# Systemd (II)

## 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 (la definició dels quals es troba 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. Aquest target és útil per intentar reparar algun error del sistema si aquest no pot arrencar de forma estàndard. Seria similar a un altre target anomenat "**emergency.target**", però aquest és més "radical" perquè munta la partició arrel en mode només lectura (això serveix per arrencar sistemes que el "rescue" potser no podria).

**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.mount", la qual és generada a partir dels paràmetres del kernel. Ho estudiarem més endavant

**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 hagi de treballar en xarxa s'haurà d'iniciar com a mínim en aquest target

NOTA: Existeix un altre tarjet relacionat amb la xarxa anomenat "pre-network.target" que està pensat per iniciar serveis abans de què qualsevol tarja de xarxa es configuri. El seu propòsit principal és fer-lo servi amb serveis de tipus tallafocs, per tal d'establir les regles abans de què la configuració de xarxa funcioni. Aquests serveis hauran de tenir una línia *Before=network-pre.target* i també una línia *Wants=network-pre.target* al seu arxiu de configuració

NOTA: Existeix un altre tarjet relacionat amb la xarxa anomenat simplement "network.target" que només indica que l'stack software de xarxa ja s'ha carregat en memòria però això no implica que les interfícies s'hagin configurat encara. Aquest target està més pensat pel procés d'apagada de la màquina per tal de realitzar aquest procés de forma ordenada: ja que l'ordre d'apagada és al revés que el d'arrencada, qualsevol unit que tingui una línia *After=network.target* s'apagarà abans que la xarxa es descarregui i això farà que aquesta unit s'apagui sense interrompre cap connexió que estigui pendent

**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.

**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 -t target*

Es pot canviar el target actual a un altre simplement executant: *systemctl isolate nomTargetDesti.target* 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*

> NOTA: La comanda anterior simplement revincula 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. En aquest sentit, a *man kernel-command-line* es poden consultar encara més paràmetres del kernel interessants relacionats amb la posada en marxa automàtica de "services" i "targets" Systemd a través de la configuració del gestor d'arranc emprat.

En el cas de voler iniciar sempre un servei determinat automàticament quan el sistema arribi al target "a.target", caldrà escriure les directives *WantedBy=a.target* (o *RequiredBy=a.target*) a la secció "[Install]" del fitxer de configuració del servei en qüestió i llavors executar *sudo systemctl enable nomServei* ; fent això es crearà un enllaç a aquest arxiu de configuració dins de la carpeta "**/usr/lib/systemd/system/a.target.wants**".

De forma complementària, si el que volem és que amb l'inici d'un determinat servei (o d'altres tipus d'units) amb *systemctl start* automàticament "s'entri" en un target determinat -anomenem-lo "a.target"-, des del fitxer de configuració del servei en qüestió caldrà escriure les directives *Wants=a.target*, *Requires=a.target* i, opcionalment, *Before=a.target* o *After=a.target* (aquestes darreres directives s'asseguren, respectivament, d'iniciar primer el servei en qüestió per tot seguit arribar al target "a.target" o bé d'arribar primer al target "a.target" per tot seguit iniciar el servei en qüestió; si no s'indica cap, l'arribada al target es farà de forma paral·lela a la posada en marxa del servei). D'altra banda, també existeix la directiva *Conflicts=a.target* , la qual s'assegura de **no** estar (i per tant, sortir-ne si cal) en el target "a.target" per poder iniciar el servei en qüestió.

D'altra banda, els arxius de configuració dels targets només tenen seccions *[Unit]* (i molt pocs 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).

Suspensions i hibernacions

Hi ha comandes específiques per passar a determinats estats ("poweroff", "reboot", etc) que es poden fer servir en comptes de la comanda *systemctl isolate* genèrica (excepte per anar a l'estat "rescue", no cal ser "root" per executar-les (sempre que no hi hagi cap sessió d'usuari iniciada més enllà de la nostra). Així:

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

> NOTA: A la comanda *systemctl reboot* li podem afegir varis paràmetres interessants que serveixen per modificar determinats valors de variables EFI concretes (tal com es podria fer també amb la comanda *efibootmgr*) per així alterar el comportament de la UEFI al proper arranc (no obstant, però, aquests paràmetres només funcionen en sistemes UEFI que hagin arrencat mitjançant el gestor d'arranc *systemd-boot)*:
>
> *--firmware-setup* : Indica que es al següent reinici s'entrarà directament dins el panell de control UEFI
> *--boot-loader-menu=nºsegons* : Indica que al següent reinici es mostrarà el menú del gestor d'arranc durant els segons indicats. Si s'indica 0, es deshabilitarà el "timeout" del menú
> *--boot-loader-entry=entryID* : Indica que es reiniciarà directament en l'entrada indicada del menú del gestor d'arranc Es pot indicar un identificador de l'entrada o "help" per veure la llista d'entrades disponibles

