

Corso RedHat per sistemisti INFN

RHEL/SL/CentOS 7

Systemd

introduzione a systemd

systemd units

systemctl

target units

SysV Init vs systemd

sintassi dello unit file

Introduzione a systemd



Init system: il processo PID 1

PID 1 e' il primo processo eseguito dal kernel
questo processo esegue tutti gli altri processi
la sua funzione principale:

- gestire la fase di boot e inizializzare il sistema

altre funzioni:

- gestire start/stop/restart dei servizi

- gestire i processi orfani

- gestire la fase di shutdown del sistema

PID 1 e' un daemon che rimane attivo fino allo shutdown

l'implementazione storicamente tra le piu' diffuse e' SysV Init

SysV Init: problemi

- SysV Init ha alcuni problemi storici
- Lentezza
 - fa partire tutti i servizi prima di rendere il sistema disponibile
 - fa partire tutti i servizi in modo sequenziale
 - lo startup di ciascun servizio e' guidato da shell script
- Controllo limitato sui processi
 - impossibile gestire il daemon che fa double fork
 - ed i suoi sottoprocessi
 - limitata gestione dinamica dei servizi
 - limitato supporto di dipendenze e priorit 

Ripensare PID 1

Il processo di inizializzazione del sistema deve

- far partire meno cose subito
 - demandare lo startup di cose non indispensabili a dopo
- fare piu' in fretta
 - parallelizzazione dello startup
- migliorare il controllo sui processi dei servizi

systemd fa questo, e molte altre cose

systemd: caratteristiche (1/3)

Parallelizzazione dello startup e gestione dinamica tramite

- Socket-based activation

- Bus-based activation (D-Bus)

Gestione della attivazione (e disattivazione) dinamica di servizi, tramite

- Device-based activation

- Path-based activation

Gestione delle dipendenze tra servizi

- dipendenze positive o negative (requirements, conflicts)

- start condizionato di servizi

systemd: caratteristiche (2/3)

Definizione di uno stato del sistema

- estensione del concetto di runlevel

- analisi della transazione tra due stati per garantire consistenza

- supporto di “snapshot” dello stato dei servizi

Service management ed isolamento

- controllo delle risorse e dei processi (cgroups)

- private /tmp

systemd: caratteristiche (3/3)

Supporto ad altre funzioni:

Mount and automount management (anche vfs /proc, /sys, ...)

Swap management, Timer management, Host name

Time management, Log di sistema, Disk encryption,

Container registration, Read ahead, GPT partition discovery

...

Compatibilita' (non completa) con SysV Init

Init script come sorgente alternativa di configurazione

systemd interpreta le configurazioni LSB

Diffusione di systemd

- systemd e' stato sviluppato da Lennart Poettering e Kay Sievers, di RedHat
- systemd e' stato adottato come sistema di inizializzazione da diverse distribuzioni
 - Fedora
 - Red Hat
 - Debian
 - OpenSuse
 - Ubuntu
- ci sono numerose alternative , alcune segnano il passo
 - runit, OpenRC, Upstart, ...

systemd: il dibattito

systemd ha guadagnato molte critiche

- sostanzialmente systemd viola la filosofia Unix:
 - do one thing and do it well
- vuole gestire cose che sono “compito” di altri pacchetti
 - non si limita alla inizializzazione
 - va molto oltre: gestione del sistema
- e' un grosso e complesso binario
 - molto complicato capire cosa faccia
 - monolitico, appare Windows-like
 - desktop oriented
- log in formato binario (corruttibili)
- team di sviluppo non responsivo sui bug (Torvald)
- imposto dall'alto (dalle distribuzioni)

Parte della comunita' ha fatto opposizione di principio, parte esprime critiche a fianco di commenti favorevoli

systemd units



- systemd gestisce oggetti che si chiamano units
 - una unit e' definita da un nome e da un tipo:
`<unit_name>.<unit_type>`
- la unit rappresenta una risorsa gestibile del sistema
 - servizi
 - risorse di rete, device, mount point, path
 - raggruppamenti di risorse (target units)
- ogni unit e' associate ad uno unit file, che ne specifica tipologia e configurazione
- le operazioni di management sulle units sono fatte tramite l'utility `systemctl`

lo unit file contiene i parametri di configurazione della unit (ad es.: dipendenze, environment, eseguibile, ...)

