

Monitorización

Administración de Sistemas

Unai Lopez Novoa
unai.lopez@ehu.eus



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Contenido

1. Introducción
2. Gestión de recursos
3. Análisis de rendimiento
4. Registros del sistema (logs)



Introducción

- Para asegurar el buen funcionamiento de una aplicación, hay 4 elementos a observar:

Latencias

Tráfico

Errores

Saturación



Introducción

- Latencia:
 - El tiempo necesario para completar una petición.
 - Importante distinguir entre peticiones satisfactorias y fallidas.
- Tráfico:
 - Métrica de la demanda de un sistema.
 - Suelen ser específicas de un sistema.
 - Ejemplos:
 - Peticiones HTTP / segundo.
 - Transacciones / segundo.



Introducción

- Errores:
 - Resultado de peticiones fallidas.
 - Pueden ser explícitos o implícitos
 - *Ejemplo:* Al solicitar una web a un servidor:
 - Error explícito: error 404.
 - Error implícito: recuperar una web diferente a la esperada.
- Saturación:
 - Medida de la capacidad de un recurso en uso.
 - Importante observar los recursos más limitados.
 - Latencia puede ser un indicador de saturación.



Introducción

- Cómo observar cada elemento:

Latencias

Monitorizar logs de aplicación.

Tráfico

Monitorizar logs de aplicación y red.

Errores

Monitorizar logs de aplicación y del sistema.

Saturación

Monitorizar hardware (CPU, memoria, ...) y red.



Monitorización de CPU

- Comando **top**
 - Uso de recursos del sistema en tiempo real
- Comando **ps**
 - Listado de procesos y su uso de recursos
- Comando **pstree**
 - Árbol de procesos del sistema



Gestión de procesos

- Prioridades de los procesos
 - El planificador del SSOO asigna intervalos de tiempo a los procesos según su prioridad.
 - Esto se controla según 2 valores:
 - Prioridad (**PR**): puede tomar valores en el rango -100 a 39.
 - Valor "nice" (**NI**): puede tomar valores en el rango -20 a 19.
 - *Columnas PR y NI en el comando Top.*
 - *Para ambos, cuanto más negativo el valor, mayor prioridad.*
 - En Linux, los procesos se consideran de 2 tipos:
 - Procesos normales (la mayoría de los lanzados por usuarios).
 - Procesos de tiempo real (generalmente, los esenciales para el SSOO).



Gestión de procesos

- Prioridades de los procesos normales
 - Se calcula: **PR** = 20 + **NI**
 - P.e. si el valor NI de un proceso es -20, su prioridad es 0
 - P.e. si el valor NI es 19, su prioridad es 39.
 - Los procesos normales ocupan el rango 0-39 de prioridades.
 - Por defecto, el valor "nice" (NI) es 0.
 - Un usuario normal puede modificar el valor NI entre 0 y 19.
 - Puede reducir la prioridad sobre el resto de procesos del sistema
 - El usuario *root* puede modificar el valor NI -20 y 19.



Gestión de procesos

- Prioridades de los procesos normales
 - Comando **nice**
 - Lanza un comando con un valor Nice concreto
 - Sintaxis: `nice -n valor comando`
 - Valor es relativo (define cuánto más o menos)
 - Ejemplo: `nice -n 10 ./miScript`
 - Comando **renice**
 - Cambia el valor Nice de un proceso (o grupo) en ejecución
 - Un usuario normal (no root) sólo puede incrementar el valor Nice.
 - Y cada cambio que haga es irreversible
 - Sintaxis: `renice -n valor -p PID [-g grupo]`
 - Valor es absoluto
 - Ejemplo: `renice -n 15 -p 7552`



Gestión de procesos

- Prioridades de los procesos en tiempo real
 - Se calcula: **PR** = - 1 - *prioridad_tiempo_real*.
 - El valor *prioridad_tiempo_real* toma valores entre 1 y 99.
 - P.e. si *prioridad_tiempo_real* es 50, PR vale -51.
 - El valor "nice" no se tiene cuenta.
 - En el comando top, si PR=-100, se muestra como 'rt' (real time).
- Comando **chrt**
 - Lanza un proceso con una prioridad de tiempo real
 - Sintaxis: `chrt --rr <prioridad_tiempo_real> <programa>`
 - Ejemplo: `chrt --rr 20 ./miPrograma`
 - Lanzaría ./miPrograma con PR=-21



Gestión de procesos

- Prioridades de los procesos: comando **ps**

- El comando **ps** puede mostrar datos en diferentes formatos
 - Cada formato puede mostrar una misma prioridad con diferentes valores.

