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 l'arxiu "**/etc/systemd/system.conf**" ; molts valors per defecte hi són allà establerts. Al llarg d'aquest document anirem estudiant alguns d'ells.

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

"Units": tipus i ubicació del seus arxius de configuració

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*
.target : Defineix un grup d'units, l'estat de les quals (aturades/enceses) determina l'estat del sistema
.mount : Descriu un punt de muntatge gestionat per Systemd
.automount : Descriu un punt d'automuntatge associat a un *.mount*
.timer: Descriu la temporització/activació d'una tasca programada gestionada per Systemd
.device : Descriu un dispositiu hardware reconegut pel kernel (via udev o sysfs) gestionat per Systemd
.swap : Descriu una partició o fitxer d'intercanvi gestionat per Systemd
.path : Descriu una carpeta o fitxer monitoritzat per l'API Inotify del kernel
.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

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 homònims 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 (l'últim guanya!)

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

NOTA: També existeix la possibilitat de tenir arxius *.conf dins d'una carpeta anomenada "service.d" (o "socket.d", "target.d", etc) dins de "/usr/lib...", "/run/..." o "/etc/...". Aquests arxius *.conf serien similars als descrits al paràgraf anterior però són més genèrics (afecten a totes les units del tipus en qüestió) i, per tant, tenen menys prioritat (és a dir, són sobreescribibles tant per aquells com pels arxius de configuració canònics de cada unit concreta).

NOTA: 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). En parlarem més endavant.

Systemd també permet definir serveis que no estan associats al sistema global sinó que únicament formen part de la sessió d'un usuari estàndard.

NOTA: Això és possible gràcies a què, just després de cada inici de sessió realitzat al sistema, es posa en marxa automàticament (gràcies al mòdul PAM "pam_systemd") una instància particular de Systemd (en forma de dimoni, *systemctl status user@nºUID.service)* per aquell usuari en qüestió. La primera instància que aparegui és l'encarregada d'executar la comanda *systemd --user* , la qual es qui permet realment aquest funcionament individual per totes les sessions d'usuaris que s'inicïin a partir de llavors (el procés amb PID 1 pròpiament dit és *systemd –system*). Cal dir que les instàncies es paren automàticament a mesura que l'usuari associat surt de la seva sessió però el procés *systemd --user* finalitzarà només després d'haver-se tancat el darrer inici de sessió existent al sistema. Veieu *man user@.service* i *systemctl cat user@nºUID.service* per més informació.

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 de menor a major):

" **/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 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* ... que veurem tot seguit 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* (veure més avall) 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. Es pot indicar també, de totes formes, alguna unit no llistada a *Wants=* o *Requires=* (limitant-se llavors només a una definició temporal)

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. Es pot indicar també, de totes formes, alguna unit no llistada a *Wants=* o *Requires=* (limitant-se llavors només a una definició temporal)

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 no s'iniciarà. 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, en executar la comanda *systemctl enable* (veure més avall) , l'unit en qüestió s'activarà automàticament als propers reinicis.

NOTA: 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 d'una unit (que anomenarem "pepito.service") tingués 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 si l'unit en qüestió fallés en voler-se iniciar automàticament en arribar a algun target dels indicats, farà que aquests targets no es puguin assolir. La carpeta "/etc/systemd/system/nomTarget.requires" és on es troba el link de l'unit en aquest cas.

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. Els tipus més comuns 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, els quals es podran executar un cop (amb *systemctl start*, veure més avall) fins finalitzar (moment en el qual l'unit tornarà a estar "stopped"). 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, caldria llavors executar manualment *systemctl stop* per indicar a Systemd que l'script/comanda ha finalitzat.

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*

NOTA: No caldria escriure la ruta absoluta de l'executable si aquesta es troba a la llista de rutes que mostra la comanda *systemd-path*, però en general es recomana escriure-la per evitar sorpreses
NOTA: No es permeten escriure redireccionadors (">",">>","<"."|") ni el símbol "&" per passar a segon pla
NOTA:Si la ruta de l'executable comença amb un guió ("-"), valors de retorn de la comanda diferents de 0 (que normalment es considerarien senyal d'error) es consideraran com a vàlids.
NOTA: Podem utilitzar fins i tot la directiva **SuccessExitStatus=*nº*** 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. Un exemple de possible ús: l'enviament d'un correu just després d'haver-se posat en marxa el servei corresponent. 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* Si aquesta directiva no s'especifica, els processos associats a la unit en qüestió seran finalitzats per defecte mitjançant l'enviament de la senyal indicada a la directiva **KillSignal=*nº*** , la qual, si tampoc està especificada, s'assumirà que és SIGTERM (nº15)