- uno unit file ha il nome della unit che definisce: `<unit-name>.<unit-type>`
 - es: `ntpd.service`

collocazione degli unit file:

- **`/lib/systemd/system/`** (default dir per i packages)
- **`/etc/systemd/system/`** (sysadmin, override del file in default dir)
 - viene mantenuto in caso di update del pacchetto
- **`/run/systemd/system/` (modifiche non permanenti, via switch `--runtime`)**
 - prioritata' sulla default location, non su `/etc/systemd/system`

Tipi di unit (1/3)

service

- Una service unit descrive come gestire un servizio o una applicazione. Questo include come farla partire, fermarla, in quali circostanze deve essere fatta partire automaticamente, le sue dipendenze e priorit  in relazione alle altre unit.

socket

- Una socket unit descrive un socket di rete, un socket unix o una FIFO che systemd utilizza per implementare l'attivazione dinamica dei servizi socket-based. Ad una socket unit deve essere associata una service unit, usualmente con lo stesso nome, che verr  attivata in occasione di attivita' sul (sui) socket che questa unit definisce

device

- Una device unit descrive un device che udev ha designato come gestito da systemd. Una possibile funzione   la attivazione/disattivazione dinamica di servizi che dipendono da hardware. Queste unit sono create dinamicamente per effetto di udev.

mount

- Una unit di tipo mount definisce un mount point che deve essere gestito da systemd. Le entry in /etc/fstab generano mount unit automaticamente.

Tipi di unit (2/3)

automount

- La unit di tipo automount serve a gestire mount point montati automaticamente on demand (esattamente come fa autofs). Alla unit automount deve corrispondere una unit di tipo mount, che viene attivata quando si fa un accesso al mount point.

swap

- La swap unit definisce aree swap sotto il controllo di systemd (in modo simile a unit di tipo mount).

target

- La unit di tipo target viene usata per definire configurazioni complessive dello stato di attivazione delle unit. Utilizzati per definire lo stato del sistema dopo il boot o per modificare lo stato (come gli init levels). Lo stato, cioè l'insieme delle unit da attivare e da disattivare in relazione ad uno stato, viene definito tramite relazioni di dipendenza tra la target unit e le altre unit.

path

- La path unit definisce un path a cui collegare una unit per implementare la path-based unit activation. Per default la path unit ha lo stesso nome della service unit a cui è connessa.

Tipi di unit (3/3)

timer

- Definisce un timer gestito da systemd, a cui e' possibile associare una unit che verra' attivata alla scadenza del timer. Timer opera in modo analogo a cron.

snapshot

- systemd supporta la definizione di snapshot dello stato di attivazione delle unit in un dato momento, attraverso la creazione di una snapshot unit, che riporta le corrette dipendenze per rigenerare lo stato del sistema alla sua creazione. Le snapshot non hanno unit file e non vengono mantenute al reboot.

slice

- La unit di tipo slice e' utilizzata per definire Linux Control Groups a cui associare i processi generati da una unit, in modo da poterne limitare le risorse dedicate. Per default le service unit sono messe nella system.slice

scope

- Create automaticamente da systemd in base ad informazioni ricevute dalla bus interface. Utilizzate per gestire insiemi di processi di sistema creati esternamente (non tramite systemd). Non hanno unit file. Raggruppa un pool di worker process per motivi di gestione e controllo risorse.