- Formato BSD:

- Comando: **ps al**

F	UID	PID	PPID	PRI	NI	VSZ	RSS	WCHAN	STAT	TTY	TIME	COMMAND
...												
0	1000	2033	1104	21	1	7208	2708	- SN+.	pts/0		0:00	ping www.ehu.eus

- Formato Unix:

- Comando: **ps -u unai -o pid,user,pri,nice,args**

PID	USER	PRI	NI	COMMAND
...				
2033	unai	18	1	ping www.ehu.eus

Se muestra un valor de prioridad (PRI) diferente para un mismo proceso (PID=2033).



Gestión de procesos

- Comando **kill**

- Envía señales a procesos (no sólo matarlos)
- Sintaxis: **kill <opciones> PID**
 - Opciones: -l Mostrar las señales disponibles
 -señal Mandar una señal al proceso
 - Hay 3 formas de indicar una señal:
 - Con su número -19
 - Con el prefijo SIG -SIGSTOP
 - Sin el prefijo SIG -STOP
 - Señales: -STOP Parar el proceso
 -CONT Reanudar el proceso (parado con STOP)
 -KILL Matar el proceso
 ...



Gestión de procesos

- Comando **ulimit**

- Limitar el uso de recursos
- Los límites sirven para la Shell en uso
- Sintaxis: `ulimit -<opción> [límite]`
 - Opciones:
 - a Lista los límites establecidos
 - f Máximo número de ficheros creados por la Shell
 - m Máxima memoria disponible
 - t Máximo tiempo de CPU (segundos)

- Ejemplo:

```
unai@unai-server:~$ ulimit -a
core file size          (blocks, -c) 0
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 31543
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
```



Gestión de procesos

- Fichero `/etc/security/limits.conf`

- Permite hacer una configuración permanente de límites
- Cada línea tiene el siguiente formato:

usuario/grupo tipo-de-límite ítem valor

- Donde:

- | | |
|------------------|--|
| • usuario/grupo | Nombre del usuario o grupo (comienza con @) |
| • tipo-de-límite | soft/hard |
| • ítem | Puede ser: cpu, nproc, maxlogins, fsize, ... |
| • valor | Valor para el ítem definido |

- Ejemplos:

```
@student hard nproc 20
@faculty soft nproc 20
@faculty hard nproc 50
ftp      hard nproc 0
```

- Mas información: `man limits.conf`



Gestión de procesos

- Comando **cpulimit**

- Permite limitar el % de uso constante de CPU de un proceso
 - ulimit y limits.conf sólo permiten limitar el tiempo total de uso CPU
 - nice y renice permiten reducir la prioridad pero no fijar un umbral
- Está en los repositorios Debian
- Uso: `cpulimit --pid PID --limit <límite>`
 - Donde <límite> es el límite de % CPU máximo que queremos permitir
- Más información:
 - <https://www.tecmint.com/limit-cpu-usage-of-a-process-in-linux-with-cpulimit-tool/>



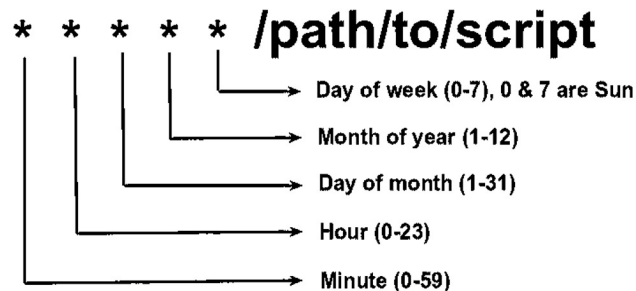
Planificación de tareas

- Comando **crontab**

- Una línea por tarea programada
- Sintaxis: **crontab** <opciones>
 - Opciones:

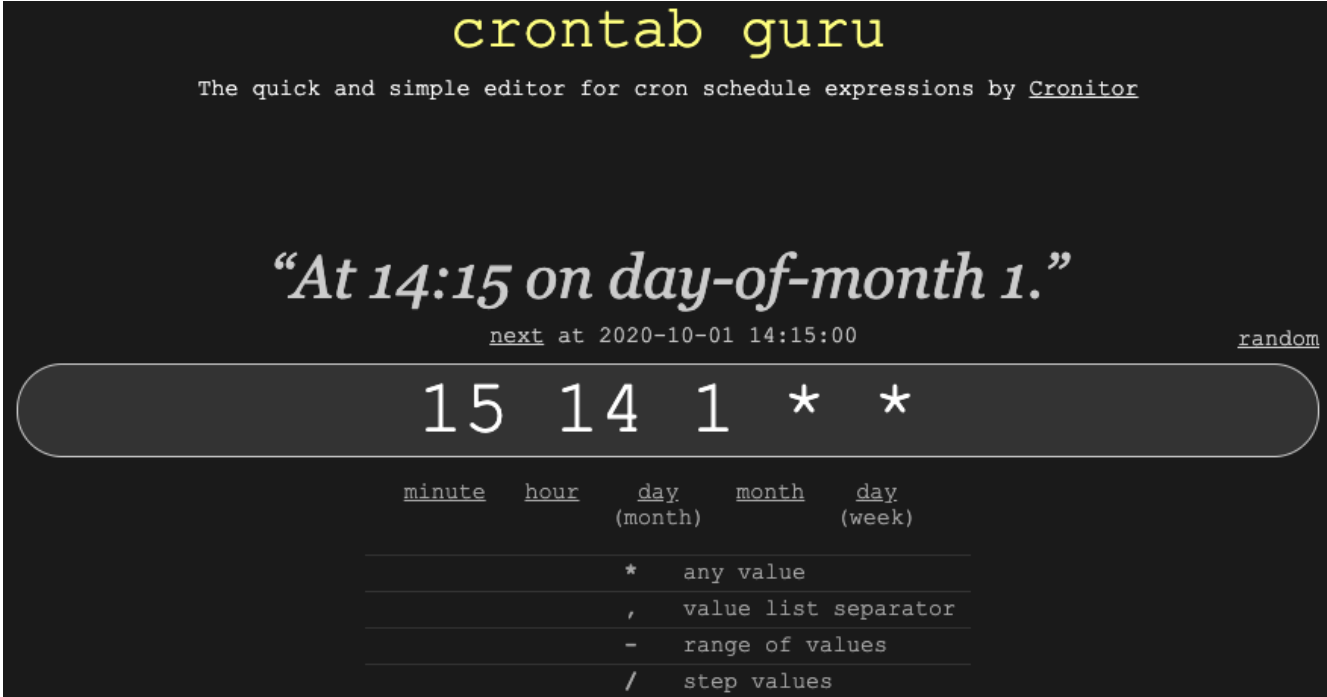
-l	Mostrar las tareas programadas
-e	Editar las tareas programadas
-r	Elimina las tareas programadas

- Cada entrada de cron es una línea sigue la estructura:



Planificación de tareas

- Comando **crontab**
 - <https://crontab.guru/>
 - Editor online de entradas cron



The screenshot shows the 'crontab guru' website. At the top, it says 'crontab guru' in yellow, followed by 'The quick and simple editor for cron schedule expressions by Cronitor'. Below this, a large quote reads: "At 14:15 on day-of-month 1." Underneath the quote, it says 'next at 2020-10-01 14:15:00' and a 'random' link. A large rounded rectangle contains the cron expression '15 14 1 * *'. Below this, a table explains the fields: minute, hour, day (month), month, and day (week). The table lists the symbols used in the cron expression: '*' for any value, ',' for value list separator, '-' for range of values, and '/' for step values.

<u>minute</u>	<u>hour</u>	<u>day</u> (month)	<u>month</u>	<u>day</u> (week)
		*	any value	
		,	value list separator	
		-	range of values	
		/	step values	



Planificación de tareas

- Comando **at**

- Controla las tareas a ejecutar por atd
- Para programar una tarea
 - Desde Shell: `at HORA` (donde HORA es una hora en formato HH:MM)
 - Se abre el Shell de at, escribir el/los comando(s) deseado(s):
 - P.e. `ls /home/unai -l`
 - Cerrar la Shell de at (Ctrl + D)
 - La salida estándar (stdout) se envía por mail usando sendmail
 - Conviene revisar `/var/spool/mail/<usuario>`
- Otras opciones:
 - Desde Shell: `at -l` Listado de tareas pendientes
 - Desde Shell: `at -d <ID>` Eliminar tarea (obtener ID con -l)



