UADER | FCyT

Sistemas Operativos

Práctica

Lic. Exequiel Aramburu

exequiel.aramburu@uader.edu.ar

Proceso

Programa != Proceso

GNU/Linux como Sistema Operativo Multitarea

- Múltiples procesos pueden ejecutarse “simultáneamente” sin interferir entre sí.

- Cada proceso “cree” que se está ejecutando solo en el sistema y que tiene acceso a todos los recursos del mismo.

- Todos los procesos tienen siempre un determinado *estado* en el tiempo.

- El sistema operativo mantiene información de todos sus procesos y de su *estado.*

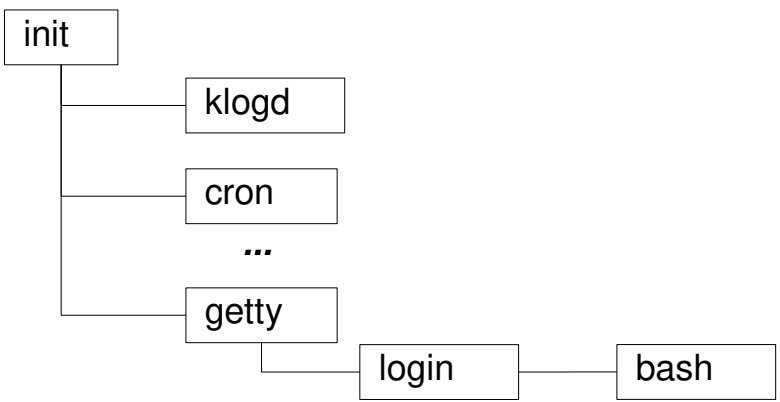
- Para cada proceso deben reservarse recursos y asignarse prioridades.

Procesos en GNU/Linux

- Init es el primer proceso a ejecutarse por el núcleo. Corre con un identificador de proceso (PID) igual a 1.

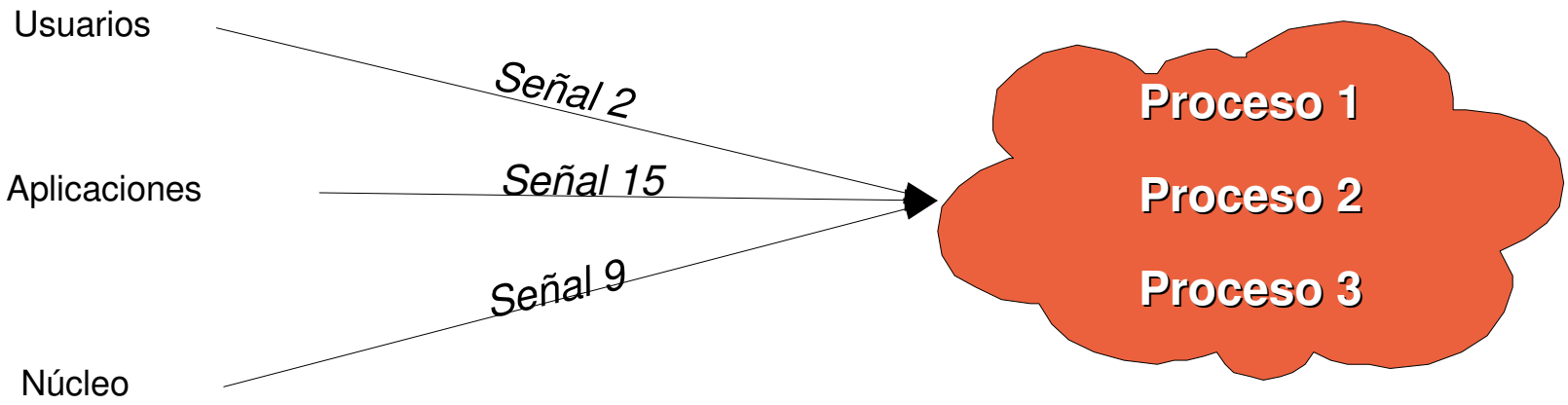
- Debe encontrarse siempre en ejecución ya que es el “Padre” de todos los demás procesos.

