Systemd (I)

Introducció

Systemd és vàries coses:

-El procés Init (PID 1) del sistema
-El gestor de dimonis
-Un intermediari entre aplicacions d'usuari i certes parts de l'API del kernel de Linux

La configuració general de Systemd es troba a /etc/systemd/system.conf ; molts valors per defecte hi són allà establerts.

Per saber la versió actual de Systemd que hi ha funcionant al sistema, fer ***systemctl --version***

"Units": tipus i ubicació

Tot el que és gestionat per Systemd s'anomena "unit" i cada unit és descrita per un arxiu de configuració propi, el qual tindrà una extensió diferent segons el tipus de d'unit que es tracti:

.*service* : Descriu la configuració d'un dimoni
.*socket* : Descriu la configuració d'un socket (de tipus UNIX o bé TCP/IP) associat a un .service
.*device* : Descriu un dispositiu hardware reconegut pel kernel (via udev o sysfs) gestionat per Systemd
.*mount* : Descriu un punt de muntatge gestionat per Systemd
.*automount* : Descriu un punt d'automuntatge associat a un .mount
.*swap* : Descriu una partició o fitxer d'intercanvi gestionat per Systemd
.*target* : Defineix un grup d'units (s'utilitza a mode de "metapaquet" d'units)
.*path* : Descriu una carpeta o fitxer monitoritzat per l'API Inotify del kernel
.*timer*: Descriu la temporització/activació d'una tasca programada (usant el programador Systemd)
.*slice* : Defineix un grup d'units associades a processos per tal d'administrar i limitar els recursos comuns (CPU, memòria, discos, xarxa). Usa internament els anomenats "cgroups" del kernel
.*snapshot* : Descriu la configuració general del sistema en un moment concret (útil per tornar-hi després de haver fet algun canvi. Cal tenir en compte, però, que aquestes units (que es creen amb la comanda *systemctl snapshot*) no sobreviuen al tancament d'una sessió d'usuari

Els arxius de configuració de les units (siguin del tipus que siguin) poden estar repartits en tres carpetes distintes:

/usr/lib/systemd/system: Per units proporcionades pels paquets instal.lats al sistema
/run/systemd/system: Per units generades en temps real durant l'execució del sistema. No persistents
/etc/systemd/system: Per units proporcionades pels administrador del sistema

Els arxius presents a /etc/... **sobreescriuen** els arxius **homònims** que estiguin a /run/... els quals sobreescriuen els que estiguin a /usr/lib/... (o en algunes distribucions, /lib/...). Si no tenen el mateix nom, tots els fitxers de les tres carpetes es mesclen ordenats pel seu nom de forma numericoalfabètica i es van llegint en aquest ordre fins el final.

D'altra banda, si dins de /usr/lib/..., /run/... o /etc/... hi ha una carpeta anomenada com una unit seguit del sufixe ".d", qualsevol fitxer amb extensió *.conf que hi hagi a dins serà llegit just després dels fitxers de configuració de la unit pertinent. Això serveix per poder **afegir (o sobreescriure)** opcions de configuració concretes (les presents en aquests fitxers) sense haver de tocar les configuracions "genèriques" de la unit. Per exemple: l'arxiu "/usr/lib/systemd/system/beep.service.d/foo.conf" pot ser útil per modificar la configuració definida a "/usr/lib/systemd/systemd/beep.service" (i d'aquesta manera, fer possible que un paquet pugui canviar la configuració establerta per un altre) i l'arxiu "/etc/systemd/system/beep.service.d/foo.conf" pot ser útil per modificar la configuració definida a "/usr/lib/systemd/system/beep.service" (i d'aquesta manera, fer possible que un administrador pugui canviar certes parts de la configuració de la unit preempaquetada al sistema sense haver de reemplaçar-la completament). Aquests fitxers "override" (concretament amb el nom

"/etc/systemd/system/nomUnit.d/override.conf") es poden generar d'una manera molt còmoda i ràpida amb la comanda *systemctl edit nomUnit*

Algunes "unit" contenen un símbol @ al seu nom (per exemple, nom@cadena.service); això significa que són instàncies d'una unit-plantilla, el fitxer de configuració de la qual és el que no conté la part "cadena" al seu nom (així: nom@.service) . La part "cadena" és l'identificador de la instància (de fet, dins del fitxer de configuració de la unit-plantilla el valor "cadena" substitueix totes les ocurrències de l'especificador especial %i).