NOTA: Recordem que en el cas d'un servei de tipus "oneshot", si s'especifica la directiva *RemainAfterExit=yes*, la comanda indicada a ExecStop= s'executarà quan executem manualment *systemctl stop*, i si no s'especifica, la comanda indicada a ExecStop= s'executarà automàticament quan just finalitzi l'execució de la comanda

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/apagada, que s'ha retornat un valor !=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)

NOTA: Amb la directiva **RestartSec = *nºs*** es pot indicar el número de segons que Systemd s'esperarà en reiniciar el servei (si així ho marca la directiva Restart=) a comptar després de què s'hagi aturat. Per defecte són 100ms.

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" si no ho aconsegueix (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"

User={ *nomUsuari | UID* } ; Group={ *nomGrup | GID* }

Defineix, respectivament, l'usuari o el grup en què s'executaran els processos associats al servei en qüestió (si no es defineix cap grup, s'utilitza el grup principal de l'usuari). Per als serveis del sistema (serveis administrats pel PID 1 -és a dir, per *systemd --system* -), el valor per defecte és "root". Per als serveis d'usuari (és a dir, serveis administrats per *systemd –user*), no es permet canviar la identitat d'usuari (així que aquesta directiva no té gaire sentit)

WorkingDirectory=*/ruta/carpeta*

Indica el directori de treball del servei en qüestió. Si no s'especifica aquesta directiva, el valor per defecte és "/" (en el cas de serveis de sistema) o $HOME (en el cas de serveis d'usuari). Si la ruta es precedeix amb un símbol "-", el fet de què la carpeta corresponent no existeixi no s'interpretarà com un error.

NOTA: Si s'ha indicat la directiva *User=*, es pot escriure "~" com a valor d'aquesta directiva, equivalent així a la ruta de la carpeta personal de l'usuari indicat a User=.

StandardOutput= *{ null | tty | journal | socket | file:/ruta/fitxer.txt | append:/ruta/fitxer.txt }*

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 (és a dir, que si el programa en qüestió imprimís alguna cosa a la pantalla del terminal en executar-se en primer pla, aquesta sortida es redireccionarà al Journal en executar-se via arxiu .service) . 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). El valor "file" indica la ruta del fitxer de text concret on s'imprimirà la sortida (si no existeix es crearà i, en qualsevol cas, se sobreescriu) i el valor "append" és similar a "file" però permet afegir contingut sense sobreescriure.

NOTA: El fet de què per defecte la sortida estàndar vagi a parar al Journal es pot canviar de forma general per totes les units a la directiva *DefaultStandardOutput* del fitxer "/etc/systemd/system.conf"
NOTA: També existeix la directiva **StandardError= { null | tty | journal | socket | file:/ruta/fitxer.txt | append:/ruta/fitxer.txt }**, similar a *StandardOutput=* però per la sortida d'error

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 carregades a RAM però "inactives"
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)

NOTA: Com a valor del paràmetre *--state* també es pot posar qualsevol valor vàlid de la columna SUB (es pot obtenir la llista completa fent *systemctl --state help*)

NOTA: 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 (la llista sencera es pot veure fent *systemctl --state=help*): "plugged", "mounted", "running", "exited", "waiting","listening"...

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-. 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 són usades només com a dependència d'alguna altra unit (i per tant no han d'executar-se per sí mateixes)

"disabled" : La unit està desactivada i, per tant, no es posarà en marxa als següents reinicis (gràcies a la inexistència en "/etc/systemd/system/*nomTarget*.target.wants" de l'enllaç corresponent). 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)

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

"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 "generator"

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

Activa/Desactiva/Reinicia la unit indicada immediatament seguint les indicacions escrites al seu arxiu de configuració corresponent. També existeix l'opció *reload* per rellegir l'arxiu de configuració en qüestió sense haver de reiniciar la unit però no totes les units la suporten

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 s'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 mostra en pantalla aquest estat.

systemctl is-active nomUnit[.service]

Retorna $?=0 si la unit indicada està activa i 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 mostra en pantalla aquest estat.

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.

