Nombre: ..... Fecha: / 12 /2010

Grupo: 1 ☐ 2 ☐ 3 ☐ 4 ☐

PRÁCTICA 18
GESTIÓN DE PROCESOS Y TAREAS

Una de las tareas más importantes del SO es la gestión de los procesos que se están ejecutando en una máquina. El hecho de que todos los procesos deban compartir los recursos hardware disponibles (memoria RAM, CPU) hace que el SO juegue un papel primordial en gestionar esos recursos para que los procesos se ejecuten de forma simultánea (al menos de cara al usuario) y compatible.

Una posible definición de proceso es que un proceso es un programa que se encuentra en ejecución. Cada proceso, durante su ejecución, guarda información sobre su "contexto" que incluye, entre otras cosas, información sobre su proceso padre, los recursos del sistema que se están consumiendo (segmentos de memoria asignados), permisos de seguridad, etc.

Antes de empezar la práctica, visita el enlace http://structio.sourceforge.net/guias/AA_Linux_colegio/procesos-y-tareas.html y lee el punto 3.1 del mismo.

1. Inicia Ubuntu. Desde la línea de mandatos, lee la página del manual sobre el mandato ps. Ejecuta ps y visualiza los procesos que se están ejecutando en este momento. El primer valor que aparece es el identificador de proceso (PID). El segundo es la terminal que está asociada a ese proceso. Después también podemos observar el tiempo acumulado de uso de CPU, y finalmente el nombre del programa que ha dado lugar a este proceso. Apunta los procesos activos y sus valores.

2. Por defecto, ps sólo ha mostrado los procesos asociados con la misma terminal e iniciados por el mismo usuario. Vamos a recuperar las opciones que nos permiten mostrar todos los procesos. Ejecuta el mandato ps –e. Comprueba la lista de procesos que están corriendo en tu máquina. ¿Cuál lleva el PID igual a 1? ¿Qué procesos se están ejecutando en las terminales tty1, tty2…tty6?

Recuerda lo que sucedía al pulsar "Ctrl + Alt + F1" hasta "Ctrl + Alt + F6"; comprueba el nombre de cada una de esas terminales. Vuelve a la terminal en que se encuentra el entorno gráfico. ¿Qué proceso está corriendo sobre tty7?

3. El mandato ps todavía nos puede ofrecer más información sobre los procesos en ejecución. Por ejemplo, ¿quién ha iniciado cada uno de los procesos en nuestra máquina? Vamos a usar las siguientes opciones del mandato ps: "a" nos permite conocer todos los procesos que tienen una terminal asociada; "x" aquellos que no tienen terminal; la opción "u" nos muestra la salida en un formato más legible. Teclea el mandato ps aux.

Observa en las cabeceras la información que has obtenido de cada proceso. Busca en http://unixhelp.ed.ac.uk/CGI/man-cgi?ps el significado de aquellas columnas que desconozcas y apúntalo en tu informe (por ejemplo, RSS,

VSZ). Busca también en el anterior enlace y apunta en tu informe, en la sección "PROCESS STATE CODES", el significado del status de cada uno de los procesos que están en ejecución.

4. Todos lo mandatos y opciones que hemos visto hasta ahora ofrecían información estática sobre los procesos. Esta información se extrae del directorio "/proc" del sistema. Hay algunas aplicaciones que también nos permiten conocer en tiempo real las características de cada proceso. Ejecuta el mandato man top. ¿Qué hace el mandato top? Ejecuta el mismo.

Como puedes observar, la información sobre el sistema se refresca cada 3 segundos (se puede modificar ese parámetro). Por lo demás, la interfaz de usuario de top no es especialmente agradable, aunque sí resulta sencillo modificar ciertas opciones y ajustarla a nuestros requisitos. Pulsa "q" para salir de top.

5. Ejecuta ahora el mandato "top –u alumno". ¿Cómo ha cambiado la salida del mandato? ¿A quién pertenecen los procesos que observas ahora?

