Práctica 6: Monitorización del Sistema

Objetivos

En esta práctica estudiaremos los aspectos básicos sobre la monitorización del sistema que incluye principalmente la gestión y monitorización de procesos y los archivos de log del sistema. Además veremos cómo automatizar la ejecución de tareas.

Contenidos

Preparando el entorno...

En esta práctica necesitaremos una máquina virtual con el sistema CentOS 7 instalado. Según se explica en la práctica 3, haced un clon enlazado de la instalación base disponible en el laboratorio.

Monitorización y Gestión de Procesos

En primer lugar estudiaremos cómo monitorizar los procesos que hay en ejecución en el sistema y los recursos que consumen.

Control de Trabajos en la Shell

El control de trabajos iniciados en la shell permite consultar, detener y reanudar su ejecución. La shell mantiene una lista de trabajos arrancados que puede consultarse con el comando `jobs`. Cada trabajo tiene un identificador, un `PID` del proceso asociado y la orden de ejecución.

Ejercicio 1. Para probar diferentes acciones sobre los procesos vamos a usar un comando gráfico que permita comprobar fácilmente el estado del proceso asociado. Escribir el siguiente contenido en un archivo (`progreso.sh`) y darle permisos de ejecución. Finalmente, ejecutar el script anterior y comprobar su funcionamiento.

Ejercicio 2. *Detener e interrumpir la ejecución de un proceso*. La combinación Ctrl+c envía la señal de interrupción (SIGINT) que interrumpe el proceso y Ctrl+z lo detiene (SIGTSTP). Comprobar el efecto de ambos sobre la ejecución del programa progreso.sh del Ejercicio 1.

Listado 1. Archivo progreso.sh

```
#!/bin/bash
( for i in `seq 00 99` ; do
    echo $i
    sleep 0.5
  done ) |
zenity --progress
```

Ejercicio 3. *Ejecuciones en primer y segundo plano.* Cuando ejecutamos un comando decimos que está en primer plano (*foreground*) y en posesión de la terminal de control asociada, i.e. recibe la entrada introducida por el terminal incluidas las señales generadas por teclado (ej. Ctrl+c). Equivalentemente, un proceso en segundo plano mantiene su ejecución pero cede el terminal de control a la shell. Los comandos `fg` y `bg` permiten poner en primer (*foreground*) y segundo plano (*background*) un proceso, respectivamente:

- Enviar un proceso a segundo plano: (i) ejecutar `progreso.sh`; (ii) Ctrl+z; (iii) `bg`
- Recuperar la ejecución en primer plano y terminar su ejecución Ctrl+c

Ejercicio 4. Es posible arrancar un proceso en segundo plano directamente añadiendo & al final de la orden. Arrancar el comando de prueba en segundo plano varias veces:

- Consultar lista de trabajos en ejecución en la shell se puede consultar con el comando `jobs` (ver el efecto de la opción -l)
- Cuando hay varios procesos en segundo plano se puede especificar el trabajo sobre el que actúan (%n) las órdenes `bg` y `fg`. Poner selectivamente procesos en primer plano.
- Consultar la página de manual bash en relación a `jobs`, `fg` y `bg`.

Ejercicio 5. *Terminal de control.* Los procesos tienen asociados un terminal de control al que se envían las salidas estándar y de error y que recoge la entrada estándar si es necesaria. Ejecutar el siguiente comando en primer y segundo plano; y analizar dónde se muestra la salida estándar y qué sucede cuando lee de la entrada estándar.

Listado 2. Ejecución de procesos en segundo plano que realiza operaciones de entrada/salida

```
$ (sleep 1; date ; cat - > /tmp/entrada; sleep 1; date ) &
```

Ejercicio 6. Arrancar el comando de prueba (`progreso.sh`) en segundo plano, cerrar la ventana del terminal donde se arranco y comprobar qué sucede. Ejecutar el mismo comando con la orden `nohup` (`man nohup`) también en segundo plano y repetir el ejercicio.

Ejercicio 7. El comando kill sirve para enviar señales a un proceso:

- Consultar `kill -l` para ver las señales disponibles.
- `kill` puede usar un PID o un job (%n). Repetir los ejercicios 3 y 4 usando el comando `kill` en lugar de Ctrl+c y Ctrl+z.