systemctl



- **systemctl** e' l'utility che serve a interagire con systemd
- attraverso systemctl e' possibile
 - eseguire e fermare unit
 - abilitare o disabilitare lo start di unit al boot
 - controllare lo stato delle unit e del sistema
 - modificare lo stato del sistema
 - operare sui file di configurazione delle unit
- systemctl supporta la **bash completion**
- se non si specifica la unit type, per default systemctl assume sia una unit di tipo service

system state overview

- Lo stato del sistema (dal punto di vista di systemd) si visualizza tramite
 - # **systemctl status**
- l'output mostra
 - stato corrente, eventuali job queued e unit fallite
 - la gerarchia di slice dei processi eseguiti da systemd
- Per visualizzare i job pending:
 - # **systemctl list-jobs**
- Per visualizzare i parametri di config del manager:
 - # **systemctl show**

units state overview

# systemctl list-units	(solo le unit active)
# systemctl list-units -a	(tutte)
# systemctl list-units --failed	(quelle fallite)

lo stato delle unit e' definito da

LOAD: indica se lo unit file e' stato letto ed e' corretto

ACTIVE: stato che indica se la unit sia stata attivata

SUB: low level state dipendente dalla unit type

Per visualizzare tutti gli unit file presenti su disco:

```
# systemctl list-unit-files
```

list-unit-files non include le unit generate dinamicamente da systemd, ma visibili tramite **list-units** (es: slice units, mount units, ...)

unit control: start/stop/status

```
# systemctl start <unit>
# systemctl stop <unit>
# systemctl restart <unit>
# systemctl try-restart <unit> (restart solo se running)
# systemctl reload <unit>
# systemctl reload-or-restart <unit>

# systemctl status <unit>
# systemctl is-active <unit>
# systemctl is-enabled <unit>
# systemctl is-failed <unit>
```

uniut control: enable/disable

```
# systemctl enable <unit>  
# systemctl disable <unit>  
# systemctl preset <unit> (reset al valore di default)  
# systemctl reenable <unit>
```

L'abilitazione di una unit corrisponde alla creazione di un soft link in:

```
/etc/systemd/system/<current-target>.wants/<unit>
```

che punta allo unit file della unit stessa.

unit control: mask/unmask

```
# systemctl mask <unit>
```

```
# systemctl unmask <unit>
```

systemd supporta la funzionalita' di unit masking
una unit masked e' una unit che, anche se attivata, non fa nulla
Il masking si realizza con la creazione di un soft link:

/etc/systemd/system/<unit> -> /dev/null

essendo in /etc/systemd/system/, fa override del link nella
standard directory

systemd trova uno unit file vuoto e non fa nulla
l'operazione di unmask rimuove semplicemente il link

unit control: dependencies

- systemd supporta il concetto di dipendenze
 - la configurazione supporta la specifica di dipendenza nei due versi:
 - questa unit richiede che la unit-X sia eseguita
 - l'esecuzione di questa unit e' richiesta dalla unit-Y
 - systemd utilizza queste ed altre informazioni per ordinare e parallelizzare lo startup delle unit

systemctl list-dependencies [--reverse]

- le dipendenze possono essere stringenti o no, questo si definisce nello unit file (vedremo come).

systemd: configuration

La configurazione di systemd e' in
`/etc/systemd/system.conf`

Contiene una serie di parametri che normalmente non e' necessario modificare; alcuni curiosi:

- `CtrlAltDelBurstAction=`
defines what action will be performed if user presses Ctrl-Alt-Delete more than 7 times in 2s...

Per visualizzare il valore corrente dei parametri:

`systemctl show`

Esercitazione parti 1 e 2
Soluzione

target units



- La target unit definisce una collezione di units
- contiene solo dipendenze
 - definisce l'insieme dei service che devono essere fatti partire quando la unit viene fatta partire
 - in questo senso, una target unit definisce uno stato del sistema
- Le dipendenze sono tipicamente definite negli unit file delle unit che fanno parte dello stato (tramite WantedBy=<target-unit>)
 - alla abilitazione del service viene creato un link in /etc/systemd/system/<target>.wants/<target>

working with target units (cont.)

