LPI 103.5. Crear, monitoritzar o matar processos

Wiki:

http://acacha.org/mediawiki/index.php/LPI_103.5

Autor: Sergi Tur Badenas

Objectius

103.5. Crear, monitoritzar o matar processos	
	▪ **Objectiu**: Els candidats han de ser capaços de gestionar els processos a un nivell bàsic. ▪ **Pes**: 4
	**Àrees Clau de Coneixement**: ▪ Executar processos en primer i en segon pla. ▪ Indicar a un programa que s'executi després de finalitzar la sessió. ▪ Fer un seguiment dels processos actius. ▪ Seleccionar i ordenar els processos per a la seva visualització. ▪ Enviar senyals als processos.
	La següent és una llista parcial de fitxers, termes i utilitats utilitzades: ▪ & ▪ bg ▪ fg ▪ jobs ▪ kill ▪ nohup ▪ ps ▪ top ▪ free ▪ uptime ▪ killall
	**Apunts**: LPI 103.5. Crear, monitoritzar o matar processos

Autor: Sergi Tur Badenas

Procés

En informàtica un procés és una instància d'una aplicació que esta essent executada per una computadora

- **Característiques**
  - Una aplicació o programa és solament una col·lecció passiva d'instruccions que no esdevenen un procés fins que el programa és executat.
  - Una sola aplicació o programa pot arribar a executar múltiples processos al mateix temps
  - Linux és un sistema operatiu multitasca i també multi-proces (suport per a múltiples CPUs) i pot executar diferents processos al mateix temps
  - És possible llançar vàries instàncies d'un mateix programa simultàniament.

Autor: Sergi Tur Badenas

Elements d'un procés

- **Procés = estructura de dades del SO**
  - Contenen (meta)informació sobre els programes que està executant el sistema operatiu.
  - Elements (dades) més importants:
    - **Identificador del procés (PID)**
    - **Identificador del procés pare (PID)**
    - **Estat del procés**
    - **Prioritat** del procés
    - **Senyals** pendents de ser processades.
    - ...
  - Apunts tema 16 oposicions

Autor: Sergi Tur Badenas

PID

- **Process IDentifier (PID)**
  - Tot procés s'identifica amb un número únic, el PID.
  - L'encarregat de gestionar els processos és el **kernel** i disposa d'una **taula de processos** on cada procés es identificat pel seu PID.
    - **Una aplicació no pot crear un procés directament!**. Es necessari demanar al sistema operatiu que crei el procés i el gestioni. Es disposa de dos **crides de sistema**:
      - **fork**: permet que un procés crei un altre procés. Aquest nou procés es anomenat procés fill
      - **exec**: permet executar una aplicació externa. Substitueix el procés en execució per un altre procés. No confondre amb l'ordre exec.

Autor: Sergi Tur Badenas

Tipus de processos

- **Tipus de processos**
  - **Procés fill**: Jerarquia de processos. Creats per fork.
  - **Procés orfe**: el pare ha finalitzat la seva execució abans que el fill.
  - **Procés zombie**: Procés que ha acabat però que resta a l'espera d'una resposta del pare.
  - **Procés parat**: es poden parar processos, pausa (Ctrl+z).
  - **Procés dimoni (servei)**: s'executa permanentment i en segon terme. No té interfícies d'usuari associades.

Autor: Sergi Tur Badenas

Jerarquia de processos Unix.

- **Unix/Linux**
  - Els processos només es poden executar a partir de processos ja existents
  - Els processos acabin tenint una jerarquia (relacions pare/fill).
  - La crida de sistema que permet en Unix crear un nou procés és fork (forquilla). Fork crea un procés fill que és una còpia casi exacte del procés pare i que s'executa en paral·lel al procés pare
- **Dos conceptes:**
  - **Herència**: El procés hereta totes les propietats del pare (de fet és una còpia exacte del seu pare).
  - **Sincronització entre processos pare i fill**: Mecanismes IPC com senyals