NOTA: Dins de la carpeta "**/usr/lib/systemd/system-shutdown**" poden haver arxius ***.shutdown,** que són Bash scripts executables que s'executaran just abans de l'apagada/reinici del sistema (és a dir, just en posar en marxa els serveis "poweroff.service" o "reboot.service"). Qui executarà aquests scripts és el binari **/usr/lib/systemd/systemd-shutdown**, el qual és invocat sempre per aquests serveis, que el col.loquen com a PID 1 i és el responsable de desmuntar dels sistemes de fitxers, deshabilitar la swap, matar els processos que quedin pendents, etc. Als scripts executats per /usr/lib/systemd/systemd-shutdown podem utilitzar un paràmetre ($1) que pot valer "poweroff" o "reboot" depenent de l'acció que realitzarà i que ens podria servir per distingir què volem que faci aquest script segons l'acció indicada. Tots els scripts s'executen en paral.lel. Cal tenir en compte, finalment, que el sistema de fitxers en aquell moment roman muntat però en mode només lectura.

D'altra banda, amb la comanda *systemctl suspend* podem suspendre el sistema (o dit d'una altra manera, ens permeten arribar al target "suspend.target"; no cal ser "root" si només tenim iniciada sessió nosaltres) i amb la comanda *systemctl hibernate* la podem posar a hivernar (o dit d'una altra manera, en permeten arribar al target "hibernate.target"; no cal ser "root"). "Suspendre" significa que manté 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. "Hibernar" 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.

NOTA: Dins de la carpeta "**/usr/lib/systemd/system-sleep**" poden haver arxius ***.sleep**, que són Bash scripts executables que s'executaran just abans de la hibernació o suspensió del sistema (és a dir, just en posar en marxa internament els serveis "systemd-hibernate.service" o "systemd-suspend.service", els quals, per cert, mai han de ser invocats directament amb *systemctl start ...* sinó fent servir les comandes explicades al paràgraf anterior: *systemctl hibernate* o *systemctl suspend*). Qui executarà aquests scripts és el binari **/usr/lib/systemd/systemd-sleep**, el qual és invocat sempre per aquests serveis i admet dos paràmetres que podem fer servir en aquests scripts com $1 i $2 respectivament. El primer paràmetre pot valer "pre" o "post" depenent de si la màquina està anant a la suspensió/hibernació o n'està tornant, respectivament. El segon paràmetre pot valer "suspend" o "hibernate" depenent de l'acció que realitzarà i que ens podria server per distingir què volem que faci aquest script segons l'acció indicada. Tots els scripts s'executen en paral.lel.

Si es vol realitzar una tasca llarga i assegurar-se de què la màquina no es pugui suspendre o apagar mentrestant, es pot invocar la comanda corresponent a aquesta tasca així: *systemd-inhibit comanda_llarga* Amb el paràmetre *--what=accio* es pot afegir l'acció concreta a inhibir, on "acció" pot ser, entre altres, la paraula "shutdown" (equivalent a apagada o reinicii) o "sleep" (equivalent a hibernació o suspensió). La comanda *systemd-inhibit --list* mostra les tasques que tenen aquest truc en marxa. Trobareu més informació als primers paràgrafs de https://www.freedesktop.org/wiki/Software/systemd/inhibit/

Apagades retardades

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* Com es pot veure, en aquest cas caldrà fer servir una comanda específica (la qual, en realitat, no és més que un enllaç a la mateixa comanda *systemctl poweroff* coneguda sols que indicant-li uns paràmetres que no estan -encara- disponibles a la pròpia comanda *systemctl poweroff*). Per cancel·lar una apagada/reinici previst, caldrà executar *sudo shutdown -c*

NOTA: El mecanisme intern utilitzar per "traspassar" aquests paràmetres de la comanda *shutdown* a *systemctl poweroff* és mitjançant el canal D-Bus de sistema. De fet, qui s'encarrega de gestionar les aturades (retardades o immediates) és el servei D-Bus "org.freedesktop.login1", implementat pel dimoni *systemd-logind*. Estudiarem els serveis D-Bus aviat.

Es pot comprovar si hi ha alguna aturada prevista en un futur consultant el contingut de l'arxiu "**/run/systemd/shutdown/scheduled**". Aquest fitxer consta de tres directives: **USEC**, que val el nombre de microsegons comptats des del 1-1-1970 que representa el moment exacte quan la màquina s'apagarà (sabent que la comanda *date -d @'xxx'* "tradueix" el valor indicat -però en segons!!- a una data humanament llegible, es pot executar el següent per saber la data de la propera apagada: *date -d @'$(head -1 /run/systemd/shutdown/scheduled |cut -c6-15)'* ); **WARN_ALL**, que és una directiva booleana que, si val 1, enviarà un missatge a tots els usuaris que tinguin iniciada una sessió al sistema (i també al Journal) avisant de la futura apagada i **MODE**, que indica si realment volem fer una aturada (valor "poweroff") o un reinici (valor "restart"), entre altres. Com es pot veure, en aquesta funcionalitat no intervé per res cap "timer".

