Tema

2.1

Sistema Operativos (SSOO)

Definición y Gestión de procesos

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Índice

- Definición de proceso
- Gestión de procesos
- Estados de los procesos
- Gestión de procesos en Linux

Definición de proceso

Un proceso puede informalmente entenderse como **un programa en ejecución**.

- Formalmente un proceso es **"Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados"**.

Definición de proceso

La gestión de procesos es una de las tareas fundamentales de cualquier sistema operativo

El sistema operativo debe:

- Asignar recursos a los procesos
- Permitir el intercambio de información entre los mismos
- Proteger los recursos de un proceso del resto
- Y facilitar la sincronización de los procesos

Definición de proceso

¿Es lo mismo un programa que un proceso?

Definición de proceso

Para que el SO pueda llevar estas tareas, este mantiene una estructura de datos para cada proceso que describe:

- Su estado
- Los recursos que posee

Esto permite al sistema operativo imponer un control sobre los procesos (es decir, gestionar los procesos)

Gestión de procesos

Principal problema:

El cuello de botella que se forma por la diferencia de velocidad entre el procesador y los dispositivos de entrada y salida (E/S).

Si un proceso que está usando el procesador necesita realizar una operación de E/S (por ejemplo, escribir en disco duro), el SO debe dar paso a otro proceso para que el procesador no esté inactivo.

El SO sería algo parecido a un policía de tráfico.

Gestión de procesos

En resumen, el SO debe gestionar el uso que cada proceso hace de los recursos hardware del sistema (en este tema, el procesador y el disco duro), repartiendo los tiempos que necesitan cada uno de los procesos.

Estados de los procesos

Un proceso puede estar en los siguientes estados principales:

- **Ejecución:** es un proceso que está haciendo uso del procesador.
- **Bloqueado:** no puede ejecutarse hasta que un evento externo sea llevado a cabo.
- **Listo:** ha dejado disponible al procesador para que otro proceso pueda ocuparlo.
- **Nuevo**: proceso que se acaba de crear, pero que aún no ha sido admitido en la lista de procesos ejecutables por el sistema operativo (no ha sido aún cargado en la memoria principal).
- **Terminado**: el proceso ha sido excluido por el sistema operativo, bien porque se detuvo o bien porque fue abandonado por alguna otra razón

Gestión de procesos en Linux

¿Cómo se gestionan los procesos en Linux?

La orden más importante es la orden **ps**.

- Lista todos los procesos en ejecución del sistema
- Y toda su información asociada

```
mcardenas@mcardenas:~$ ps
  PID TTY          TIME CMD
 1279 tty1     00:00:00 bash
 1393 tty1     00:00:00 ps
mcardenas@mcardenas:~$
```

Gestión de procesos en Linux

Con **ps** a secas podemos ver los procesos del usuario (si en vez del UID queréis verlo con el username se pone **ps u**, y además sale **%CPU** y **%MEM**)

¿Cómo identifico mi UID?

$**echo $UID**

```
mcardenas@mcardenas:~$ echo $UID
1000
mcardenas@mcardenas:~$
```

```
mcardenas@mcardenas:~$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mcarden+  1279  0.0  0.1  21472  4984 tty1     S    19:47   0:00 -bash
mcarden+  1403  0.0  0.1  38376  3656 tty1     R+   19:55   0:00 ps u
mcardenas@mcardenas:~$
```

Gestión de procesos en Linux

- Ver todos los procesos del sistema escribo: **ps -ef**
- Para ver las primeras líneas: **ps -ef | head -5**
- Valores importantes mostrados en ps (ps –ef):
  - **UID** – usuario que ha lanzado el proceso
  - **PID**: ID del proceso
  - **PPID**: ID del proceso padre
  - **CMD**: orden que ha lanzado el proceso

```
mcardenas@mcardenas:~$ ps -ef | head -5
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 19:47 ?        00:00:01 /sbin/init maybe-ubiquity
root           2     0  0 19:47 ?        00:00:00 [kthreadd]
root           4     2  0 19:47 ?        00:00:00 [kworker/0:0H]
root           6     2  0 19:47 ?        00:00:00 [mm_percpu_wq]
```