- Este proceso realiza varias acciones como vimos anteriormente y finalmente ejecuta varias instancias (forks) del proceso *getty*.



Gestión de Procesos

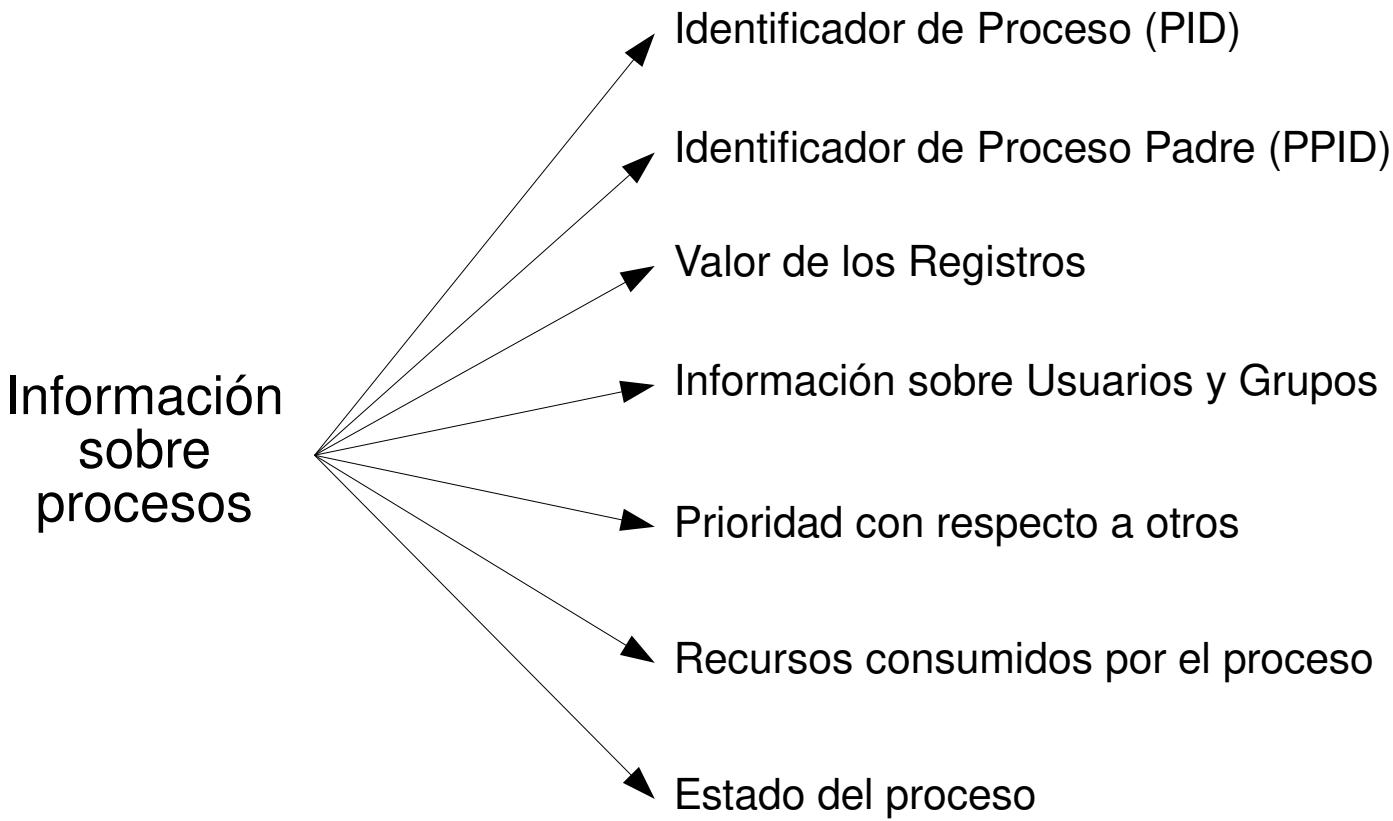
Constantemente el núcleo y las aplicaciones de usuarios envían *señales* a los procesos para modificar su *estado*.