Autor: Sergi Tur Badenas

Procés init

- **El primer procés en executar-se és el nucli (kernel)**
  - És executat pel gestor d'arrancada
  - El procés del nucli que gestiona tota la resta de processos s'anomena **scheduler** (planificador de processos). No el podeu visualitzar (té el PID 0)
- **Procés init**
  - És el procés pare de tota la resta de processos. PID 1
  - Ho veurem amb més detall a l'objectiu:
    - 101.2 Arrancada del sistema

Autor: Sergi Tur Badenas

Procés init

- **El podeu veure amb pstree**

```
$ pstree -p | head
init(1)-+-NetworkManager(848)-+-dhclient(4278)
        |                     `-{NetworkManager}(2682)
        |-acpid(1076)
        |-apache2(2270)-+-apache2(2322)
        |               |-apache2(2323)
        |               |-apache2(2325)
        |               |-apache2(2326)
        |               `-apache2(2327)
        |-atd(1087)
        |-avahi-daemon(802)---avahi-daemon(803)
```

- L'opció **-p** mostra el PID del procés
- L'ordre head permet visualitzar el principi d'un fitxer o flux de dades

Autor: Sergi Tur Badenas

pstree

- **Permet visualitzar la jerarquia de processos**

```
$ pstree
...
     ├─gnome-terminal─┬─8*[bash]
     │                ├─bash─┬─pstree
     │                │      └─sleep
```

  - Localitzeu l'interpret d'ordres on heu executat pstree
  - Executeu: **$ sleep 10&**
  - Torneu a executar pstree
  - Vegeu com el procés sleep és fill de bash i germà de pstree
  - També hi ha l'opció: $ ps **--forest**

Autor: Sergi Tur Badenas

pstree

- Mostrar el pid i l'usuari:

```
$ pstree -up | less
```

- Mostrar les ordres completes:

```
$ pstree -a | less
```

- Ordres similars (--forest f de ps):

```
$ ps -aux -forest
$ ps -ejH | more
```

```
$ top -pa
 init,1
   |-NetworkManager,6507
   |-acpid,5156 -c /etc/acpi/events -s
/var/run/acpid.socket
   |-apache2,6712 -k start
   |   |-apache2,6785 -k start
   |   |-apache2,6786 -k start
```


Autor: Sergi Tur Badenas

Manipulació de processos

- **Crear processos. 2 tipus:**
  - **Primer pla (foreground – fg)**: és el mode per defecte amb el qual executem ordres a l'interpret d'ordres. Les ordres bloquegen l'execució de l'interpret

```
$ sleep 10     $ evince fitxerPDF.pdf     $ ooimpress Transpas.odp
```

  - **Segon pla (background – bg):** es pot fer amb el símbol **&** (o l'ordre bg)
    - Ens proporciona el PID
    - Es per poder gestionar el procés
      - p. ex. Matar el procés amb kill

```
$ sleep 10 &
[1] 2514
$
[1]+  Fet
        sleep 10
```


Autor: Sergi Tur Badenas

Control de tasques

- **Ordres built-in de bash per gestionar processos**
  - **jobs:** mostra els processos que s'estan executant a l'interpret
  - **fg**: passa un procés de segon terme a primer terme
  - **bg:** passa un procés de primer terme a segon terme
  - Aquestes ordres formen part de bash (com p. ex. l'ordre cd)
  - Seguim l'exemple:
    - Control de tasques Unix

Autor: Sergi Tur Badenas

Control de tasques

El mode per defecte en que s'executa una ordre a la línia d'ordres és en primer pla. Per executar una comanda en segon pla, s'utilitza l'operador **&**

```
$ ordre &
```

Per comprovar-ne el funcionament, s'utilitza l'ordre `sleep`. Aquesta ordre s'espera un temps especificat en segons abans de finalitzar la seva execució. Per exemple:

```
$ sleep 10
```

Bloquejarà la terminal durant 10 segons. Si s'executa:

```
$ sleep 10&
```

La terminal no es bloqueja i apareixerà un missatge similar a

```
[1] 13578
```

On **[n]** ([1] en l'exemple) és el número de procés que s'està executant en segon terme i **13578** és el PID del procés.