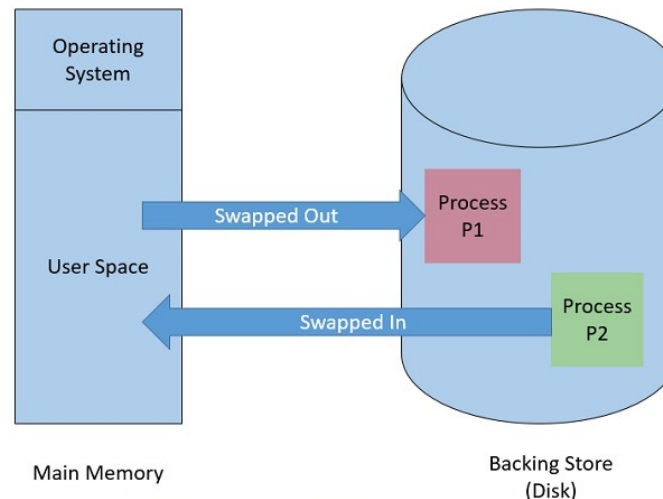
Ejercicio 1

- Instalar el paquete stress-ng
 - Suite de programas para evaluar el rendimiento
- Ejecutar stress-ng durante 2 minutos con 4 hilos CPU. Mientras está en ejecución, cambiar la prioridad de uno de sus procesos a la mínima posible.
 - ¿Qué sucede? Utilizar **top**.
 - Probar a cambiar la prioridad del mismo a la máxima posible.
- Ejecutar stress-ng durante 3 minutos con 1 hilo CPU. Mientras está en ejecución, limitar su uso de CPU al 50%.
 - Verificar con **top**.



Gestión de memoria

- La mayoría de sistemas operativos modernos utilizan memoria virtual.
 - Utilizan un espacio de disco como extensión de la memoria principal.
 - Espacio de intercambio o Swap.
 - Se organiza en páginas que se intercambian entre memoria y disco.



Gestión de memoria

- La mayoría de sistemas operativos modernos utilizan memoria virtual.
 - Un uso excesivo de Swap puede degradar el rendimiento.
 - Valores referencia de latencias en una jerarquía de memoria¹:

Tipo de memoria	Latencia
Caché L1 en CPU	1 ns
Caché L2 en CPU	4 ns
RAM	100 ns
SSD	16.000 ns
HDD	2.000.000 ns



Gestión de memoria

- Monitorizar la memoria

- Comando **top**

- Utilizar Shift+m para ordenar por consumo descendente de memoria

- Comando **vmstat**

- Campos relativos a la memoria:

procs		-----memoria-----				---swap--		-----io----		-sistema--		-----cpu-----				
r	b	swpd	libre	búfer	caché	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	8972	1097844	776348	5687216	0	0	0	3	1	2	0	0	100	0	0
0	0	8972	1097712	776348	5687216	0	0	0	0	42	48	0	0	100	0	0
0	0	8972	1097712	776348	5687216	0	0	0	0	39	48	0	0	100	0	0

swpd: Memoria swap en uso

libre: Memoria libre

buff: Memoria usada como buffer

cache: Memoria usada como cache

si: Memoria swap retirada de disco(Swap In)

so: Memoria swap llevada a disco (swap out)



Gestión de discos y ficheros

- Monitorización

- Comando **df**

- Listado de sistemas de ficheros y espacio disponible
 - Sintaxis: `df <opciones>`
 - Ejemplo: `df -h`
 - Muestra los tamaños en kB, MB, ... en lugar de en bytes (-h para Human readable)

- Comando **du**

- Tamaño de una rama del sistema de ficheros (p.e., de un directorio)
 - Sintaxis: `du <opciones> directorio`
 - Ejemplos: `du -sh /home`
 - Muestra el tamaño total del directorio /home sin listar todo su contenido



Gestión de discos y ficheros

- Monitorización

- Comando **lsof**

- Muestra los ficheros en uso por los procesos del sistema (**list open files**)
 - Útil para resolver el error “resource is busy” al desmontar una partición.

- Comando **iostat**

- Muestra estadísticas de uso y tasas de transferencia de los dispositivos de almacenamiento.
 - Uso: `iostat -p <disco>`
 - Ejemplo: `iostat -p /dev/sda`

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	2.26	26.52	258.30	127.34	438959	4275565	2107840
sda1	2.24	26.01	258.30	127.34	430614	4275564	2107840
sda14	0.00	0.02	0.00	0.00	272	0	0
sda15	0.01	0.43	0.00	0.00	7088	1	0



Gestión de red

- Monitorización

- Comando **netstat**

- Muestra información sobre las conexiones y rutas de red
 - Mostrar conexiones activas: `netstat -a | more`
 - Mostrar tabla de rutas: `netstat -r`

- Comando **nethogs**

- Muestra conexiones y ratio de tráfico enviado/recibido
 - Requiere instalar el paquete sysstat

NetHogs version 0.8.6-3

PID	USER	PROGRAM	DEV	SENT	RECEIVED
2561	unai	wget	ens4	55.561	84852.891 KB/sec
2312	unai	sshd: unai@pts/2	ens4	1.038	0.296 KB/sec
460	root	/usr/bin/google_osco..	ens4	0.011	0.011 KB/sec
?	root	unknown TCP		0.000	0.000 KB/sec
TOTAL				56.609	84853.198 KB/sec



Gestión de red

- Monitorización

- Comando **tcpdump**

- Es un analizador de tráfico para conexiones TCP/IP
 - Uso más común: captura de tráfico para posterior análisis.
 - Comenzar a capturar tráfico y guardar en un fichero
 - Sintaxis: `tcpdump -i <interfaz> -Z <usuario> -w <ficheroCaptura>`
 - Ejemplo: `tcpdump -i ens4 -Z unai -w miCaptura`
 - Las interfaces disponibles se pueden mostrar con: `ip link`
 - Visualizar un fichero de captura de tráfico:
 - Sintaxis: `tcpdump -nr <ficheroCaptura>`
 - Se puede añadir el parámetro `-ttt` para incluir la diferencia de tiempo entre cada paquete.



Gestión de red

- Comando **telnet**

- Útil para comprobar si un servicio remoto está a la escucha.
 - Sintaxis: `telnet <IP> <puerto>`

- Comando **netcat**

- Herramienta para leer de y escribir en conexiones de red.
- Utilidad: Abrir una conexión a la escucha en un puerto.
 - Sintaxis: `nc -l <puerto>`
- Utilidad: Conectarse a una IP/puerto y escribir en él.
 - Sintaxis: `nc <IP> <puerto>`



Ejercicio 2

- *Realizar este ejercicio en parejas, seréis A y B*
- A inicia una captura de tráfico con tcpdump en su MV.
 - La captura se debe guardar en un fichero
- B hace ping 5 veces a la máquina virtual de A.
- A para la captura de tráfico.
- Buscad paquetes relacionados con ping en el fichero de captura.
 - ¿Cuántos paquetes encontráis?



“Esta aplicación tarda mucho en ejecutarse”



Análisis de rendimiento

- Código Python de ejemplo
 - Implementación de la sucesión de Fibonacci (0, 1, 1, 2, 3, 5, 8, 13, ...)
 - Calcula el valor del número 34 en la sucesión

```
def fibo(n):  
    if n <= 1:  
        return n  
    return fibo(n-1) + fibo(n-2)  
  
if __name__ == "__main__":  
    print(fibo(34))
```

- URL: <https://github.com/ulopeznovoa/AS-profiling.git>
 - Fichero **fibonacci.py**



Análisis de rendimiento

- **time**: Mide cuánto tarda en ejecutarse un comando/script

- Uso:

```
time <comando>
```

- Salida:

```
real 0m2.345s  
user 0m1.876s  
sys  0m0.469s
```

- Interpretación:

- **real**: Tiempo total de ejecución (como si fuese medido con un cronómetro)
- **user**: Tiempo dedicado a ejecutar código de la aplicación (p.e. cálculos)
- **sys**: Tiempo dedicado a llamadas del sistema (p.e. acceder a ficheros)



Análisis de rendimiento

- **Profiler:** herramienta de análisis que mide el uso de recursos durante la ejecución de un software.
 - Mide el tiempo que pasa un software en determinadas funciones.
 - Ayuda a identificar *cuellos de botella*: porciones de código en las que el software pasa la mayor parte del tiempo.
 - Muestra el flujo de llamadas entre funciones.
 - Complementario a un debugger.



Análisis de rendimiento

- Los profilers son específicos para 1 o varios lenguajes.
- Algunos ejemplos:
 - **C / C++:** gprof, perf
 - **Python:** cProfile, py-spy, Scalene.
 - Uso de cProfile:

```
python3 -m cProfile fichero.py
```
 - **Matlab:** profiler integrado.
 - **Javascript:** Chrome DevTools.