**EXERCICIS:**

**1.-a)** Executa, des de la sessió d'un usuari qualsevol iniciada en una màquina virtual qualsevol, la comanda *systemctl* adient per anar al target de suspensió. ¿Què passa?

**b)** Reactiva el sistema i ara executa la comanda *systemctl* adient per entrar-hi en el target "multi-user". ¿Què passa ara?

**c)** Reinicia la màquina virtual i entra ràpid en el mode d'edició del seu menú del Grub (recorda que això es fa pulsant ESC per mostrar el menú i tot seguit pulsant "e" per editar l'entrada sel.leccionada). Un cop dins del mode d'edició, afegeix el paràmetre del kernel *systemd.unit=rescue.target* i continua seguidament amb l'arranc (pulsa "F10"). ¿Què passa?

> NOTA: Atenció, l'apartat anterior només funcionarà si l'usuari "root" té contrasenya definida al sistema!! Si no és el cas, una alternativa és escriure el paràmetre *init=/bin/bash* o bé, prèviament configurar el target "rescue" per a què accepti logins sense contrasenya; això últim simplement consisteix en executar la comanda *sudo systemctl edit rescue.service* i escriure el següent: *[Service]*
> *Environment=SYSTEMD_SULOGIN_FORCE=1*
> NOTA: Atenció, l'apartat anterior només funcionarà en Fedora si SELinux està en mode "permissive". Per fer això cal editar prèviament l'arxiu "/etc/selinux/config" (com a root) establint la línia *SELINUX=permissive* i reiniciar el sistema.

Tal com s'ha comentat a la teoria, existeixen més paràmetres del kernel destinats a Systemd a part de *systemd.unit=*; per exemple, *systemd.mask=nomUnit* pot servir per enmascarar una determinada unit (o més, si s'escriu vàries vegades) en aquell arranc concret per tal de depurar alguna errada del sistema mentre que *systemd.wants=nomUnit* pot servir pel contrari: per iniciar units addicionals automàticament en aquell arranc en concret. Aquests dos paràmetres, entre d'altres, són interpretats per un programa específic anomenat *systemd-debug-generator* Per més informació es pot consultar la pàgina *man kernel-command-line*

**cII)** Reinicia la màquina virtual de nou per entrar de la forma habitual al sistema. Deshabilita el servei "chronyd" (a Fedora) o "cron" (a Ubuntu), s'haurà posat en marxa sol i reinicia de nou per comprovar que ja no estigui funcionant. Torna a reiniciar però ara entra ràpid de nou en el mode d'edició del seu menú del Grub i, un cop allà dins, afegeix el paràmetre del kernel *systemd.wants=chronyd.service* (o *systemd.wants=cron.service*) i continua amb l'arranc (pulsa "F10"). Quan acabi l'arranc, ¿el servei "chronyd" estarà encès?

**2.-** Crea (o modifica, si ja el tenies fet d'exercicis anteriors) un fitxer anomenat "fiufiu.service" dins de "/etc/systemd/system" amb el següent contingut (les modificacions al fitxer preexistent estan en negreta) i tot seguit executa *systemctl enable fiufiu* i reinicia la màquina. :

> *[Unit]*
> *Description=Cold turkey*
> ***Requires=network-online.target***
> ***After=network-online.target***
> *[Service]*
> *Type=simple*
> *ExecStart=nc -l -p 5555*
> *Restart=on-success*
> ***[Install]***
> ***WantedBy=multi-user.target***

**a)** ¿Està el servei automàticament encès després el reinici i pots connectar-hi, per tant, directament amb la comanda *nc 127.0.0.1 5555* sense haver de fer res més?

**b)** ¿Què mostra la comanda *ls -l /etc/systemd/system/multi-user.target.wants/fiufiu.service*? ¿Què hi ha a la carpeta "/etc/systemd/system/multi-user.target.wants"? ¿I a "/etc/systemd/system/graphical.target.wants"?