Para enviar señales utilizaremos el comando *kill*

Gestión de Procesos

Gestión de Procesos



Herramientas para trabajar con procesos

Para visualizar procesos

- ps
- pstree
- top
- htop

Para enviar señales a procesos

- kill
- killall
- nice
- renice

Herramientas para la administración de procesos

Examinando la salida del comando ps

- Usuario con el que se ejecuta el proceso.
- Identificador de Proceso (PID)
- % de uso de CPU y de Memoria
- Terminal en la que se ejecuta el comando.
- Fecha de Inicio
- Comando que se ejecuta

```
exequiel@Exequiel-PC:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 169824 11812 ?        Ss   abr20   0:20 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    abr20   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   abr20   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   abr20   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   abr20   0:00 [kworker/0:0H-events_highpri]
root           9  0.0  0.0      0     0 ?        I<   abr20   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    abr20   0:00 [rcu_tasks_rude_]
root          11  0.0  0.0      0     0 ?        S    abr20   0:00 [rcu_tasks_trace]
root          12  0.0  0.0      0     0 ?        S    abr20   0:01 [ksoftirqd/0]
root          13  0.1  0.0      0     0 ?        I    abr20   4:14 [rcu_sched]
root          14  0.0  0.0      0     0 ?        S    abr20   0:00 [migration/0]
root          15  0.0  0.0      0     0 ?        S    abr20   0:00 [idle_inject/0]
root          16  0.0  0.0      0     0 ?        S    abr20   0:00 [cpuhp/0]
root          17  0.0  0.0      0     0 ?        S    abr20   0:00 [cpuhp/1]
root          18  0.0  0.0      0     0 ?        S    abr20   0:00 [idle_inject/1]
root          19  0.0  0.0      0     0 ?        S    abr20   0:00 [migration/1]
root          20  0.0  0.0      0     0 ?        S    abr20   0:00 [ksoftirqd/1]
root          22  0.0  0.0      0     0 ?        I<   abr20   0:00 [kworker/1:0H-events_highpri]
root          23  0.0  0.0      0     0 ?        S    abr20   0:00 [cpuhp/2]
root          24  0.0  0.0      0     0 ?        S    abr20   0:00 [idle_inject/2]
root          25  0.0  0.0      0     0 ?        S    abr20   0:00 [migration/2]
root          26  0.0  0.0      0     0 ?        S    abr20   0:00 [ksoftirqd/2]
root          28  0.0  0.0      0     0 ?        I<   abr20   0:00 [kworker/2:0H-events_highpri]
root          29  0.0  0.0      0     0 ?        S    abr20   0:00 [cpuhp/3]
root          30  0.0  0.0      0     0 ?        S    abr20   0:00 [idle_inject/3]
root          31  0.0  0.0      0     0 ?        S    abr20   0:00 [migration/3]
root          32  0.0  0.0      0     0 ?        S    abr20   0:00 [ksoftirqd/3]
root          34  0.0  0.0      0     0 ?        I<   abr20   0:00 [kworker/3:0H-kblockd]
```

Herramientas para la administración de procesos

top – muestra la lista de procesos que consumen más recursos en el sistema

Sintaxis:

top <opciones>

Opciones:
-s:deshabilita el modo interactivo.
-c: muestra el path absoluto con el que fue llamado cada comando de la lista.
-d <segundos>.permite especificar los segundos de espera entre muestreos.

Ejemplos:

$top -c
$top -d 1

Herramientas para la administración de procesos

htop – muestra la lista de procesos y nos permite gestionarlos.

Sintaxis:

htop

Comandos en modo interactivo:

F1 – Muestra la ayuda
F6 – Permite ordenar los procesos por consumo de memoria, consumo de cpu, usuario o PID.
F10 – Sale de htop.

Ejemplos:

htop

Herramientas para la administración de procesos

kill – permite enviar señales a un proceso

Sintaxis:

kill <opciones> <señal> PID

Opciones:

-l: muestra una tabla con las señales disponibles y sus respectivos números.
-s señal: permite especificar una señal por su nombre para enviar a un proceso.
-n señal: permite especificar una señal por su número para enviar a un proceso.