- Se puede enviar una señal a un proceso por nombre (a todos lo procesos que encajen) con el comando `pkill`. Arrancar varios procesos de prueba y eliminarlos todos enviando la señal SIGKILL con el comando `pkill`.

Monitorización de procesos

Ejercicio 1. La herramienta principal para ver los procesos en ejecución del sistema es `ps`. Ejecutar la orden sin argumentos y comprobar su salida.

Ejercicio 2. Estudiar la página de manual de `ps`, especialmente las opciones más comunes (ej. aux ó `-elf`) y el significado de los datos mostrados por cada proceso:

- Mostrar todos los procesos del usuario actual en formato extendido
- Mostrar los procesos del sistema, incluyendo el identificador del proceso, del grupo, la sesión, el estado y la línea de comandos.
- Observar el identificador de proceso, grupo y sesión de los procesos. ¿Qué identificadores comparten la shell y los programas que se ejecutan en ella?¿Cuál es el identificador de grupo de procesos cuando se crea un nuevo proceso?

Ejercicio 3. Otra forma sencilla de localizar procesos específicos (por ejemplo para enviar una señal) es mediante los comandos `pgrep` y `pidof`. Estudiar el uso de `pgrep`, buscar por ejemplo los procesos que encajen con `gnome.` ¿Cuántos procesos hay en el sistema cuyo nombre termina por `daemon`?.

Ejercicio 4. La herramienta `top` es muy útil porque muestra un resumen del sistema que incluye: carga (ver comando `uptime`), estado de la memoria (`free`) y procesos (`ps`). Esta herramienta permite filtrar por usuario, enviar señales a procesos, ordenar la lista de procesos según diferentes criterios y configurar la información mostrada.

- Ejecutar `top` y estudiar su uso (man top; pulsar h una vez arrancada).
- Ejecutar un nuevo terminal y arrancar varios procesos de prueba. Usando `top`, filtrar los procesos, pararlos y finalmente eliminarlos enviando la señal adecuada.

Ejercicio 5. El comando `lsof` permite ver los descriptores que tiene abierto un proceso, consultar la página de manual y probar las siguientes opciones:

- Procesos tienen abierto para escritura `/var/log/messages.`
- Procesos que tienen abierto algún fichero o directorio de `/etc` (`+D`).
- Ficheros que tiene abierta la shell que está usando (`-p`).
- **Nota:** Las conexiones de red (sockets) se pueden ver con `-i`.

Monitorización de la memoria y entrada/salida

Ejercicio 1. El consumo de la memoria virtual se puede obtener con `free`(1). Consultar el consumo de memoria del sistema. ¿Qué significa la memoria contabilizada bajo la sección *`buffers/cached`*?

Ejercicio 2. De la misma forma el comando `vmstat` permite recoger información sobre el rendimiento dinámico del sistema. Estudiar su uso (`man vmstat`) y el significado de la información que muestra en cada columna.

Ejercicio 3. Las estadísticas de rendimiento del subsistema de entrada salida se pueden mostrar con el comando `iostat`, que además del estado de la CPU, muestra la actividad de cada dispositivo:

- Estudiar su uso y el significado de cada métrica mostrada.
- Mostrar las estadísticas de E/S en un terminal cada segundo. En otro terminal escribir un archivo de 100M y observar el rendimiento del sistema.

Para saber más...

- Los procesos se estructuran en grupos y sesiones, que permite enviar señales de forma coordinada. Estudiar cómo se genera esa estructura de dependencia según se crean procesos.

- Estudiar las señales de control de la terminal `SIGTTIN`, `SIGTTOU`, `SIGTSTP` y `SIGHUP`. ¿Cuándo se generan?. ¿Cuál es el comportamiento por defecto del proceso asociado a cada señal?

- La herramienta `htop` es una versión mejorada de top, con un interfaz más amigable y algunas funciones extendidas. Probar su funcionamiento.

- La herramienta `iotop` permite monitorizar la entrada/salida de forma similar a top. Probar su funcionamiento.

Automatización de Procesos (cron y at)

Es el sistema de automatización permite ejecutar procesos a una hora y con una determinada periodicidad (e.g. *reconstruir los índices de archivos del sistema los lunes a las 2am*). Normalmente el demonio responsable es `cron`, aunque CentOS usa `anacron` para equipos que se apagan regularmente.