**c)** ¿Quin és el significat de les línies *Requires=* i *After=* afegides?

**3.-a)** Executa *systemctl is-enabled ufw* si estàs a Ubuntu (o *systemctl is-enabled firewalld* si estàs a Fedora) i comprova que el servei en qüestió estigui activat (si no ho estigués, activa'l). Tot seguit executa la comanda *systemctl show -p "Wants" multi-user.target | grep -E "(ufw|firewalld)"* ¿Què significa el que veus? Pista: observa el contingut de la carpeta "/etc/systemd/system/multi-user.target.wants"

**aII)** Executa ara *systemctl disable ufw* (o *systemctl disable firewalld*) i tot seguit torna a executar la comanda *systemctl show -p "Wants" multi-user.target | grep -E "(ufw|firewalld)"* ¿Què veus ara? ¿Per què? Pista: observa el missatge que apareix a pantalla en deshabilitar els serveis ufw/firewalld

**b)** Dedueix i digues per què la línia *Before=* de la unit "ufw.service" (a Ubuntu) o "firewalld.service" (a Fedora), corresponent al tallafocs del sistema (la configuració del qual es pot saber mitjançant *systemctl cat ufw/firewalld*) té el valor que té.

**c)** Després de llegir per a què serveix la comanda *nm-online* (i en especial el seu paràmetre -s) a la seva pàgina del manual, observa la sortida de *systemctl cat NetworkManager-wait-online.service* i dedueix per a què serveix tenir iniciat aquest servei automàticament a l'arranc del sistema i quin sentit té el valor de les seves línies *Requires=, After= i Before=*.

**cII)** Després de llegir per a què serveix la comanda *systemd-networkd-wait-online* a la seva pàgina del manual, observa la sortida de *systemctl cat systemd-networkd-wait-online.service* i dedueix per a què serviria tenir iniciat aquest servei automàticament a l'arranc del sistema (cosa que no ho està per defecte) i quin sentit té el valor de les seves línies *Requires=, After=, Before=* i *Conflicts=*

**d)** Llegeix la pàgina del manual del servei "systemd-time-wait-sync"per saber què passaria si l'activem (*enable*) i si, a més, afegim la *línia After=time-sync.target* dins de la secció *[Unit]* de l'arxiu de configuració d'una unit qualsevol de tipus "service", a més de la línia *TimeoutStartSec=5m* dins de la secció *[Service]*

> NOTA: La línia *TimeoutStartSec=* és necessària perquè el servei "systemd-time-wait-sync" espera infinitament fins que es pugui sincronitzar l'hora (és a dir, es pot no arribar mai al target "time-sync", amb la qual cosa els serveis marcats com *After=time-sync.target* no arribarien a iniciar mai)

**4.-a)** Crea un target nou (anomenat "manolo.target") on el sistema haurà d'entrar just després d'activar "graphical.target" (és a dir, ha de ser l'últim target en activar-se). La idea serà assegurar de què s'entra en aquest target per tal d'executar un determinat servei (l'anomenarem "manolo.service") l'últim de tots. Per fer això, has de fer això:

* Crea un nou fitxer anomenat "/etc/systemd/system/manolo.target" amb el següent contingut:

```
[Unit]
Description=Manolo is kind
Requires=graphical.target
After=graphical.target
Conflicts=rescue.target
AllowIsolate=yes
```

* Crea un nou fitxer anomenat "/etc/systemd/system/manolo.service" amb el següent contingut:

```
[Unit]
Description=Manolo is superb
Requires=manolo.target
After=manolo.target
[Service]
ExecStart=printf "MANOLO\n"
RemainAfterExit=yes
[Install]
WantedBy=manolo.target
```

**b)** Executa *systemctl enable manolo.service* i reinicia la màquina. ¿Creus que el servei "manolo" estarà funcionant automàticament o no? ¿Per què ho creus? Comprova-ho executant la comanda *systemctl status manolo.service* (o també observant si apareix la paraula "MANOLO" al Journal). PISTA: La resposta de perquè el servei "manolo" estarà funcionant (o no) es troba en el que mostra la comanda *systemctl list-units -t target | grep "manolo"* i en entendre el significat de la línia *WantedBy=*

**c**) Tot seguit executa *systemctl start manolo.service* . Després d'observar que el servei s'hagi posat en marxa correctament (en la sortida de la comanda *systemctl status manolo.service* o també observant si apareix la paraula "MANOLO" al Journal), ¿creus que la comanda *systemctl list-units -t target | grep "manolo"* mostrarà alguna cosa diferent respecte l'apartat anterior? Per què ho creus? PISTA: La resposta es troba en entendre el significat de les línies *Requires=* i *After=* de l'arxiu "manolo.service"

**d)** Executa *systemctl set-default manolo.target* i torna a reiniciar la màquina. ¿Creus que el servei "manolo" ara estarà funcionant automàticament o no? ¿Per què ho creus? Comprova-ho executant la comanda *systemctl status manolo.service* (o també observant si apareix la paraula "MANOLO" al Journal). PISTA: La resposta es troba en entendre el significat de la línia *WantedBy=*

**5.-**Només mirant el contingut d'aquest "service" (que anomenarem "pepi.service") i les NOTES indicades, digues en quin moment s'executaria si fem *sudo systemctl enable pepi.* Comprova-ho reiniciant la màquina i observant tot seguit la sortida de *journalctl -u pepi*

```
[Unit]
Description=Pepi
DefaultDependencies=no
[Service]
Type=oneshot
ExecStart=date
[Install]
WantedBy=final.target
```

