Linux Day 2023

systemd, un amico prezioso

28 ottobre 2023

Giuseppe Sacco <eppesuig@debian.org>

systemd, un amico prezioso

Cosa vedremo:
cos'è systemd
storia controversa dell'adozione
unit, servizi, *socket*, timer, rete, ...
modalità utente
qualche esempio

systemd, un amico prezioso

cos'è systemd

systemd, un amico prezioso

systemd è principalmente un sistema per la gestione dell'avvio dei computer, inoltre:

- gestisce vari *target* (vagamente simili ai *runlevel*),
- permette priorità tra i servizi attivi sui vari *target*,
- è integrato con udev,
- permette il controllo delle risorse,
- include un sistema di dipendenze sofisticato,
- gestisce i *mount point* globali e in *namespace* privati,
- gestisce i timer con calendari sofisticati,
- gestisce le postazioni/terminali,
- gestisce i file di log di sistema, ...

systemd, un amico prezioso

storia controversa della sua adozione

systemd, un amico prezioso

Nel 2010 i due autori — *Lennart Poettering* e *Kay Sievers* —, che lavoravano per Red Hat, volevano:

to improve the software framework for expressing dependencies, to allow more processing to be done concurrently or in parallel during system booting, and to reduce the computational overhead of the shell.

systemd, un amico prezioso

Poettering e *Sievers* sono due sviluppatori che avevano già operato a basso livello sviluppando udev, PulseAudio e Avahi, e cercando di portare cambiamenti radicali alla struttura del sistema. Sono due che hanno spesso trovato la via più diretta, lasciando perdere la compatibilità POSIX. Quindi, anche per il carattere degli autori, c'è stato inizialmente un grosso scontro a proposito di systemd.

Ma ....

systemd, un amico prezioso

… alla fine systemd è stato adottato da varie distribuzioni:

- 2010 Fedora Linux
- 2011 Arch Linux, OpenSUSE
- 2013 Manjaro Linux
- 2014 CentOS, Debian, Ubuntu, Red Hat, SUSE
- 2018 Linux Mint

Non è invece adottato da: Devuan, antiX, MX Linux, Nitrux, Void Linux

systemd, un amico prezioso

File di configurazione

systemd, configurazione

Tutto systemd si basa su file di configurazione in formato testo. Sono nella forma:

```
[sezione1]
parametro1=valore…
parametro2=valore…

[sezione2]
parametro3=valore…
```

systemd, configurazione

Questi file sono forniti con i pacchetti Linux per l'integrazione del pacchetto stesso con systemd. Ad esempio un server ftp avrà un file che dice a systemd come controllarlo. Non vanno confusi con i file di configurazione del pacchetto stesso, cioè del server ftp.

I file forniti dai pacchetti sono normalmente in `/usr/lib/systemd/system` e `/usr/lib/systemd/user`, ma vengono cercati anche in `/etc/systemd/system` e `/etc/systemd/user`. (E in altre directory, ma qui semplifico.)

systemd, configurazione

I file in `/usr/lib/systemd/`**`system`** sono quelli dei servizi non legati ad un utente particolare, invece quelli in `/usr/lib/systemd/`**`user`** sono quelli eseguiti dagli utenti.

Esempio: un server web avrà la sua configurazione in `/usr/lib/systemd/`**`system`** mentre il salvaschermo avrà la sua configurazione in `/usr/lib/systemd/`**`user`**. Se invece si tratta di processi privati di un solo utente, la configurazione andrà in `~/.config/systemd/user`.
Infine, le configurazioni dei programmi interni di systemd sono normalmente in `/etc/systemd`.

systemd, configurazione

Per modificare i file di configurazione si usa il meccanismo dei *drop-in*. Vale a dire che si crea un file ulteriore con solo i parametri da modificare o aggiungere.