Comandes per gestionar units (principalment de tipus "service")

A continuació mostrem algunes de les comandes més importants per gestionar units principalment (que no exclusivament) de tipus "service":

systemctl [list-units] [-t {service|socket|...}] [--all | --failed | --state=inactive]

Mostra l'estat de les units que estan "actives" (del tipus indicat; si no s'indica, apareixen totes).
Si s'escriu --*state=inactive* es mostra l'estat de les units que estan "inactives"
Com a valor del paràmetre --*state* també es pot posar qualsevol valor vàlid de la columna SUB
Si s'escriu --*failed* es mostra l'estat de totes les units amb errors
Si s'escriu --*all* es mostra l'estat de totes les units ("actives", "inactives", amb errors i altres)

La diferència entre les columnes LOAD, ACTIVE i SUB la diu la sortida de la pròpia comanda:

LOAD = Indica si la unit ha sigut carregada en RAM. Possibles valors: "loaded", "error", "masked"
ACTIVE = Estat genèric de la unit. Possibles valors: "active", "inactive", "failed", "(des)activating"
SUB = Estat més concret de la unit; depén del tipus de unit. Possibles valors: "plugged","mounted","running","exited","waiting","listening",etc

systemctl [-t {service|socket|...}] list-unit-files

La subcomanda *list-units* només mostra les units que Systemd ha intentat llegir i carregar en memòria. Ja que Systemd només llegeix les units que ell pensa que necessita, això no inclou necessàriament totes les units disponibles al sistema. Per veure totes les units, incloent aquelles que Systemd no ha intentant ni tan sols carregar, cal utilitzar *list-unit-files*. Aquesta subcomanda mostra l'"estat de càrrega" de cada unit; possibles valors són:

"enabled" o "enabled-runtime" : La unit s'activarà al següent reinici -i subsegüents- .
NOTA: Això s'aconsegueix gràcies a l'existència d'un enllaç a l'arxiu de configuració de la unit en qüestió dins de la carpeta "/etc/systemd/system/*nomTarget*.target.wants", creat en algun moment previ amb la comanda *systemctl enable* (veure més avall) o bé manualment amb *ln -s*

"static" : La unit no té secció "[Install]" en el seu fitxer de configuració. Això fa que les comandes *systemctl enable* (i sobre tot *systemctl disable*) no funcionin. Per tant, el fet de què la unit estigui activada o no en un determinat "target" dependrà de l'existència "estàtica" del seu enllaç corresponent dins de la carpeta "/etc/systemd/system/*nomTarget*.target.wants" . Aquest tipus d'units solen estar associades a les que realitzen una acció "oneshot" o bé a les que són usades només com a dependència d'alguna altra unit (i per tant no han d'executar-se per sí mateixes)

"generated": La unit s'activarà mitjançant un mecanisme automàtic especial anomenat "generator", executat en arrencar el sistema. Cada unit en aquest estat té el seu propi "generator".

"transient" : La unit és temporal i no sobreviurà al següent reinici

"disabled" : La unit està desactivada i, per tant, no es posarà en marxa als següents reinicis (gràcies a la inexistència de l'enllaç corresponent dins de "/etc/systemd/system/*nomTarget*.target.wants"). Tampoc podrà ser iniciada automàticament mitjançant altres sistemes (com ara via socket, via D-Bus o bé via endollament de hardware. No obstant, podrà ser posada en marxa en qualsevol moment "manualment" executant *systemctl start (*veure més avall)