Es poden executar diversos cops l'ordre sleep i consultar la taula de treballs (jobs) en segon terme utilitzant l'ordre **jobs**:

```
$ sleep 10&
[1] 13691
$ sleep 10&
[2] 13692
$ sleep 10&
[3] 13695
$ sleep 10&
[4] 13696

$ jobs
[1]   Running                 sleep 10 &
[2]   Running                 sleep 10 &
[3]-   Running                 sleep 10 &
[4]+   Running                 sleep 10 &
```

En qualsevol moment es pot aturar un procés amb la combinació de tecles Ctrl+Z (envia la senyal aturar). Escriviu:

Autor: Sergi Tur Badenas

Control de tasques

En qualsevol moment es pot aturar un procés amb la combinació de tecles Ctrl+Z (envia la senyal aturar). Escriviu:

```
$ sleep 50
```

I s'aturarà l'ordre amb la combinació Ctrl+Z (envia la senyal aturar). Apareixerà el següent per pantalla:

```
[1]+  Stopped                 sleep 50
```

L'ordre sleep ha quedat suspesa tal i com es pot veure executant:

```
$ jobs
[1]+  Stopped                 sleep 50
```

Es pot reprendre la seva execució en primer pla amb:

```
$ fg sleep
```

O es pot reprendre en segon pla amb:

```
$ bg sleep
[1]+ sleep 50 &
```


Autor: Sergi Tur Badenas

Manipulació de processos

- **Acabar (terminar) un procés**
  - 3 formes d'acabar prematurament un procés que s'està executant en **primer pla**:
    - **Ctrl-c**: Envia el senyal 2 (SIGINT). **La majoria** d'aplicacions estan programades per finalitzar l'execució quan reben aquesta senyal. Algunes comandes no ho fan així ho requereixen d'una doble confirmació abans de finalitzar el procés)
    - **Ctrl+**\: S'envia el senyal 3 (SIGQUIT). També depèn de com estigui programada l'aplicació.
    - Utilitzar **kill**
  - L'única forma de finalitzar un procés que s'executa en segon pla és utilitzar la comanda kill.

Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **ps (process status)**
  - Permet consultar la taula de processos
  - Diferents implementacions de l'ordre ps. Difereixen sobretot en la forma d'utilitzar les opcions:
    - **Opcions UNIX**: les opcions han d'estar totes agrupades i han d'estar precedides per un guió (-)
    - **Opcions BSD**: poden estar agrupades i no utilitzen mai el guió (-).
    - **Opcions llargues de GNU**: precedides per dos guions (--)

Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **Processos associats a la terminal i l'usuari actuals:**

```
$ ps
PID TTY          TIME CMD
31852 pts/7    00:00:00 bash
31884 pts/7    00:00:00 ps
```

  - Dos processos:
    - L'interpret d'ordres en si (bash) (PID 31852)
    - El propi procés de l'ordre ps (PID 31884)
    - **TTY (terminal associada):** on s'ha executat ps (ordre $ tty)
    - **TIME:** Temps acumulat d'ús de CPU
    - **CMD**: la **com**an**d**a que ha generat el procés

```
$ id -u
1000
$ echo $EUID
1000
```

  - **EUID**: **E**ffective **U**ser **Id**entifier. Variable d'entorn

Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **Mostrar tots els processos**
  - Unix:

```
$ ps -e
$ ps -ef
$ ps -eF
$ ps -ely
```

  - BSD:

```
$ ps ax
$ ps axu
```

```
$ ps aux | head
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2528  1508 ?        Ss   10:19   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   10:19   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   10:19   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   10:19   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   10:19   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S<   10:19   0:00 [migration/1]
root         7  0.0  0.0      0     0 ?        S<   10:19   0:00 [ksoftirqd/1]
root         8  0.0  0.0      0     0 ?        S<   10:19   0:00 [watchdog/1]
root         9  0.0  0.0      0     0 ?        S<   10:19   0:00 [events/0]
```


Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

Opcions

- a: Amb l'estil BSD, per defecte només es mostren els processos de l'usuari que executa l'ordre ps. L'opció a treu aquesta restricció i mostra tots els processos associats a una terminal
- x: Combinat amb a mostra tots els processos, associats o no a una terminal.
- u: Mostra informació orientada a l'usuari. El resultat es certes columnes extres (USER, %CPU, MEM...) respecte a ax
- -e: genera una llista amb informació de tots els processos.
- -l: mostra una llista llarga (long format) d'informació dels processos
- -f: mostra certa informació extra com el PPID i el UID. Realment
- -y: combinat amb -ls no mostra els FLAGS.
- -F: encara més informació extra respecte -f

Autor: Sergi Tur Badenas

Mostra la taula de processos. ps

- **ps -aux**

```
$ ps -aux > psmenysaux
Warning: bad ps syntax, perhaps a bogus '-'?
See http://procps.sf.net/faq.html
```

  - L'ordre $ps -aux normalment dona el mateix resultat si poseu $ ps aux
  - Excepte l'avís anterior (a l'ordre anterior hem redireccionat la sortida estàndard a un fitxer però l'error estàndard s'ha mostrat per la terminal).
  - Això és així per que realment ps -aux us hauria de mostrar segons l'estàndard POSIX i UNIX, tots els processos amb una terminal associada (-a) per a l'usuari (-u) x. Si no teniu usuari x al sistema l'ordre suposa que realment volieu fer ps aux sense guió.

Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **Paginar:**

```
$ ps aux | more
$ ps aux | less
```

  - Buscar un procés:

```
$ ps ax | grep firefox
 3044 ?        Sl     9:23 /usr/lib/firefox-3.5.4/firefox
 4373 pts/1    R+     0:00 grep firefox
```

    - Cal ignorar la línia del propi grep
    - També podem utilitzar **pidof**:

```
$ pidof firefox
3044
```

    - Mostrar els processos d'un usuari:

```
$ ps -u root
```

    - Mostrar tots d'un usuari (els executat com root o com un altre usuari però amb SUID)

```
$ ps -U root -u root
```


Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- Mostrar el nom d'un procés a partir del seu PID:

```
$ ps -p 7293 -o comm=
bash
```

- Mostrar un procés pel nom de l'ordre

```
$ ps -C syslogd -o pid
  PID
 5280
```

- Indicar els camps de la sortida

```
$ ps -eo user,pid,ppid,uid,gid,command,stat,tty,time,%cpu,ni,rss,%mem,share,stime
 USER       PID  PPID   UID   GID COMMAND                     STAT TT           TIME %CPU  NI   RSS %MEM - STIME
 root         1     0     0     0 /sbin/init                  Ss   ?        00:00:01  0.0   0  1496  0.1 - 16:38
 root         2     0     0     0 [kthreadd]                  S<   ?        00:00:00  0.0  -5     0  0.0 - 16:38
 root         3     2     0     0 [migration/0]               S<   ?        00:00:00  0.0   -     0  0.0 - 16:38
 root         4     2     0     0 [ksoftirqd/0]               S<   ?        00:00:00  0.0  -5     0  0.0 - 16:38
```


Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **Username (USER)**: el nom de l'usuari que executa el programa
- **Process ID (PID)**: l'identificador de procés.
- **Parent PID (PPID)**: El PID del procés pare.
- **UID**: l'identificador de l'usuari sota el qual s'executa el procés.
- **GID**: l'identificador del grup (**G**roup **ID**entifier) que executa el procés
- **COMMAND**: mostra l'ordre que ha generat aquest procés.
- **STAT**: mostra l'estat del procés.
- **Teletype (TTY o TT)**: el codi que identifica la terminal on s'està executant el procés. Els **dimonis** (serveis Linux), són processos especials que no tenen cap terminal associada ("?")
- **TIME**: indica el total de temps acumulat de CPU

Autor: Sergi Tur Badenas

Mostrar la taula de processos. ps

- **%CPU**: és (o era) el tant per cent de CPU ocupada pel procés en el moment que vam executar ps
- **NI**: el codi de nivell de prioritat del procés.
- **Memòria**: Hi ha múltiples capçaleres relacionades amb la memòria.
- **RSS**(**Resident Set Size**): memòria utilitzada pel procés i les seves dades.
- **%MEM**: és el percentatge de memòria que el programa està (o millor dit estava) consumint durant l'execució de ps
- **SHARE**: Memòria compartida amb altres processos (com p. ex. biblioteques compartides)
- **START o STIME**: Mostra la data d'execució del procés.