Le target unit sono utilizzate per

- accorpare unit
- definire punti di sincronizzazione per lo startup
 - incluso il punto di arrivo (il runlevel di Init)

Come punti di sincronizzazione, tipicamente sono definiti in modo gerarchico

- es. `graphical.target` dipende da `multi-user.target`

L'operazione di "isolation" di una target unit consiste in:

- fermare tutte le unit attive (i services) da cui la target unit non dipende
- attivare tutte le unit da cui la target unit dipende

In questo modo si porta il sistema ad uno stato definito

```
# systemctl isolate <target-unit>
```

E' equivalente a quanto si faceva con init <N>

default target unit

- La default target unit e' la target che definisce lo stato che il sistema raggiunge al bootstrap
 - analoga al default init level di SysV Init
 - di norma: graphical.target o multi-user.target

systemctl set-default <target-unit>

systemctl get-default

- Il default runlevel e' definito attraverso lo unit file

/etc/systemd/system/default.target

che e' un link che punta allo unit file reale della target unit selezionata tramite get-default

snapshot unit

- La snapshot unit viene creata tramite il comando
systemctl snapshot <snap-name>
- questo crea dinamicamente una unit che contiene solo dipendenze verso le unit attive al momento della sua creazione
- in questo modo il sistema puo' essere ripristinato allo stato definito dalla snapshot
 - tramite **# systemctl isolate <snap-name>**
 - in questo senso la snapshot e' molto simile ad una target unit
- la snapshot unit e' definita solo runtime
 - si perde la sua definizione al reboot

Systemd vs Init



supporto per SysV Init scripts

- systemd supporta lo startup dei servizi definiti in Init script
 - migrazione graduale da Init a systemd
- systemd supporta gli script tramite systemd-sysv-generator
 - questa utility fa parte dei system generator di systemd
 - sono utility che vengono eseguite in uno stadio precoce di systemd e generano dinamicamente service units
 - in modo analogo vengono generate altre unit dinamiche, ad esempio mount units da /etc/fstab

supporto per SysV Init scripts (cont.)

- systemd-sysv-generator legge gli script in `/etc/init.d/*`
 - interpreta script e le direttive Linux Standard Base
 - valuta e genera dipendenze
 - genera delle service unit corrispondenti
- la generazione dinamica delle service unit e' rifatta ad ogni reload
 - l'esecuzione delle unit cosi' generate deve comunque aspettare la `basic.target`
- systemd supporta anche lo script `/etc/rc/rc.local` (attenzione!)
 - attraverso la unit `rc-local.service`

SysV Init runlevel e target units

systemd definisce alcune target-units che sono corrispondenti ai runlevel di SysV Init:

- init 0 -> systemctl isolate poweroff.target
- init 1 -> systemctl isolate rescue.target
- init 3 -> systemctl isolate multi-user.target
- init 5 -> systemctl isolate graphical.target
- init 6 -> systemctl isolate reboot.target

il supporto dei runlevel e' parziale: no user defined runlevel 2 e 4
esistono le target unit: runlevel[0-6].target

- sono symlink ai target sopra descritti, 2 e 4 -> multi-user.target

Target per il boot

Esistono target utilizzabili per un bootstrap minimale

- **emergency.target**
 - solo systemd e una shell
 - usata automaticamente in caso di problemi con fsck al boot
- **rescue.target**
 - oltre a systemd e shell, fa partire servizi minimali e fa il mount dei file system

Pensate per essere eseguite tramite il kernel parameter

- `systemd.unit=<target>`

systemd supporta una modalita' interattiva

in questa modalita' systemd non parallelizza, e chiede conferma per ogni unit da attivare

questa modalita' si attiva specificando il kernel parameter
`systemd.confirm_spawn=1`

