Systemd

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 <u>després</u> 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 <u>abans</u> 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

Targets

Podem definir un "target" com un "estat" del sistema definit per un determinat conjunt de serveis posats en marxa (i uns altres que no). La idea és que, en arrencar el sistema, s'arribi a un determinat "target" (i, opcionalment, a partir d'allà, poder passar a un altre si fos necessari). A continuació es llisten els "targets" més importants (tots ells ubicats dins de /usr/lib/systemd/system):

poweroff.target (o "runlevel0.target") Si s'arriba a aquest "target", s'apaga el sistema

reboot.target (o "runlevel6.target"): Si s'arriba a aquest "target", es reinicia el sistema

rescue.target (o "runlevel1.target"): Si s'arriba a aquest "target", s'inicia el sistema en mode text, sense xarxa i només per l'usuari root. Seria similar a un altre target anomenat "**emergency.target**", però l'"emergency" és més "radical" que el "rescue" perquè gràcies a muntar la partició arrel en mode només lectura permet arrencar sistemes que el "rescue" potser no pot.

multi-user.target (o "runlevel3.target") : En aquest cas s'inicia el sistema en mode text però amb xarxa i multiusuari (el target per defecte en servidors)

graphical.target (o "runlevel5.target") : En aquest cas, s'inicia el sistema en mode gràfic amb xarxa i multiusuari (el target per defecte en sistemes d'escriptori) Implica haver passat pel target "multi-user" prèviament.

Altres targets predefinits que s'instal.len amb Systemd i que cal conèixer són:

ctrl-alt-del.target
Target activat quan es pulsa CTRL+ALT+SUPR. Per defecte és un enllaç a "reboot.target"

sysinit.target
Target que executa els primers scripts d'arranc

sockets.target
Target que activa, en arrencar, totes les units de tipus "socket". Es recomana, per tant, que tots els arxius de configuració d'una unit "socket" tinguin a la seva línia Wants= aquest target indicat (o bé WantedBy=)

timers.target
Target que activa, en arrencar, totes les units de tipus "timer". Es recomana, per tant, que tots els arxius de configuració d'una unit "timer" tinguin a la seva línia Wants= aquest target indicat (o bé WantedBy=)

paths.target
Target que activa, en arrencar, totes les units de tipus "path". Es recomana, per tant, que tots els arxius de configuració d'una unit "path" tinguin a la seva línia Wants= aquest target indicat (o bé WantedBy=)

swap.target
Target que habilita la memòria swap

basic.target
Target que posa en marxa tots els target relacionats amb punts de muntatge, memòries swaps, paths, timers, sockets i altres unitats bàsiques necessàries pel funcionament del sistema.

initrd-fs.target

El generador systemd-fstab-generator afegeix automàticament les units indicades a la directiva Before= d'aquesta unit a la unit especial "sysroot-usr.mount" (a més de tots els punts de muntatge existents a /etc/fstab que tinguin establertes les opcions "auto" i "x-initrd.mount"). Veure més endavant una explicació de les units de tipus mount.

initrd-root-fs.target

El generador systemd-fstab-generator afegeix automàticament les units indicades a la directiva Before= d'aquesta unit a la unit especial "sysroot-usr.mount", la qual és generada a partir dels paràmetres del kernel. Veure més endavant una explicació de les units de tipus mount.

local-fs.target

El generador systemd-fstab-generator afegeix automàticament les units indicades a la directiva Before= d'aquesta unit a totes les units de tipus "mount" que es refereixen a punts de muntatge locals. També afegeix a aquest target les dependències de tipus Wants= corresponents als punts de muntatge existents a /etc/fstab que tenen l'opció "auto" establerta. Veure més endavant una explicació de les units de tipus mount.

network-online.target

Target que s'activa automàticament tan bon punt el subsistema de xarxa és funcional. Qualsevol servei que necessiti treballar en xarxa s'haurà d'iniciar com a mínim en aquest target

printer.target

Target que s'activa automàticament tan bon punt una impressora és endollada o apareix disponible durant l'arranc. Aquí on se sol iniciar, per exemple, el servei Cups.

sound.target

Target que s'activa automàticament tan bon punt una tarja d'àudio és endollada o apareix disponible durant l'arranc.

bluetooth.target

Target que s'activa automàticament tan bon punt un controlador Bluetooth és endollat o apareix disponible durant l'arranc.

smartcard.target

Target que s'activa automàticament tan bon punt un controlador Smartcard és endollat o apareix disponible durant l'arranc.

system-update.target

Target especial utilitzada per actualitzacions del sistema. El generador systemd-system-update-generator redireccionarà el procés d'arranc automàticament a aquest target si la carpeta /system-update existeix

umount.target

Target que desmunta tots els punts "mount" i "automount" durant l'apagada del sistema

final.target

Target utilitzat durant l'apagada del sistema que pot fer-se servir per apagar els últims serveis després de què els serveis "normals" ja s'han aturat i els punts de muntatge s'han desmuntat.

Per saber el target on ens trobem en aquest moment podem fer:

systemctl get-default

Cal tenir en compte que múltiples targets poden estar activats a la vegada. Un target activat indica que Systemd ha intentat iniciar totes les units associades a aquest target. Això vol dir que la comanda anterior només ens diu quin és el target "final" on hem arribat, però al llarg del camí des de l'arranc de la màquina fins arribar a aquest target "final" s'han anat activant diferents targets a mode de "graons" intermitjos. Per veure tots els targets activats, cal fer **systemctl list-units –type=target**

Es pot canviar el target actual a un altre simplement executant...:

systemctl isolate nomTargetDesti.target

...però per canviar el target per defecte on s'anirà a parar automàticament a cada arranc del sistema es pot fer:

systemctl set-default nomTargetDefecte.target

La comanda anterior, en realitat l'únic que fa és revincular el link /etc/systemd/system/default.target a l'arxiu *.target adient.