NOTA: A "final.target" s'arriba just abans dels targets "shutdown"/"reboot"/"halt"/"kexec" (veieu *man bootup*)
NOTA: Si s'escriu "/opt/unScript.sh" en "/usr/lib/systemd/systemd-shutdown" en lloc de fer el que l'exercici proposa, seria executat molt tard perquè el sistema de fitxers arrel ja estaria remuntat en mode només lectura.
NOTA: *DefaultDependencies=no* fa que s'ignorin totes les dependències i executa "el primer" en l'inici i "el darrer" en l'apagada (respectant sempre les directives "Before" i "After", això sí).

**6.-a)** Crea un script executable dins de la carpeta "/usr/lib/systemd/system-sleep" amb el següent contingut i prova de suspendre el sistema i tornar-lo a "despertar". ¿Què passarà?:

```
#!/bin/bash
if [[ "$1" == "pre" ]] ; then
        echo "we are suspending or hibernating at $(date)..." >> /tmp/systemd_suspend_test.txt
elif [[ "$1" == "post" ]] ; then
        echo "...and we are back from $(date)" >> /tmp/systemd_suspend_test.txt
fi
```

**b)** Executa la comanda *systemd-inhibit firefox* ¿Què mostra llavors la comanda *systemd-inhibit --list* i per què? Comprova-ho executant (amb el mateix usuari) la comanda *systemctl suspend* . ¿Què passa? ¿I si tanques el firefox i ho tornes a provar?

**c)** Estableix una aturada programada d'aquí dos minuts i comprova, abans de què aquesta es faci efectiva, quin és el contingut de la directiva USEC a l'arxiu "/run/systemd/shutdown/scheduled" i fes la transformació del seu valor a un format de data humanament llegible. ¿Quina comanda hauries de fer servir per tal d'apagar la màquina a les 12:00 i quin serà llavors el contingut de la directiva USEC?

Boot chain

Per saber el conjunt d'units (service, socket, targets...) que han hagut d'iniciar-se (o parar-se) per arribar a iniciar un target (o service!) determinat es pot utilitzar la comanda: *systemctl list-dependencies nomTarget.target* (o *nomUnit.service*)

> 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*

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

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

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 de la UEFI, del gestor d'arranc, del kernel, de l'initrd i de 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 *systemd-analyze* que també convé conèixer, més enllà de l'àmbit de l'arranc del sistema poden ser les següents (però n'hi ha moltes més: *syscall-filter, calendar, timespan, dump, log-level, cat-config* ... consulteu la seva pàgina del manual):

*systemd-analyze security [nomUnit] :* Mostra l'estat de "hardering" i confinament dels diferents serveis Systemd (o de la unit concreta que s'hagi indicat, oferint en aquest cas informació més detallada) amb l'objectiu d'informar sobre el seu "grau d'exposició"

*systemd-analyze verify /ruta/fitxer/conf/unaunit.service:* Mostra si l'arxiu de configuració de la unit indicada conté algun error sintàctic o de dependències o de permisos d'execució, etc.

D'altra banda, es pot veure si, un cop iniciat el sistema, encara queden tasques pertanyents a l'arranc per completar executant la comanda *systemctl list-jobs*

**EXERCICIS:**

**1.- a) ¿**Quins targets han d'haver-se iniciat per a què el servei "gdm" es pugui posar en marxa? (això ho pots saber amb la comanda s*ystemctl list-dependencies* ...)

**b)** Executa la comanda *systemctl list-dependencies --reverse gdm.service* Què veus?

**c)** ¿Què fa la comanda *tree /etc/systemd/system* i per a què podria servir-te?

**d)** Consulta la pàgina *man bootup* i digues què mostra l'esquema gràfic que apareix sota l'apartat "System manager bootup" i el que mostra un esquema similar sota l'apartat "User session bootup".

**2.-a)** Executa *systemd-analyze plot > una imatge.svg* i seguidament obre amb un visor qualsevol aquesta imatge per observar quina unit bloqueja més temps l'arranc del teu sistema. Prova de desactivar-la (esperem no trencar res!) i reinicia. Executa ara *systemd-analyze blame* per comprovar si el temps total d'arranc ha disminuït efectivament.