systemctl vs service

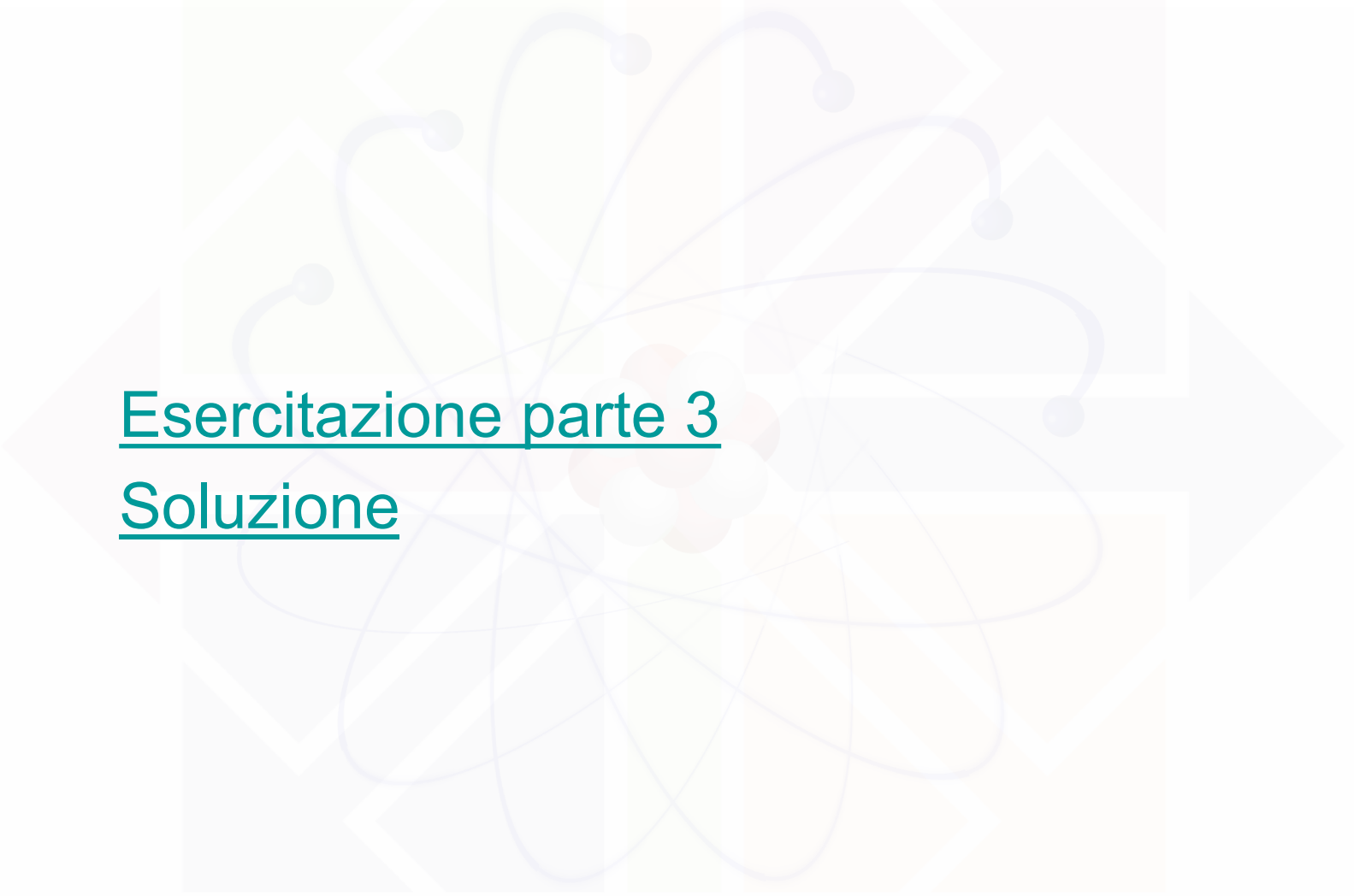
service	systemctl	Description
<code>service name start</code>	<code>systemctl start name.service</code>	Starts a service.
<code>service name stop</code>	<code>systemctl stop name.service</code>	Stops a service.
<code>service name restart</code>	<code>systemctl restart name.service</code>	Restarts a service.
<code>service name condrestart</code>	<code>systemctl try-restart name.service</code>	Restarts a service only if it is running.
<code>service name reload</code>	<code>systemctl reload name.service</code>	Reloads configuration.
<code>service name status</code>	<code>systemctl status name.service</code> <code>systemctl is-active name.service</code>	Checks if a service is running.
<code>service --status-all</code>	<code>systemctl list-units --type service --all</code>	Displays the status of all services.

systemctl vs service (cont.)

