systemd

systemd è un gestore di sistema e di servizi per Linux, compatibile con gli initscript SysV e LSB. **systemd** fornisce una notevole capacità di parallelizzazione, usa socket e D-Bus per l'avvio dei demoni, offre un avvio su richiesta dei demoni, tiene traccia dei processi con l'utilizzo del control groups di Linux, supporta lo snapshotting e il restore dello stato del sistema, mantiene i punti di mount e di automount e implementa un elaborato servizio di controllo logico basato sulle relazioni delle dipendenze.

Contents

Uso base di systemctl

Il principale comando usato per controllare systemd è `systemctl`. Alcuni degli utilizzi possibili sono l'esame dello stato del sistema e la gestione del sistema e dei servizi. Vedere `man 1 systemctl` per maggiori dettagli

Suggerimento: Si può usare il seguente comando `systemctl` con lo switch `-H <user>@<host>` per controllare un'istanza di systemd su una macchina remota. Si userà SSH per connettersi all'istanza remota di systemd.

Nota: `systemadm` è il frontend grafico ufficiale per `systemctl`. E' fornito dal pacchetto systemd-ui-git AUR di AUR .

Analizzare lo stato del sistema

Lista della unità attive:

```
$ systemctl
```

oppure:

```
$ systemctl list-units
```

Lista delle unità che hanno avuto problemi:

```
$ systemctl --failed
```

I file delle unità disponibili possono essere visti in `/usr/lib/systemd/system/` e `/etc/systemd/system/` (questi ultimi hanno la precedenza sui primi). Si possono vedere le unità installate con:

```
$ systemctl list-unit-files
```

Usare le unità

Le unità possono essere, per esempio, servizi (`.service`), punti di montaggio (`.mount`), dispositivi (`.device`) oppure i sockets (`.socket`).

Quando si usa `systemctl`, occorre generalmente specificare sempre il nome completo dell'unità compreso il suffisso, per esempio `sshd.socket`. Esistono tuttavia delle scorciatoie quando si specifica l'unità nei seguenti comandi `systemctl`:

- Se non si specifica il suffisso, per systemctl sarà sottointeso `.service`. Per esempio, `netcfg` e `netcfg.service` sono equivalenti.
- I punti di montaggio saranno automaticamente tradotti nella appropriata unità `.mount`. Per esempio, specificare `/home` è equivalente a `home.mount`.
- Come i punti di montaggio, anche i dispositivi sono automaticamente tradotti nell'appropriata unità `.device`, quindi specificare `/dev/sda2` è equivalente a `dev-sda2.device`.

Vedi `man systemd.unit` per dettagli.

Attivare immediatamente una unità:

```
# systemctl start <unit>
```

Fermare immediatamente una unità:

```
# systemctl stop <unit>
```

Far ripartire una unità:

```
# systemctl restart <unit>
```

Chiedere ad una unità di ricaricare la sua configurazione:

```
# systemctl reload <unit>
```

Mostrare lo stato di una unità, compreso se sta funzionando o no:

```
$ systemctl status <unit>
```

Controllare se una unità è già attivata o no:

```
$ systemctl is-enabled <unit>
```

Attivare l'avvio automatico al boot:

```
# systemctl enable <unit>
```

Nota: I servizi senza una sezione `[Install]`, sono solitamente evocati automaticamente da altri servizi. Ma se occorre installarli manualmente usare il seguente comando rimpiazzando `foo` con il nome del servizio.

```
# ln -s /usr/lib/systemd/system/"foo".service
/etc/systemd/system/graphical.target.wants/
```

Disattivare l'avvio automatico al boot:

```
# systemctl disable <unit>
```

Mostra la pagina del manuale associata a una unità (non supportato dai files .unit):

```
$ systemctl help <unit>
```

Ricaricare systemd, controllo per nuove o modificate unità:

```
# systemctl daemon-reload
```

Power management

`polkit` è indispensabile per la gestione energetica. Se ci si trova in una sessione locale di `systemd-logind` e non ci sono altre sessioni attive, il seguente comando funzionerà senza i privilegi di root. Altrimenti (per esempio se un altro utente è loggato in una console), systemd automaticamente richiederà la password di root.