NOTA: 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*

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...:

```
cups.service - CUPS Scheduler
   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 línies de journald -u (es pot usar els paràmetres homònims -n nº i -o xxx)
```

*Els valors possibles per la línia "Loaded:" són els mateixos que apareixen a la columna LOAD de *list-units* Tot seguit 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 a la comanda s'indica un PID en comptes de una unit, es pot veure la mateixa informació. Indicar el PID és útil quan, per exemple, es vol conèixer la unit a la què està associat un determinat procés (es mostra a línia Loaded:"), tot i que aquesta informació també es pot conèixer executant simplement *ps -e -o pid,comm,unit* o similar

systemctl cat nomUnit[.service]

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

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

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,..."* sols es mostren les parelles clau<->valor desitjades.

systemctl help nomUnit[.service]

Obre la pàgina de man associada a la unit indicada (al seu fitxer de configuració ha d'estar 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" envia directament el senyal en qüestió a tots els processos del grup mentre que "stop" executa la forma oficial de tancar el servei, configurada mitjançant l'ordre indicada amb *ExecStop=* al fitxer de servei corresponent. Normalment "stop" hauria de ser suficient; "kill" és útil quan es penja.

EXERCICIS:

Tots els exercicis es faran en una màquina virtual on l'usuari pugui tenir permisos d'administrador.

0.-Systemd pot no ser el procés INIT del nostre sistema Linux: encara que sigui el més extès amb diferència, pots trobar distribucions que facin servir sistemes INIT alternatius. Digues, de les següents comandes, quines serveixen per saber si el procés INIT del sistema és Systemd (o no): *file /sbin/init ; pgrep ^systemd$; man init*

1.-a) Executa *systemctl list-units -t service | grep ufw* (si estàs a Ubuntu) o *systemctl list-units -t service | grep firewalld* (si està a Fedora). ¿Què vol dir la paraula "loaded"? ¿I "active"? Confirma que a la sortida de *systemctl status ufw* (o *systemctl status firewalld)* també es mostra l'estat "loaded" i "active".

b) Ara enmascara la unit ufw/firewalld. ¿Què passa realment quan s'enmascara una unit? Pista: consulta on apunta el recentment creat arxiu "/etc/systemd/system/ufw.service" (o "/etc/systemd/system/firewalld.service")

c) Executa *systemctl list-units -t service | grep ufw* (o *systemctl list-units -t service | grep firewalld*) de nou. ¿Per què encara apareix la paraula "active"? Confirma-ho executant de nou *systemctl status ufw / systemctl status firewalld*

d) Si ara executes *systemctl stop ufw* (o *systemctl stop firewalld*), ¿què mostra ara la comanda *systemctl list-units -t service | grep ufw* (o *systemctl list-units -t service | grep firewalld*) ? ¿Per què? ¿Què hauries de fer per a què veiessis alguna cosa? Pista: fes servir el paràmetre *--all*

e) Executa la comanda *systemd-delta* . ¿Què significa la relació indicada entre els dos fitxers que apareixen a cada línia [MASKED]? ¿I entre els dos fitxers que apareixen a cada línia [EXTENDED]?

> NOTA: També hi podria haver alguna parella de fitxer en línies [OVERRIDEN] o fins i tot [EQUIVALENT]

f) Intenta iniciar l'unit ufw (o firewalld) encara enmascarada. ¿Pots? Desenmascara-la i torna-ho a provar. ¿Pots ara?

g) Executa la comanda *systemctl edit ufw* (o *systemctl edit firewalld*) i, a l'editor de text que apareix, escriu les línies següents i guarda. Tot seguit (aquí no cal executar *systemctl daemon-reload* ja que la comanda *systemctl edit* s'encarrega d'executar-la automàticament sempre), executa la comanda *systemctl cat ufw* (o *systemctl cat firewalld*). ¿Què veus? ¿I si executes *systemctl status ufw* (o *systemctl status firewalld*) quina descripció veus? ¿I si tornes a executar *systemd-delta* ?

> *[Unit]*
> *Description=Hola amigo*

> NOTA: L'editor obert per defecte en executar la comanda *systemctl edit* ve definit per la variable d'entorn $SYSTEMD_EDITOR. Si aquesta variable no està definida (ni tapoc $EDITOR ni $VISUAL), s'intentarà obrir per ordre els següents editors i el primer trobat serà l'utilitzat: *nano, vim, vi*

2.-a) Crea un fitxer anomenat "/etc/systemd/system/pepe.service" amb el següent contingut ...:

> *[Unit]*
> *Description=Pepe es colega*
> *[Service]*
> *Type=oneshot*
> *ExecStart=ls -l*
> *ExecStop=df -h*
> *[Install]*
> *WantedBy=multi-user.target*