"masked" o "masked-runtime" : La unit està enmascarada (és a dir, està desactivada i, per tant, no es posarà en marxa als següents reinicis ni automàticament, però a més, tampoc podrà ser posada mai en marxa manualment amb *systemctl start* ni tan sols si és una dependència d'un altre servei)

systemctl {start|stop|restart} nomUnit[.service]

Activa/Desactiva/Reinicia la unit indicada immediatament seguint les indicacions escrites al seu arxiu de configuració corresponent.
NOTA: Si la unit no fos de tipus ".service", llavors caldrà indicar el seu tipus explícitament darrera el seu nom (per exemple, *systemctl start nomUnit.socket*). Aquesta norma és extensiva per la resta de comandes

systemctl {enable|disable} nomUnit[.service]

Activarà/Desactivarà automàticament la unit indicada a partir del següent reinici (i següents)
NOTA: En realitat el que fa *enable/disable* és crear/eliminar un enllaç dins de la carpeta "/etc/systemd/systemd/*nomTarget*.target.wants" al fitxer de configuració de la unit en qüestió (on "nomTarget" ve definit a la directiva WantedBy de la secció "[Install]" de dit fitxer).

systemctl {mask|unmask} nomUnit[.service]

Enmascara/Desenmascara la unit indicada.
NOTA: Això ho aconsegueix vinculant el fitxer de configuració ubicat a /etc/... de la unit en qüestió a /dev/null

systemctl is-enabled nomUnit[.service]

Retorna $?=0 si la unit indicada està configurada per activar-se en els següents reinicis (és a dir, si està en els estats -llistats per *systemctl list-unit-files*-: "enabled", "enabled-runtime", "static", "generated" o "transient") i a més, mostra en pantalla aquest estat.

systemctl is-active nomUnit[.service]

Retorna $?=0 si la unit indicada està activa i, a més, mostra en pantalla aquest estat (valor llistat a la columna ACTIVE de *systemctl list-units*)

systemctl is-failed nomUnit[.service]

Retorna $?=0 si la unit indicada va fallar en intentar activar-se i, a més, mostra en pantalla aquest estat. Si no volem que es mostrin els estats en pantalla (això també va per *is-enabled* i *is-active*), es pot afegir el paràmetre *-q*
NOTA: Una unit pot estar en estat "failed" por múltiples raons: perquè el procés ha acabat amb un codi d'error diferent de 0, perquè ha finalitzat de forma anormal, perquè s'ha superat un timeout determinat, etc.

systemctl status {nomUnit[.service] |PID}

Mostra l'estat i informació variada sobre la unit o procés indicat. Si s'indica una unit es pot veure...:

Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; vendor preset: disabled)
Active: active (running) since Sat 2017-11-18 20:48:06 CET; 4h 2min ago
Docs: man:cupsd(8)
***Main PID**: 745 (cupsd)*
Status: "Scheduler is running..."
Tasks: 1 (limit: 4915)

CGroup: /system.slice/cups.service
└─745 /usr/sbin/cupsd -l
Darreres línias de journald -u (es pot usar els paràmetres homònims -n nº i -o xxx)

Els valors per la línia "Loaded:" són els mateixos que apareixen a la columna LOAD de *list-units*
Seguidament s'indiquen els valors de l'estat actual i el predefinit pel paquet, que seran un dels detallats anteriorment en parlar de *list-unit-files*
Els valors per la línia "Active:" són els mateixos que apareixen a la columna ACTIVE de *list-units*
El punt ("●") és blanc si la unit està "inactive" ; vermell si "failed" o verd si "active"