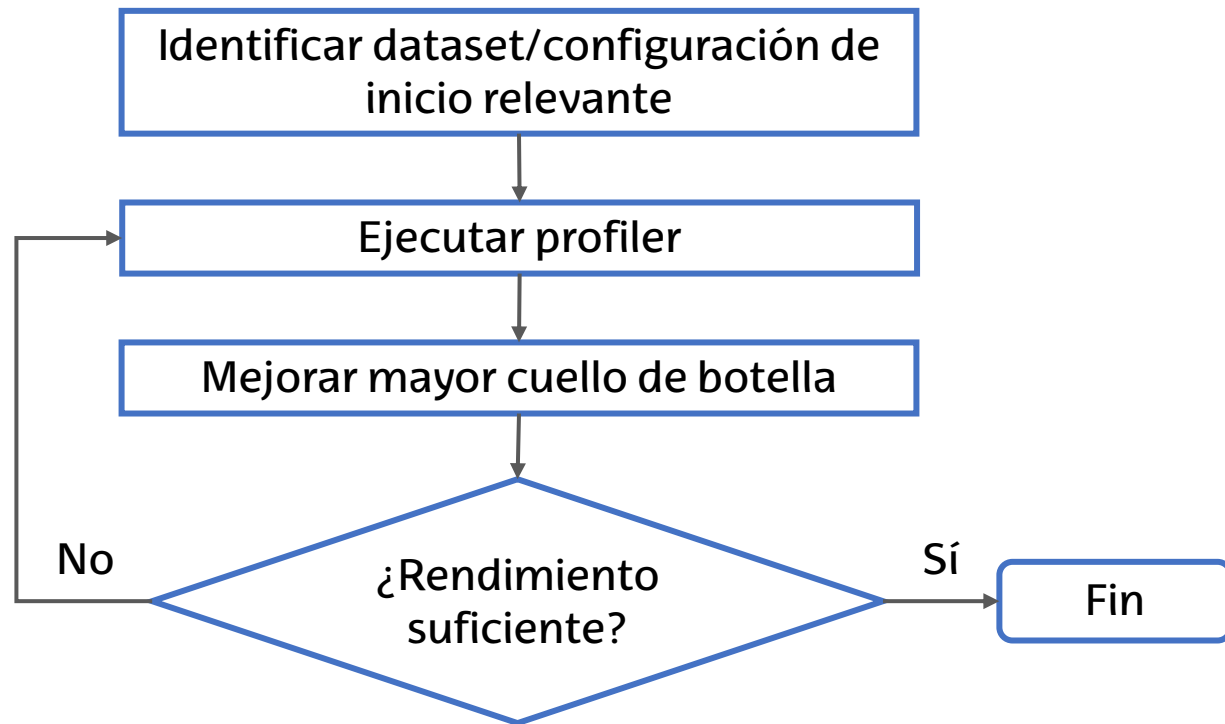
Análisis de rendimiento

- Hay 2 tipos principales de profilers:
 - 1) Basados en trazas (*Tracing profilers*):
 - Registran cada llamada y retorno a funciones
 - Suelen provocar *overhead* (incremento del tiempo de ejecución)
 - P.e. cProfile en Python
 - 2) De muestreo (*Sampling profilers*):
 - Inspeccionan el estado del programa en intervalo (p.e. cada 10 ms)
 - Menos precisos, poco impacto en el tiempo de ejecución
 - P.e. Scalene en Python



Análisis de rendimiento

- Procedimiento habitual de mejora de rendimiento:



Análisis de rendimiento

- El rendimiento de un programa está relacionado con su complejidad computacional:
 - La notación O grande define el rendimiento teórico
 - Los profilers miden el rendimiento real y uso de recursos
- Ejemplo:
 - Complejidad temporal de algoritmos de ordenación¹:

Algoritmo	Complejidad	Rendimiento
Bubble Sort	$O(n^2)$	Muy lento
Quick Sort	$O(n \log n)$	Rápido en la mayoría de casos
Radix Sort	$O(n d)$	Depende del tipo de datos



¹Fuente: Comparison of Sorting Algorithms - <https://medium.com/@tssovi/comparison-of-sorting-algorithms-298fdf037c8f>

Ejercicio 3

- Descargar el fichero **slow.py**
 - URL: <https://github.com/ulopeznovo/AS-profiling.git>
- Medir el tiempo de ejecución del programa y obtener un desglose de las llamadas con cProfile
- Analizar el código y reducir su tiempo de ejecución
 - Eliminar código innecesario
 - Mejorar código no óptimo