Gestión de procesos en Linux

- Con **ps –u joselito** se pueden ver qué procesos está ejecutando el usuario **joselito**
- Con **ps –e** se pueden ver todos los procesos del sistema

```
mcardenas@mcardenas:~$ ps -u mcardenas
  PID TTY          TIME CMD
 1258 ?        00:00:00 systemd
 1263 ?        00:00:00 (sd-pam)
 1279 tty1     00:00:00 bash
 1443 tty1     00:00:00 ps
mcardenas@mcardenas:~$ ps -u mcardenas | head -5
  PID TTY          TIME CMD
 1258 ?        00:00:00 systemd
 1263 ?        00:00:00 (sd-pam)
 1279 tty1     00:00:00 bash
 1444 tty1     00:00:00 ps
mcardenas@mcardenas:~$ _
```

Gestión de procesos en Linux

Otra opción muy común para lanzar la orden **ps** es **ps aux**

- La lista muestra muchos procesos, para ello usa el comando en conjunción con **grep** (si quiero filtrar) o con **more** (paginar)
  - **ps aux | grep bash**
  - **ps aux | more**
- También, para ver las primeras líneas:
  - **ps aux | head -5**

Gestión de procesos en Linux

Valores importantes mostrados en **ps aux**:

USER – usuario que ha lanzado el proceso

PID: ID del proceso

%CPU y **%MEM**: % uso de CPU y memoria

TIME: tiempo de CPU acumulado

STAT

```
mcardenas@mcardenas:~$ ps aux | head -10
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.2  77804  8860 ?        Ss   19:47   0:01 /sbin/init maybe-ubiquity
root         2  0.0  0.0      0     0 ?        S    19:47   0:00 [kthreadd]
root         4  0.0  0.0      0     0 ?        I<   19:47   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        I<   19:47   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0     0 ?        S    19:47   0:00 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        I    19:47   0:00 [rcu_sched]
root         9  0.0  0.0      0     0 ?        I    19:47   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    19:47   0:00 [migration/0]
root        11  0.0  0.0      0     0 ?        S    19:47   0:00 [watchdog/0]
mcardenas@mcardenas:~$
```

Gestión de procesos en Linux

Estado del proceso (STAT) → ver **man ps**

S: (interruptible sleep): en pausa, esperando un evento para continuar

R: (running o runnable): en ejecución

T: Marks a stopped process. Parado

U: process in uninterruptible wait

I (iddle): Marks a process that is idle (sleeping for longer than about 20 seconds)

Z: Marks a dead process (a “zombie') ver man ps

Gestión de procesos en Linux

Procesos **padre** y **procesos** hijos

- Para verlos, lanzar varias instancias del shell con la orden bash, y ver con **ps -ef** los PID y el PPID
- **ps -ef | grep bash** (ver su PID y el PPID)

Para ver quién es su padre:

- **ps -p 538** (o el PPID que sea)
- **bash** (lanzo otro proceso hijo bash)
- **ps -ef | grep bash** (ver su PID y el PPID, que será el PID del primer bash)

Gestión de procesos en Linux

Para mostrar el árbol de procesos se usa la orden **pstree**

Posiblemente habrá que instalarlo con **apt-get install**

Forma de ejecutarlo:

- **pstree -p** (aparecen los PIDs)
- **pstree 1581** (el árbol de un proceso en concreto con PID 1581)
- También se puede ver algo parecido con **ps axjf**

Gestión de procesos en Linux

Más información de los procesos:

< High-priority (not nice to other users)

N Low-priority (nice to other users)

+ Is in the foreground process group

Gestión de procesos en Linux

La orden **top**: Muestra valores actualizándose instantáneamente

- Parecido al monitor del sistema y al administrador de tareas (task manager)
- Pero desde aquí no se pueden administrar los procesos

Opciones en **top**:

top –d 5 (Donde 5 es el número de segundos a transcurrir entre cada muestreo)

top –o %CPU (Donde %CPU es el valor por el que vamos a ordenar los procesos)

Gestión de procesos en Linux

Trabajos: listado de trabajos (procesos creados por un usuario) ejecutándose

- También se le llaman **tareas**
- Se usa la orden **jobs**

A cada trabajo se le asigna un número **N** como identificador.