Si s'indica un PID en comptes de una unit, es pot veure la mateixa informació, però aquesta manera pot ser útil per conèixer la unit a la què està associat un determinat procés, per exemple (encara que per conèixer aquesta informació també es podrien observar els valors de la columna "unit" mostrada per la comanda *ps* si així s'hi indica amb el paràmetre *-o*):

cups.service - CUPS Scheduler
*Loaded: loaded (**/usr/lib/systemd/system/cups.service**; enabled; vendor preset:*
Active: active (running) since Sat 2017-11-18 20:48:06 CET; 4h 2min ago
Docs: man:cupsd(8)
Main PID: 745 (cupsd)
Status: "Scheduler is running..."
Tasks: 1 (limit: 4915)
CGroup: /system.slice/cups.service
└─745 /usr/sbin/cupsd -l
Darreres línias de journald _PID= (es pot usar els paràmetres homònims -n nº i -o xxx)

systemctl show {nomUnit[.service] |PID}

Mostra la configuració actual de la unit (obtinguda a partir del fitxer general system.conf i del fitxer de configuració propi de la unit) indicada en un format adient per ser processat per màquines. Amb el paràmetre -p "nomClau,unAltreNom,..." es poden obtenir només les parelles clau<->valor desitjades.

systemctl daemon-reload

Recarrega tots els fitxers de configuració d'units noves o modificades des de la darrera vegada que es va posar en marxa Systemd (incloent els generators).

systemctl help nomUnit[.service]

Obre la pàgina de man associada a la unit indicada (al seu fitxer de configuració deu venir indicada)

systemctl edit nomUnit[.service]

Crea (amb l'editor de text predeterminat del sistema) un fitxer de configuració (inicialment buit) per la unit indicada anomenat"/etc/systemd/system/nomUnit.*tipusUnit*.d/override.conf" per sobreescriure (o ampliar) la configuració ja existent per ella. Un cop guardats els canvis, recarrega la unit automàticament amb aquesta nova configuració. Si es volgués editar directament el fitxer "/etc/systemd/system/nomUnit.*tipusUnit*", cal afegir llavors el paràmetre *--full*

systemctl cat nomUnit[.service]

Mostra la configuració final actual resultant d'haver llegit els diferents fitxers de configuració possibles de la unit indicada

systemd-delta

Mostra quins fitxers de configuració d'units estan sobreescrits o ampliats (de /usr/lib a /etc i/o amb fitxers "overrides"), enmascarats, redireccionats (amb la línia Alias= de la secció [Install]),etc i per quins

systemctl kill [--signal=nº] nomUnit[.service]

Envia una senyal concreta (indicada amb el paràmetre *–signal* ; per defecte és la nº15, SIGTERM) a tots processos associats a la unit indicada.
NOTA: `kill` goes directly and sends a signal to every process in the group, however `stop` goes through the official configured way to shut down a service, i.e. invokes the stop command configured with `ExecStop=` in the service file. Usually `stop` should be sufficient. `kill` is the tougher version, for cases where you either don't want the official shutdown command of a service to run, or when the service is hosed and hung in other ways.

Systemd també permet definir serveis per a què no estiguin associats al sistema global sinó que únicament formin part de la sessió d'un usuari estàndar, generant-se una instància particular del servei per cada usuari actiu a la màquina. D'aquesta manera, cada instància s'iniciarà automàticament després d'iniciar la sessió d'un usuari i es pararà en sortir-ne.

NOTA: Això és possible gràcies a què just després del primer inici de sessió que es realitzi al sistema es posa en marxa (gràcies al mòdul PAM "pam_systemd") la comanda *systemd –user (*qui es qui permetrà aquest funcionament individual per totes les sessions d'usuaris que s'inicïin a partir de llavors), a més del procés amb PID 1 pròpiament dit, que és *systemd –system*. El procés *systemd –user* finalitzarà automàticament just després d'haver-se tancat el darrer inici de sessió existent al sistema.

Els fitxers de configuració de les units "de tipus usuari" es troben en altres carpetes de les de les units "de sistema". Concretament (es mostren en ordre de precedència ascendent):

/usr/lib/systemd/user: Per units proporcionades pels paquets instal.lats al sistema
~/.local/share/systemd/user: Per units de paquets que han sigut instal.lades a la carpeta personal
/etc/systemd/user: Per units proporcionades pels administrador del sistema
~/.config/systemd/user : Per units construïdes pel propi usuari

NOTA: La variable especial %h es pot utilitzar dins dels fitxers de configuració de les units "d'usuari" per tal d'indicar la ruta de la carpeta personal de l'usuari en qüestió.