6. Vuelve a ejecutar top. Apunta el PID del proceso Xorg. Dirígete a la carpeta "/proc". Ejecuta ls. ¿A quién pertenecen los distintos directorios presentes en el mismo? Busca el directorio cuyo nombre coincida con el PID de Xorg. Muévete al directorio. Lista el contenido del mismo.

Comprueba qué tipo de fichero es limits (file limits). Observa su contenido con ayuda de "sudo less limits". ¿Qué tipo de información contiene?

Puedes leer en http://www.linux.com/archive/feature/126718 algunas de las peculiaridades del directorio "/proc". Apunta algunas de sus principales características.

7. Veamos ahora algunos atajos de teclado que nos permiten gestionar procesos. Comprueba la función del mandato yes. Aunque el mismo pueda no parecer de gran utilidad a nosotros nos va a servir para comprobar cómo podemos detener y "matar" procesos. Ejecuta el mandato "yes hola". Observa que el mensaje aparece infinitas veces.

8. Vamos a "matar" esa tarea. Intenta salir de la tarea con "q". La tecla "q" nos permite salir de ciertas aplicaciones en ejecución, pero no acabar con una tarea. Teclea el atajo "Ctrl + C". El mismo debería terminar con el proceso activo. ¿Qué ha sucedido? El atajo de teclado "Ctrl + C" se encarga de terminar (o matar) una tarea. El atajo "Ctrl + Z" se encarga únicamente de detenerla (aunque el proceso siga "vivo" y se pueda retomar en el estado en que se detuvo). Finalmente, el atajo "Ctrl + Y" suspende un proceso hasta que el mismo disponga de una nueva entrada.

9. Podemos ahora redirigir la salida del mandato a un fichero (observa que esto podría darnos serios problemas de memoria en nuestra máquina). Para poder redirigir la salida de mandatos a un fichero sin peligro de que eso colapse nuestra memoria, Linux dispone de un fichero llamado /dev/null.

Comprueba las propiedades de "/dev/null". Apunta el tipo de fichero que es (c), sus permisos, propietario y tamaño. ¿Podemos escribir al mismo? ¿Podemos leer su contenido?

Ejecuta el mandato "yes adios > /dev/null".

Como puedes observar, la tarea en ejecución no permite seguir utilizando la terminal. Teclea de nuevo "Ctrl + C" para detenerlo de nuevo. ¿Qué tamaño ocupa ahora en disco el fichero /dev/null? ¿Dónde ha ido a parar toda la información que hemos enviado?

10. Ejecutamos de nuevo el madato "yes que tal > /dev/null".

11. Abre una nueva terminal y localiza el PID del proceso "yes" iniciado. Puedes usar ps o top. Comprueba la utilidad del mandato kill. Como puedes observar, kill nos permite mandar señales a un proceso. El tipo de señales que permite mandar lo puedes encontrar, por ejemplo, en http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_12_01.html.

Lee las Secciones 12.1.1.3 y 12.1.2 del anterior enlace. Apunta algunas de las señales más comunes que se pueden enviar a procesos en ejecución y la forma de hacerlo (por teclado, o por medio de kill). Interrumpe el proceso activo "yes" por medio del mandato kill (kill –s 15 PID). Observa la diferencia entre SIGKILL y SIGTERM.

12. Envía la señal SIGTERM al proceso yes antes iniciado (kill –s señal PID). Vuelve a ejecutar el proceso "yes que tal > /dev/null" y, desde la otra terminal, envíale ahora una señal de SIGKILL. Comprueba que el resultado externo ha sido el mismo que antes.

13. No todos los procesos que ejecutamos en una terminal tienen que ejecutarse en primer plano (bloqueando así la terminal). También podemos hacer lo que se conoce como ejecución en segundo plano. Puedes leer en http://www.mastermagazine.info/termino/5040.php información acerca de las principales diferencias entre ejecutar un proceso en primero o segundo plano (esencialmente tiene que ver con la prioridad del mismo).