Spegnimento e riavvio del sistema:

```
$ systemctl reboot
```

Spegnimento del sistema:

```
$ systemctl poweroff
```

Sospensione del sistema:

```
$ systemctl suspend
```

Ibernazione del sistema:

```
$ systemctl hibernate
```

Stato ibrido di riposo (o suspend-to-both):

```
$ systemctl hybrid-sleep
```

Avviare DM con systemd

Per avviare il login grafico, abilitare il demone del suo Display Manager corrispondente (per esempio KDM). Attualmente esistono i .service per GDM, KDM, SLiM, XDM and LXDM e LightDM.

```
# systemctl enable kdm
```

Questo dovrebbe funzionare. Se non dovesse, potrebbe essere ancora presente l'impostazione di default.target di una vecchia installazione:

```
# ls -l /etc/systemd/system/default.target
```

```
/etc/systemd/system/default.target -> /usr/lib/systemd/system/graphical.target
```

è sufficiente eliminare il link simbolico e systemd utilizzerà il default.target predefinito (per esempio `graphical.target`).

```
# rm /etc/systemd/system/default.target
```

Usare systemd-logind

Per controllare lo stato della propria sessione utente, si può utilizzare `loginctl`. Tutte le azioni di PolicyKit come la sospensione del sistema o il montaggio di dispositivi esterni funzioneranno automaticamente.

```
$ loginctl show-session $XDG_SESSION_ID
```

Configurazione nativa

Nota: E' possibile che ci sia bisogno di creare questi files. Tutti i file devono avere i permessi a `644` e il proprietario `root:root`

Hostname

L'hostname è configurato in `/etc/hostname`. Il file può contenere il nome del dominio di sistema, se esiste, tuttavia nel momento di scrivere `hostnamectl` non si può configurare il FQDN. Per configurare l'hostname corto:

```
# hostnamectl set-hostname myhostname
```

See `man 5 hostname` e `man 1 hostnamectl` per dettagli.

Esempio di file:

```
/etc/hostname
```

```
myhostname
```

Impostazioni locali

Note: Before you set the default locale, you first need to enable locales available to the system by uncommenting them in `/etc/locale.gen` and then executing `locale-gen` as root. The locale set via `localectl` must be one of the **uncommented** locales in `/etc/locale.gen`.

Le impostazioni locali di default sono configurate in `/etc/locale.conf` Per configurare il locale di default:

```
# localectl set-locale LANG="it_IT.utf8"
```

Vedere `man 1 localectl` e `man 5 locale.conf` per dettagli.

Per ulteriori informazioni vedere Locale.

Esempio di file:

`/etc/locale.conf`

```
LANG=it_IT.utf8
```

Console virtuale

La console virtuale (la mappatura della tastiera, i caratteri della console e la mappatura della console) è configurata in `/etc/vconsole.conf`:

`/etc/vconsole.conf`

```
KEYMAP=it
FONT=
FONT_MAP=
```

Nota: Da systemd -194, si utilizzano i caratteri integrati nel kernel e la mappatura della tastiera "us" se `KEYMAP=` e `FONT=` sono vuoti o non configurati.

Un altro modo di configurare la mappatura della tastiera è:

```
# localectl set-keymap it
```

`localectl` può essere usato anche per configurare la tastiera in ambiente X:

```
# localectl set-x11-keymap it
```

Vedere `man 1 localectl` e `man 5 vconsole.conf` per maggiori dettagli.

Per maggiori informazioni vedere i font per console e configurazione tastiera.

Orario di sistema

Systemd usa **UTC** di default per l'orologio hardware.

Suggerimento: E' generalmente consigliato avere il demone Network Time Protocol attivo per mantenere l'orologio di sistema sincronizzato con Internet e con l'orologio del bios.

Orologio hardware in localtime

Se si vuole cambiare l'orologio hardware ad usare l'orario locale (**ALTAMENTE SCONSIGLIATO**):

```
# timedatectl set-local-rtc true
```

Se si vuole riconvertirlo a UTC:

```
# timedatectl set-local-rtc false
```

Attenzione che, se l'orologio hardware è configurato a localtime, la gestione dell'orario legale è persa. Se l'orario legale cambia finché il computer è spento il proprio orario sarà sbagliato al prossimo riavvio (ulteriori approfondimenti). I kernel recenti configurano l'orario di sistema dall'orologio hardware direttamente al boot, presupponendo che l'orologio hardware sia impostato a UTC. Questo significa che se l'orologio hardware è su locale l'orologio di sistema sarà prima configurato sbagliato e poi corretto ad ogni boot. Questa è la causa di alcuni noiosi bugs (il tempo che torna indietro è raramente una buona cosa).

Una ragione per permettere che l'orologio hardware sia settato sull'orario locale è quella del dual boot con Windows (il quale usa localtime). Tuttavia, Windows è in grado di gestire l'orario con l'orologio hardware settato su UTC con un semplice aggiustamento del registro di sistema. Quindi, è consigliato configurare Windows affinché usi UTC, piuttosto che Linux usi localtime. Se si usa UTC su Windows, ricordarsi inoltre di disabilitare l'opzione di Windows "Internet Time Update", al fine di evitare confusione tra windows e l'orologio hardware, nel tentativo di sincronizzazione con l'orario internet. Al contrario, occorre lasciare che Linux sincronizzi l'orario di sistema con l'orario internet attivando il demone NTP, come suggerito precedentemente.