Una altra característica de les units "d'usuari" és que poden ser gestionades per part d'aquest usuari sense que hagi de ser administrador del sistema; això ho pot fer amb les mateixes comandes *systemctl* ... ja conegudes només que afegint el paràmetre *--user* Així, per exemple, per arrencar un servei automàticament cada cop que s'inicïi la nostra sessió caldrà executar *systemctl --user enable nomUnit* ; per veure l'estat de totes les nostres units "d'usuari" caldrà fer *systemctl --user list-units* ; per recarregar les units modificades caldrà fer *systemctl --user daemon-reload*, etc

Seccions i directives comunes en els fitxers de configuració de les units

L'estructura interna dels fitxers de configuració de les units està organitzada en seccions, distingides cadascuna per un encapçalament case-sensitive envoltat de claudàtors (*[Encapçalament]*). Dins de les seccions es defineixen diferents directives (també case-sensitive) en la forma de parelles *NomDirectiva=valor* , on el valor pot ser una paraula, una frase, una ruta, un número, t*rue/yes* o f*alse/no*, una data, etc, tot depenent del seu significat.

NOTA: També poden existir directives on no s'escrigui cap valor (és a dir, així: *NomDirectiva=*). En aquest cas, s'estarà "resetejant" (és a dir, anul.lant) el valor que prèviament s'hagués donat en algun altre lloc

La primera secció (encara que l'ordre no importa) sempre sol ser l'anomenada **[Unit]** i s'utilitza per definir dades sobre la pròpia unit en sí com a unit que és i la relació que té aquesta amb altres units. Algunes de les seves directives més habituals són:

Description=Una breu descripció de la unit

El seu valor és retornat per diferent eines Systemd

Documentation=man:sshd(8) https://ruta/pag.html

Proporciona una lista d'URIS que apunten a documentació de la unit.
La comanda *systemctl status* les mostra

Wants=unservei.service unaltre.service untarget.target ...

Llista les units que seria bo que estiguessin iniciades per a què l'unit en qüestió pugui funcionar correctament. Si no ho estan ja, Systemd les iniciarà en paral.lel juntament amb l'unit en qüestió; si es vol indicar un cert ordre en comptes d'iniciar totes en paral.lel, es pot utilitzar les directives After= o Before=. Si alguna de les units llistades falla en iniciar-se, l'unit en qüestió s'iniciarà igualment

Requires=unservei.service unaltre.service untarget.target ...

Llista les units que imprescindiblement han d'estar iniciades per a què l'unit en qüestió pugui funcionar correctament. Si no ho estan ja, Systemd les iniciarà en paral.lel juntament amb l'unit en qüestió; si es vol indicar un cert ordre en comptes d'iniciar totes en paral.lel, es pot utilitzar les directives After= o Before=. Si alguna de les units llistades falla en iniciar-se, l'unit en qüestió també fallarà automàticament

BindsTo=unservei.service unaltre.service untarget.target ...

Similar a Requires= però, a més, fa que l'unit en qüestió s'aturi automàticament si alguna de les units associades finalitza.

Before=unservei.service unaltre.service untarget.target ...

Indica, de les units llistades a les directives Wants= o Requires=, quines no s'iniciaran en paral.lel sinò després de la unit en qüestió. Si aquí s'indiqués alguna unit que no es trobés llistada a Wants= o Requires=, aquesta directiva no es tindrà en compte.

After=unservei.service unaltre.service untarget.target ...

Indica, de les units llistades a les directives Wants= o Requires=, quines no s'iniciaran en paral.lel sinò abans de la unit en qüestió. Si aquí s'indiqués alguna unit que no es trobés llistada a Wants= o Requires=, aquesta directiva no es tindrà en compte.