chkconfig	systemctl	Description
<code>chkconfig name on</code>	<code>systemctl enable name.service</code>	Enables a service.
<code>chkconfig name off</code>	<code>systemctl disable name.service</code>	Disables a service.
<code>chkconfig --list name</code>	<code>systemctl status name.service</code> <code>systemctl is-enabled name.service</code>	Checks if a service is enabled.
<code>chkconfig --list</code>	<code>systemctl list-unit-files --type service</code>	Lists all services and checks if they are enabled.
<code>chkconfig --list</code>	<code>systemctl list-dependencies --after</code>	Lists services that are ordered to start before the specified unit.
<code>chkconfig --list</code>	<code>systemctl list-dependencies --before</code>	Lists services that are ordered to start after the specified unit.

Incompatibilita' con SysV Init

- parziale supporto dei runlevel
- systemctl non supporta comandi custom
 - ne' personalizzazione dello startup/shutdown di un servizio
- systemctl non puo' comunicare con processi fatti partire a mano
- systemd ferma solo processi running
 - SysV Init script vengono eseguiti anche se il processo non e' running
- i systemd services non possono leggere da standard input
- i systemd services non ereditano alcun contesto della shell da cui vengono fatti partire
- timeout di 5 minuti hard coded per i servizi eseguiti tramite SysV Init script



Esercitazione parte 3 Soluzione

Unit file: sintassi



Sintassi dello unit file

Lo unit file e' costituita da una sequenza di:

[session]
direttiva1=valore1
direttiva2=valore2

L'ordine non e' importante.

Le direttive possono essere sovrascritte nelle *snippet* (vedremo dopo):

direttiva1=nuovo-valore

o anche rimosse, lasciando vuoto il valore:

direttiva2=

Informazioni complete: **man systemd.unit, man systemd.<unit-type>**

[Unit] session

la [Unit] session e' solitamente la prima, ed include (tra le altre):

- **Requires**=unit1 unit2 ...
units che devono essere attivate prima della current
- **Wants**=unit1 unit2
units che systemd tenta di attivare prima della current
- **BindsTo**=unit1 unit2 ...
come Requires, ma la current unit viene fermata quando si fermano queste
- **Before/After**=unit1 unit2...
Le units vengono fatte partire prima/dopo la current (se attivate)
- **Conflicts**=unit1 unit2...
la corrente unit non viene attivata se una di queste e' attiva
- **Condition**...=...
se le condizioni non sono soddisfatte, la current unit viene ignorata
es: in quotaon.service: **ConditionPathExists=/usr/sbin/quotaon**
- **Assert**...=...
se le condizioni non sono soddisfatte, la current unit viene dichiarata failed

[Install] section

Usualmente indicata per ultima, include:

- **WantedBy=unit1...**
altro modo per indicare una dipendenza, usualmente riferita a target units;
se la current unit e' attiva, viene creata una directory unit1.wants con un symlink allo unit file della current unit
in questo modo tutte le dipendenze della unit1 stanno scritte come link nella dir unit1.wants
- **RequiredBy=unit1 unit2...**
come WantedBy, ma requisito stringente
- **Alias=...**
specifica un alias per la unit corrente (es: syslog/rsyslog, runlevel*,...)
- **Also=unit1 unit2...**
le unit specificate saranno abilitate/disabilitate assieme alla current, come un gruppo

Usata per fornire direttive per le service units

- **Type**=simple | forking | oneshot | dbus | notify | idle
indica il tipo di processo (should be defined)
- **RemainAfterExit**=yes|no (per Type=oneshot)
- **PidFile**=... (per Type=fork)
- **BusName**=... (per Type=dbus)
- **ExecStart**=... (nome dell'eseguibile)
- **ExecStop**=... (eseguibile per lo stop; per default: kill <pid>)
- **Restart**=always | on-success | on-failure|...
definisce se e quando systemd debba fare restart

Session specifiche di unit type

[Socket]

ListenStream, ListenDatagram, ListenFIFO:

direttive per specificare il tipo di socket

Service=... se il service associato ha nome diverso

[Path]

PathExists, PathExistsGlob, PathChanged (alla close()),

PathModified (ad ogni write()), **DirectoryNonEmpty**

direttive per attivare il service associato in caso di eventi su file system

Service=... se il service associato ha nome diverso

[Mount]

Unit per demandare a systemd la gestione del mount di un device

What=, Where=, Type=, Options=, ...