NOTA: Una altra manera d'entrar al final de l'arranc del sistema en un determinat target per defecte és afegir la línia *systemd.unit=nomTargetDesti.target* a la llista de paràmetres del kernel indicada a la configuració del gestor d'arranc

Hi ha una sèrie de comandes específiques per passar a determinats estats (poweroff, reboot, etc) que es poden fer servir en comptes de la comanda *systemctl isolate* genèrica. Per exemple:

sudo systemctl rescue : Similar a *systemctl isolate rescue.target*

sudo systemctl poweroff (o **sudo poweroff** a seques): Similar a *systemctl isolate poweroff.target*

sudo systemctl reboot (o **sudo reboot** a seques): Similar a *systemctl isolate reboot.target*

Si es vol aturar (-P) o reiniciar (-r) la màquina en un moment futur determinat (hh:mm), llavors caldrà executar la comanda: **sudo shutdown {-P | -r } *hh:mm***

Si es vol aturar (-P) o reiniciar (-r) la màquina d'aquí a una certa quantitat de minuts, llavors caldrà executar la comanda: **sudo shutdown {-P | -r } +*m***

D'altra banda, amb la comanda **sudo systemctl suspend** podem suspendre el sistema i amb la comanda **sudo systemctl hibernate** la podem posar a hivernar. El primer significa que es guarda tot l'estat del sistema a la RAM i s'apaga la majoria de dispositius de la màquina; quan s'engega de nou, el sistema restaura el seu estat previ de la RAM sense haver de reiniciar-se de nou: aquest procés és molt ràpid però té l'inconvenient de que obliga a mantenir amb alimentació elèctrica la màquina tota l'estona. El segon significa que es guarda tot l'estat del sistema al disc dur (si té espai lliure) i s'apaga del tot la màquina: quan s'engega de nou, el sistema restaura el seu estat previ des del disc dur sense haver de reiniciar-se de nou: aquest procés és força lent però té l'avantatge de no haver de mantenir amb alimentació elèctrica la màquina.

Si es vol realitzar una tasca llarga i assegurar-se de què la màquina no es suspendrà o apagarà mentrestant, es pot invocar la comanda corresponent a aquesta tasca així: *systemd-inhibit comanda_llarga* La comanda *systemd-inhibit --list* mostra les tasques que tenen aquest truc en marxa.

Per a què l'inici *amb systemctl start* d'un determinat servei (o target) es produeixi dins d'un target determinat -anomenem-lo "a.target"- des del propi fitxer de configuració del servei en qüestió cal escriure les directives *Wants=a.target*, *Requires=a.target* i/o *After=a.target* (aquestes directives s'asseguren d'arribar al target "a.target" per iniciar llavors el servei en qüestió). D'altra banda, també existeix la directiva *Conflicts=a.target* , la qual s'assegura de **no** estar en el target "a.target" per poder iniciar el servei en qüestió.

En el cas de voler iniciar sempre un servei determinat automàticament en el target "a.target", llavors caldrà escriure a més les directives *WantedBy=a.target* o *RequiredBy=a.target* del fitxer de configuració del servei (en aquest darrer cas, en fer *systemctl enable nomServei* es crea un enllaç al seu arxiu de configuració dins de "/lib/systemd/system/a.target.wants").

Els arxius de configuració dels targets només tenen seccions [Unit] (i molt poques la secció [Install]). En aquest sentit, és interessant consultar els arxius corresponents, per exemple, a multi-user.target o graphical.target: només trobem les directives Description, Documentation, Wants, Requires, After, Conflicts i AllowIsolate (i a partir d'elles podem deduir les dependències que hi ha entre targets...encara que per això hi ha comandes específiques que de seguida veurem).

Boot chain

Per saber la jerarquia de dependències de targets per arribar a iniciar un target (o service!) determinat es pot utilitzar la comanda:

NOTA: Una manera alternativa de obtenir una informació similar seria executant *systemctl show -p "Wants" nomTarget.target && systemctl show -p "Requires" nomTarget.target* . També es pot executar *systemctl status*

systemctl list-dependencies nomTarget.target (o nomUnit.service)

Les dependències mostrades es corresponen a les units que han sigut "required" o "wanted" per les units superiors. Les dependències recursives només es mostren pels targets intermitjos; si es volen veure també pels service, mounts, paths, socket, etc intermitjos cal incloure el paràmetre **--all** a la comanda anterior.

També es poden mostrar quines units depenen per funcionar del correcte inici d'un target (o service!) determinat amb la comanda:

NOTA: Una manera alternativa de obtenir una informació similar seria executant *systemctl show -p "WantedBy" nomTarget.target && systemctl show -p "RequiredBy" nomTarget.target*

systemctl list-dependencies --reverse nomTarget.target (o nomUnit.service)

Altres paràmetres interessants d'aquesta comanda són **--before** i –**after**, els quals serveixen per mostrar les units que depenen per funcionar del correcte inici anterior o posterior d'un target, respectivament.

D'altra banda, respecte l'arranc del sistema podem obtenir una informació més detallada sobre els temps que triga cada unit en carregar-se i l'ordre en què ho fa gràcies a la comanda *systemd-analyze*, la qual té vàries possibilitats

systemd-analyze : Mostra el temps total emprat en l'arranc del sistema i quina part d'aquest temps ha sigut emprat en tasques del kernel, quina part en ús de l'initrd i quina part en tasques d'usuari

systemd-analyze blame : Mostra els temps disgregats per servei. Cal indicar que aquests temps són "en paral.lel", així que la suma total que surt serà sempre molt superior al temps real emprat en l'arranc.

systemd-analyze dot [nomTarget.target] | dot -T {png | svg} -o foto.{png|svg} : Genera una sortida que si es passa a l'aplicació "dot" (pertanyent al paquet "graphviz") generarà finalment un gràfic

(en format png o svg) on es poden visualitzar totes les dependències del target (o servei!) indicat (o, si no s'indica, del "default.target"; també es poden indicar comodins al nom del target/servei).

systemd-analyze plot [nomTarget.target] > something.svg : Genera un gràfic on es mostra els temps d'execució i de bloqueig de cada unit durant l'arranc fins arribar al target (o servei!) indicat (o, si no s'indica, del "default.target")

systemd-analyze critical-chain [nomTarget.target] : Mostra l'arbre de dependències bloquejants pel target (o servei!) indicat. El temps mostrat després de "@" indica el temps que fa que la unit està activa; el temps mostrat després de "+" indica el temps que la unit ha trigat en activar-se.