Autor: Sergi Tur Badenas

Estat del processos

- **Model de 7 estats (amb paginació)**

```
 R “Running”. Procés en curs d’execució (l'està utilitzant la CPU)
 T “sTopped”. Procés parat (amb les tecles Ctrl + Z per exemple)
 S “Sleeping”. Procés dormit, en espera del processador.
 D “Device”. Procés dormit en espera d’un recurs (generalment una
  entrada/sortida). Els processos en aquest estat no poden interrompre’s.
 Z “Zombie”. Procés terminat. Seguirà en aquest estat fins que el seu pare
ho noti i recuperi el seu codi de retorn.
```

  - El segon caràcter del camp STAT està posat a W si el procés s’ha mogut a l’espai de paginació.
  - Finalment apareix una N o un < com a tercer caràcter de la columna STAT si el procés és respectivament de menor o de major prioritat.

Autor: Sergi Tur Badenas

Estat del processos

Autor: Sergi Tur Badenas

Senyals

- **Mecanisme bàsic de comunicació entre processos**
  - Mecanisme IPC (Inter-Process Communication) proporcionat pel sistema operatiu.
  - Hi ha 64 senyals que s'identifiquen per un número o un nom simbòlic (**SIG+Nom_del_senyal**)
  - Cada procés pot programar l'acció que durà a terme per a cada senyal, excepte 2 senyals:
    - **SIGKILL(9)**: permet finalitzar un procés de forma abrupta
    - **SIGSTOP(19)**: permet aturar temporalment un procés.

Autor: Sergi Tur Badenas

Senyals

- **Senyals més freqüents**
  - **INT (2)**: Interrupt. Es enviada quan polsem CTRL+c. Els programes poden ignorar la senyal, simplement sortir o realitzar tasques abans d'aturar-se
  - **KILL (9)**: elimina el procés immediatament i incondicionalment.
  - **TERM (15)**: Terminate. Finalitza el procés de forma controlada (si està implementada la senyal)
  - **TSTOP (19)**: Para l'execució temporalment (Ctrl+z). Es pot tornar a executar l'aplicació, per exemple amb les ordres **fg** o **bg** o la senyal **CONT(18)**

Autor: Sergi Tur Badenas

Enviar senyals. kill

- **Permet enviar senyals a un procés**
  - Sintaxi:
```
kill [ -signal | -s signal ] pid ...
```
  - Mostrar totes les senyals:
```
$ kill -l
```
  - Intentar aturar un procés correctament (SIGTERM per defecte):
```
$ ps aux | grep "nom_proces"
sergi      4643  0.0  0.0   3036   ............
$ kill 4643
```
  - Si està penjat i no respon el podem matar amb:
```
$ kill -9 4643
```
  - Mostrar el nom d'una senyal a partir del codi de senyal:
```
$ kill -l 19
STOP
```


Autor: Sergi Tur Badenas

Enviar senyals. kill

- Només podem matar processos nostres o qualsevol procés si som root
- Matar tots els processos:

```
$ kill -9 -1
```

- **killall**
  - Envia la senyal segons el nom:

```
$ killall mozilla
$ kill -9 mozilla
```

  - L'ordre pidof és un enllaç a killall

```
$ ls -la /bin/pidof
lrwxrwxrwx 1 root root 16 2008-11-18 08:54 /bin/pidof -> ../sbin/killall5
```