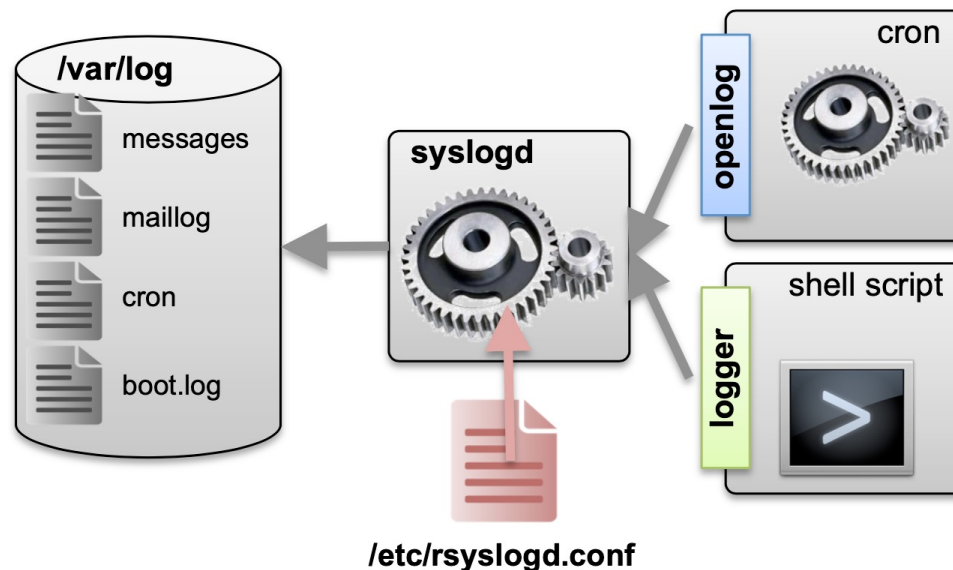
Logs

- El kernel de Linux, los servicios y las aplicaciones generan eventos constantemente:
 - Información sobre su estado, fallos/anomalías, ...
 - Errores de arranque
 - Acceso a información (seguridad)
- Una gestión adecuada de esta información es esencial para descubrir y solucionar problemas
- Todos estos eventos suelen estar gestionados por un único servicio
 - En Unix/Linux es syslog



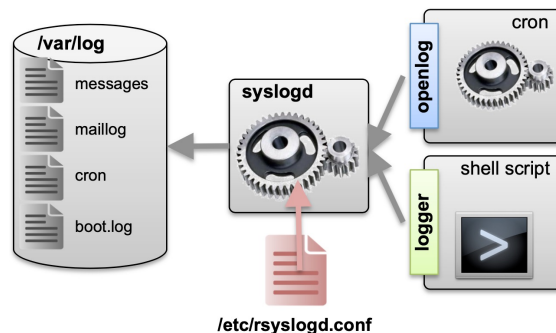
Syslog

- Es el recolector de eventos empleado por el kernel, servicios y aplicaciones
- Flexible, seguro y fácil de usar
- Está compuesto por los siguientes elementos:



Syslog

- Partes de syslog:
 - **syslogd**: Servicio del sistema. Recibe los mensajes del resto de servicios y aplicaciones y los añade al registro.
 - **openlog**: Librerías para usar syslog desde una aplicación.
 - P.e., `openlog` (C/C++), `sys::syslog(openlog(),syslog())` (Perl)
 - **logger**: Comando del sistema para enviar mensajes a syslog.
 - **rsyslogd.conf**: Fichero de configuración.
 - Ver siguiente transparencia.



Syslog

- **rsyslogd.conf**

- Listado de acciones a realizar en función de los mensajes recibidos.
- Tiene una línea por acción, con el formato:
 - entidad.nivel acción
- **Entidad:** lista de valores definidos por el sistema
 - P.e.: Kern, user, daemon (otro servicio), auth (login, su, ssh), mail, cron, ...
- **Nivel:** tipo de notificación
 - emerg, alert, crit, err, warning, notice, info, debug, * (todos los niveles)
- **Acción:**

• <nombre-de-fichero>	Escribir el mensaje a ese fichero.
• <nombre-dominio>/<IP>	Enviar el mensaje al syslogd del nodo indicado.
• <nombre-usuario>	Enviar mensaje al usuario, si está conectado.
• *	Enviar mensaje a todo usuario conectado.