Altres opcions de la comanda systemd-analyze són *syscall-filter* , *verify* , *dump* , *set-log-level* o *time*

Plantilles

Una plantilla és un fitxer de configuració de tipus "service" que té la particularitat de permetre posar en marxa variants d'un mateix servei sense haver d'escriure un arxiu "service" diferent per cada variant. Bàsicament, per utilitzar una plantilla cal fer els següents passos:

1.-L'arxiu "service" que farà de plantilla s'ha d'anomenar "nomServei@.service" . És a dir, cal indicar el símbol arroba abans del punt

2.-El contingut d'aquest arxiu plantilla pot ser exactament igual que el d'un arxiu "service" estàndar

3.-A l'hora d'iniciar, parar, habilitar, deshabilitar, veure l'estat, etc d'una plantilla, caldrà indicar l'identificador concret de la variant amb la què volem treballar. Aquest identificador s'estableix la primera vegada que arrenca la variant i simplement consisteix en una cadena entre l'arroba i el punt, així: *systemctl start nomServei@**identificador.**service* A partir d'aquí, aquest identificador es farà servir de la mateixa manera per la resta de tasques relacionades amb la gestió d'aquesta variant

NOTA: An instance file is usually created as a symbolic link to the template file, with the link name including the instance identifier. In this way, multiple links with unique identifiers can point back to a single template file. When managing an instance unit, systemd will look for a file with the exact instance name you specify on the command line to use but if it cannot find one, it will look for an associated template file.

4.-La gràcia de les plantilles és que l'identificador indicat al punt anterior es pot utilitzar dinàmicament dins del contingut de l'arxiu plantilla (concretament mitjançant el símbol "**%i**"), de manera que segons el valor concret de l'identificador indicat (disponible, ja diem, mitjançant "%i"), es posarà en marxa el servei escoltant en un port diferent, o bé utilitzant un arxiu de configuració diferent, o el que ens convingui.

NOTA: Altres símbols especials que es poden indicar en un arxiu de configuració d'una plantilla poden ser

%p: Represents the unit name prefix (this is the portion of the unit name that comes before the @ symbol)
%n: Represents the full resulting unit name (%p plus %i)
%u: The name of the user configured to run the unit.
%U: The same as above, but as a numeric UID instead of name.
%H: The host name of the system that is running the unit.
%%: This is used to insert a literal percentage sign.

Posem un exemple. Imaginem que tenim un determinat servidor web que volem executar amb dues configuracions diferents a la vegada. La solució seria crear un fitxer plantilla anomenat per exemple "servidorweb@.service" amb un contingut similar al següent:

[Unit]
Description=My HTTP server
[Service]
Type=simple
ExecStart=/usr/sbin/webServer --config-file /etc/%i.conf
[Install]
WantedBy=multi-user.target

Amb aquest arxiu, es podria iniciar llavors el servidor dues vegades, cadascuna indicant el nom del fitxer de configuració desitjat, així:

$ sudo systemctl start servidorweb@config1.service
$ sudo systemctl start servidorweb@config2.service

Les comandes anteriors el que faran serà executar, respectivament, les comandes:

/usr/sbin/webServer –config-file /etc/config1.conf
/usr/sbin/webServer –config-file /etc/config2.conf

Sockets

Un aspecte molt interessant de Systemd és que permet que un servidor no estigui permanentment encés sinó que només arrenqui "sota demanda" (és a dir, quan detecti una connexió, normalment externa). D'aquesta manera, aquest servidor no consumeix més recursos que els mínims imprescindibles, en el moment just. Per aconseguir això, el que passa és que sí que hi ha un component "escoltant" tota l'estona possibles intents de connexions, però aquest component no és pas la unit "service" en sí sinò un "gos guardià" que només despertarà la unit "service" quan calgui. Aquest "gos guardià" és la unit de tipus "socket".

Cada fitxer de configuració d'una unit "socket" ha de tenir exactament el mateix nom que el fitxer de configuració de la unit "service" que vol despertar (és a dir, si tenim el servei "a.service", el socket corresponent haurà d'anomenar-se "a.socket"). La idea és tenir la unit "socket" sempre encesa (*systemctl enable a.socket*) però la unit "service" no (*systemctl disable a.service*); quan es detecti una connexió, el "socket" automàticament encendrà la unit "service" (això es pot veure fent *systemctl status a.service* mentre existeix la connexió) i l'apagarà de nou passat un determinat temps sense activitat (per defecte 5 minuts). Òbviament, si paréssim el "socket" (*systemctl stop a.socket*) o el deshabilitéssim pel proper reinici (*systemctl disable a.socket*) ja no hi hauria "gos guardià" amatent i, per tant, el servei ja no es posaria mai en marxa automàticament.