NOTA: El més típic és tenir una unit A que necessita que la unit B estigui funcionant prèviament per tal de poder-se posar en marxa. En aquest cas, simplement caldria afegir les línies *Requires=B* i *After=B* a la secció [Unit] de l'unit A. Si la dependència és opcional, es pot substituir *Requires=B* per *Wants=B*

Conflicts=unservei.service unaltre.service ...

Llista les units que no poden estar funcionant al mateix temps que l'unit en qüestió. Iniciar una unit amb aquesta directiva causarà que les aquí llistades s'aturin automàticament.

ConditionXXXX=...

Hi ha un conjunt de directives que comencen per "Condition" que permeten a l'administrador comprovar certes condicions abans d'iniciar l'unit. Si la condició no es compleix, la unit és ignorada. Alguns exemples són:

ConditionKernelCommandLine=param[=valor]
ConditionACPower={yes| no}
ConditionPathExists=[!]/ruta/fitxer/o/carpeta
ConditionPathExistsGlob=[!]/ruta/fitxers/o/carpetes
ConditionPathIsDirectory=[!]/ruta/carpeta

ConditionPathIsSymbolicLink=[!]/ruta/enllaç
ConditionPathIsMountPoint=[!]/ruta/carpeta
ConditionPathIsReadWrite=[!]/ruta/fitxer/o/carpeta
ConditionDirectoryNotEmpty=[!]/ruta/carpeta
ConditionFileNotEmpty=[!]/ruta/fitxer
ConditionFileIsExecutable=[!]/ruta/fitxer

AssertXXXX=...
Igual que amb "ConditionXXX", hi ha un conjunt de directives que comencen per "Assert" que permeten a l'administrador comprovar certes condicions abans d'iniciar l'unit. La diferència és que aquí, si la condició no es compleix, s'emet un error.

OnFailure=unaunit.service unaaltra.service ...
Indica les units que s'activaran quan la unit en qüestió entri en estat "failed". Aquesta directiva pot fer-se servir, per exemple, per executar una unit que envïi un correu electrònic quan la unit en qüestió, que podrà ser un servei, falli.

AllowIsolate=yes
Aquesta directiva només té sentit per units de tipus target. Si el seu valor és "yes" (per defecte és "no") indica que el target en qüestió admetrà que se li apliqui la comanda *systemctl isolate* (veure més avall)

D'altra banda, la darrera secció (encara que l'ordre no importa) d'un arxiu de configuració d'una unit sempre sol ser l'anomenada **[Install]** , la qual, atenció, és opcional. S'utilitza per definir com i quan l'unit pot ser activada o desactivada. Algunes de les seves directives més habituals són:

WantedBy=untarget.target unaltre.target ...
Indica els targets on l'unit en qüestió s'activarà en executar la comanda *systemctl enable* Quan s'executa aquesta comanda, el que passa és que per cada target indicat aquí apareixerà, dins de cada carpeta "/etc/systemd/system/nomTarget.wants" respectiva, un enllaç simbòlic apuntant al propi arxiu de configuració de l'unit en qüestió. L'existència d'aquest enllaç és el que realment activa de forma efectiva un servei automàticament. Eliminar els links de totes les carpetes "nomTarget.wants" pertinents implica desactivar la unit (que és el que fa, de fet, la comanda *systemctl disable* a partir de la llista de targets que troba a la línia *WantedBy=*). Per exemple, si l'arxiu de configuració de la unit en qüestió (que anomenarem pepito.service) té una línia com WantedBy=multi-user.target, en executar systemctl enable pepito.service apareixerà dins de la carpeta /etc/systemd/system/multi-user.wants un link apuntant a aquest arxiu de configuració

RequiredBy=untarget.target unaltre.target ...
Similar a WantedBy= però on la fallada de l'unit en qüestió en executar *systemctl enable* farà que els targets indicats aquí no es puguin arribar a assolir. La carpeta on es troba el link de l'unit en aquest cas s'anomena "/etc/systemd/system/nomTarget.requires"

Alias=unaltrenom.tipusUnit
Permet a l'unit en qüestió ser activada amb *systemctl enable* utilitzant un altre nom diferent