Vantaggi del *drop-in*: se il pacchetto viene aggiornato con una nuova configurazione in `/usr/lib/systemd`, non si perde nessuna modifica locale, perché i file *drop-in* verrebbero aggiunti alla nuova configurazione anziché alla vecchia.

systemd, configurazione

Esempio di *drop-in* per il servizio interno `systemd-timesyncd` che ha la configurazione in `/etc/systemd/timesyncd.conf`. I file *drop-in* sono con estensione `conf` e si trovano in una directory che si chiama come il file stesso più un «`.d`» finale:

- `/etc/systemd/timesyncd.conf.d/*.conf`
- `/usr/lib/systemd/timesyncd.conf.d/*.conf`

systemd, configurazione

Un esempio con un *drop-in*:

/etc/systemd/timesyncd.conf :

```
[Time]
#NTP=
#FallbackNTP=0.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org
3.debian.pool.ntp.org
#RootDistanceMaxSec=5
#PollIntervalMinSec=32
#PollIntervalMaxSec=2048
#ConnectionRetrySec=30
#SaveIntervalSec=60
```

/etc/systemd/timesyncd.conf .d/debian.conf:

```
[Time]
NTP=0.it.pool.ntp.org 1.it.pool.ntp.org 2.it.pool.ntp.org 3.it.pool.ntp.org
```

systemd, un amico prezioso

Cosa abbiamo visto:
come mai è nato systemd
dove sono i file di configurazione
una prima occhiata alla sintassi

systemd, caratteristiche

Le *unit*

systemd, caratteristiche /1

unit: un file di configurazione che descrive uno tra: *service*, *socket*, *device*, *mount point*, *auto mount point*, *swap file* o partizione, *target*, *path* controllato, timer. (Più altri due che non elenco.) La sua sintassi è del tipo:

```
[Unit]
Description=Unit di esempio
Requires=aula-sys.service proiettore.service
[Install]
WantedBy=LinuxDay2023.target
```

systemd, caratteristiche /2

Target

una delle *unit* elencate è quella di tipo *target*.
Si tratta di un raccoglitore di altri servizi che saranno tutti attivi una volta raggiunto il *target* in oggetto.
Esempi di target: `graphical.target`, `local-fs.target`, `multi-user.target`, `network-online.target`, `emergency.target`

systemd, caratteristiche /3

Alcuni corrispondono a quelli di sysV:

```
$ ls -l /usr/lib/systemd/system/runlevel*
lrwxrwxrwx 1 root root 15 20 set 14.15 /usr/lib/systemd/system/runlevel0.target -> poweroff.target
lrwxrwxrwx 1 root root 13 20 set 14.15 /usr/lib/systemd/system/runlevel1.target -> rescue.target
lrwxrwxrwx 1 root root 17 20 set 14.15 /usr/lib/systemd/system/runlevel2.target -> multi-user.target
lrwxrwxrwx 1 root root 17 20 set 14.15 /usr/lib/systemd/system/runlevel3.target -> multi-user.target
lrwxrwxrwx 1 root root 17 20 set 14.15 /usr/lib/systemd/system/runlevel4.target -> multi-user.target
lrwxrwxrwx 1 root root 16 20 set 14.15 /usr/lib/systemd/system/runlevel5.target -> graphical.target
lrwxrwxrwx 1 root root 13 20 set 14.15 /usr/lib/systemd/system/runlevel6.target -> reboot.target
```

systemd, caratteristiche /4

Ma sono molti di più:

```
$ ls /usr/lib/systemd/system/*target
/usr/lib/systemd/system/basic.target                /usr/lib/systemd/system/printer.target
/usr/lib/systemd/system/blockdev@.target            /usr/lib/systemd/system/reboot.target
/usr/lib/systemd/system/bluetooth.target            /usr/lib/systemd/system/remote-cryptsetup.target
/usr/lib/systemd/system/boot-complete.target        /usr/lib/systemd/system/remote-fs-pre.target
/usr/lib/systemd/system/cryptsetup-pre.target       /usr/lib/systemd/system/remote-fs.target
/usr/lib/systemd/system/cryptsetup.target           /usr/lib/systemd/system/remote-veritysetup.target
/usr/lib/systemd/system/ctrl-alt-del.target         /usr/lib/systemd/system/rescue-ssh.target
/usr/lib/systemd/system/default.target              /usr/lib/systemd/system/rescue.target
/usr/lib/systemd/system/emergency.target            /usr/lib/systemd/system/rpcbind.target
[…]
/usr/lib/systemd/system/network.target              /usr/lib/systemd/system/umount.target
/usr/lib/systemd/system/nfs-client.target           /usr/lib/systemd/system/usb-gadget.target
/usr/lib/systemd/system/nss-lookup.target           /usr/lib/systemd/system/veritysetup-pre.target
/usr/lib/systemd/system/nss-user-lookup.target      /usr/lib/systemd/system/veritysetup.target
/usr/lib/systemd/system/paths.target                /usr/lib/systemd/system/virt-guest-shutdown.target
/usr/lib/systemd/system/poweroff.target             /usr/lib/systemd/system/wg-quick.target

$ ls -l /usr/lib/systemd/system/default.target
lrwxrwxrwx 1 root root 16 20 set 14.15 /usr/lib/systemd/system/default.target -> graphical.target
```