Per canviar el port on escolta un "socket" (entre altres coses) cal modificar la configuració del "socket" pròpiament dit i això no depén de la configuració del servidor en qüestió. Els arxius de configuració de cada "socket" es poden trobar, com qualsevol altra unit, o bé dins de la carpeta /lib/systemd/system o bé dins de /etc/systemd/system i es pot utilitzar igualment la comanda *systemctl edit a.socket* per tal de generar arxius "override". La secció que ens interessa en aquests tipus de fitxers és la secció **[Socket]**, la qual pot contenir alguna de les següents directives més importants:

ListenStream=[IP:]nºport

Indica el número de port TCP per on escoltarà el socket. Opcionalment, es pot indicar una IP concreta per especificar que només escoltarà en el port oferit per aquella IP i cap més.

NOTA: Es poden indicar vàries línies *ListemStream* per fer que el socket escolti en varis ports a la vegada. D'altra banda, com que aquesta línia pot estar escrita en diferents fitxers, si es vol assegurar que només s'escolti en un port concret sense tenir en compte altres línies que pugui haver llegit Systemd prèviament, es pot afegir primer una línia *ListemStream* buida (així: *ListemStream=*) i després la línia *ListenStream* associada al port desitjat; el que fa la línia *ListemStream* buida és "resetejar" totes les línies *ListemStream* anteriors

ListenDatagram=[IP:]nºport

Indica el número de port UDP per on escoltarà el socket. Opcionalment, es pot indicar una IP concreta per especificar que només escoltarà en el port oferit per aquella IP i cap més.

ListenSequentialPacket=/ruta/arxiu.socket

Indica el socket de tipus UNIX per on s'escoltarà. Només serveix per comunicacions entre processos de la mateixa màquina

Service=unNomAlternatiu

Si el nom de l'arxiu "service" no és igual que el nom de l'arxiu "socket", aquí es pot indicar llavors el nom que té l'arxiu "service" per a què el socket el sàpiga trobar.

Accept=yes

Si s'indica, fa que es generi una instància del servei diferent per cada connexió. Útil quan es fan servir plantilles (veure més endavant). Si el seu valor és no (per defecte) només una instància del servei gestionarà totes les connexions.

La comanda *systemctl status *.socket* ens permet saber quants i quins sockets estan escoltant ara mateix; el valor Accepted mostra quantes connexions s'han realitzat en total des de què el socket ha sigut iniciat i el valor Connected mostra quantes connexions estan actualment actives

Gestió dels noms locals de la màquina:

El nom "static" és el nom de la màquina tradicional, escollit per l'usuari a la instal.lació del sistema i guardat al fitxer /etc/hostname. El nom "transient" és un nom dinàmic mantingut pel kernel i inicialitzat a partir del nom "static" per defecte; pot canviar via DHCP o mDNS durant l'execució del sistema. El nom "pretty" és una petita descripció de la màquina per presentar-la a l'usuari.

A més de l'arxiu /etc/hostname hi ha altres arxius importants, com ara /etc/os-release (el qual emmagatzema el nom de la distribució, la seva versió, la seva pàgina web, etc), /etc/machine-id (el qual conté un identificador únic per cada màquina, utilitzat a altres eines) i /etc/machine-info (el qual conté informació extra sobre la màquina com ara el seus noms "transient" i "pretty", etc).

Les comandes més importants relacionades amb aquests fitxers són:

hostnamectl [status] : Mostra els tres tipus de nom de màquina a més d'altra informació emmagatzemada als arxius /etc/os-release, /etc/machine-id i /etc/machine-info (i /proc/cmdline)

hostnamectl set-hostname nom [--transient | --pretty | --static] : Canvia el nom de la màquina. Si no s'indica cap paràmetre, s'aplica el nom indicat a tots tres tipus: "transient", "pretty" i "static"; si s'indica algun en concret, només es canvia aquest.

La comanda *hostnamectl* en realitat no és més que un client D-Bus que realitza les peticions mitjançant aquesta via (ja siguin de consulta o d'actualització de dades) al servei *systemd-hostnamed*, el qual es posa en marxa només quan és rep la petició (no està tota l'estona funcionant). Això significa que la mateixa funció que podem fer amb la comanda *hostnamectl* es pot realitzar mitjançant qualsevol eina que permeti fer crides al canal D-Bus, com ara *dbus-send*, *gdbus* o la pròpia de Systemd, *busctl*. Concretament, el funcionament bàsic d'aquesta darrera comanda és aquest:

busctl {--system | --user } call ***nom.Servei /nom/Objecte nom.Interfície mètode [signatures paràmetres]***
busctl {--system | --user } get-property ***nom.Servei /nom/Objecte nom.Interfície propietat***
busctl {--system | --user } set-property ***nom.Servei /nom/Objecte nom.Interfície propietat signatura valor***
busctl {--system | --user } introspect ***nom.Servei /nom/Objecte [nom.Interfície]***

on "signatura" vol dir el tipus de paràmetre indicats, que pot indicar-se amb una "i" si és de tipus sencer de 32 bits, una "s" si és de tipus "string", "b" si és de tipus booleà, "ai" si és un array de sencers, "as" si ho és de cadenes, etc (per saber tots els tipus possibles, consultar https://dbus.freedesktop.org/doc/dbus-specification.html#type-system)

Gestió de les "locales":

Una "locale" és una configuració regional definida pel sistema que inclou principalment el llenguatge i pel mapejat del teclat però també el format dels números, hores, dates, telèfons, monedes, unitats de mesura, etc. Per esbrinar la "locale" actualment definida es pot executar la comanda...:

localectl [status]

...la qual simplement mostra el contingut dels arxius /etc/locale.conf i /etc/vconsole.conf. El que es veu representen les diferents variables d'entorn que defineixen un determinat aspecte de la configuració regional. La variable LANG representa la configuració genèrica. Per conèixer les implicacions exactes d'una determinada variable (és a dir, els valors i formats que defineix) es pot executar **locale -ck LC_NOMVARIABLE** Per veure totes les locales instal.lades al sistema es pot fer:

localectl list-locales

Per instal.lar una nova "locale" (per exemple, el francès, "fr_FR", a Ubuntu es pot executar la comanda **check-language-support -l *fr_FR*** ; aquesta comanda mostra tots els paquets que caldrien instal.lar per disposar d'aquesta "locale" plenament incorporada al sistema. A Fedora només cal instal.lar el metapaquet adient a la "locale" desitjada, el qual té el nom de "langpacks-XX" (on "XX" representa la "locale" escollida)