La forma de hacer que un programa se ejecute en segundo plano es escribiendo el programa en el intérprete de mandatos seguido de un símbolo &; comprueba el siguiente mandato:

$yes otra vez > /dev/null &

14. Vamos a hacer ahora uso del mandato jobs. Comprueba en primer lugar su función por medio de "help jobs". La diferencia entre un "job" y un "process" es que los "jobs" son obligatoriamente iniciados desde una terminal y están asociados a ella (son procesos "hijos" de la terminal).

Ejecuta el mandato "jobs" en la misma terminal que has ejecutado "yes". ¿En qué estado se encuentra el proceso?

Compruébalo también con los mandatos ps y top. Observa el porcentaje de CPU que consume.

15. Entre los procesos de Linux siempre existe una jerarquía definida, ya que cada proceso debe tener un proceso padre (excepto el proceso de inicio o init). Esta jerarquía adquiere relevancia ya que "matar" a un proceso padre por lo general conlleva acabar también con los procesos hijos. En algunos casos, un proceso padre y sus hijos pueden incluso compartir memoria. Comprueba la jerarquía de procesos en tu máquina por medio del mandato pstree (puedes ver alguna de sus opciones en man ps). Comprueba sus ancestros. Apunta en tu informe de qué procesos desciende "yes".

16. Vuelve a ejecutar yes en la misma terminal y también en segundo plano (yes mensaje > /dev/null &). Vuelve a comprobar el árbol de procesos por medio de "pstree -h". Usando top, anota en tu informe el porcentaje de CPU (aproximado) que suman estos dos procesos yes.

17. Vuelve a comprobar el estado de los procesos iniciados en esta shell por medio de "jobs". Comprueba que aparecen las dos tareas iniciadas y que en la segunda aparece el símbolo + indicando que es la última que se ha ejecutado.

18. El hecho de que las tareas estén ejecutándose en segundo plano, impide que les podamos enviar una señal de teclado (por ejemplo, Ctrl + Z, Ctrl + C). Compruébalo.

A través de "jobs", cada tarea que se está ejecutando desde nuestra terminal recibe un nuevo número (1, 2…). Esos números pueden ser usados con fg (foreground) para traer dichas tareas a primer plano (fg 1, fg 2), o para mandarlas a segundo plano (bg 1, bg 2…).

19. Mata la segunda tarea iniciada. Por ejemplo, puedes ejecutar top y capturar su PID, y enviarle una señal de kill.

20. Comprueba por medio de jobs que sólo queda una tarea activa.

21. Ejecuta el mandato "yes mensaje2 > /dev/null". Por medio del teclado (Ctrl + Z), o por medio de kill (con la señal SIGSTOP ó 19 y con el PID correspondiente) envía al proceso una señal de "detenido".

22. Comprueba que el proceso está detenido por medio de jobs.

23. Para recuperar una tarea detenida sólo tienes que ejecutar fg (foreground) o bg (background), dependiendo de que quieras que la tarea se ejecute en primer o segundo plano. Comprueba con fg que la tarea vuelve a primer plano.

24. Por medio de una señal o del atajo de teclado detén de nuevo la tarea. Vuelve a iniciarla, esta vez en segundo plano (bg). Vuelve a detenerla sin traerla a primer plano. Para ello debes usar obligatoriamente señales por medio de kill.

25. Crea nuevas tareas por medio de "yes" en primer y segundo plano y prueba a detenerlas y reanudarlas por medio de bg, fg y las señales de kill.