Un trabajo (proceso) puede ejecutarse en:

- Modo **foreground** (primer plano) o
- **Backgroung** (segundo plano)

Gestión de procesos en Linux

Background: $ **nombre_del_proceso &**

Ejemplo:

- Orden **sleep**: la orden sleep ejecutaría un proceso después del tiempo especificado

Por ejemplo: **sleep 10; date**

Mostraría la fecha a los diez segundos

```
mcardenas@mcardenas:~$ sleep 10; date
```

```
mcardenas@mcardenas:~$ sleep 10; date
mar feb 11 20:53:48 UTC 2020
mcardenas@mcardenas:~$
```

Gestión de procesos en Linux

Background: $ **nombre_del_proceso &**

Ejemplo:

- Orden **sleep**: la orden sleep ejecutaría un proceso después del tiempo especificado

Por ejemplo: **sleep 10; date**

Mostraría la fecha a los diez segundos

Prueba:

- **sleep 60 & > /dev/null**
- Para verlo: **jobs** (identificado con [1])
- **ps aux | grep sleep** (identificado con su PID)

Gestión de procesos en Linux

```
mcardenas@mcardenas:~$ sleep 10; date
mar feb 11 20:53:48 UTC 2020
mcardenas@mcardenas:~$ sleep 60 & > /dev/null
[1] 1524
mcardenas@mcardenas:~$ jobs
[1]+  Running                 sleep 60 &
mcardenas@mcardenas:~$ ps aux | grep sleep
mcarden+  1524  0.0  0.0   6176   836 tty1     S     20:57   0:00 sleep 60
mcarden+  1526  0.0  0.0  13136  1088 tty1     S+    20:58   0:00 grep --color=auto sleep
mcardenas@mcardenas:~$
```

Suspender un trabajo en foreground: [ctrl-z]

- **sleep 60**
- ctrl-z

Ver **jobs**

Gestión de procesos en Linux

Para pasar un trabajo de foreground a background (y viceversa):

- Primero suspenderlo [ctrl-z]
- Lo veo en **jobs**

Escribir luego

- **$ bg %n** (**fg** para foreground), siendo **n** el número de trabajo que se ve en **jobs**
- No hace falta poner %

Gestión de procesos en Linux

Si **sleep 60** es el trabajo **2**, lo paso a foreground así:

- **fg 2**

Lo suspendo otra vez: [Ctrl-z]

Lo veo en jobs

Lo paso a background así:

- **bg 2**

Parar un trabajo foreground antes de que acabe: [Ctrl-C]

- No lo veré en jobs

Gestión de procesos en Linux

Otro ejemplo:

- Ejecuto **top** en foreground
- ¿Qué diferencia hay entre pulsar [Ctrl+c] y [Ctrl+z]
- Una vez suspendido reanudarlo en foreground
- Pasarlo a background desde foreground
- ¿Cómo lo paro ahora, estando en background?

Gestión de procesos en Linux

No puedo usar [Ctrl+c]

- Debo matar la **tarea**: kill %1

Otra opción sería matar el **proceso**. Para ello debería buscar antes el PID y luego hacer un kill a ese PID, sin el %

- **ps aux | grep top**
- **kill 1767** (1767 = PID del proceso)

Fijaos que no se pone el %

¿Qué pasa si se pone **kill 1** en vez de **kill %1**?

Gestión de procesos en Linux

¿Quién puede matar un proceso?

- Sólo el propietario de ese proceso
- Recordad que lo puedo saber por el UID asociado al proceso

Cuando se ejecuta la orden kill, el SO envía una señal al proceso (señal 15 por defecto o de terminación de proceso)

- Existen más de 20 señales
- Existen algunos procesos como el propio Shell que no mueren cuando reciben esta señal
- ¿Qué hacemos entonces para matarlo?