> NOTA: Una altra manera d'aconseguir el mateix és, per exemple, veure les units que hi ha "enabled" (amb la comanda *sudo systemctl list-unit-files -t service | grep "enabled"*) i a partir d'allà anant desactivant/enmascarant les que considerem. Si hi ha algun servei que no coneixem què fa, sempre es pot consultar la descripció que dóna la comanda *systemctl status nomServei* i anar als enllaços de documentació que allà s'hi mostren

**b)** Instal.la el paquet "graphviz" i genera un gràfic PNG amb les dependències del target "multi-user". Fes una llista de les dependències allà mostrades i adjunta la captura del gràfic

**c)** Executa *systemd-analyze security systemd-journald.service* i descriu què obtens

Variables d'entorn

Els serveis, en executar-se, no hereten cap "context" de l'usuari que els invoca (tal com el proporcionat per qualsevol variable definida a ".bashrc" o similar); és a dir, cada servei s'executa "net" i independent, sense cap variable d'entorn predefinida. Això és perquè l'usuari que posa en marxa el servei en qüestió ("root" en el cas dels serveis del sistema, un usuari estàndard en el cas dels serveis d'usuari) en realitat l'únic que fa és indicar al dimoni Systemd (*systemd --system* o *systemd --user*, respectivament) que ell posi en marxa el servei desitjat sense cap "record" de qui ho demana. D'aquesta manera s'aconsegueix disposar sempre d'un entorn controlat, conegut i comú per qualsevol servei que es posi en marxa, independentment de qualsevol "ingerència" externa.

En realitat, el descrit al paràgraf anterior no és ben bé així: Systemd sí que estableix per defecte un conjunt selecte de variables d'entorn, el valor de les quals serà tingut en compte per totes les units, a no ser que s'indiqui algun valor diferent de forma explícita. Aquestes variables propagades a qualsevol procés invocat per Systemd són **LANG** (Systemd estableix el seu valor a partir de la configuració present a l'arxiu "locale.conf" o bé via el paràmetre del kernel *locale.LANG=*), **PATH** (Systemd estableix el valor fixe de "*/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin*") i **HOME, USER, SHELL** (només establertes, però, per les units que tenen indicada la directiva *User=* i totes les d'usuari).

> * Si es vol afegir alguna variable més a aquesta llista de variables d'entorn per defecte reconegudes per totes les units del sistema sense distinció, es pot fer de diferents formes:

**1.**-Especificant-les a la línia ***DefaultEnvironment=*** de l'arxiu "**/etc/systemd/system.conf**" amb la sintaxi "*NOMVARIABLE1=valor1 NOMVARIABLE2=valor2 ...*"

**2.**-Especificant-les a través del paràmetre del kernel *systemd.setenv=NOMVARIABLE=valor*

**3**.-Especificant-les amb la comanda *systemctl set-environment NOMVARIABLE=valor*

NOTA: La comanda *systemctl unset-environment NOMVARIABLE[=valor]*, al seu torn, elimina la variable d'entorn indicada per totes les units del sistema. Si s'indica un valor concret, només s'eliminarà si té aquell valor específic

**5.** Systemd també té en compte el mòdul PAM "pam_env", així que les possibilitats que ofereix aquest mòdul per definir variables d'entorn també s'apliquen.

* Si, en canvi, es vol afegir alguna variable a aquesta llista de variables d'entorn reconegudes però que només afecti als processos pertanyents a una unit del sistema concreta (és a dir, als indicats a la seva directiva *ExecStart=*, *ExecStop=*, etc i relacionats), es pot fer de la següent manera (que serà la més habitual):

**4.** Especificant-les dins de la secció *[Service]* (i també *[Socket]*, tal com veurem aviat) de l'arxiu de configuració de la unit corresponent mitjançant les directives:

***Environment=NOMVARIABLE1="valor1" [NOMVARIABLE2="valor2"] [...]***
Pren com a valor una llista d'ítems *NOMVARIABLE=valor* , separats entre sí per un espai en blanc, que definiran l'entorn on s'executaran els processos corresponents a la unit en qüestió. Aquesta directiva es pot indicar més d'un cop; es tindran en compte llavors tots els ítems indicats indistintament, tot i que si una mateixa variable apareix repetida, es prendrà com el seu valor vàlid el darrer indicat. D'altra banda, si s'especifca aquesta directiva amb un valor buit (així, *Environment=*), els possibles valors previ que s'haguessin pogut indicar en línies anteriors s'esborraran.

NOTA: Tingueu en compte que les variables d'entorn indicades amb aquesta directiva no són adequades per passar secrets (com ara contrasenyes, material clau, ...) als processos del servei perquè estan exposades a clients no privilegiats mitjançant D-Bus IPC i, generalment, no s'entenen com a dades que requereixen protecció. Cal utilitzar la directiva ***LoadCredential=*** per passar dades de manera segura als processos corresponents a una unitat.