Ejercicio 1. Las programación de trabajos generales se especifica en /etc/crontab. Cada línea especifica una variable o un trabajo, mediante 5 parámetros temporales. Un * en uno de los parámetros significa para todos los posibles valores. Determinar la configuración para ejecutar un comando:

- 15 minutos después de la medianoche todos los sábados.
- El primer día de cada mes a las 3:30 AM.

Ejercicio 2. Consultar los archivos del directorio `/etc/cron.d` e interpretar su contenido.

Ejercicio 3. Cron está a disposición de los usuarios:

- `crontab -e`, permite editar una nueva entrada. Planificar la ejecución de un comando (date >> /tmp/salida) cada 5 minutos (*/5).
- comprobar la planificación de trabajos con `crontab -l`.

Ejercicio 4. El comando `at` sirve para planificar trabajos específicos que se ejecutan una sola vez, planificar la ejecución de un comando (ej. date > /tmp/at.ejemplo):

- `at <hora>`, la hora se puede especificar de muchas formas ver (`man at`), por ejemplo `at now + 3 minutes`
- Escribir los comandos que se ejecutarán, terminar con Ctrl+d
- Consultar los trabajos planificados con `atq` (`atrm` elimina comandos)

Logs del Sistema

Una parte importante del mantenimiento de un sistema es monitorizar las acciones que ocurren en él. Los archivos de log recogen esta información y son gran ayuda cuando es necesario analizar un

problema. En general los procesos usan una infraestructura común de *logging* vía el interfaz *syslog(3)*. Estos mensajes se procesan por los servicios de *logs* del sistema, en CentOS 7 hay dos servicios configurados por defecto: `journalctl` (systemd) y `rsyslogd`**.**

Cada uno de estos servicios ofrece características diferentes y pueden usarse conjuntamente.

Ejercicio 1. Comprobar que el paquete `rsyslog` está instalado, que el servicio está programado para arrancarse en los niveles multiusuario y que está en ejecución.

Ejercicio 2. El archivo de configuración es `/etc/rsyslog.conf`. Estudiar la sección `RULES`, cada regla hace referencia <servicio>.<severidad>. Los servicios son `authpriv`, `cron`, `kern`, `mail`, `news`, `user`, y `uucp`; y los niveles de severidad, de menor a mayor son: `debug`, `info`, `notice`, `warn`, `err`, `crit`, `alert`, `emerg`. Estudiar la página de manual de syslog, sección 3 para determinar el significado de los tipos anteriores.

Ejercicio 3. Estudiar los archivos de log en `/var/log`, ejemplo `boot.log`, `messages` y `secure`. Además el comando `dmesg` sirve para mostrar los mensajes del kernel.

Ejercicio 4. Los archivos de log pueden crecer demasiado lo que dificulta su manejo. Por defecto la utilidad logrotate rota los logs cada semana. Estudiar los contenidos de `/etc/logrotate.conf` y los archivos de configuración específicos de cada servicio en `/etc/logrotate.d`.

Ejercicio 5. `systemd` introduce una nueva filosofía de logging, con cierta controversia, que reemplaza el sistema de log en texto plano por una base de datos en binario. El comando principal de acceso al *journal* del systema es `journalctl`. Ejecutar este comando para mostrar todos los mensajes del sistema, incluidos todos los servicios y usuarios.

Ejercicio 6. La ventaja principal del almacenamiento binario de los mensajes de *log* es la posibilidad de indexarlo, usar las siguientes opciones de búsqueda:

- La opción -p permite ver los mensajes por prioridad: `debug` (7), `info` (6), `notice` (5), `warning` (4), `err` (3), `crit` (2), `alert` (1), and `emerg` (0). Ejemplo mostrar los mensajes de error.
- La opción -b muestra los mensajes del arranque actual
- Las opciones `--since` y `--until` permiten ajustar el rango temporal. Buscar los mensajes de error producidos por el sistema en los últimos 5 minutos.
- Otras opciones importantes son `-x`, para mostrar los mensajes completamente, `-e` para mostrar las últimas líneas y `-f` para activar el modo continuo.

Ejercicio 7. El comportamiento de journal se puede configurar en `/etc/systemd/journald.conf`, ver `journald.conf(5)`. ¿Dónde se almacenan los mensajes generados por `journal`?