Per establir temporalment (mentre el terminal actual estigui obert) una "locale" només cal fer **export LANG=fr_FR.utf8** Per establir permanentment una "locale" (gravant-se a l'arxiu /etc/locale.conf) en canvi cal fer el següent (el canvi es veurà al següent inici de sessió):

localectl set-locale LANG=en_GB.utf8 (o LC_*=...)
NOTA: Es pot modificar a mà l'arxiu ~/.config/locale.conf per sobreescriure aquesta configuració general

Per veure tots els mapejats de teclat disponibles es pot fer:

localectl list-keymaps

Per establir permanentment un mapejat de teclat (gravant-se a l'arxiu /etc/vconsole.conf) es pot fer:

localectl set-keymap en_GB
NOTA: Existeix el paràmetre del kernel vconsole.keymap=... que permet sobreescriure durant l'arranc del sistema la configuració escrita a /etc/vconsole.conf

Cal aclarir que la comanda anterior només funciona en terminals virtuals degut a què en entorns gràfics la configuració del teclat ve determinada per la indicada al panell de control de l'escriptori.

La comanda *localectl* en realitat no és més que un client D-Bus que realitza les peticions mitjançant aquesta via (ja siguin de consulta o d'actualització de dades) al servei *systemd-localed*, el qual es posa en marxa només quan és rep la petició (no està tota l'estona funcionant). Això significa que la mateixa funció que podem fer amb la comanda *localectl* es pot realitzar mitjançant qualsevol eina que permeti fer crides al canal D-Bus, com ara *dbus-send*, *gdbus* o la pròpia de Systemd, *busctl*

Gestió dels "seats":

Systemd-logind is a system service that manages user logins. It is responsible for the following:

*Keeping track of users and sessions, their processes and their idle states,
*Creating control groups for user processes,
*Providing PolicyKit-based access for users to operations such as system shutdown or sleep,
*Implementing a shutdown/sleep inhibition logic for applications,
*Handling of power/sleep hardware keys,
*Multi-seat management, session switch management, and device access management for users,
*Automatic spawning of text logins (gettys) on virtual terminal (console) activation and user runtime directory management.

Its config file is /etc/systemd/logind.conf, where most interesting directives are:

NAutoVTs=nº : Number of available virtual terminals. If you want all [Fx] keys to start a getty, increase the value of NAutoVTs to 12. Remember you can also pre-activate gettys which will be running from boot simply doing *systemctl enable getty@tty9.service*

KillUserProcesses=yes : Configures whether the processes of a user should be killed when she or he completely logs out (i.e. after her/his last session ended). Defaults to "no" because of popular using of tmux and similar software. You can specify a list of usernames affected by this directive with **KillOnlyUsers=** directive (or, alternatively, not affected by it with **KillExcludeUsers= .** Former defaults to empty list and latter (which has precedence in case of conflict) defaults to "root"

IdleAction= { ignore | poweroff | reboot | suspend | hibernate | lock } : Configures the action to take when the system is idle. Defaults to "ignore". The system will execute the action after all user sessions report that they are idle, no idle inhibitor lock is active, and the time configured with **IdleActionSec=** has expired (this last directive configures the delay after which the action configured in IdleAction= is taken after the system is idle)

HandlePowerKey= Triggered when the power key/button is pressed. Defaults to poweroff
HandleSuspendKey= Triggered when the suspend key/button is pressed. Defaults to suspend
HandleHibernateKey= Triggered when the hibernate key/button is pressed. Defaults to hibernate
HandleLidSwitch= Triggered when the lid is closed, except in cases below. Defaults to suspend
HandleLidSwitchDocked= Triggered when the lid is closed if the system is inserted in a docking station or more than one display is connected. Defaults to ignore
The specified action for each event can be the same as in IdleAction directive

Comandes interessants són **loginctl list-users , loginctl list-sessions , loginctl list-seats , loginctl show-user *nomUsuari*** o **loginctl show-session nº .** També, per sistemes "multiseat", poden ser útils les comandes **loginctl seat-status *seatX*** o **loginctl attach *seatX /sys/devices/.../videoCard***

Gestió del temps:

Els sistemes operatius distingeixen dos tipus de rellotges diferents:

*Un rellotge de temps real (RTC), sovint referit com a "hardware clock" perquè sol ser un circuit integrat a la placa base, el qual és completament independent de l'estat del sistema operatiu i funciona fins i tot quan la computadora està apagada gràcies a l'alimentació elèctrica proporcionada per la pila botó.

*Un rellotge de sistema, sovint referit com a "software clock", el qual és mantingut pel kernel. El seu valor inicial es basa en l'obtingut del RTC però una vegada el sistema ha arrencat i el rellotge del sistema inicialitzat, aquest és completament independent del RTC. No obstant, degut a què a cada reinici es torna a "sincronitzar" amb el RTC, el "software clock" només és autònom durant el funcionament del sistema.

Al sistemes Linux moderns, en comptes de la comanda *date*, per conèixer l'hora i data del sistema és més complet (ja que ens mostra tant l'hora i data local com la UTC -la del "meridià de Greenwitch"- i la RTC -que ha de ser sempre igual a la UTC-, a més de la zona horària) fer servir la comanda:

timedatectl [status]

Per canviar ja sigui la data, l'hora o les dues coses, tant a nivell de "software clock" com de "hardware clock" a la vegada, la comanda a executar és aquesta:

timedatectl set-time "nºany-nºmes-nºdia nºhora:nºmin:nºseg"

La comanda anterior no funcionarà si hi ha algun servei NTP habilitat a la màquina (veure més endavant). D'altra banda, cal tenir en compte que l'hora que s'ha d'indicar ha de ser sempre la LOCAL, ja que *timedatectl* ja en deduirà la corresponent hora UTC/RTC convenientment.