Gestión de procesos en Linux

Intenta matar la shell

- Kill de la muerte: **sudo kill -9 PID**

Ejemplo:

- Identificar **con ps –ef** y grep el PID del shell

También se puede hacer poniendo solo **ps**

Intentar matarlo con un kill

- Matarlo con el **kill -9**

Gestión de procesos en Linux

Al igual que Apache, multitud de servicios necesitan ser reiniciados, y la mayoría de ellos responde al argumento ‘HUP’ (Hang up) de kill.

Mediante el siguiente comando, el servicio perteneciente a Apache, se reiniciará y volverá a cargar el fichero de configuración, permitiéndonos ver si los cambios han surtido efecto y volviendo a dar servicio a los usuarios.

- **kill –HUP [PID de Apache]**

Gestión de procesos en Linux

Si queremos matar un proceso por su nombre y no por su PID usaremos la orden **killall**

- **killall sleep**
- **killall Firefox**

Esta orden mataría el proceso Firefox y todos los que dependen de él.

- Otro ejemplo con **sleep** y **shell** scripts: la orden **who** (también w sólo) dice quién está en el sistema.
- Mediante **(sleep 3600; who >> log) &** crearíamos un log con el registro de usuarios en el sistema una hora después.

Gestión de procesos en Linux

¿Cómo funciona?

- Se crea un proceso en modo background que duerme (suspende su operación) durante 3600 segundos; luego se despierta, ejecuta la orden who y coloca la salida en un fichero de nombre log.

Otro ejemplo: mediante este script:

```
(while true
do
sleep 60
finger
done) &
```

Finger dice qué usuarios han hecho login en el sistema

Gestión de procesos en Linux

Prioridad: se usa la orden **nice**

- La orden **nice** en Linux sirve para ejecutar un programa con su prioridad de programación modificada.
- Se le asignará de esta forma más o menos tiempo de CPU y se ejecutará por lo tanto más lento (o más rápido)
- Los valores van desde el -20 (mayor prioridad al proceso) al 19 (menor prioridad)
- Solo los usuarios root pueden modificar la prioridad de todos los procesos.
- Los demás usuarios podrán modificar la prioridad de los procesos de los que son propietarios.

Gestión de procesos en Linux

Prioridad

Aumentar la prioridad:

- **nice -n-20 orden** (-20 es la máxima prioridad)

Disminuir la prioridad

- **nice -n19 orden** (19 es la mínima prioridad)

Necesitamos ser root (o sudo) para cambiar la prioridad

Gestión de procesos en Linux

Prioridad

Ejemplo:

sudo su

nice -n-19 sleep 120 &

ps –l (ver su prioridad PRI)

nice -n19 sleep 130 &

ps –l (ver su prioridad PRI)

Gestión de procesos en Linux

Prioridad

- Podemos restablecer la prioridad de este proceso, con pid 898, con el comando **renice**
- **renice 898** (898 sería el pid de sleep; verlo con **ps al**)
- Su prioridad volvería a ser 0

Gestión de procesos en Linux

Prioridad

- Si se observa que un usuario del sistema está ocupando mucho tiempo de CPU, se puede asignar a todos los procesos que este lance una prioridad más baja (o más alta)

Haga la siguiente prueba:

- Como root: **renice +20 -u alumno**
- Salir de root (logout)
- Y al volver a ser el usuario, lanzad **top &**
- Ver su prioridad con **ps al**

Gestión de procesos en Linux

Otras órdenes:

time: la orden time se utiliza para determinar la duración de ejecución de un determinado comando.

- Da el tiempo del sistema, el tiempo del usuario y el tiempo real
- Ya lo veremos con los algoritmos de gestión de procesos

Bibliografía

- **CARRETERO**, Jesús, GARCÍA, Félix, DE MIGUEL, Pedro, PÉREZ, Fernando. Sistemas Operativos: una visión aplicada. McGraw-Hill, 2001.
- **STALLINGS**, William. Sistemas operativos: aspectos internos y principios de diseño. 5ª Edición. Editorial Pearson Educación. 2005. ISBN: 978-84-205-4462-5.
- **TANENBAUM**, Andrew S. Sistemas operativos modernos. 3ª Edición. Editorial Prentice Hall. 2009. ISBN: 978-607- 442-046-3.

marlon.cardenas@ufv.es

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior