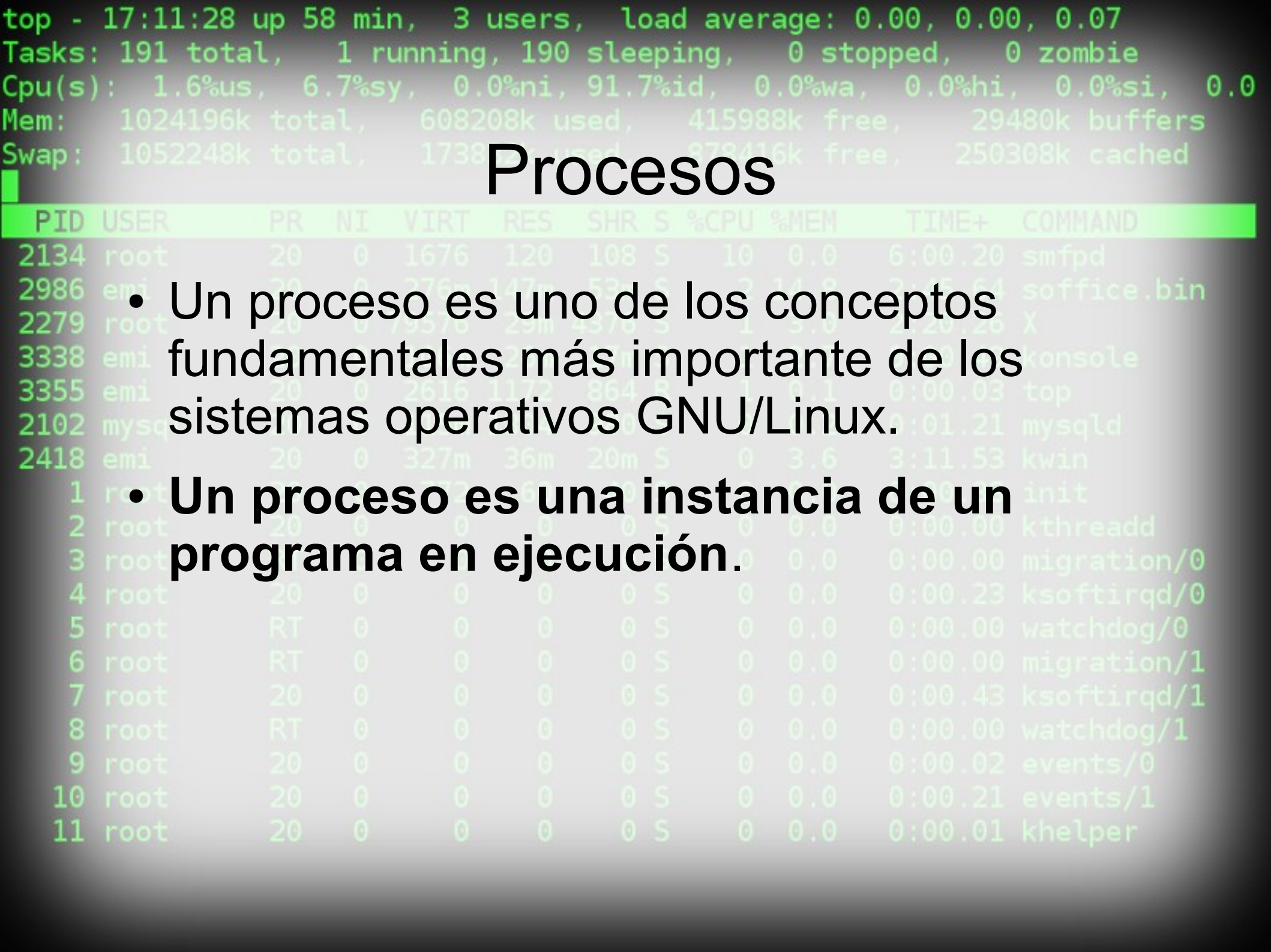
Gestión de Procesos en GNU/Linux

2013

Ing. Emiliano Marini
www.linuxito.com

Objetivo

- Analizar **técnicas** y **herramientas** para gestionar **procesos** en GNU/Linux:
  - Listar procesos en ejecución
  - Obtener información de procesos
  - Iniciar y detener procesos
  - Determinar y alterar la prioridad de ejecución de procesos
  - Gestionar procesos en segundo plano



Procesos

- Un proceso es uno de los conceptos fundamentales más importante de los sistemas operativos GNU/Linux.
- **Un proceso es una instancia de un programa en ejecución**.

Crear procesos

- ¿Cómo crear un proceso?
  - Iniciar una sesión:

```
man login
```

  - Ejecutar un programa:

```
mkdir /tmp/prueba
cd /tmp/prueba
touch doc.txt
nano doc.txt
```

Identificadores de un proceso

- Process ID (**PID**): Identificador único.
- User ID (**UID**) y Group ID (**GID**): Usuario y grupo al que el proceso pertenece:
  - UID y GID **reales** (heredados del padre)
  - UID y GID **efectivos** (los procesos con UID efectivo igual a 0 son privilegiados ya que se ejecutan como superusuario)
- Parent Process ID (**PPID**): PID del proceso padre.

Listar procesos

- El comando **ps** muestra un snapshot de los procesos actuales:

```
man ps
ps
ps u
ps -l
ps aux
ps -U root -u root u
```

ps

- Utilizando **ps** se puede obtener información relacionada a los procesos, por ejemplo:
  - Identificadores (USER, PID, UID, GID, PPID)
  - Hora de inicio (START)
  - % de uso de memoria (%MEM) y CPU (%CPU)
  - Tiempo de CPU acumulado (TIME)
  - Estado (STAT)
  - Terminal asociada (TTY)
  - Comando (CMD/COMMAND)

Estado de procesos

D	Uninterruptible sleep (usually IO)
R	Running or runnable (on run queue)
S	Interruptible sleep (waiting for an event to complete)
T	Stopped, either by a job control signal or because it is being traced
X	Dead (should never be seen)
Z	Defunct ("zombie") process, terminated but not reaped by its parent

Estado de procesos
Información adicional

<	High-priority (not nice to other users)
N	Low-priority (nice to other users)
L	Has pages locked into memory (for real-time and custom IO)
s	Is a session leader
l	Is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+	Is in the foreground process group

El proceso init

- En Linux cada proceso tiene un proceso padre.
- “init” es el primer proceso que crea el kernel Linux cuando inicia el sistema (boot)
- Todos los procesos son hijos de **init** (de forma directa o indirecta).
- El proceso **init** no puede ser matado (kill), excepto cuando se apaga el sistema.
- El proceso **init** siempre tiene el PID = 1.

Árbol de procesos

- Es posible visualizar la jerarquía de procesos en forma de árbol a partir de init (PID = 1):

```
pstree
ps jf
ps axjf
ps -ejH
```

- Si se especifica un PID, el árbol se inicia desde tal proceso. Si se especifica un usuario válido se mostrará la jerarquía de todos los procesos del mismo.

Demonios (daemons)

- En un momento determinado pueden existir en el sistema procesos invocados por el usuario actual, invocados por otros usuarios, o invocados por el sistema operativo (daemons).
- Los demonios son procesos que necesitan ejecutarse en segundo plano (background) por largos períodos de tiempo y no requieren una terminal que los controle (TTY).

```
man 4 tty
```

- No interactúan directamente con los usuarios, sino en forma de servicio.

Señales

- Mensajes que se envían a un proceso para notificar eventos importantes.
- Por su naturaleza, los procesos son interrumpidos y forzados a manejarlas inmediatamente.
- Cada señal se identifica con un número entero al igual que un nombre simbólico:

```
kill -l
man 7 signal
```

Terminar procesos

- El comando kill se utiliza para enviar señales a procesos.
- Por defecto envía la señal TERM (Termination Signal).
- ¿Cómo matar un proceso?
  - Enviar SIGTERM:

    ```
    kill -15 [pid]
    ```

  - Si no responde, enviar SIGKILL:

    ```
    kill -9 [pid]
    ```

[off-topic] Everything is a file

- **"Everything is a file"** describes one of the defining features of Unix, and its derivatives, that a wide range of **input/output resources** such as documents, directories, hard-drives, modems, keyboards, printers and even some inter-process and network communications **are simple streams of bytes exposed through the filesystem name space**.
- The same set of tools, utilities and APIs can be used on a wide range of resources. When a file is opened a file descriptor is created. The **file path becoming the addressing system** and the **file descriptor being the byte stream I/O interface**.

[off-topic] Everything is a file

- A range of **pseudo and virtual filesystems** exists which exposes information about processes and other system information in a hierarchical file-like structure.
- An example of this purely **virtual filesystem** is under **/proc** that exposes many system properties as files.
- All of these "files" have **standard Unix file attributes** such as an owner and access permissions, and can be queried by the same classic Unix tools and filters.

“Destripar” procesos

- El pseudo sistema de archivos **proc** se utiliza como interfase a las estructuras de datos del **kernel**.
- Generalmente es montado en /proc.
- En su mayor parte es de sólo lectura, pero algunos archivos permiten modificar variables del kernel.
- Existe un subdirectorio numérico por cada proceso, cuyo nombre coincide con el **PID** de los mismos.

/proc

```
man proc

ls -l /proc

ls -l /proc/[pid]

(cat /proc/[pid]/cmdline; echo) | tr '\0' ' '

ls -l /proc/[pid]/cwd
cd /proc/[pid]/cwd; pwd

(cat /proc/[pid]/environ; echo) | tr '\0' '\n'

ls -l /proc/[pid]/exe

ls -l /proc/[pid]/fd
```

Experimento: manipular la entrada estándar de otro proceso

- Ejecutar el editor de texto “vi”:

```
touch /tmp/prueba/doc2.txt
vi /tmp/prueba/doc2.txt
[INSERT]
```

- Desde otra terminal inyectar texto:

```
ps u
echo hola >> /proc/[pid]/fd/0
```

Más /proc

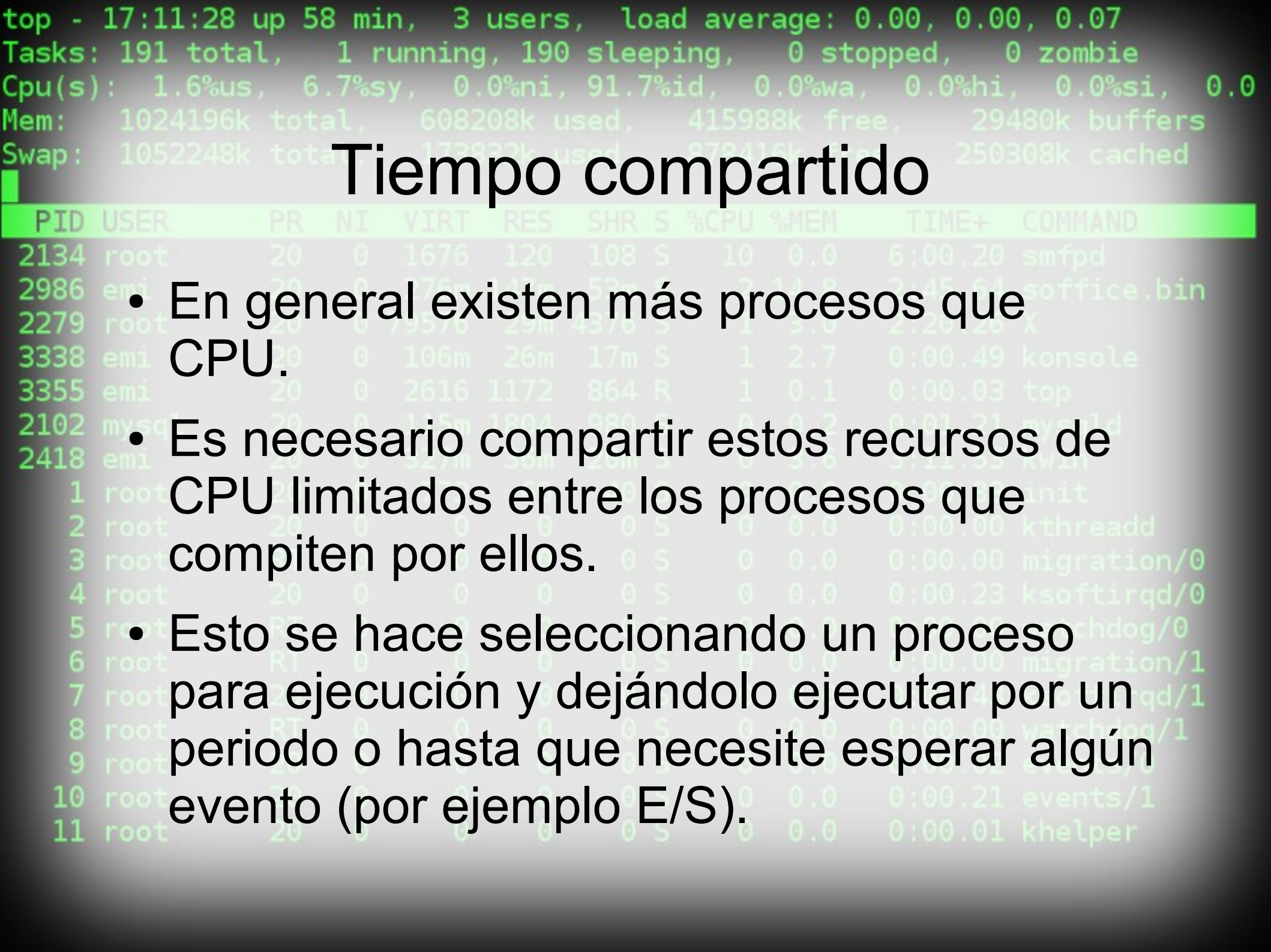
```
cat /proc/[pid]/limits
cat /proc/[pid]/maps
cat /proc/[pid]/mountinfo
ls -l /proc/[pid]/root
cat /proc/[pid]/smaps
cat /proc/[pid]/stat
cat /proc/[pid]/statm
cat /proc/[pid]/status
cat /proc/[pid]/task
```

¿Qué pasa con el directorio /proc/[pid] cuando matamos el proceso?