systemd, caratteristiche /5

Esempio, `graphical.target`:

```
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target
display-manager.service
AllowIsolate=yes
```

Domanda bonus: Come mai si usa `display-manager.service` e non la unit specifica per gdm3, lightdm, xdm, ecc?

systemd, caratteristiche /6

`Requires=multi-user.target`

- oltre alla *unit* attuale vanno avviate quelle in Requires
- se viene fermata la *unit* elencata, viene fermata pure questa

`Wants=display-manager.service`

- oltre alla *unit* attuale vanno avviate quelle in Wants

`Conflicts=rescue.service rescue.target`

- questa *unit* va spenta se si avvia una di quelle *unit* e viceversa

`After=multi-user.target rescue.service ...`

- questa *unit* parte dopo che quelle elencate sono avviate, se sono da avviare

systemd, un amico prezioso

Servizi

un file di configurazione che termina in `.service`
contiene informazioni su un processo controllato da systemd

systemd, servizi /1

Esempio, `fio.service`:

```
[Unit]
Description=Flexible I/O Tester as service
After=network.target
Documentation=man:fio(1)

[Service]
Type=forking
PIDFile=/run/fio.pid
ExecStart=/usr/bin/fio --server --daemonize /run/fio.pid

[Install]
WantedBy=multi-user.target
```

systemd, servizi /2

`Type=forking`

- indica come avviene internamente l'avvio della *unit*. `forking` vuol dire che il comando invocato uscirà subito dopo aver avviato un altro processo che rimarrà attivo come servizio.

`ExecStart=/usr/bin/fio --server --daemonize /run/fio.pid`

- è il comando da eseguire. Ci possono essere più righe di questo tipo e vengono tutte eseguite

`[Install]`

- questa *unit* va collegata come `Wants=` al *target* indicato. Se si esegue `systemctl enable unit.service` allora viene fatto il *link* da `/etc/systemd/system/multi-user.target.wants/`

systemd, servizi /3

Altro esempio di un servizio che fa un ping continuo:

```
[Unit]
Description=ping che attraversa il tunnel openvpn verso ufficio e lo tiene attivo
After=openvpn-client@ufficio.service
Requires=openvpn-client@ufficio.service

[Service]
Type=simple
ExecStart=/bin/ping -i 55 -4 192.168.234.95
StandardOutput=null
PrivateDevices=yes
PrivateTmp=yes
ProtectHome=yes
ProtectSystem=full
RestrictAddressFamilies=AF_INET
NoNewPrivileges=yes
IPAddressAllow=192.168.234.0/24
IPAddressDeny=any

[Install]
WantedBy=multi-user.target
```

systemd, servizi /4

`Type=simple`

- indica come avviene l'avvio della *unit*. `simple` vuol dire che la *unit* risulterà attiva appena verrà eseguito il comando.

`StandardOutput=null`

- Indica che l'output del comando non deve andare nel log

`PrivateDevices=yes e PrivateTmp=yes`

- Monta dei file system `/dev` e `/tmp` vuoti presi da tmpfs

`ProtectHome=yes e ProtectSystem=full`

- Monta i file system `/home /usr /boot /efi` e `/etc` in sola lettura

systemd, servizi /5

`RestrictAddressFamilies=AF_INET`

- questo servizio potrà solo aprire socket IPv4

`IPAddressAllow=192.168.74.35/24 e IPAddressDeny=any`

- indica che questo servizio potrà solo comunicare verso gli indirizzi IPv4 di questa LAN

`NoNewPrivileges=yes`

- Indica che il servizio non potrà ottenere privilegi ulteriori, ad esempio eseguendo programmi con il bit setuig

Per la sicurezza usare: `systemd-analyze security` *`unit`*`.service`

systemd, un amico prezioso

Cosa abbiamo visto:
cos'è un *target*
cos'è una *unit* di tipo service
alcuni semplici file di *unit*
controllo delle risorse

systemd, un amico prezioso

Mount e automount

gestione dei *file system* e dei relativi *mount point*

(con una piccola digressione su generatori e modelli)

systemd, mount /1

Systemd gestisce i *mount i point* e i *file system*

- può montare un *file system* all'avvio
- può montare un *file system* solo quando usato
- può smontare un *file system* quando non usato
- può creare un *file system* all'avvio
- può gestire anche lo *swap*
- può allargare un *file system* all'avvio

systemd, mount /2

Systemd ha un generatore che legge `/etc/fstab` e crea le *unit* per i *file system*: `systemd-fstab-generator`. Si possono inserire parametri per questo generatore in `/etc/fstab`, ad esempio `x-systemd.before`, `x-systemd.automount` e `x-systemd.idle-timeout`

Per vedere le *unit* generate:

```
$ systemctl list-units --type=mount
```

Per dettagli sulla singola *unit*:

```
$ systemctl status home.mount
$ sudo journalctl --unit home.mount
```

systemd, mount /3

Tutte le *unit* di tipo `.mount` definiscono un *file system* e sono create dal generatore oppure fatte a mano. Ci sono anche le *unit* `.automount` che vogliono un *mount point* da controllare e che montano il *file system* solo quando qualcuno va a cercarvi i file o a scriverci. Dopo un certo periodo il volume viene nuovamente smontato.

Ad esempio, in molti sistemi universitari le home degli utenti sono montate solo quando l'utente proprietario vi accede.

systemd, mount /3

Analogamente a quello che genera *unit* a partire da `/etc/fstab`, ci sono altri generatori:

```
$ ls -l /usr/lib/systemd/system-generators
totale 468
-rwxr-xr-x 1 root root 84536 11 gen  2023 nfs-server-generator
-rwxr-xr-x 1 root root   899 21 giu 21.41 openvpn-generator
-rwxr-xr-x 1 root root  1036 14 set 10.35 postgresql-generator
-rwxr-xr-x 1 root root 39264 11 gen  2023 rpc-pipefs-generator
-rwxr-xr-x 1 root root 31328 18 mag 11.00 snapd-generator
-rwxr-xr-x 1 root root 35336 20 set 14.15 systemd-cryptsetup-generator
-rwxr-xr-x 1 root root 14792 20 set 14.15 systemd-debug-generator
-rwxr-xr-x 1 root root 43616 20 set 14.15 systemd-fstab-generator
-rwxr-xr-x 1 root root 22832 20 set 14.15 systemd-getty-generator
-rwxr-xr-x 1 root root 31032 20 set 14.15 systemd-gpt-auto-generator
-rwxr-xr-x 1 root root 14776 20 set 14.15 systemd-hibernate-resume-generator
-rwxr-xr-x 1 root root 22936 20 set 14.15 systemd-integritysetup-generator
-rwxr-xr-x 1 root root 14640 20 set 14.15 systemd-rc-local-generator
-rwxr-xr-x 1 root root 14776 20 set 14.15 systemd-run-generator
-rwxr-xr-x 1 root root 14640 20 set 14.15 systemd-system-update-generator
-rwxr-xr-x 1 root root 31096 20 set 14.15 systemd-sysv-generator
-rwxr-xr-x 1 root root 31248 20 set 14.15 systemd-veritysetup-generator
```