- Per maggiori informazioni vedere Time.

Moduli del Kernel

Ad oggi, tutti i moduli necessari da caricare sono gestiti automaticamente da udev, cosicché, se non si vuole usare qualcuno dei moduli fuori dal kernel, non c'è bisogno di mettere i moduli che dovrebbero essere caricati al boot in nessun file di configurazione. Tuttavia, ci sono casi in cui si vuole caricare un modulo extra durante il processo di avvio oppure evitare il caricamento di un altro perché il computer funzioni correttamente.

Moduli extra da caricare all'avvio

I moduli extra al kernel da caricare durante il boot sono configurati in una lista statica in `/etc/modules-load.d/`. Ogni file di configurazione ha il nome nello stile di `/etc/modules-load.d/<programma>.conf/`. I files di configurazione dovrebbero semplicemente contenere una lista con i nomi dei moduli da caricare, separati dal tasto INVIO. Le linee vuote o quelle dove il primo carattere che non è uno spazio è # oppure ; sono ignorate. Ad

esempio:

```
/etc/modules-load.d/virtio-net.conf

# Carica virtio-net.ko al boot
virtio-net
```

vedere `man 5 modules-load.d` per maggiori dettagli.

Configurare le opzioni dei moduli

Ulteriori opzioni per i moduli devono essere impostati in `/etc/modprobe.d/modprobe.conf`.

Esempio:

- Abbiamo il file `/etc/modules-load.d/loop.conf` con all'interno il modulo `loop` da far caricare durante la fase di boot.
- in `/etc/modprobe.d/modprobe.conf` specifichiamo le opzioni addizionali, es. `options loop max_loop=64`

In seguito, l'opzione appena impostato potrebbe essere verificata tramite `cat /sys/module/loop/parameters/max_loop`

Blacklisting

Per evitare il caricamento di alcuni moduli del kernel si utilizza la stessa modalità degli initscripts in quanto è attualmente gestito da kmod. Vedere Blacklist dei moduli per maggiori dettagli.

Montaggio del filesystem

La configurazione di default controlla e monta i filesystems prima di avviare i servizi ai quali occorre che siano montati. Per esempio, systemd automaticamente si assicura che il montaggio di filesystems come NFS o Samba avvenga solo dopo che la rete è attiva. Pertanto, il montaggio dei filesystems, sia locali che remoti, specificati in `/etc/fstab` dovrebbe funzionare senza problemi.

Vedere `man 5 systemd.mount` per dettagli.

Automount

- Se si possiede una grande partizione `/home`, sarebbe meglio permettere ai servizi che non dipendono da essa di avviarsi durante il controllo di `/home` da parte di fsck. Ciò può essere ottenuto aggiungendo la seguente opzione alla riga della partizione di `/home` in `/etc/fstab`:

```
noauto,x-systemd.automount
```

Questo controllerà e monterà `/home` al primo accesso, e il kernel memorizzerà in un buffer tutti gli accessi ai file in `/home` finché non sarà pronta.

Nota: questo renderà il filesystem `/home` di tipo `autofs`, che è ignorato da mlocate di default. La velocità di montaggio automatico di `/home` non può essere maggiore di un secondo o due, a

seconda del proprio sistema, quindi è possibile che non valga la pena di utilizzare questo trucco.

- Lo stesso si può applicare al montaggio dei filesystems remoti. Se si desidera montarli successivamente all'accesso occorrerà usare i parametri `noauto,x-systemd.automount`. In aggiunta, si può usare l'opzione `x-systemd.device-timeout=#` per specificare un tempo di timeout in caso la risorsa di rete non sia disponibile.
- Se si hanno filesystems criptati con keyfiles, si può comunque aggiungere il parametro `noauto` alla corrispondente riga in `/etc/crypttab`. Systemd non aprirà il supporto corispondente all'avvio, ma aspetterà finché non avverrà un accesso e quindi automaticamente aprirà il keyfile specifico prima di montarlo. Questo potrebbe far risparmiare un po' di secondi al boot se si sta per esempio utilizzando un supporto RAID criptato, in quanto systemd non deve aspettare finché il dispositivo non sarà disponibile. Per esempio:

`/etc/crypttab`

```
data /dev/md0 /root/key noauto
```

LVM

Se si possiede un volume LVM non attivato durante l'initramfs, attivare il servizio `lvm-monitoring` fornito dal pacchetto lvm2:

```
# systemctl enable lvm-monitoring.service
```

ACPI power management

Systemd gestisce degli eventi ACPI relativi all'alimentazione. Possono essere configurati attraverso le seguenti opzioni di `/etc/systemd/logind.conf`:

- `HandlePowerKey` : specifica che azione deve essere eseguita quando viene premuto il bottone di avvio
- `HandleSuspendKey` : specifica che azione deve essere eseguita quando viene premuto il bottone di sospensione
- `HandleHibernateKey` : specifica che azione deve essere eseguita quando viene premuto il bottone di ibernazione
- `HandleLidSwitch` : specifica che azione deve essere eseguita quando il coperchio del portatile viene chiuso.