...i tot seguit (després de *systemctl daemon-reload)* executa *systemctl start pepe* ¿Per què no veus cap sortida? ¿Què veus, en canvi, si fas *journalctl -u pepe* i per què? A partir de la sortida de la comanda anterior, dedueix què veuries si fas *systemctl -n 0 status pepe* i comprova-ho.

b) Què hauries de modificar de l'arxiu anterior per a què la comanda *df -h* no s'executés just després de *ls -l* sinò només quan s'escriguís *systemctl stop pepe*? Pista: consulta l'explicació de la directiva *RemainAfterExit=* a la teoria. Prova-ho utilitzant les comandes *systemctl --full edit pepe , systemctl daemon-reload* i, un altre cop, *journalctl -u pepe*

c) ¿Per a què serveix la línia *WantedBy=*... ? O dit d'una altra manera: ¿quina relació té aquesta línia amb la comanda *systemctl enable*? Pista: observa el contingut de la carpeta "/etc/systemd/system/multi-user.target.wants"

d) ¿Què hauries de canviar de l'arxiu anterior per a què la sortida de les comandes executades per la unit (ja sigui a ExecStart= o a ExecStop=) no vagi a parar al Journal sinó al terminal "/dev/pts/1"? Prova-ho.

3.-a) Crea un script anomenat "/opt/yeah.sh" amb el següent contingut (i dóna-li permisos d'execució):

```
#!/bin/bash
while [[ true ]] ;  do
        curl -s ipinfo.io/ip
        sleep 3
done
```

b) Crea un fitxer anomenat "/etc/systemd/system/pepa.service" amb el següent contingut i tot seguit (després de fer *systemctl daemon-reload)* executa *systemctl start pepa . S*i fas llavors *journalctl -f -u pepa,* ¿què veus? ¿Per què? ¿I si executes *systemctl stop pepa* , què veus llavors al Journal ? ¿Per què? ¿I si descomentes la línia que apareix comentada, executes *systemctl daemon-reload* i tornes a fer *systemctl start pepa*? ¿Per què?

```
[Unit]
Description=Pepa es colega
[Service]
Type=simple
ExecStartPre=systemd-cat -t PEPA -p crit  echo "Començo"
ExecStart=/opt/yeah.sh
ExecStop=systemd-cat -t PEPA -p crit  echo "Acabo"
#StandardOutput=null
```

4.-a) Crea un fitxer anomenat "fiufiu.service" dins de "/etc/systemd/system" amb el següent contingut i seguidament (després de fer *systemctl daemon-reload)* posa'l en marxa amb la comanda *systemctl start fiufiu* . Comprova llavors amb *systemctl status fiufiu* (o també amb *ss -tnl*) que s'hagi iniciat correctament

```
[Unit]
Description=All we are saying is give peace a chance
[Service]
Type=simple
ExecStart=ncat -l -p 5555
Restart=on-success
```

b) Executa la comanda *journalctl -f -u fiufiu* i seguidament, obre un altre terminal per executar-hi la comanda *ncat 127.0.0.1 5555*. Escriu-hi alguna cosa a la connexió oberta per aquest client Netcat i observa a la vegada el que apareix en temps real al Journal. ¿Què passa? ¿Per què?

c) Finalitza el client Netcat (amb CTRL+C, per exemple). Recorda que quan finalitzes un client Netcat, el comportament per defecte del servidor Netcat de l'altre extrem és finalitzar també. No obstant, el servei "fiufiu" continua funcionant (i per tant, pots tornar a connectar-t'hi de nou amb el client Netcat) ¿Per què? (per saber-ho pots observar els missatges de logs mostrats a la finestra on funciona *journalctl -f -u fiufiu*)

cII) Ara comenta la línia *Restart=...* que apareix al fitxer "fiufiu.service", reinicia el servei i torna a executar el client Netcat un parell de vegades. La primera vegada haurà de connectar-se sense problemes com sempre però la segona ja no. ¿Per què? (comprova-ho amb systemctl status fiufiu i raona la resposta)

d) ¿La comanda *systemctl enable fiufiu* funcionarà? ¿Per què?