Queste unit devono avere un nome derivato dal mount point

- senza la prima /, con le altre cambiate in dash
- (vedi esempi: **# systemctl list-units -t mount**)

Il file /etc/fstab viene analizzato da systemd per generare dinamicamente le relative mount units (preferred method!)

[Automount]

direttive per la gestione del servizio di automount tramite systemd

Il nome deve essere codificato dal mount point (vedi mount unit)

deve necessariamente esistere una mount point unit associata

[Swap]

supporta direttive per la gestione dello swap (analoga a mount)

[Timer]

supporta direttive per specificare i timer per l'attivazione della service unit associata (analogo a at/cron)

[Slice] session

La [Slice] session non viene associata a unit di tipo slice.

Lo scopo della Slice session e' di specificare opzioni di resource control per l'insieme dei processi eseguiti da una unit

Internamente il controllo delle risorse viene eseguito tramite C-Groups

Le direttive specificate nella session [Slice] sono supportate anche nelle session [Scope], [Service], [Socket], [Mount] e [Swap]

- **CPUAccounting, CPUQuota, MemoryLimit, TaskMax...**

direttive per specificare i limiti della slice

- **Slice=...**

Nome della slice unit in cui mettere la unit (default: system.slice)

Permette di implementare un albero gerarchico di nomi

- la root slice si chiama “-”
- il separatore della gerarchia dei nomi e' “-”:

Template unit file

- systemd supporta la creazione dinamica di unit multiple sulla base di un template
- si possono configurare unit in cui parte della configurazione sono informazioni dinamiche accessibili a runtime
- nello unit file si possono utilizzare degli indicatori che vengono sostituiti a run time dai valori della istanza; es:
 - Description=Getty on %I
- il template file ha un nome del tipo
 - <template-name>@.service
- Le sue istanze avranno nome:
 - <template-name>@<instance-name>.service
- Per eseguire una istanza:
 - systemctl start <template-name>@<instance-name>.service
- Per eseguire una istanza automaticamente (ad esempio in corrispondenza di una target) , si deve inserire un link col nome della istanza e che punti al template unit file nella directory <target-name>.wants

Modifiche parziali allo unit file

- Se esiste una directory
/[lib|etc]/systemd/system/<unit>.d
i file di estensione .conf ivi contenuti vengono letti per eventuali
aggiunte o modifiche di parametri alla configurazione (snippets)
- L'operazione puo' essere eseguita a mano o tramite il comando
systemctl edit <unit>
- Dopo aver fatto una modifica si deve eseguire
systemctl daemon-reload
 - se si utilizza systemctl edit il daemon-reload viene eseguito automaticamente

`systemctl cat <unit>`

mostra il file di configurazione

- se esistono delle modifiche (snippet) le mostra
- per ogni file mostra il nome del file ed il contenuto
- mostra solo i file prioritari (override)

`systemctl show <unit> [--all]`

mostra il valore di tutti i parametri applicabili alla unit

- `--all`: mostra anche i parametri vuoti

Avete un eseguibile in /root/ses

- exec name: /root/ses/ses
- l'eseguibile espone un servizio di echo, gira in foreground e si mette in ascolto sulla porta TCP 0.0.0.0:5555
- funziona correttamente se eseguito interattivamente (provare)