Autor: Sergi Tur Badenas

Monitoritzar processos en temps real. top

- **Top**
  - Mostra els processos ordenats per percentatge d'ús de CPU (de més a menys) i de forma continuada:

```
$ top
top - 11:34:41 up  1:13,  2 users,  load average: 0.14, 0.23, 0.22
Tasks: 155 total,   2 running, 153 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.8%us,  0.5%sy,  0.0%ni, 97.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2063192k total,  1192316k used,   870876k free,    70848k buffers
Swap:  4594548k total,        0k used,  4594548k free,   424452k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
6554 root      20   0  356m  24m 8776 R    3  1.2   4:20.86 Xorg
7283 sergi     20   0  102m  28m  14m S    2  1.4   0:04.04 gnome-terminal
7279 sergi     20   0  333m 175m  25m S    1  8.7   4:33.38 firefox
6202 root      10 -10  1768  408  332 S    1  0.0   0:03.42 mouseemu
7000 sergi     20   0 54368  24m  15m S    1  1.2   0:06.38 gnome-panel
7025 sergi     20   0 21460 3100 1756 S    1  0.2   0:07.22 gnome-screensav
   1 root      20   0  3056 1888  564 S    0  0.1   0:01.58 init
```

  - Existeixen alternatives/millores a top com **atop** o **htop**

Autor: Sergi Tur Badenas

Monitoritzar processos en temps real. top

- **Capçalera**

```
top - 11:37:13 up  1:16,  2 users,  load average: 0.31, 0.25, 0.23
 Tasks: 155 total,   1 running, 154 sleeping,   0 stopped,   0 zombie
 Cpu(s):  1.9%us,  0.8%sy,  0.0%ni, 97.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
 Mem:   2063192k total,  1193276k used,   869916k free,    71252k buffers
 Swap:  4594548k total,        0k used,  4594548k free,   424468k cached
```

  - **up:** és la quantitat de temps que porta encesa la màquina.
  - **users**: nombre d'usuaris diferents (inclou un mateix usuari que estigui "logat" en diferents terminals)
  - **load average**: Càrrega mitjana del sistema (tots els valors respecte 1)
  - **Tasks**: estadístiques de les tasques.
  - **Cpu(s)**: estadístiques de la CPU.
  - **Mem**: estadístiques de la memòria del sistema
  - **Swap**: Estadístiques de la memòria swap.

Autor: Sergi Tur Badenas

Monitoritzar processos en temps real. top

- **Mostrar colors**: premeu z
- **Ajuda**: premeu h
- Manipular processos
  - **Matar un procés**: prémer k i indicar el PID del procés
  - **Canviar la prioritat d'un procés**: prémer r i indicar el PID
- Canviar les columnes de sortida:
  - **f:** permet afegir/treure columnes
  - **o:** permet canviar l'ordre
  - **u**: mostrar només les tasques d'un usuari
  - **n**: limitar el nombre de processos a mostrar
  - **i**: mostrar només el processos actius

Autor: Sergi Tur Badenas

Prioritat dels processos

- **Valors nice**
  - Indiquen la prioritat d'un procés. **Rang: -20 a 20**
    - Valor més alt de prioritat és -20 (màxima prioritat)
    - Valor més baix de prioritat és 20 (mínima prioritat).
    - La prioritat per defecte és 0
  - Els processos amb més prioritats utilitzen més recursos. Els processos amb prioritat mínima (+20), només s'executaran quan el sistema no executa cap altra tasca.
  - Els usuaris, només poden modificar els processos dels quals són propietaris en un interval de **0 a 20**.
  - L'usuari root, pot canviar la prioritat de qualsevol procés a qualsevol valor.

Autor: Sergi Tur Badenas

Modificar la prioritat d'un procés. renice

- **Permet modificar la prioritat d'un procés**
  - Sintaxi:

```
$ renice prioritat [ [-p] pid ... ] [ [-g] pgrp ...] [ [-u] user ... ]
```

  - On:
    - **prioritat**: és el valor de la prioritat assignada al procés
    - **pid**: és l'identificador del procés (si volem actuar sobre més d'un procés, hem d'escriure la opció –p)
    - **pgrp**: és l'identificador del grup de proces (hem d'utilitzar la opció –g si n'hi ha més d'un).
    - **user**: nom d'usuari del propietari del procés (utilitzem –u si volem actuar sobre més d'un).