systemd, mount /3

Ad esempio, quello di openvn crea una *unit* di tipo service per ciascuna configurazione che si trova in `/etc/openvpn`. Per distinguerle, le *unit* si chiameranno `openvpn@`*`nomeconf`*`.service` e saranno tutti link ad un modello di *unit* chiamato `openvpn@.service` che utilizza **`%i`** al posto del nome della configurazione (i sta per istanza):

```
$ cat /usr/lib/systemd/system/openvpn@.service
[...]
[Service]
Type=notify
PrivateTmp=true
WorkingDirectory=/etc/openvpn
ExecStart=/usr/sbin/openvpn --daemon ovpn-%i --status /run/openvpn/%i.status 10 --cd
/etc/openvpn --config /etc/openvpn/%i.conf --writepid /run/openvpn/%i.pid
PIDFile=/run/openvpn/%i.pid
[...]
```

systemd, un amico prezioso

Cosa abbiamo visto:

unit `mount` e `automount`

generatori di *unit*

modelli di *unit*

systemd, un amico prezioso

Socket

usare systemd per ascoltare su un socket e poi attivare la *unit* corrispondente

systemd, socket /1

Un socket è un canale di comunicazione, come ad esempio quelli TCP, forse i più conosciuti, o le *named pipe*. Su questo canale di norma c'è un processo *server* che riceve le comunicazioni dai *client* e ne soddisfa le richieste.

systemd, socket /2

Una *unit* che deve ascoltare su un socket, alla partenza fa tante cose e solo dopo un po' si mette in attesa di una richiesta, ma non è detto che questa arrivi subito. Intanto però consuma RAM e CPU.

Systemd cerca di fare in modo che le *unit* vengano attivate solo quando c'è veramente una richiesta da parte di un *client*.

systemd, socket /3

Systemd permette di fare una *unit* di tipo `.socket` corrispondente alla *unit* di tipo `.service`. La seconda non viene attivata subito e la prima indica a systemd che tipo di socket creare. Una volta ricevuta una connessione, systemd avvia la *unit* di tipo `.service` e le passa la comunicazione.

systemd, socket /4

Questo ha permesso a systemd di accelerare molto l'avvio delle macchine, difatti con sysV si doveva attendere che tutti i *daemon* fossero attivi, mentre ora vengono attivati solo alla bisogna.

systemd, socket /5

Questo modo di operare è detto *socket activation* e va previsto nell'applicazione *server*, difatti questa anziché mettersi in ascolto sul socket, deve collaborare con systemd per ricevere le richieste.
In passato un meccanismo simile era quello di inetd: le applicazioni demandavano a inetd la gestione del socket. Systemd simula inetd, quindi le applicazioni che usavano inetd sono già usabili con systemd. Per le altre c'è `systemd-socket-proxyd`.

systemd, socket /6

Facciamo un esempio: attiviamo un *server web* come utente normale quando viene fatta una richiesta sulla porta TCP 4080.
Il server web che viene usato nell'esempio è nginx.

systemd, socket /7

File di configurazione ~/.config/nginx/nginx.conf:

```
worker_processes auto;
pid /home/giuseppe/.config/log/nginx/giuseppe.pid;
error_log /home/giuseppe/.config/log/nginx/error.log;

events {
     worker_connections 768;
}

http {
     server {
          listen [::]:4080 ipv6only=on;
          listen 4080;
     }
     root /home/giuseppe/public_html;
     access_log /home/giuseppe/.config/log/nginx/access.log;
}
```

systemd, socket /8

Unit per systemd `~/.config/systemd/user/nginx.service`:

```
[Unit]
Description=Server web nginx configurato in ~/.config/nginx/nginx.conf

[Service]
Environment=NGINX=3:4;
ExecStart=/usr/sbin/nginx -c /home/giuseppe/.config/nginx/nginx.conf
WorkingDirectory=~
```

systemd, socket /9

Unit per systemd `~/.config/systemd/user/nginx.socket`:

```
[Unit]
Description=Socket per avviare nginx solo alla prima richiesta
[Socket]
# ascolta con un socket IPv6
ListenStream=4080
# ascolta con un socket IPv4
ListenStream=0.0.0.0:4080
# indica che il socket IPv6 non va utilizzato anche per le
# connessioni IPv4
BindIPv6Only=ipv6-only
After=network-online.target
Requires=network-online.target

[Install]
WantedBy=sockets.target
```