***EnvironmentFile=[-]/ruta/fitxer***
Similar a la directiva *Environment=* però en aquest cas les variables d'entorn s'obtenen del fitxer de text indicat (el qual tindrà cada parella *NOMVARIABLE=valor* escrita a una línia diferent). Les línies d'aquest fitxer que siguin buides o que comencin per "#" o ";" seran ignorades. Si s'indica el símbol "-" al davant de la ruta del fitxer indicat, en el cas de què aquest fitxer no existís no s'enviarà cap missatge d'advertència o error. Aquesta directiva es pot indicar més d'un cop; es tindran en compte llavors el contingut de tots els fitxers indicats indistintament, tot i que si una mateixa variable apareix repetida, es prendrà com el seu valor vàlid el darrer llegit (segons l'ordre de les directives *EnvironmentFile=* escrites). D'altra banda, si s'especifica aquesta directiva amb un valor buit (així, *EnvironmentFile=*), els possibles valors previ que s'haguessin pogut indicar en fitxers llegits anteriorment s'esborraran. Cal tenir en compte, a més, que variables indicades mitjançant *EnvironmentFile=* tenen preferència a les indicades mitjançant *Environment=*

NOTA: Si qualsevol variable sindica a la directiva ***UnsetEnvironment=***, serà esborrada tot i haver estar indicada a qualsevol dels punts anteriors

Si la mateixa variable d'entorn s'establís en diferents llocs dels mencionats anteriorment, l'ordre d'aplicació és l'indicat amb els números mostrats a la llista anterior; és a dir, el darrer (el mòdul PAM) "guanyarà" a qualsevol dels llocs anteriors. En qualsevol cas, es ,pot comprovar la llista de variables d'entorn reconegudes (i el valor actualment aplicat) per les units del sistema amb la comanda *systemctl show-environment*

Les units de tipus "usuari" (és a dir, les ubicades dins de la carpeta "~/.config/systemd/user") poden també obtenir de diversos llocs la llista de variables d'entorn reconegudes (és a dir, el seu "context"):

*Per les units d'usuaris de tots ells sense distinció:

**1.**Indicant "*NOMVARIABLE1=valor1 NOMVARIABLE2=valor2 ..."* com a valor de la línia ***DefaultEnvironment=*** de l'arxiu "**/etc/systemd/user.conf**"

**2.**Alternativament, creant dins de la carpeta "**/etc/systemd/system/user@.service.d**" un fitxer anomenat de qualsevol forma però amb extenxió "*.conf" i amb un contingut similar a:

*[Service]*
*Environment="NOMVARIABLE1=valor1"*
*Environment="NOMVARIABLE2=valor2"*
*...*

NOTA: També existeixen les carpetes "**/usr/lib/environment.d**" i "**/etc/environment.d**", similars en significat a la carpeta mencionada al proper punt, "~/.config/environment.d" però aplicades de forma genèrica per qualsevol unit d'usuari. Per més informació consulteu *man environtment.d*

*Per les units d'un determinat usuari:

**3.** Indicant diferents línies de la forma *NOMVARIABLE=valor* dins d'un fitxer anomenat de qualsevol forma però amb extenxió "*.conf" ubicat a la carpeta "**~/.config/environment.d**"

**4.** Alternativament, executant les comandes *systemctl --user set-environment NOMVARIABLE=valor* o *systemctl –user import-environment NOMVARIABLE*. però només afectaria a les units iniciades a partir de llavors

NOTA: La comanda *systemctl –user import-environment NOMVARIABLE* serveix per a què Systemd reconegui el valor d'una determinada variable d'entorn establert fora d'ell (per exemple, dins de l'arxiu ".bashrc"). Un cas molt comú és el d'importar la variable PATH,

**5.** Alternativament, fent servir les directives *Environment=* i *EnvironmentFile=* aplicades a la configuració de la unit particular

En qualsevol cas, es ,pot comprovar la llista de variables d'entorn reconegudes (i el valor actualment aplicat) per les units d'usuari amb la comanda *systemctl –user show-environment*

**EXERCICIS:**

**1.-a)** Crea un fitxer anomenat "pirripipi.service" dins de "/etc/systemd/system" amb el següent contingut...:

*[Unit]*
*Description=Instant karma*
*[Service]*
*Type=oneshot*
*Environment=ONE="one" TWO="two two"*
*ExecStart=echo $ONE ${TWO}*

...i inicia la unit corresponent. ¿Quin és el valor de les variables ONE i TWO i on/com ho has pogut comprovar?

NOTA: Les claus a ${TWO} s'indiquen per incloure els espais en blanc del valor de la variable

**b)** ¿Què passa amb els valors registrats d'aquestes variables si ara executes *systemctl set-environment ONE=three TWO=four* i tornes a iniciar la unit anterior? ¿Per què? ¿I si ara comentes la línia *Environment=* i tornes a iniciar la unit?