Autor: Sergi Tur Badenas

Modificar la prioritat d'un procés. renice

- **Exemples**
  - Baixar la prioritat d'un procés amb PID 785:

```
$ renice +15 785
```

  - Modificar la prioritat dels processos d'un usuari:

```
# renice +20 -u alumne1
```

  - També podem modificar la prioritat amb:
    - Ordre top: prement r
  - Consultar la prioritat amb ps:
    - Columna NI

Autor: Sergi Tur Badenas

nice

- **Permet executar un procés amb una prioritat concreta**

```
$ nice -n 19 dd if=/dev/cdrom of=~/mdk1.iso
```

  - Per defecte defineix una prioritat de 10
  - L'opció –n servei per establir un valor de prioritat.
  - L'anterior ordre no satura gens el sistema

Autor: Sergi Tur Badenas

nohup

- **Executa una comanda immune als hangups**
  - Evita que un procés engegat en una terminal especifica s'aturi al finalitzar la terminal. El nou procés no té cap relació amb la terminal. L'ordre és una ordre built-in de bash.
  - Per comprovar-ho, executeu dos terminals. En una executeu:

```
$ sleep 10000
```

  - Comproveu a l'altre terminal que el procés sleep s'està executant amb:

```
$ ps aux | grep sleep
```

  - Tanqueu la terminal on heu executat sleep 10000.
    - Comproveu com també s'atura el procés sleep
    - Passa el mateix si executeu sleep en background. Per evitar-ho:

```
$ nohup sleep 10000
```


Autor: Sergi Tur Badenas

time

- **Conèixer el temps que triga en executar-se un ordre**

```
$ time ls -la
total 20
drwxr-xr-x  2 sergi sergi 4096 2008-10-18 11:30 .
drwxr-xr-x 69 sergi sergi 4096 2008-10-18 13:22 ..
-rwxr-xr-x  1 sergi sergi  347 2008-10-12 20:57 iconv.sh
...
real   0m0.187s
user   0m0.004s
sys    0m0.000s
```

- Temps total d'execució: també conegut com a temps real.
- Temps d'ús de la CPU d'usuari: temps d'execució de la comanda en espai d'usuari.
- Temps d'ús de la CPU de sistema: temps d'execució de la comanda en espai de sistema.

Autor: Sergi Tur Badenas

lsof

- **Mostra els fitxers oberts d'un sistema. Els podem mostrar tots amb:**

```
$ sudo lsof | more
COMMAND     PID        USER   FD      TYPE     DEVICE      SIZE       NODE NAME
init          1        root  cwd       DIR        8,2      4096          2 /
init          1        root  rtd       DIR        8,2      4096          2 /
```

  - Es pot utilitzar per localitzar quin procés bloqueja un fitxer i matar el procés:

```
$ sudo lsof /u/abe/foo

$ sudo kill -HUP `lsof -t /u/abe/bar`
```

    - Útil quan no podem desmuntar una unitat per que està en ús

Autor: Sergi Tur Badenas

disown

- **Permet dessasociar una tasca de la terminal on s'ha executat**

```
# sleep 30&
[1] 12642
# sleep 30&
[2] 12643
# sleep 30&
[3] 12644
# jobs
[1]   Running                 sleep 30 &
[2]-  Running                 sleep 30 &
[3]+  Running                 sleep 30 &
# disown %1
# jobs
[2]-  Running                 sleep 30 &
[3]+  Running                 sleep 30 &
```