Le azioni specificate possono essere una qualsiasi di `ignore`, `poweroff`, `reboot`, `halt`, `suspend`, `hibernate`, `hybrid-sleep` o `kexec`.

Se queste opzioni non sono configurate, systemd userà quelle di default: `HandlePowerKey=poweroff`, `HandleSuspendKey=suspend`, `HandleHibernateKey=hibernate`, and `HandleLidSwitch=suspend`.

Su sistemi che non hanno una configurazione grafica o semplici gestori di finestre come i3 o

awesome, può rimpiazzare il demone acpid che è solitamente usato per gestire questi eventi ACPI.

Nota: Digitare `systemctl restart systemd-logind.service` perchè le modifiche abbiano effetto.

Nota: Systemd non riesce a gestire gli eventi di alimentazione di rete e di batteria, sicché è ancora richiesto l'uso di Laptop Mode Tools o altri strumenti acpid similari.

Nell'attuale versione di systemd, le opzioni `Handle*` saranno applicate al sistema finché non saranno "inibite" (temporaneamente spente) da un programma, come un gestore di alimentazione in un DE. Se questi inibitori non sono usati, si può finire in una situazione in cui systemd sospende il sistema , poi al risveglio gli altri gestori di alimentazione lo sospendono di nuovo.

Attenzione: Attualmente, i gestori di alimentazione delle versioni più recenti di KDE e GNOME sono gli unici che possiedono i necessari comandi inibitori. Quando li avranno anche gli altri, occorrerà settare l'opzione `Handle` a `ignore` se si voglio gestire gli eventi ACPI con Xfce , acpid o qualsiasi altro programma.

Nota: Systemd può anche usare altri backends per la sospensione (come Uswsusp o TuxOnIce), in aggiunta al backend di default del *kernel*, al fine di mettere il computer in uno stato di sleep o ibernazione.

For `systemctl hibernate` to work on your system you need to follow instructions at Hibernation and possibly at Mkinitcpio Resume Hook (`pm-utils` does not need to be installed).

Sleep hooks

Systemd non usa pm-utils per mettere la macchina a riposo quando usa `systemctl suspend`, `systemctl hibernate` oppure `systemctl hybrid-sleep`, quindi gli hooks di Pm-utils incluso qualsiasi hook personalizzato si sia creato non funzioneranno. Tuttavia, systemd fornisce due meccanismi simili per avviare script personali in base a questi eventi.

Suspend/resume service files

Dei service files possono essere agganciati a suspend.target, hibernate.target e sleep.target per eseguire azioni prima o dopo la sospensione lo l'ibernazione. Files separati dovrebbero essere creati per azioni degli utenti e azioni di sistema (root). Per attivare i servizi degli utenti, `# systemctl enable suspend@<user> && systemctl enable resume@<user>`. Esempi:

`/etc/systemd/system/suspend@.service`

```
[Unit]
Description=User suspend actions
Before=sleep.target

[Service]
```

```
User=%I
Type=forking
Environment=DISPLAY=:0
ExecStartPre= -/usr/bin/pkill -u %u unison ; /usr/local/bin/music.sh stop ;
/usr/bin/mysql -e 'slave stop'
ExecStart=/usr/bin/sflock

[Install]
WantedBy=sleep.target
```

```
/etc/systemd/system/resume@.service

[Unit]
Description=User resume actions
After=suspend.target

[Service]
User=%I
Type=simple
ExecStartPre=/usr/local/bin/ssh-connect.sh
ExecStart=/usr/bin/mysql -e 'slave start'

[Install]
WantedBy=suspend.target
```

Per azioni di sistema o root (attivare con `# systemctl enable root-suspend`):

```
/etc/systemd/system/root-resume.service

[Unit]
Description=Local system resume actions
After=suspend.target

[Service]
Type=simple
ExecStart=/usr/bin/systemctl restart mnt-media.automount

[Install]
WantedBy=suspend.target
```

```
/etc/systemd/system/root-suspend.service

[Unit]
Description=Local system suspend actions
Before=sleep.target

[Service]
Type=simple
ExecStart=-/usr/bin/pkill sshfs

[Install]
WantedBy=sleep.target
```

Un paio di consigli a proposito di questi servizi (maggiori informazioni in `man systemd.service`):

- Se si usa `Type=OneShot` si possono usare linee `ExecStart=` multiple. Altrimenti solo una linea ExecStart è permessa. Si possono aggiungere più comandi con `ExecStartPre` oppure separandoli con un punto e virgola (vedere il primo esempio più su -- notare gli spazi

prima e dopo il punto e virgola... sono indispensabili!).