Syslog

- rsyslogd.conf

- Ejemplo:

```
# First some standard log files.  Log by facility.  
#  
auth,authpriv.* /var/log/auth.log  
*.*;auth,authpriv.none -/var/log/syslog  
#cron.* /var/log/cron.log  
mail.* -/var/log/mail.log  
  
...
```

- En Ubuntu, por defecto es: /etc/rsyslog.d/50-default.conf
 - Los cambios en este fichero requieren reiniciar el servicio rsyslog.



Syslog

- Syslog escribe en ficheros en /var/log:
 - syslog Eventos generales, ni críticos ni de depuración
 - kern.log Registros del kernel
 - auth.log Información de inicio/fin de sesiones
- Hay ficheros en /var/log/ que **no** gestiona syslog
 - /wtmp Registra accesos de los usuarios y reinicios
 - Está en formato binario
 - Es utilizado por los comandos last y uptime
 - /lastlog Contiene el último acceso de cada usuario
 - ...



Gestión de logs

- Los logs son una herramienta fundamental para el control y reparación del sistema.

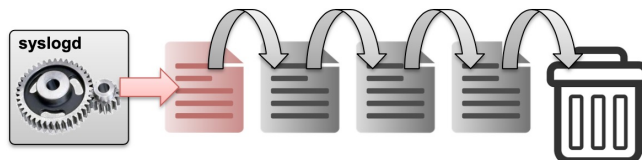
Pero...

- Cuanta más información de logs, mayor uso de disco.
 - Los logs pueden llegar a consumir un espacio significativo.
 - Puede ser costoso buscar información/datos concretos entre miles de líneas.



Gestión de logs

- Rotación de logs
 - Periódicamente cambiar el fichero donde se escriben los logs, cambiando a escribir en uno nuevo y borrando el más antiguo.



- Se puede hacer de manera manual con un script.
 - Ejemplo:

```
#!/bin/bash
cd /var/log/
mv messages.2 messages.3
mv messages.1 messages.2
mv messages messages.1
cat /dev/null > messages
chmod 600 messages

#Reiniciar syslog
service rsyslog restart
```

Gestión de logs

- Rotación de logs
 - Se puede implementar la rotación con un servicio del sistema.
 - Evita errores humanos al crear scripts.
 - Servicio **logrotate**
 - Se configura con los siguientes ficheros:

/etc/logrotate.conf
Por defecto, para todos los servicios

```
# rotate log files weekly, monthly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# compressed log files
compress
...
```

/etc/logrotate.d/
Sobrescribe logrotate.conf
para un servicio concreto

```
/var/log/dpkg.log {
    monthly
    rotate 12
    compress
    notifempty
    create 0664 root

adm
}
```



Gestión de logs

- Analizando logs
 - Para depuración:
 - Útil para obtener más información cuando algo va mal
 - Activar modo verboso de las aplicación
 - P.e. activar flag -d, /etc/init.d/ssh sshd -d
 - Importante: desactivar el modo verboso al volver a producción
 - Para monitorización:
 - Problema: abundante información, de la cual mucha puede no ser útil
 - Utilizar herramientas para buscar los mensajes relevantes, p.e.:
 - Swatch: Programa Perl que busca patrones en los logs¹
 - LogWatch: Genera resúmenes para su envío por e-mail.
 - Soluciones más complejas, p.e., pila ELK.



¹: Swatch: The Simple Log Watcher: <https://www.linuxtoday.com/blog/swatch-the-simple-log-watcher.html>

Ejercicio 4

- Enviar un mensaje al log del sistema desde el terminal.
 - Comprobar que se ha añadido correctamente.
- Añadir una regla a syslog: los mensajes de usuario de tipo "debug" se escriben en /var/log/ej3.log
- Enviar un mensaje de usuario de tipo debug y comprobar que se escribe en un /var/log/ej3.log
- Devolver syslog a su situación anterior
 - Eliminar la regla recién creada



Bibliografía

- Pablo Abad Fidalgo, José Ángel Herrero Velasco. “Advanced Linux System Administration”, OCW UNICAN, 2018:
 - Topic 8: Resource management & Topic 9: Logging
 - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
 - <https://ocw.unican.es/course/view.php?id=38>
- Dan Sullivan. “Google Cloud Associate Cloud Engineer: Get Certified”, Udemy, 2022.
 - <https://www.udemy.com/course/google-certified-associate-cloud-engineer-2019-prep-course/>
- MIT, The Lost Semester. Lecture 7 “Debugging and Profiling”
 - <https://missing.csail.mit.edu/2020/debugging-profiling/>
- Consultados en julio 2020, septiembre 2023 y septiembre 2025