5.-a) Instal.la el paquet "apache2" (a Ubuntu) o "httpd" (a Fedora) i observa, executant la comanda *systemctl cat apache2* (o *systemctl cat httpd*, respectivament) si hi apareix la directiva *Restart=*, i en cas que sí, quin valor té (si no apareix, el seu valor per defecte associat és llavors "no") . Executa llavors *sudo systemctl --signal=9 kill apache2* (o *httpd*) i comprova amb *systemctl status apache2* (o *httpd*) si el servei reinicia sol o no. ¿Per què passa el que passa?

b) Executa la comanda *systemctl edit apache2* (o *systemctl edit httpd*)i, a l'editor de text que apareix, escriu el següent:

A Ubuntu:
[Service]
Restart=no

A Fedora:
[Service]
Restart=on-failure

Després de fer *systemctl daemon-reload* (i de comprovar que la modificació és efectiva amb *systemd-delta* o també *systemctl cat apache2/ systemctl cat httpd)*, inicia el servei Apache un altre cop i comprova que efectivament estigui iniciat. Torna'l a matar un altre cop amb *sudo systemctl --signal=9 kill apache2* (o *httpd*) i torna a comprovar un altre cop si el servei ha reiniciat automàticament o no. ¿Què passa ara i per què?

c) Executa de nou la comanda *systemctl edit apache2* (o *httpd*), esborra el contingut escrit de l'apartat anterior i escriu ara aquestes altres línies noves:

[Unit]
OnFailure=sonar.service

NOTA: Si afegíssis les línies anteriors a les que hi havia escrites de l'apartat b) en lloc de substituir-les, el que es demana en aquest apartat no funcionaria. La raó està explicada a la descripció de la directiva *OnFailure=* a *man systemd.unit* : "A service unit using *Restart=* enters the failed state only after the start limits are reached" És a dir, la directiva *OnFailure=* no s'aplica si el servei logra reiniciar-se sol després d'una fallada.

El fitxer de configuració de la unit "sonar.service" cridada a la línia *OnFailure=* anterior haurà d'estar ubicat, com tots, dins de la carpeta "/etc/systemd/system" i haurà de tenir un contingut similar al següent. ¿Què passarà llavors cada cop que l'Apache finalitzi degut a alguna situació inesperada (com per exemple seria una senyal kill 9)?

[Unit]
Description=Sonar
[Service]
User=usuari
Group=usuari
Environment=XDG_RUNTIME_DIR=/run/user/1000
Type=oneshot
ExecStart=play /home/usuari/sonar.ogg

NOTA: La comanda *play* s'encarrega de reproduir el fitxer de so indicat i admet molts formats possibles, no només "ogg" ("mp3", "wav",...). Forma part del paquet "sox". Òbviament, la ruta indicada de l'arxiu de so és suggerida: pot ser qualsevol altra

NOTA: El valor de les línies *User=*, *Group=* ha de ser el nom de l'usuari que actualment tingui iniciada la sessió i el valor de la variable d'entorn XDG_RUNTIME_DIR (indicada a la línia *Environment=*) ha de ser "/run/user/XXX" on "XXX" ha de ser el UID de l'usuari que actualment tingui iniciada la sessió (aquesta dada es pot conèixer fàcilment simplement executant la comanda *id* en un terminal). La raó de ser d'aquestes tres línies és l'explicada a la següent nota.
NOTA: La comanda *play*, com qualsevol altre programa que reprodueixi sons en sistemes moderns, no emet ell per si sol el so sinó que l'envia a un altre programa anomenat PulseAudio, el qual s'encarrega de rebre en un sol lloc tots els sons dels diferents programes, gestionar-los, barrejar-los, controlar-los, etc (per això pots escoltar alhora un vídeo de Youtube i el Spotify, per exemple). El detall important aquí és que PulseAudio s'inicia a la sessió d'escriptori de cada usuari particular (és a dir, s'executa una instància de PulseAudio per cada usuari diferent). Si deixéssim l'arxiu "sonar.service" sense les línies *User=*, *Group=* i sense la definició de la variable XDG_RUNTIME_DIR, aquesta "unit" s'executaria per defecte en un entorn propi de "root", però "root" precisament no inicia sessió en cap escriptori, així que, com que no hi hauria cap instància de PulseAudio associat a ell, la comanda *play* enviaria el so enlloc. Amb aquestes tres línies indiquem que volem executar "sonar.service" sent un determinat usuari amb tot el seu entorn associat (incloent la seva instància PulseAudio particular), i així la comanda *play* enviarà el so a la instància PulseAudio adequada (sempre que tinguem la sessió iniciada amb aquest usuari en particular).

d) Modifica l'arxiu "sonar.service" anterior per a què ara la seva línia *ExecStart=* tingui aquest valor: *notify-send Error "Apache reiniciat!"*. ¿Què passarà ara cada cop que l'Apache finalitzi degut a alguna situació inesperada? ¿Per què creus que encara cal establir la variable XDG_RUNTIME_DIR?