- Un comando preceduto da '-' causerà una uscita con stato non-zero per essere ignorato e trattato come un comando successivo.
- Il miglior modo per trovare gli errori in questi servizi è naturalmente con `journalctl`.

Servizi combinati di Suspend/resume

Con il servizio combinato suspend/resume, un singolo collegamento fa tutto il lavoro per le varie fasi (sleep/resume) e per i differenti targets (suspend/hibernate/hybrid-sleep).

Esempi e spiegazioni:

`/etc/systemd/system/wicd-sleep.service`

```
[Unit]
Description=Wicd sleep hook
Before=sleep.target
StopWhenUnneeded=yes

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=-/usr/share/wicd/daemon/suspend.py
ExecStop=-/usr/share/wicd/daemon/autoconnect.py

[Install]
WantedBy=sleep.target
```

- `RemainAfterExit=yes`: Dopo averlo avviato, il servizio è considerato attivo fino a quando non è esplicitamente fermato.
- `StopWhenUnneeded=yes`: Quando è attivo, il servizio sarà fermato se nessun altro servizio necessita di esso. Nell'esempio specifico, sarà fermato dopo che sleep.target sarà fermato.
- Perché sleep.target venga richiamato da suspend.target, hibernate.target e hybrid-sleep.target e sleep.target esso stesso è un StopWhenUnneeded service, il collegamento garantisce l'avvio e la fermata corretta per diversi compiti.

Hooks in /usr/lib/systemd/system-sleep

Systemd avvia tutti gli eseguibili in `/usr/lib/systemd/system-sleep/` e passa due argomenti a ognuno di essi:

- Argument 1: `pre` o `post`, a seconda che la macchina si stia addormentando o svegliando
- Argument 2: `suspend`, `hibernate` oppure `hybrid-sleep`, dipende da quello che è stato invocato.

Al contrario di Pm-utils, systemd avvierà questi scripts contemporaneamente e non uno dopo l'altro.

L'output di ogni script personalizzato sarà registrato da `systemd-suspend.service`, `systemd-hibernate.service` oppure `systemd-hybrid-sleep.service` cosicché sarà possibile vedere i propri outputs nel journal:

```
# journalctl -b -u systemd-suspend
```

Nota che è possibile usare anche `sleep.target`, `suspend.target`, `hibernate.target` oppure `hybrid-sleep.target` per agganciare le unità allo stato di sospensione invece di usare degli scripts personalizzati.

Un esempio di script personalizzato:

`/usr/lib/systemd/system-sleep/example.sh`

```
#!/bin/sh
case $1/$2 in
  pre/*)
    echo "Going to $2..."

    ;;
  post/* )
    echo "Waking up from $2..."
    ;;
esac
```

Non dimenticare di rendere lo script eseguibile:

```
        +
# chmod a+x /usr/lib/systemd/system-sleep/example.sh
```

Vedere `man 7 systemd.special` e `man 8 systemd-sleep` per maggiori dettagli.

File temporanei

Systemd-tmpfiles mantiene la configurazione dei file in `/usr/lib/tmpfiles.d/` e in `/etc/tmpfiles.d/` per descrivere la creazione, la pulizia e la rimozione dei files e delle cartelle temporanee e volatili che normalmente risiedono in cartelle come `/run` o `/tmp`. Ogni file di configurazione prende il nome nello stile di `/etc/tmpfiles.d/<program>.conf`. Ciò sovrascriverà ogni file in `/usr/lib/tmpfiles.d/` con lo stesso nome.

I tmpfiles sono normalmente forniti assieme al service files per creare cartelle che certi demoni si aspettano di trovare esistenti. Per esmpio il demone Samba si aspetta che esista la cartella `/run/samba` per ottenere i permessi corretti. Il corrispondente tmpfile è qualcosa del genere:

`/usr/lib/tmpfiles.d/samba.conf`

```
D /run/samba 0755 root root
```

I tmpfiles possono anche essere usati per scrivere valori in certi files al boot. Per esempio, se si usa `/etc/rc.local` per disabilitare il risveglio del sistema attraverso dispositivi USB con `echo USBE > /proc/acpi/wakeup`, si può usare in alternativa il seguente tmpfile:

`/etc/tmpfiles.d/disable-usb-wake.conf`

```
w /proc/acpi/wakeup - - - - USBE
```

Vedi `man 5 tmpfiles.d` per i dettagli.

