Compilador Recursos Registro de Compilación Depuración Resultados Cerrar

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Carlos\Desktop\matriz.exe
- Output Size: 161.34765625 KiB
- Compilation Time: 0.50s

Line: 20 Col: 21 Sel: 0 Lines: 134 Length: 2953 Insertar Done parsing in 0.031 seconds

Escribe aquí para buscar

C:\Users\Carlos\Desktop\matriz.exe

Numero de filas de la matriz A=7
 Numero de columnas de la matriz A=7
 Numero de filas de la matriz B=7
 Numero de columnas de la matriz B=7

Matriz A

6	9	8	5	9	2	4
1	8	3	9	3	8	7
8	6	8	9	4	1	1
7	6	1	5	8	7	5
9	6	3	1	3	1	7
5	9	2	4	3	7	7
3	4	7	1	4	8	3

Matriz B

2	6	6	2	7	4	8
3	4	8	5	5	3	5
7	1	2	5	6	5	2
6	1	6	7	8	6	4
7	4	3	1	6	1	2
1	6	8	6	9	2	7
4	3	2	3	2	9	4

SUMA

8	15	14	7	16	6	12
4	12	11	14	8	11	13
15	7	10	14	10	6	5
13	7	7	12	16	13	19
16	10	6	2	9	2	9
6	15	10	14	13	5	14
7	7	9	8	6	17	7

RESTA

4	3	2	3	2	-2	-4
-2	4	-5	4	-2	5	1
1	5	6	4	-2	-4	-4
1	5	-5	-2	0	1	2
2	1	0	6	-3	0	5
4	3	-6	2	-5	1	0
-1	1	5	0	2	-1	-1

MULTIPLICACION

12	54	48	10	63	8	32
3	32	24	45	15	24	42
56	8	16	45	14	5	5
42	6	6	35	44	42	28
63	24	9	1	18	1	14
5	54	16	48	16	6	49
12	12	14	9	8	72	12

Transpuesta Matriz A

6	1	8	7	9	5	3
9	1	6	6	6	9	4
8	3	8	1	3	2	7
5	9	9	5	1	8	3
9	1	4	8	3	4	4
2	8	1	7	1	3	8
4	7	1	6	7	7	3

Transpuesta Matriz B

2	3	7	6	7	1	4
6	4	1	1	4	6	3
6	8	2	6	3	8	2
2	5	5	7	1	6	3
7	5	6	8	6	9	2
4	3	5	6	1	2	9
8	6	5	4	2	7	4

Compilation results...

III.CONCLUSIÓN

En esta práctica analizamos y comprendimos como el S.O se encuentra estructurado en una primera etapa, el desarrollo de procesos y la compresión de los hilos de ejecución a través de los cuales, el sistema delega funciones y opera en forma multi funcional. También estudiamos la forma en la que el sistema gestiona las interrupciones, este se encarga de controlar los accesos al procesador, verificar el estatus de un proceso y determinar su ejecución de acuerdo al nivel de importancia, cabe destacar que no todas las interrupciones son controladas por el SO, ya que existen interrupciones enmascaradas y que son exclusivas del hardware de cada computadora.

IV.REFERENCIAS

[https://es.abcdef.wiki/wiki/Exec_\(system_call\)](https://es.abcdef.wiki/wiki/Exec_(system_call))

<https://es.stackoverflow.com/questions/179414/como-funciona-la-funci%C3%B3n-fork>

<https://es.puuteri.org/802810-how-does-getpid-work-LPUWGF>

<https://www.youtube.com/watch?v=pRMIcy5q6wU>

<https://docs.microsoft.com/es-es/windows/win32/procthread/creating-processes>

<https://docs.microsoft.com/es-es/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa?redirectedfrom=MSDN>