6.-a) Crea, a la carpeta personal del teu usuari de la màquina real (suposarem que és "asix2"), i sent aquest usuari, el següent script amb el nom de "yeye.sh" i dóna-li permisos d'execució:

```
#!/bin/bash
if [[ $(id -nu) != "asix2" ]];then
    echo "Not asix2 user, exiting." &&  exit 1
fi
echo "Started" > /tmp/xyz
for i in {1..3}; do
    sleep 10
    echo "${i}0 seconds" >> /tmp/xyz
done
echo "Finished" >> /tmp/xyz
```

aII) ¿Què fa, aquest script? (pots executar-lo directament per provar-ho) ¿Per a què serveix el condicional del començament de l'script?

aIII) Crea l'arxiu "yeye.service" dins de la carpeta "/home/asix2/.config/systemd/user" (si cal, crea les subcarpetes necessàries) amb el següent contingut...:

```
[Unit]
Description=Run service as a user
[Service]
Type=simple
ExecStart=/home/asix2/yeye.sh
```

... i posa'l en marxa, sempre sent l'usuari "asix2" (fixa't que no cal ser "root" mai!), amb la comanda *systemctl --user start yeye* ¿Què mostra tot seguit la comanda *systemctl –user status yeye*? ¿I la comanda *tail -f /tmp/xyz*? ¿I la comanda *ps -e | grep yeye*? ¿I la comanda *systemctl –user cat yeye*? ¿Què passa quan t'esperes mig minut i tornes a fer *systemctl --user status yeye*, què veus llavors?

NOTA: Cada cop que modifiques l'arxiu "yeye.service" hauràs d'executar la comanda *systemctl --user daemon-reload*

b) Crea un arxiu anomenat "sonar2.service" dins de la carpeta "/home/asix2/.config/systemd/user" amb el següent contingut...:

[Unit]
Description=Sonar2
[Service]
Type=oneshot
ExecStart=play /home/asix2/sonar.ogg

... i posa'l en marxa, sempre sent l'usuari "asix2" (fixa't que no cal ser "root" mai!), amb la comanda *systemctl --user start sonar2* (òbviament, la ruta indicada de l'arxiu de so és suggerida: pot ser qualsevol altra) ¿Què passa? ¿Per què creus que no ha calgut especificar ara les línies *User=*, *Group=* i *Environment=XDG_RUNTIME_DIR* que vam haver d'escriure en un "service" similar a l'apartat c) de l'exercici anterior? (mira't les notes d'aquell apartat).

7.-a) De nou a la màquina virtual de treball, crea un arxiu anomenat "restaurar.service" dins de la carpeta "/home/usuari/.config/systemd/user" amb el següent contingut...:

[Unit]
Description=Restaurar carpeta personal
[Service]
Type=oneshot
ExecStart=/opt/restaurar.sh
[Install]
WantedBy=default.target

...on el codi de l'script "restaurar.sh" invocat (que ha de tenir permisos d'execució!) ha de ser aquest...

#!/bin/bash
*rm -rf /home/usuari/**
#Si la còpia estigués en un pendrive o disc extern, caldria muntar-lo abans: mount /dev/sdb1 /mnt ...
cd / && tar -xzf /opt/copiaUsuari.tgz
#...i desmuntar-lo tot seguit: umount /dev/sdb1

b) Habilita el servei executant *systemctl --user enable restaurar*

c) Crea un fitxer anomenat "canari.txt" dins de la teva carpeta personal i tot seguit guarda a "/opt" una "còpia de seguretat" comprimida de tota la teva carpeta personal, així (suposant que el teu usuari es diu "usuari"): *sudo tar -czf /opt/copiaUsuari.tgz /home/usuari*

d) Esborra el fitxer "canari.txt" i tot seguit reinicia la màquina. ¿Torna a aparèixer de nou aquest fitxer on hi era abans d'esborrar-lo? ¿Per què?

e) Torna a repetir exactament l'apartat anterior per confirmar que no ha sigut només un cop el que ha passat.

f) ¿Quina és la "gràcia" d'haver habilitat el servei "restaurar" només per l'usuari "usuari"? ¿Què passaria si executessis en un moment donat la comanda *systemctl --user start restaurar* essent un altre usuari?