Nota: Questo metodo potrebbe non funzionare per impostare le opzioni in `/sys` poiché il

servizio `systemd-tmpfiles-setup` può eseguirsi prima che i moduli delle periferiche appropriate siano caricati. In questo caso si potrebbe verificare che il modulo abbia un parametro per l'opzione che si desidera impostare con `modinfo <module>` e impostare questa opzione con un file di configurazione in `/etc/modprobe.d`. In caso contrario si dovrà scrivere una regola udev per impostare l' attributo appropriato non appena viene visualizzato il dispositivo.

Unità

Un file di configurazione di unità racchiude informazioni riguardanti un servizio, un socket, un dispositivo, un punto di montaggio, un punto di automontaggio, un file o una partizione di swap, un obbiettivo di avvio, un percorso del filesystem oppure un controllo schedulato da systemd. La sintassi è ispirata da "XDG Desktop Entry Specification" files di tipo .desktop, che a loro volta richiamano i files .ini di windows.

Vedi `man 5 systemd.unit` per maggiori informazioni.

Gestire le dipendenze

Con systemd le dipendenze possono essere risolte progettando le unità correttamente. Il caso più tipico è che l'unità `A` richieda l'unità `B` per poter funzionare prima che `A` parta. In questo caso aggiungere `Requires=B` e `After=B` alla sezione `[Unit]` di `A`. Se la dipendenza è opzionale aggiungere, invece, Wants=B e After=B. Notare che `Wants=` e `Requires=` non includono `After=`, il che significa che se `After=` non è specificato, le due unità saranno avviate in parallelo.

Le dipendenze sono di solito posizionate sui .service e non sui .target. Per esempio, `network.target` è richiamato qualsiasi sia il servizio che configuri l'interfaccia di rete, quindi avviare successivamente la propria unità personalizzata è sufficiente in quanto `network.target` è avviato comunque.

Type

Ci sono parecchi tipi differenti di avvio da considerare quando si scrive un servizio personalizzato. Ciò è configurato tramite il parametro `Type=` nella sezione `[Service]`. Vedere `man systemd.service` per una spiegazione più dettagliata.

- `Type=simple`(default): systemd presuppone che il servizio deve essere avviato immediatamente. Il processo non può essere suddiviso. Non usare questo tipo se altri servizi hanno bisogno di essere disposti con questo servizio, a meno che non sia attivato dal socket.
- `Type=forking`: systemd presuppone che il servizio deve essere avviato prima il processo sia suddiviso e il genitore sia concluso. Per i classici demoni usare questo tipo a meno che non si sappia che non è necessario, come la maggior parte dei demoni usa il double-forking per segnalare che sono pronti. Occorre pure specificare `PIDFile=` perché systemd tenga traccia del processo principale.

- `Type=oneshot`: E' utile per scripts che eseguono un singolo lavoro e si concludono. Si può configurare pure con `RemainAfterExit=yes` in modo che systemd consideri il servizio ancora attivo anche dopo che il processo si è concluso.
- `Type=notify`: Identico a `Type=simple`, ma con l'accorgimento che il demone invierà un segnale a systemd quando sarà pronto. L'implementazione di riferimento per questa notifica è fornita da `libsystemd-daemon.so`.
- `Type=dbus`: Il servizio è considerato pronto quando lo specificato `BusName` appare nel sistema di bus si DBus.

Rimpiazzare le unità fornite

Per editare il file unit fornito da un pacchetto, si può creare una cartella chiamata `/etc/systemd/system/<unit>.d/` per esempio `/etc/systemd/system/httpd.service.d/` e mettere i files `*.conf` in essa per sovrascrivere e aggiungere nuove opzioni. Systemd controllerà questi files `*.conf` e li applicherà al di sopra degli originali. Per esempio, se si vuole semplicemente aggiungere un dipendenza all'unità si può creare il seguente file:

```
/etc/systemd/system/<unit>.d/customdependency.conf

[Unit]
Requires=<new dependency>
After=<new dependency>
```

Poi eseguire i comandi seguenti perché le modifiche abbiano effetto:

```
# systemctl daemon-reload
# systemctl restart <unit>
```

In alternativa si può copiare la vecchia unità da `/usr/lib/systemd/system/` in `/etc/systemd/system/` e fare qui le modifiche. Una unità in `/etc/systemd/system/` sovrascriverà sempre la stessa unità in `/usr/lib/systemd/system/`. Da notare che quando l'unità originale in `/usr/lib/` cambia a causa di un aggiornamento, le modifiche non si rifletteranno automaticamente sulla propria unità personalizzata in `/etc/`. In aggiunta occorre riattivarla manualmente con `systemctl reenable <unit>`. E' raccomandato usare il metodo con il `*.conf` descritto sopra.

Suggerimento: Si può usare `systemd-delta` per vedere quali unità sono state sovrascritte e quali esattamente sono state modificate.