NOTA: La comanda *date* també es podria fer servir per modificar la data i hora però només ho fa a nivell de "software clock". Alternativament, tenim la comanda *hwclock*, la qual pot modificar la data i hora del RTC. Però no farem servir cap d'aquestes dues ja que amb *timedatectl* ja en tenim prou.

La zona horària del sistema està indicada a un arxiu anomenat /etc/localtime, que no és més que un accés directe a /usr/share/zoneinfo/Europe/Madrid (o a l'arxiu associat a la zona indicada durant la instal.lació del sistema; també podria ser /usr/share/zoneinfo/UTC). Per saber totes les zones horàries possibles no cal, però, fer cap llistat manual del contingut de la carpeta /usr/share/zoneinfo perquè la següent comanda ja ens ho fa automàticament:

timedatectl list-timezones

Canviar de zona horària, doncs, seria tan senzill com fer que /etc/localtime apuntés a un altre arxiu que estigui dins de /usr/share/zoneinfo (aquests arxius s'instal.len amb el paquet "tzdata"). Això es pot fer "manualment" amb la comanda *ln -s* o bé, més fàcil, amb la comanda:

timedatectl set-timezone "Europe/Madrid"

La comanda *timedatectl* també és capaç de sincronitzar automàticament el nostre rellotge (tant software com hardware, recordem) amb servidors NTPs públics d'Internet (veieu http://www.pool.ntp.org) si passen dues coses: que haguem executat la comanda **timedatectl set-ntp on** i que al sistema estigui en marxa algun dimoni-client NTP com ara "chrony" o bé **systemd-timesyncd** , el qual és el client NTP per defecte de Systemd. The NTP servers contacted are determined from the global settings in */etc/systemd/timesyncd.conf* and, with more precedence, from the per-link static settings in .network files (with the NTP= option), and the per-link dynamic settings received over DHCP. If no NTP server information is acquired after completing those step, the NTP server host names or IP address defined in FallbackNTP= line from timesyncd.conf file will be used.

NOTA: Si s'executa *timedatectl set-ntp on* i només existeix *systemd-timesyncd* com a client NTP del sistema, aquest serà iniciat automàticament.

Units "mount" i generators:

L'arxiu /etc/fstab és prescindible a sistemes Systemd ja que cada punt de muntatge de dispositiu es pot definir en una unit de tipus "mount", la qual podrà estar habilitada (*enable*) a partir d'un determinat "target" d'arranc (concretament, l'indicat a la seva directiva WantedBy=). De totes maneres, per mantenir la compatibilitat, existeix un programa especial anomenat "/usr/lib/systemd/system-generators/systemd-fstab-

generator", executat automàticament per Systemd durant el seu inici, que s'encarrega de llegir les línies que estiguin presents a /etc/fstab i transformar-les en respectives units de tipus "mount" (les quals, en aquest cas, s'ubicaran dins de /run/systemd/generator).

D'altra banda, el fet de tractar els punts de muntatge de dispositius com units permet que el fet de muntar-los i desmuntar-los es transformi simplement en un inici o aturada de la seva unit corresponent.

Els arxius de les units "mount" s'han d'anomenar obligatòriament com la ruta del punt de muntatge que representen però eliminant la primera barra ("/") corresponent a la carpeta arrel i substituïnt la resta d'eventuals barres que apareguin a la ruta per guions ("-").Per exemple, si tenim una unit que representa el punt de muntatge "/mnt/pepe", el seu arxiu de configuració s'haurà d'anomenar "mnt-pepe.mount". A més de les seccions "[Unit]" i "[Install]" comunes a la resta d'units, les de tipus "mount" disposen d'una secció **[Mount]** que pot incloure, entre d'altres, les següents directives:

What=/dev/disp1 : Dispositiu a muntar. El valor d'aquesta directiva també pot ser *UUID=unUuid* o *LABEL=unaEtiqueta*, entre altres

Where=/ruta/punt/muntatge : Ha de coincidir amb el nom de la unit (canviant les barres per guions). No cal que existeixi prèviament

Type= { ext4 | vfat | xfs | ... } Sistema de fitxers del dispositiu a muntar.

Options = { defaults | ... } : Qualsevol opció de muntatge vàlida a /etc/fstab es pot indicar aquí. Si n'hi ha més d'una, s'escriuran totes seguides separades per comes.

TimeoutSec=nº : Indica la quantitat de temps (per defecte segons) que el sistema esperarà abans de considerar que el muntatge ha fallat.

NOTA: En el cas de punts de muntatge remots com ara NFS o Samba, a la secció [Unit] dels arxius *.mount sol haver les directives *Requires=network-online.target* i *After=network-online.target* (o similars) per assegurar-se de què la xarxa estigui habilitada abans d'intentar realitzar el muntatge.

Els "generators" són petits executables que Systemd posa en marxa a l'inici de la seva posada en marxa per tal de realitzar diferents tasques bàsiques variades. En general, transformen certes configuracions que no són "natives" de Systemd a units que sí que ho són. És a dir, són generadors d'units dinàmiques en cada arranc. Aquests petits programes es poden ubicar en les següents carpetes (ordenades per ordre d'exeucució i, per tant, de sobrescriviment):
/run/systemd/system-generators
/etc/systemd/system-generators
/lib/systemd/system-generators

NOTA: Note that the order of the two directories with the highest priority is reversed with respect to the unit load path, and generators in /run overwrite those in /etc. A symlink to /dev/null or an empty file can be used to mask a generator, thereby preventing it from running.