Multiprogramación

- Linux (como la mayoría de sistemas operativos modernos) puede ejecutar múltiples procesos compartiendo CPU, memoria y otros recursos entre ellos.

```
top
htop
```



Tiempo compartido

- En general existen más procesos que CPU.
- Es necesario compartir estos recursos de CPU limitados entre los procesos que compiten por ellos.
- Esto se hace seleccionando un proceso para ejecución y dejándolo ejecutar por un periodo o hasta que necesite esperar algún evento (por ejemplo E/S).

Prioridad de ejecución

- Para garantizar que procesos importantes no se queden sin CPU, la selección se efectúa de acuerdo a una prioridad.
- La columna NI en la salida del comando `top` indica la prioridad de ejecución o **nice**.
- Esta prioridad oscila desde **-20** (mayor prioridad) hasta **19** (menor prioridad).

```
man nice
ps o pid,ni,comm,args
ps -l
```

Alterar la prioridad de ejecución

- Iniciar un proceso con baja prioridad:

```
nice -n 19 grep -Ri net /
```

- Modificar la prioridad de un proceso:

```
grep -Ri net /
ps o pid,ni,comm,args
renice -n 19 -p [pid]
ps o pid,ni,comm,args
```

Procesos en segundo plano

- En Linux podemos iniciar procesos en primer plano (foreground) o en segundo plano (background).
- Un proceso iniciado en foreground monopoliza la terminal e impide iniciar más procesos desde la misma.
- Un proceso en background, una vez iniciado deja de monopolizar la terminal, y devuelve el control al usuario (prompt).

Iniciar procesos en background

- Es posible iniciar procesos en background utilizando el caracter ampersand:

```
&
```

- Es posible detener procesos utilizando:

```
Ctrl+Z
```

- A cada proceso en background se le asigna un identificador numérico.

```
man jobs
man bg
man fg
```

Alternar foreground entre procesos

```
top [Ctrl+Z]
ping 8.8.8.8 > /dev/null &
jobs
tail -f /var/log/dmesg > /dev/null &
jobs
ping 8.8.4.4 > /dev/zero [Ctrl+Z]
jobs
bg 4
jobs
fg 3 [Ctrl+Z]
jobs -l
fg
ps
```

Más herramientas...

- Listar procesos: `pgrep`, `atop`
- Enviar señales: `killall`, `pkill`, `skill`
- Esperar procesos: `wait`
- Medir procesos: `time`
- “Inmortalizar” procesos: `nohup`, `disown`

Links de interés

- Curso de Introducción a GNU/Linux
  http://www.ant.org.ar/cursos/curso_intro/book1.html
- explainshell.com http://explainshell.com/
- The Linux Documentation Project - “create and distribute a canonical set of high quality free GNU/Linux documentation”:
  - Introduction to Linux http://tldp.org/LDP/intro-linux/html/intro-linux.html
  - Bash Guide for Beginners
    http://tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html
  - Advanced Bash-Scripting Guide
    http://tldp.org/LDP/abs/html/abs-guide.html
  - GNU/Linux Command-Line Tools Summary
    http://tldp.org/LDP/GNU-Linux-Tools-Summary/html/GNU-Linux-Tools-Summary.html
  - Y más... http://tldp.org/guides.html

Referencias

- **An overview of Linux processes** - [https://www.ibm.com/developerworks/community/blogs/58e72888-6340-46ac-b488-d31aa4058e9c/entry/an_overview_of_linux_processes21]
- **Parent process** - [http://en.wikipedia.org/wiki/Parent_process]
- **Linux Filesystem Hierarchy - 1.14. /proc** - [http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html]
- **Everything is a file** - [http://en.wikipedia.org/wiki/Everything_is_a_file]
- **Prioridades de ejecución de procesos** - [http://www.ibm.com/developerworks/ssa/linux/library/l-lpic1-v3-103-6/]
- **Introduction To Unix Signals Programming** - [http://titania.ctie.monash.edu.au/signals/]