Siccome le unità saranno aggiornate di volta in volta , usare systemd-delta per la manutenzione del sistema.

Evidenziazione della sintassi per le unità di systemd con Vim

L'evidenziazione della sintassi per le unità di systemd con Vim può essere attivata installando vim-systemd dal repo ufficiale.

Targets

Quello dei "runlevels" è un concetto superato in systemd. Systemd ha il concetto di *target* (lett. "bersagli") che adempiono uno scopo simile a quello dei runlevel, ma che hanno un comportamento leggermente differente. Ogni *target* ha un nome anziché un numero ed è usato per raggiungere uno specifico scopo con la possibilità di averne più di uno attivo nello stesso momento. Alcuni *targets* sono attivati ereditando tutto dei servizi di un altro *target* e implementandolo di servizi addizionali. Ci sono *target* che simulano i runlevel di SystemVinit, è così possibile passare da un *target* all'altro utilizzando il comando `telinit RUNLEVEL`.

Conoscere il target attuale

Il seguente comando dovrebbe essere usato sotto systemd al posto di `runlevel`:

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

Creare un target personalizzato

I runlevels sono assegnati ad uno specifico scopo su l'installazione vanilla di Fedora; 0, 1, 3, 5, e 6; hanno una mappatura 1:1 con uno specifico *target* di systemd. Sfortunatamente non c'è un altrettanto buon metodo per i runlevels definiti dall'utente come 2 e 4. Se si fa uso di questi, si suggerisce di dare un nuovo nome al *target* di systemd come `/etc/systemd/system/<your target>` che prenda come base une dei runlevels esistenti (vedi `/usr/lib/systemd/system/graphical.target` come esempio), creare una cartella `/etc/systemd/system/<your target>.wants`, e fare un collegamento ai servizi addizionali da `/usr/lib/systemd/system/` che si intendono attivare.

Targets table

SysV Runlevel	systemd Target	Notes
0	runlevel0.target, poweroff.target	Ferma il sistema.
1, s, single	runlevel1.target, rescue.target	Modalità utente singolo (single user).
2, 4	runlevel2.target, runlevel4.target, multi-user.target	Definita dall'utente. Preconfigurata a 3.
3	runlevel3.target, multi-user.target	Multi-user, non-graphical. Users can usually login via multiple consoles or via the network.
5	runlevel5.target, graphical.target	Multi-user, grafica. Solitamente ha tutti i servizi del runlevel 3 con l'aggiunta di un login grafico.
6	runlevel6.target, reboot.target	Riavvio
emergency	emergency.target	Console di emergenza

Cambiare il target corrente

In systemd i targets sono esplicitati per mezzo di "target units". Si possono cambiare così:

```
# systemctl isolate graphical.target
```

Questo comando cambierà solamente il target corrente e non avrà nessun effetto sul prossimo avvio. Questo è equivalente ai comandi `telinit 3` oppure `telinit 5` in Sysvinit.

Cambiare il target predefinito all'avvio

Il target standard è `default.target`, che è abbinato in modo predefinito a `graphical.target` (che corrisponde al vecchio runlevel 5). Per cambiare il target predefinito al boot, aggiungere uno dei seguenti parametri del kernel alla linea nel bootloader:

Suggerimento: L'estensione `.target` può essere tralasciata.

- `systemd.unit=multi-user.target` (che corrisponde al vecchio runevel 3),
- `systemd.unit=rescue.target` (che corrisponde al vecchio runevel 1).

In alternativa si può lasciare il bootloader inalterato e cambiare `default.target`. Ciò può essere fatto usando `systemctl`:

```
# systemctl enable multi-user.target
```

L'effetto di questo comando si nota nell'output di `systemctl`; viene creato un link simbolico al nuovo target predefinito `/etc/systemd/system/default.target`. Questo funziona solo se:

```
[Install]
Alias=default.target
```

si trova nel file di configurazione del target. Attualmente, sia `multi-user.target` che `graphical.target` lo possiedono.

Timers

Systemd can replace cron functionality to a great extent. See systemd/Timers.

Il Journal

Fin dalla versione 38 systemd possiede un proprio sistema di log chiamato journal. Tuttavia, far funzionare il demone syslog non è più richiesto. Per leggere il log si usa:

```
# journalctl
```

Di default (quando `Storage=` è configurato a `auto` in `/etc/systemd/journald.conf`), il journal scrive in `/var/log/journal/`. La directory `/var/log/journal/` è parte di core/systemd. Se si è cancellata intenzionalmente o se qualche programma lo ha fatto systemd **non** lo riicreerà automaticamente, tuttavia sarà ricreata durante il prossimo aggiornamento di systemd. Fino a quel momento i log saranno scritti in `/run/systemd/journal`. Ciò significa che i log saranno persi al riavvio.