Also=unservei.service unaltre.service ...
Permet activar o desactivar diferents units com a conjunt. La llista ha de consistir en totes les units que també es volen tenir habilitades quan la unit en qüestió estigui habilitada

Secció [Service] (per units de tipus .service):

Depenent del tipus d'unit que tinguem ens podrem trobar amb diferents seccions específiques dins del seu fitxer de configuració, normalment escrites entre la secció [Unit] del principi i la secció [Install] del

final (si hi és). En el cas de les units de tipus "service", per exemple, ens trobem amb la secció específica anomenada **[Service]**, la qual pot incloure diferents directives com les següents:

NOTA: Les units de tipus device, snapshot i target no tenen seccions específiques

Type=maneraDarrancar

Hi ha diferents mètodes per iniciar un servei, i el mètode escollit, el qual dependrà del tipus d'executable a posar en marxa, s'ha d'indicar en aquesta directiva. Les possibilitats més comunes són:

simple: El servei nativament es queda en primer pla de forma indefinida i és Systemd qui el posa en segon pla (li crea un fitxer PID, el para quan calgui, etc). Systemd interpreta que el servei està llest tan bon punt l'executable associat es posa en marxa (encara que això sigui massa aviat perquè no estigui llest encara per rebre peticions)

forking: El servei nativament ja es posa en segon pla. Systemd interpreta que el servei està llest quan passa efectivament a segon pla. En aquest cas convé indicar també la directiva PIDFile=/ruta/fitxer.pid per a què Systemd tingui un control sobre quin procés és el que està en segon pla i el pugui identificar

oneshot: Útil per scripts, que s'executen (fent *systemctl start* igualment) un cop i finalitzen. Systemd s'esperarà fins que el procés finalitzi i interpreta que està llest quan hagi finalitzat. Es pot considerar l'ús de la directiva *RemainAfterExit=yes* per a "enganyar" a Systemd dient-li que el servei continua actiu encara que el procés hagi finalitzat; en aquest cas, la directiva *ExecStop=* no s'arribarà a fer efectiva mai.

També hi ha les possibilitats dbus (similar a "simple" però Systemd interpreta que està llest quan el nom indicat a BusName= ha sigut adquirit), idle (similar a "simple" però amb l'execució retardada fins que no s'executi res més; es pot utilitzar aquest mètode, per exemple, per emetre un so just després de la finalització de l'arranc del sistema.) i notify (el sistema més complet, on s'estableix un canal de comunicació intern entre el servei i Systemd per tal de notificar-se estats i events via l'API pròpia de Systemd sd_notify() i on Systemd interpreta que està llest quan rep l'estat corresponent a través d'aquest canal; si volem que scripts facin servir aquest mètode cal usar la comanda *systemd-notify*)

ExecStart=/ruta/executable param1 param2 ...

Indica la comanda (i paràmetres) a executar quan es realitza un *systemctl start* Si la ruta de l'executable comença amb un guió ("-"), s'acceptaran valors de retorn diferents de 0 com a vàlids.

NOTA: Podem utilitzar fins i tot la directiva **SuccessExitStatus=** per indicar quin valor considerem com a sortida exitosa del programa

ExecStartPre=/ruta/executable param1 param2 ...

Indica la comanda (i paràmetres) a executar abans de la indicada a ExecStart. Poden haver més d'una línia ExecStartPre al mateix arxiu, executant-se llavors cadascuna per ordre. La ruta de l'executable també pot anar precedida d'un guió ("-"), amb el mateix significat

ExecStartPost=/ruta/executable param1 param2 ...

Indica la comanda (i paràmetres) a executar després de la indicada a ExecStart. Poden haver més d'una línia ExecStartPost al mateix arxiu, executant-se llavors cadascuna per ordre. La ruta de l'executable també pot anar precedida d'un guió ("-"), amb el mateix significat

ExecStop=/ruta/executable param1 param2 ...