Esercizio 1

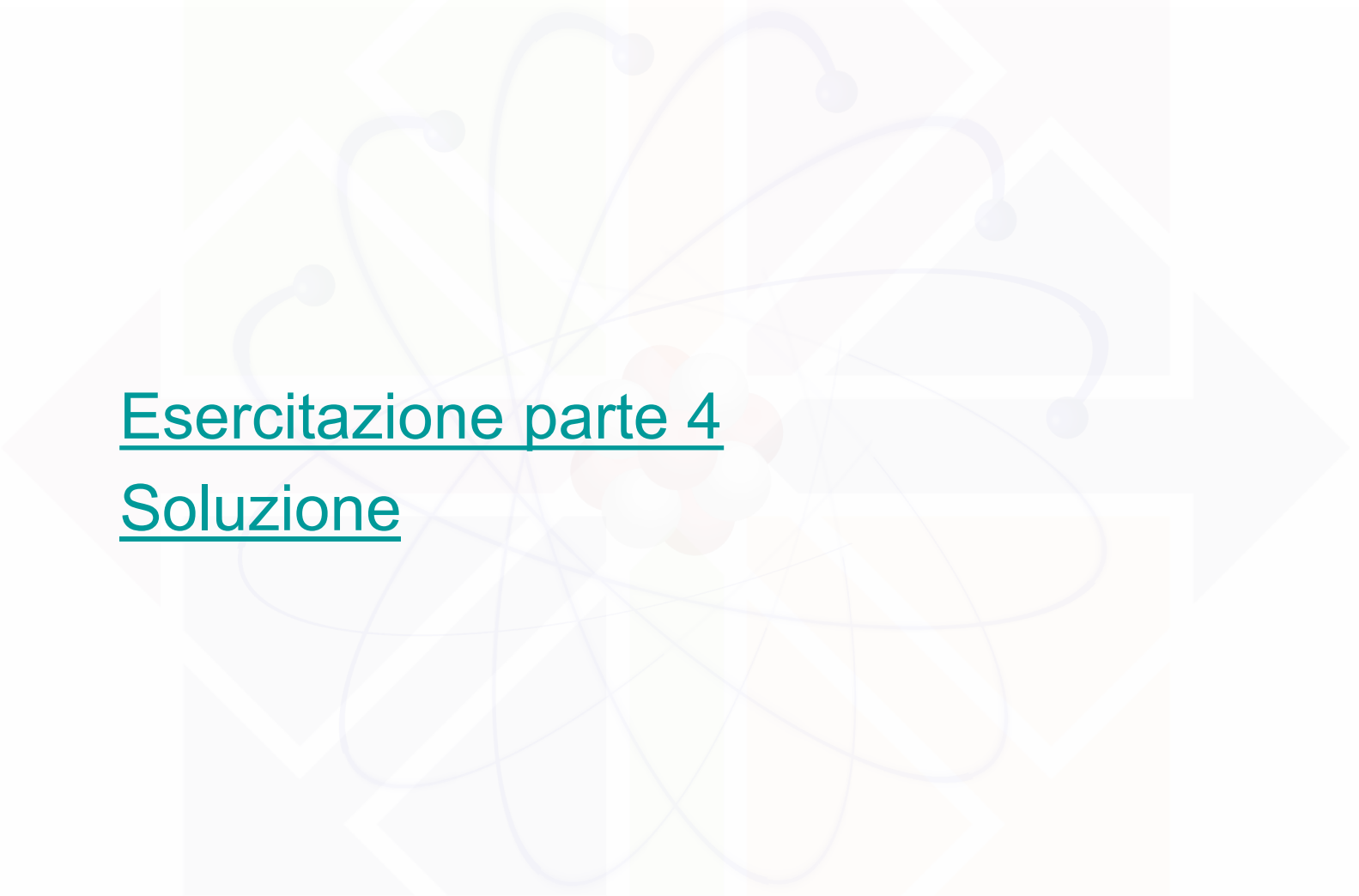
- costruire uno unit file per far gestire il servizio a systemd
 - il servizio deve ripartire in 30s se il processo muore

Esercizio 2

- costruire un altro unit file per far gestire a systemd la socket based activation
 - verificare la funzionalità

Esercizio 3

- costruire un terzo unit file per far gestire a systemd la path based activation
 - scegliere un path e creare o modificare a mano il file per verificare la funzionalità)



Esercitazione parte 4 Soluzione

Referenze in rete

- [Wiki su freedesktop.org](#) mantenuto da Lennart Poettering
- [Falsi miti su systemd](#) (L.Poettering)
- [Incompatibilita' SysV - systemd](#)