Ejemplos:

kill -9 9873
kill -SIGKILL 3482

Herramientas para la administración de procesos

killall – permite enviar señales a un proceso usando su nombre

Sintaxis:

killall <opciones> <señal> nombre

Opciones:

-l: muestra una tabla con las señales disponibles y sus respectivos números.
-s señal: permite especificar una señal por su nombre para enviar a un proceso.
-n señal: permite especificar una señal por su número para enviar a un proceso.

Ejemplos:

killall -9 kde
killall -SIGKILL amsn

Herramientas para la administración de procesos

nice – cambia la prioridad de un proceso a la hora de ejecutarlo.

Sintaxis:

nice <opciones> <proceso> argumento del proceso

Opciones:

-n: permite especificar un número entero para definir la prioridad del proceso.
--version: permite mostrar la versión actual del comando.
--help: muestra la ayuda del comando

Ejemplos:

 # nice -n -20 apt-get update

Herramientas para la administración de procesos

renice – cambia la prioridad de un proceso en ejecución

Sintaxis:

renice prioridad <-p pid> <-g grupo> <-u usuario>

Opciones:

-p pid: especifica el PID del proceso al cual queremos modificar la prioridad
-g grupo: permite cambiar el ID del grupo asociado al proceso.
-u usuario: permite cambiar el ID del usuario asociado al proceso.

Ejemplos:

$renice +1 -p 897
$renice +2 2308 -u usuario

Herramientas para la administración de procesos

free – muestra la utilización de memoria del sistema

Sintaxis:

free <opciones>
Opciones:

-m: muestra la memoria en megabytes.
-t: muestra totales de memoria en las columnas de total, usada y libre.
-s intervalo: muestra el uso de la memoria y se actualiza pasado un intervalo de tiempo.
Ejemplos:

$ free -tm
$ free -s 2

Herramientas para la administración de procesos

uptime – indica el tiempo que lleva el sistema encendido.

Sintaxis:

uptime <opciones>
Opciones:

-V: muestra la versión del programa

Ejemplos:

$ uptime
$ uptime -V

Herramientas para la administración de procesos

watch – permite ejecutar un comando periódicamente

Sintaxis:

watch <opciones> comando

Opciones:

-n: permite especificar un intervalo de tiempo para ejecutar un comando.
-t: permite no mostrar en la salida el comando que se utiliza
--help: muestra la ayuda del comando

Ejemplos:

$ watch -n 5 date
$ watch -t date

Herramientas para la administración de procesos

pgrep – busca los procesos que tienen atributos determinados.

Sintaxis:

pgrep <opciones> <patrón>
Opciones:

-G: permite especificar el nombre de un grupo para poder buscar procesos.
-I: muestra la salida del comando en formato largo agregando el nombre del proceso.
-U intervalo: permite especificar el nombre de un usuario para poder buscar procesos.
Ejemplos:

$ pgrep -G usuario
$ pgrep -L apache2

Herramientas para la administración de procesos

pstree – muestra los procesos en forma de árbol.

Sintaxis:

pstree <opciones>
Opciones:

-u: permite mostrar el nombre del usuario propietario del proceso.
-p: permite mostrar el PID del proceso.

Ejemplos:

$ pstree
$ pstree -U

Herramientas para la administración de procesos

fg, bg – permiten manipular y enviar procesos a primer y a segundo plano

Sintaxis:

fg, bg %número de trabajo

Ejemplos:

$ bg -
$ fg %1

Herramientas para la administración de procesos

nohup – ejecuta un comando y no permite que reciba más señales de hup, term y kill del sistema.

& - le indica a la shell que un proceso se ejecute en segundo plano
Sintaxis:

nohup comando <argumentos>

Opciones:

--version: muestra la versión del comando nohup.
--help: muestra la ayuda del comando

Ejemplos:

$ nohup sleep 60 &
$ yes > /dev/null &

Herramientas para la administración de procesos

jobs – permite ver los procesos detenidos o en segundo plano en una shell

Sintaxis:

jobs <opciones>

Opciones:

-l: permite ver el PID de las tareas
-p: muestra únicamente el PID de las tareas

Ejemplos:

$ jobs
$ jobs -l