Indica la comanda (i paràmetres) a executar quan es realitza un *systemctl stop* Cal tenir en compte que en el cas d'un servei de tipus "oneshot", si no s'especifica la directiva *RemainAfterExit=yes*, la comanda indicada a ExecStop s'executarà automàticament just després d'ExecStart.

ExecStopPost=/ruta/executable param1 param2 ...

Indica la comanda (i paràmetres) a executar després de la indicada a ExecStop. Poden haver més d'una línia ExecStartPost al mateix arxiu, executant-se llavors cadascuna per ordre.

Restart={ always | no | on-success | on-failure | ... }

Indica les circumstàncies sota les que Systemd intentarà reiniciar automàticament un servei que hagi finalitzat. En concret, el valor "always" indica que en qualsevol tipus de finalització es tornarà a intentar reiniciar; el valor "no" indica que en cap finalització s'intentarà reiniciar, el valor "on-success" indica que només s'intentarà reiniciar si la finalització ha sigut correcta i "on-failure" si la finalització no ho ha sigut degut a qualsevol tipus de fallada (ja sigui que s'ha sobrepassat el temps d'espera de l'arranc o l'apagada, que s'ha retornat un valor diferent de 0, etc)

NOTA: Es podria donar el cas de què un servei estigués reiniciant-se tota l'estona. Amb **StartLimitBurts=** es pot configurar el número màxim de vegades que es vol que es reinicïi i amb **StartLimitIntervalSec=** es pot configurar el temps durant el qual es comptarà aquest número màxim de vegades. Si s'arriba a aquest número dins d'aquest temps, el servei no es tornarà a reiniciar automàticament i tampoc no es podrà iniciar manualment fins passat el temps indicat (moment en el qual es torna a comptar). També existeix la directiva **StartLimitAction=** , la qual serveix per indicar l'acció a realitzar quan s'arriba al número màxim de reinicis; el seu valor per defecte és "none" però pot valer també "reboot" (reinici net), "reboot-force" (reinici abrupte) i "reboot-immediate" (reinici molt abrupte)

RestartSec = nºs

Indica el número de segons que Systemd s'esperarà en reiniciar el servei després de què s'hagi aturat (si així ho marca la directiva Restart=).

TimeoutSec=nº

Indica el número de segons que Systemd esperarà a què el servei en qüestió s'inicïi o s'aturi abans de marcar-lo com a "failed" (i reiniciar-lo si fos el cas degut a la configuració de la directiva Restart=). Es pot indicar específicament un temps d'espera només per l'inici amb la directiva **TimeoutStartSec=** i un altre temps d'espera diferent per l'apagada amb la directiva **TimeoutStopSec=**. Si no s'especifica res, s'agafa el valor per defecte (5 min) que està indicat a /etc/systemd/system.conf

RemainAfterExit=yes

Tal com ja ho hem comentat, aquesta directiva s'utilitza en serveis de tipus "oneshot" per a què la directiva ExecStop= no s'executi en acabar l'execució de la comanda sinò en fer *systemctl stop*

PIDFile= /ruta/fitxer.pid

Tal com ja ho hem comentat, aquesta directiva s'utilitza en serveis de tipus "forking" per a senyalar a Systemd quin serà el fitxer PID utilitzat pel servei de manera que el pugui controlar més fàcilment.

StandardOutput= { null | tty | journal | socket }

Indica on s'imprimirà la sortida estàndar dels programes indicats a les directives ExecStart=, i ExecStop= . El valor "null" representa el destí /dev/null. El valor "tty" representa un terminal (ja sigui de tipus virtual -/dev/ttyX- o pseudo -/dev/pts/X-), el qual haurà de ser especificat mitjançant la directiva **TTYPath=** . El valor "journal" és el valor per defecte. El valor "socket" serveix per indicar que la sortida ha de enviar-se al socket associat al servidor per tal de viatjar a l'altre extrem de la comunicació (veure més endavant).

NOTA: També existeix la directiva **StandardError= { null | tty | journal | socket }** , similar a StandardOutput= però per la sortida d'error