systemd, socket /10

```
$ systemctl --user daemon-reload
$ systemctl --user start nginx.socket
$ journalctl --user --unit nginx.service --follow &
$ time curl http://localhost:4080/pagina.html
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.22.1</center>
</body>
</html>

real 0m0,038s
user 0m0,008s
sys  0m0,000s

$ ott 17 15:06:55 uefi systemd[3585]: Started nginx.service.
ott 17 15:06:55 uefi nginx[63092]: 2023/10/17 15:06:55 [notice] 63092#63092: using inherited
sockets from "3:4;"
```

systemd, un amico prezioso

Cosa abbiamo visto:
unit di tipo `socket`
socket activation

systemd, servizi integrati /1

DNS
tra i servizi integrati in systemd c'è quello della risoluzione dei nomi.

systemd, servizi integrati /2

Il servizio si chiama `systemd-resolved` e permette di comunicare direttamente con i root DNS, oppure con i DNS locali.
Nel caso di DNS locali, può gestire DNS diversi collegati alle diverse interfacce di rete. Ad esempio, se ti attiva una VPN che da accesso alle macchine del dominio `ufficio.rete`, si può dire a `systemd-resolved` che per risolvere `nome.ufficio.rete` deve passare dal DNS ottenuto dal server di questa VPN.

systemd, servizi integrati /3

Oltre alla configurazione nel file `/etc/systemd/resolved.conf` e in eventuali *drop-in*, c'è il comando `resolvectl` che permette di dare parametri aggiuntivi.
Per utilizzarlo si deve inserire il DNS `127.0.0.53` in `/etc/resolv.conf`, cosa che avviene in automatico

systemd, un amico prezioso

Cosa abbiamo visto:
ci sono servizi forniti direttamente da systemd
uno di questi è il DNS
possono essere configurati con *drop-in*
hanno dei comandi aggiuntivi

systemd, un amico prezioso

I servizi integrati sono molti, tra i quali:

- systemd-bootchart: genera grafici sulla sequenza di avvio
- systemd-hostnamed: gestisce il nome della macchina e dei vari container
- systemd-logind: gestisce le sessioni utente
- systemd-networkd: gestisce le schede di rete e la loro configurazione
- systemd-sleep: gestisce l'ibernazione dei portatili
- systemd-timedated: gestisce data e fuso orario

systemd, journal

Journal

gestione dei file di log, dello standard output e standard error

systemd, journal /1

Systemd gestisce un file di log unico, in un formato non testuale, che viene memorizzato in `/run/log/journal` nel caso non sopravviva al boot, oppure in `/var/log/journal` se configurato per essere persistente.

Ci sono un file per il sistema e uno per ciascun utente. Per ciascuno di essi, oltre al file corrente ne vengono mantenuti alcuni precedenti.

I file sono firmati, compressi, indicizzati, con metadati
La configurazione è in `/etc/systemd/journald.conf`.
La *unit* che se ne occupa è `systemd-journald.service`.

systemd, journal /2

Per vedere cosa c'è nel log conviene limitarsi ad una *unit*. Ad esempio, per vedere le ultime righe del log di sshd:

```
$ sudo journalctl --unit ssh --all
```

Per vederle man mano che vengono aggiunte:

```
$ sudo journalctl --unit ssh --follow
```

Per vedere i messaggi di un intervallo di tempo:

```
$ sudo journalctl --unit ssh --since "2023-10-28 10:17:16"
--until "2023-10-28 15:40:16"
```

Per cercare un problema che può essere in due diverse *unit*

```
$ sudo journalctl --unit ssh --unit openvpn --since 10:00
```

systemd, un amico prezioso

Cosa abbiamo visto:
dove viene memorizzato il log
come interrogare il log

systemd, un amico prezioso

Cosa non abbiamo visto:

- i timer
- le sessioni utente
- le postazioni
- i container
- la gestione delle interfacce di rete
- il controllo delle risorse

Grazie a tutti

Giuseppe Sacco <giuseppe@sguazz.it>
Trovate questa presentazione su https://www.sguazz.it/~giuseppe/LinuxDay2023.pdf