- Les podem dessasociar totes amb:

```
$ disown -a
```


Autor: Sergi Tur Badenas

Altres ordres

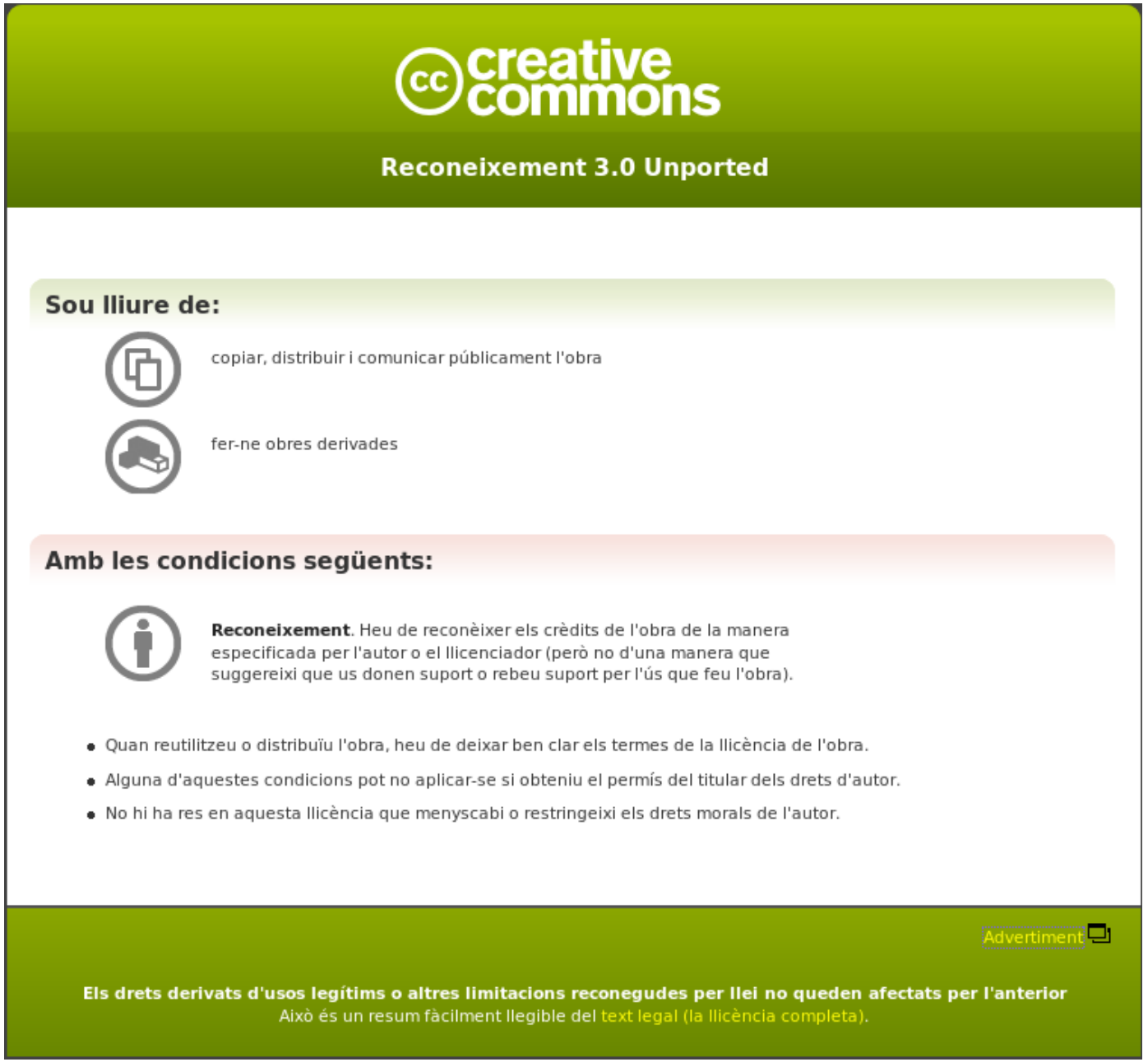
- **pgrep i pkill**
  - Similars a pidof i killall
- **free**
  - Mostra l'ús de memòria al sistema
- **uptime**
  - Mostra el temps que fa que s'executa la màquina

```
$ free
             total       used       free     shared
   buffers     cached
Mem:       1026560     669388     357172          0
     94816     240656
-/+ buffers/cache:     333916     692644
Swap:      4104596          0    4104596
```

```
$ uptime
 14:10:34 up  3:50,  5 users,  load average: 0.43, 0.20, 0.12
```


Autor: Sergi Tur Badenas

Llicència Creative Commons



http://creativecommons.org/licenses/by/3.0/deed.ca

Autor: Sergi Tur Badenas