Filtrare l' output

`journalctl` permette di filtrare l'output secondo specifici campi.

Esempi:

Mostrare tutti i messaggi di questo boot:

```
# journalctl -b
```

Tuttavia, spesso un utente può essere interessato ai messaggi non del corrente boot, ma del precedente (per esempio a causa di un non recuperabile blocco occorso). Attualmente, questa opzione non è implementata, ma esiste una discussione a systemd-devel@lists.freedesktop.org (Settembre/Ottobre 2012).

Come workaround al momento si può usare:

```
# journalctl --since=today | tac | sed -n '/-- Reboot --/{n;:r;/-- Reboot
--/q;p;n;b r}' | tac
```

,che provoca che il precedente boot sia come scaturito oggi. Tenere presente che, se ci sono molti messaggi per il giorno corrente, l'output di questo comando può essere ritardato per un bel po 'di tempo.

Seguire i nuovi messaggi:

```
# journalctl -f
```

Mostrare tutti i messaggi di un eseguibile specifico:

```
# journalctl /usr/lib/systemd/systemd
```

Mostrare tutti i messaggi di uno specifico processo:

```
# journalctl _PID=1
```

Mostrare tutti i messaggi di una specifica unità:

```
# journalctl -u netcfg
```

Vedere `man journalctl` e `systemd.journal-fields` oppure il blog di Lennert per dettagli.

Limiti alla dimensione del journal

Se il journal è persistente (non volatile) la sua dimensione è fissata al 10% della dimensione del rispettivo file system. Per esempio con `/var/log/journal` allocato su una partizione di root di 50 GiB comporterebbe 5 GiB di dati nel journal. La dimensione massima del journal persistente può essere controllata attraverso `SystemMaxUse` in `/etc/systemd/journald.conf`, così per limitarla ad esempio a 50 MiB decommentare ed editare la linea corrispondente linea:

SystemMaxUse=50M

Fare riferimento a `man journald.conf` per maggiori dettagli.

Journald coesistente con syslog

La compatibilità con il classico syslog è fornita dal socket `/run/systemd/journal/syslog`, attraverso il quale passano tutti i messaggi. Per rendere il demone syslog funzionante con il journal, si deve associarlo a questo socket anziché a `/dev/log` (official announcement). Il pacchetto syslog-ng dei repo ufficiali automaticamente fornisce la necessaria configurazione.

```
# systemctl enable syslog-ng
```

Una buona guida a `journalctl` è qui.

Correzione di errori

Lo spegnimento e il riavvio sono terribilmente lunghi

Se il processo di spegnimento impiega molto tempo (oppure sembra bloccarsi) probabilmente la colpa è da attribuirsi alla mancata chiusura di un servizio. Systemd attende del tempo per la chiusura di ogni servizio prima di chiuderlo forzatamente. Per vedere se si soffre di questo problema vedere questo articolo.

I processi di breve durata non registrano nessun output

Se `journalctl -u foounit.service` non mostra nessun output per un processo di breve durata, controllare il PID. Per esempio, se systemd-modules-load.service fallisce, e `systemctl status systemd-modules-load` evidenzia che è eseguito con PID 123, è possibile vedere l'output del journal per quel PID, per esempio `journalctl -b _PID=123`. I campi con metadata del journal come _SYSTEMD_UNIT e _COMM sono raccolti in modo asincrono e fanno affidamento sulla cartella `/proc` per il processo esistente. La sistemazione di questo richiede una sistemazione del kernel per fornire questi dati per mezzo di una connessione socket, come per SCM_CREDENTIALS.

Diagnosi dei problemi al Boot

Avviare il sistema con questi parametri sulla linea di comando del kernel:

```
systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M
```

Maggiori informazioni sul Debugging

Vedi anche

- Sito ufficiale
- Pagine di manuale
- FAQ
- Trucchi e suggerimenti
- systemd per amministratori (PDF)
- Systemd su Fedora Project
- Come correggere i problemi di systemd

- Two part articolo introduttivo nella rivista *The H Open*.
- Lennart's blog story
- status update
- status update2
- status update3
- most recent summary
- Fedora's SysVinit to systemd cheatsheet

Categories:

- Daemons and system services (Italiano)
- Boot process (Italiano)

Navigation menu

- Page
- Discussion
- View source
- History

- Create account
- Log in

Navigation

- Main page
- Categories
- Getting involved
- Wiki news
- Random page

interaction

- Help
- Contributing
- Recent changes
- Recent talks

- New pages
- Statistics
- Reports
- Requests

Tools

- What links here
- Related changes
- Special pages
- Permanent link
- Page information

In other languages

- العربية
- Deutsch
- Ελληνικά
- English
- Español
- فارسی
- Français
- □□□
- Português
- Русский
- □□□□□□□□
- □□□□□□