26. Vamos a recuperar ahora la idea de que todos los procesos dependen de su proceso "padre". Ejecuta una nueva terminal. Ejecuta varias veces seguidas el mandato "yes hola > /dev/null &". En tu otra terminal comprueba ahora la estructura de "pstree". ¿Quién es el antecesor directo de todos ellos? Ejecuta "top –u alumno" para comprobar qué tareas tienes en ejecución y cuáles son las que más recursos consumen. Cierra ahora la terminal en que se encuentran los procesos "yes" ejecutándose. Vuelve a ejecutar "pstree". ¿Aparece el "bash" correspondiente? ¿Aparecen los procesos "yes"? Ejecuta también "top –u alumno" y comprueba que las tareas ya han desparecido de tu lista de procesos en ejecución. Teniendo en cuenta que "init" es el proceso "padre" de todos los procesos que están en ejecución, imagina lo que sucedería al ejecutar "kill –s 9 1".

27. Linux también dispone de utilidades para la programación de tareas; las tareas programadas son procesos que se ejecutarán (siempre y cuando la máquina esté encendida) de forma planificada. El programa que nos permite programar tareas desde línea de mandatos se llama cron.

Puedes leer el manual de cron o los siguientes enlaces para aprender las nociones básicas sobre su sintaxis, http://es.wikipedia.org/wiki/Cron_(Unix), http://www.linuxtotal.com.mx/index.php?cont=info_admon_006 (quizá alguno de los archivos a que hacen referencia estén en lugares distintos en nuestra distribución, lo importante es que recuerdes la sintaxis y la forma de uso).

Comprueba que "cron" está activo en tu ordenador (puedes observar si aparece en pstree). El proceso "cron" se debe encontrar siempre en ejecución, para que a la hora y día que tenga programada alguna tarea pueda ejecutar la misma. También te puedes asegurar de que el mismo está en marcha por medio del mandato:

$sudo /etc/INIT.d/cron start

28. Existen diversas formas de programar nuevas tareas. El fichero en el que se encuentran las tareas programadas se llama "/etc/crontab". Es un fichero de texto, así que puedes editarlo, por ejemplo, con nano. Para ejecutarlo deberás disponer de permisos de superusuario. Ejecuta:

$sudo nano /etc/crontab

Por defecto deberían aparecer varias tareas programadas del sistema. La estructura de cada una de las líneas es la siguiente:

minuto(s) - hora - día del mes – mes - día de la semana – usuario - mandato

De esta forma, una línea:

5,10,15,20,25,30,35,40,45,50,55 * * * * alumno cd /home/alumno/Escritorio; wget http://www.larioja.com

Significará que a los minutos 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 de todas las horas (*), de todos los días del mes (*), de todos los meses y de

todos los días de la semana (*), el usuario "alumno" ejecutará los mandatos "cd /home/alumno/Escritorio; wget http://www.larioja.com". Inserta el mandato anterior en la última línea de tu fichero crontab. Guárdalo.

Crea otra nueva tarea programada, basándote en la anterior, para que cada 3 minutos de todas las horas, todos los días cada mes, de todos los meses, cualquier día de la semana, el usuario alumno realice el siguiente mandato:

Moverse a la carpeta Escritorio;
lanzar un mensaje con echo "Otra vez aquí" redirigido a un fichero de nombre "aqui" (echo "Otra vez aqui" >> aqui)

Basándote en la sintaxis del fichero crontab tal y como viene explicada en http://es.wikipedia.org/wiki/Cron_(Unix) define una tarea programada tal que el día 3 de Junio de cada año, a la hora que tú prefieras, cree una carpeta en el Escritorio de tu ordenador y dentro de ella un fichero que se llame "log" donde, por medio de echo, se envíe el mensaje "Hoy es día 3 de Junio".

Comprueba que las tareas programadas anteriores cumplen con su cometido (en particular las dos primeras). Copia las tres tareas programadas anteriores en tu informe de prácticas.

29. Para terminar, redirige todos los mandatos de la práctica a un fichero "mandatos_practica18" y enlázala junto con tu informe de prácticas en tu página de inicio.

En Windows la capacidad para gestionar procesos desde la línea de mandatos es menor. Aún así, algunos mandatos como tasklist nos pueden ofrecer ciertas facilidades al respecto.