Units "automount":

Les units de tipus "automount" han de venir acompanyades sempre d'una unit de tipus "mount" homònima. És a dir, no poden existir units "automount" sense la seva parella "mount". Les units "automount" serveixen, si estan iniciades, per no haver de tenir iniciada la corresponent unit "mount" mentre aquesta no s'utilitzi. D'aquesta manera, no es té muntat el dispositiu en qüestió inútilment. Només quan l'usuari hi accedeix, en aquell moment, la unit "mount" s'iniciarà sola i l'usuari podrà entrar al punt de muntatge sense notar que fins llavors el seu contingut no estava disponible. En aquest sentit, les units "automount" serien a les "mount" com les "socket" són a les "service".

Les units "automount" disposen, a més de les seccions [Unit] i [Install] conegudes, de la secció **[Automount]** la qual només sol estar composta d'un parell de directives només:

Where=/ruta/punt/muntatge : Ha de coincidir amb el nom de la unit (canviant les barres per guions). No cal que existeixi prèviament. Representa el punt de muntatge que s'estarà "monitoritzant" per detectar si s'hi accedeix i, per tant, si s'ha d'iniciar la unit "mount" associada

TimeoutIdleSec=nº : Indica la quantitat de temps (per defecte segons) d'inactivitat en el punt de muntatge en qüestió que Systemd s'esperarà per desmuntar-lo. Per defecte aquest temps és infinit.

Units "timer":

Les units "timer" s'usen per programar tasques o bé a) en un/s moment/s concret/s, o bé b) després d'un (o cada) cert temps. Una altra unit (generalment de tipus "service") homònima amb el "timer" ha d'existir per tal de ser executada quan el moment definit al timer arribi. Convé aclarir que aquesta unit "service" no cal que tingui cap secció [Install] perquè no cal habilitar-la explícitament: només caldria habilitar, si es volgués automatitzar la seva execució, el seu "timer" associat, el qual sí tindrà una secció [Install] (amb una línia WantedBy= generalment apuntant al target predefinit "timers.target").

A més de les seccions [Unit] i [Install], el fitxer de configuració del "timer" té una secció [Timer] que pot contenir alguna de les següents directives:

*Per un moment/s concret/s:

OnCalendar=... : Activa la unit associada al moment exacte indicat. El format general que pot tenir el valor d'aquesta directiva és: *[nomDiaSetmana] any-mes-dia hora:minuts:segons* on:

-Els dies de la setmana s'han d'escriure en anglès i en forma abreviada: Mon, Tue, Wed, ...
-Es pot escriure el comodí * a qualsevol unitat (any, mes, dia, hora, minuts, segons)
-Es poden escriure varis valors en una unitat si se separen per comes
-Es pot indicar un rang de valors en una unitat escrivint ".." entre el primer i últim valor
-Es pot indicar un interval en una unitat escrivint "/nº". Per exemple, *:*:20/15 vol dir a les *:*:20, *:*:35, *:*:50 i, com que s'acaba el minut, tornarà a començar un altre cop a *:*:20

Aquesta directiva també admet determinades paraules clau com a valors, les quals representen determinades combinacions de dia i hora. Concretament:

minutely → *-*-* *:*:00
hourly → *-*-* *:00:00
daily → *-*-* 00:00:00
monthly → *-*-01 00:00:00
weekly → Mon *-*-* 00:00:00
yearly → *-01-01 00:00:00
quarterly → *-01,04,07,10-01 00:00:00
semiannually → *-01,07-01 00:00:00

NOTA: Expressions like daily and weekly refer to specific start times and thus any timers sharing such calendar events will start simultaneously. Timers sharing start events can cause poor system performance if the timers' services compete for system resources. The **RandomizedDelaySec** option in the [Timer] section avoids this problem by randomly staggering the start time of each timer

*Després (i/o cada cert) temps:

OnBootSec=nº : Indica la quantitat de temps que Systemd s'esperarà a realitzar (una vegada) la tasca, començant a contar a partir de l'arranc del sistema (útil quan el "timer" està *enabled*)

OnStartupSec=nº : Similar a l'anterior, però l'espera es conta en relació a quan el propi Systemd ha arrencat (útil quan el "timer" està *enabled*)

OnActiveSec=nº: Indica la quantitat de temps que Systemd s'esperarà a realitzar (una vegada) la tasca, començant a contar a partir de l'activació del "timer" (útil quan el "timer" no està *enabled* i es fa un *systemctl start*)

OnUnitActiveSec=nº : Indica la quantitat de temps que Systemd s'esperarà a realitzar (de forma periòdica) la tasca, començant a contar des de la primera vegada que es va realitzar (generalment mitjançant *OnBootSec=*)

Les quantitats de temps indicades a les directives anteriors per defecte són segons però afegint algun dels següents sufixes es pot transformar en altres unitats de temps: us (microsegons), ms (milisegons), s (segons), m (minuts), h (hores), dies (d), setmanes (w), M (mesos) o y (anys). Fins i tot es poden combinar, com ara així, per exemple: "5m 20s" (5 minuts i 20 segons).

La directiva OnCalendar= no és excloent amb les altres: es poden escriure a la vegada si és necessari. Altres directives interessants són:

Persistent=yes : Si aquesta directiva val "yes", el "timer" en qüestió executarà la tasca associada inmediatament després d'arrencar el sistema si li hagués tocat executar-se abans mentre el sistema estava apagat. Se sol fer servir juntament amb la directiva OnCalendar=.
NOTA: Això s'aconsegueix gràcies a què es guarda al disc dur la darrera vegada que es va executar la tasca

WakeSystem=yes : Si aquesta directiva val "yes" permet treure el sistema de l'estat de suspensió si li toca executar una tasca programada durant aquest estat. Per defecte val *no* (*false*)

Unit=nomUnit.service : Indica la tasca a realitzar quan arribi el moment indicat pel "timer". Només cal escriure-la si el nom del fitxer de configuració de la tasca (l'arxiu "service") no és igual al del fitxer de configuració del "timer".

AccuracySec=nº : Indica la precisió del "timer" (per defecte és 1 minut). Per tasques que es vulguin realitzar cada cert segons (o a un segon determinat) cal indicar explícitament *AccuracySec=1s* Es pot arribar fins els microsegons de precisió

Com qualsevol altra unit, es poden veure la llista de timers amb la comanda *systemctl list-units -t timer* però a més disposem de la comanda específica **systemctl list-timers**, la qual informa de quan va ser la darrera vegada que es va executar la tasca, quant li queda per tornar-la a executar, etc

A l'igual que passava amb les units "service", les units "timer" també es poden executar en el context d'un usuari particular (és a dir, no cal que estiguin activats per tot el sistema en general). Per aconseguir això només cal crear l'arxiu *.timer (i l'arxiu *.service associat) dins d'alguna de les carpetes reservades per aquest context (és a dir, per ordre de sobreescritura: "/usr/lib/systemd/user", "~/.local/share/systemd/user", "/etc/systemd/user" o "~/.config/systemd/user") i executar les comandes adients (*systemctl daemon-reload, systemctl start, systemctl enable, systemctl list-timers...*) amb el paràmetre *--user*

Existeix una manera d'establir tasques programades sense haver de crear un arxiu *.timer gràcies a la comanda *systemd-run* , la qual ens permet crear "timers" (i altres tipus d'units) de tipus "transient" (és a dir, "fugaços"). Per exemple, per crear un fitxer d'aquí a 30 segons es pot fer el següent:

systemd-run --on-active=30 /bin/touch /tmp/foo

També es pot fer servir per executar un arxiu "service" preexistent que no tingui cap "timer" associat:

systemd-run –on-calendar="*-*-* 12:13:14" --unit unServei.service

Hardering:

Existeixen una sèrie de directives, que s'han d'escriure dins de la secció [Service] d'un arxiu de tipus "service", les quals permeten restringir les capacitats dels processos corresponents de manera que s'executin d'una manera més segura i controlada. Algunes d'aquestes directives són:

PrivateTmp=yes : Si val yes, la unit "service" veurà un directory /tmp particular, privat i aïllat de la carpeta /tmp genèrica del sistema, la qual històricament s'ha fet servir per tenir un espai compartit per tots els serveis i usuaris (amb el conseqüent problema de seguretat).

NOTA: Si es vol aconseguir el mateix però per usuaris en comptes de per processos (és a dir, aconseguir que cada usuari tingui una carpeta /tmp privada separada de la de la resta d'usuaris) es pot fer servir el mòdul PAM "pam_namespace" com s'explica aquí: https://airtower.wordpress.com/2010/06/26/trying-out-pam_namespace/

PrivateNetwork=yes : Si val yes, la unit "service" es desconnectarà completament de qualsevol tipus de xarxa (excepte la "loopback").

DeviceAllow=/dev/dispositiu rw : Limita l'accés al dispositiu indicat (i només a aquest) amb els permisos indicats

ProtectSystem=yes : Si val yes, la unit "service" tractarà /usr i /boot com a de només lectura. Pot tenir també el valor "full"; en aquest cas també s'hi afegirà /etc com a carpeta de només lectura.

ProtectHome=yes : Si val yes, la unit "service" no podrà entrar ni a /home ni a /root ni a /run/user. Pot tenir també el valor "read-only"

També estan les directives **ReadOnlyPaths= i InaccessiblePaths=** , les quals permeten especificar una llista de rutes absolutes (separades per espais) que seran de només lectura o inaccessibles, respectivament (i de forma recursiva per tot el seu interior).

LimitCPU=nº : Indica el número màxim de segons que la unit en qüestió podrà fer servir la CPU

LimitNPROC=nº : Indica el número màxim de processos-fill que la unit podrà generar. Una directiva similar és **TasksMAX=nº**

LimitNOFILE=nº : Indica el número màxim de descriptors de fitxers que la unit podrà generar

LimitFSIZE=nº{K|M|G} : Indica el tamany màxim que podran tenir els fitxers generats per l'unit

Tots els límits "LimitXXX" no configurats explícitament tenen un valor màxim configurat per defecte a /etc/systemd/system.conf, en les directives DefaultLimitCPU=, DefaultLimitFSIZE=, etc. Si no estan configurades allà tampoc, es fan servir els valors de fàbrica del kernel.

MemoryMax=nº{K|M|G} : Indica la quantitat màxima de RAM que podrà usar la unit en qüestió

A https://www.freedesktop.org/software/systemd/man/systemd.resource-control.html es poden consultar més directives d'aquests tipus. Per exemple, existeixen moltes directives més per limitar l'ús de la CPU o moltes altres per limitar l'entrada/sortida (I/O) de/als dispositius de bloc (discos) o interfícies de xarxa, etc.

Es pot executar un "service" (o directament qualsevol comanda) i limitar en aquell moment les seves "capacitats" mitjançant les directives anteriors però sense haver d'escriure-les a cap arxiu de configuració de cap "unit": si simplement executem amb *systemd-run* el service o comanda indicant llavors la directiva ("propietat") que volem definir per aquella execució en concret amb el paràmetre *-p*, ja ho tindrem. Per exemple, *systemd-run -p BlockIOWeight=10 updatedb* executa la comanda updatedb però limitant el pes de la seva I/O a 10

> NOTA: Un altre paràmetre interessant de systemd-run és -t, el qual permet executar comandes interactives. D'aquesta manera, per exemple, podríem executar /bin/bash indicant certs límits per tots els processos executats des d'ell.