**c)** Dedueix, observant el contingut de l'arxiu "sshd.service", d'on obté el binari *sshd* el nom i el valor de les variables d'entorn que tindrà en consideració per iniciar-se el servei SSH, i quines són aquestes

> NOTA: Existeix, tal com es pot veure també a l'arxiu "sshd.service" una variable d'entorn especial (que totes les units reconeixen) anomenada $MAINPID, la qual no s'estableix en cap fitxer extern sinó que s'estableix automàticament cada cop que la unit s'inicia al PID assignat al seu procés principal en aquell moment. Aquesta variable serveix, per exemple, per tenir la referència de a qui matar quan s'executa ExecStop= o similars.

**2.-a)** Tal com es pot veure a l'apartat titulat "Environment" de la seva pàgina del manual, la comanda *journalctl* pot ser susceptible de canviar el seu comportament segons el valor que tinguin en aquell moment diverses variables d'entorn específicament associades a aquesta comanda. Sabent això, digues, consultant aquest apartat del manual, per a què serveixen les següents variables d'aquest tipus: SYSTEMD_LOG_LEVEL, SYSTEMD_LOG_COLOR (no confondre amb SYSTEMD_COLORS) i SYSTEMD_PAGER

> NOTA: A http://systemd.io/ENVIRONMENT trobareu més variables predefinides relacionades amb Systemd com a tal

**b)** ¿Què aconseguiràs si executes la comanda *systemctl edit systemd-networkd.service* i, a l'editor que t'apareix, escrius les següents línies i les guardes?

> *[Service]*
> *Environment=SYSTEMD_LOG_LEVEL=debug*

Plantilles

Una plantilla és un fitxer de configuració de tipus "service" que té la particularitat de permetre posar en marxa diferents "instàncies" d'un mateix servei sense haver d'escriure un arxiu "service" diferent per cada instància. 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 instància amb la què volem treballar. Aquest identificador s'estableix la primera vegada que arrenca la instància 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 instància

> NOTA: Normalment un fitxer d'instància es crea en forma d'enllaç simbòlic al fitxer de plantilla, el nom del qual inclourà l'identificador de la instància. D'aquesta manera, diversos enllaços amb identificadors únics poden apuntar cap a un únic fitxer de plantilla. En gestionar una unitat d'instància, *systemctl* buscarà un fitxer amb el nom d'instància exacta que s'indiqui a la línia d'ordres però si no en trobar, cercarà el fitxer de plantilla associat.

**4.-**La gràcia de les plantilles és que el valor de 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 que hagi adquirit %i per aquella instància es podria posar en marxa el servei escoltant en un port diferent (si %i representa un número de port), o bé utilitzant un arxiu de configuració diferent (si %i representa un nom de fitxer), o el que ens convingui.

NOTA: Altres símbols especials que es poden indicar en un arxiu de configuració d'una plantilla poden ser:
**%p**: Representa el prefix del nom de la unit (és a dir, el tros del nom de la unit que va davant del símbol "@")
%n: Representa el nom complet de la unit (és a dir, %p més %i )
%u: El nom de l'usuari que executa la unit
%U: El UID de l'usuari que executa la unit
%H: El nom del sistema ("hostname") que executa la unit
**%h**: En el cas de tractar-se d'una plantilla per una unit d'"usuari" i no de sistema, indica la ruta de la carpeta personal de l'usuari en qüestió
%%: Usat per poder escriure el símbol "%" literal
Més valors es poden consultar a l'apartat "Specifiers" de *man systemd.unit*

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

**EXERCICIS:**

**1.-a)** Crea un arxiu plantilla (l'anomenarem "netcat@.service") que permeti posar en marxa diferents servidors Ncat de forma permanent (recorda el seu paràmetre *-k*) escoltant cadascun d'ells en un port diferent.

NOTA: Recorda que a la màquina on estiguis treballant hauràs d'haver instal.lat prèviament el paquet "nmap-ncat" (a Fedora) o "ncat" (a Ubuntu) per disposar de la comanda *ncat*

**b)** Inicia un servidor Ncat a partir de la plantilla anterior escoltant al port 2222 i un altre escoltant al port 3333. Comprova que, efectivament, aquests dos ports estan oberts observant la sortida de la comanda *ss -tnl* Alternativament, també pots comprovar que els dos servidors Ncat estan funcionant executant *systemctl status netcat@** o també *journalctl -u netcat@**

**c)** Connecta amb el client Ncat a un dels servidors anteriors i envia-li algun missatge. Tanca el client (amb CTRL+C) i ara torna a executar-lo per connectar a l'altre servidor; torna a enviar-li algun altre missatge i tanca'l de nou. Observa les darreres línies del Journal: ¿què hi veus?

**2.-a)** Instal·la el paquet "iperf3" i tot seguit crea un arxiu anomenat "iperf@.service" a "/etc/systemd/system" amb el següent contingut i contesta les següents preguntes relacionades:

* ¿Per a què serveixen els paràmetres *-s*, *-1* i *-p* de la comanda "iperf3"? (consulta el seu manual)
* ¿Per a què serveix la directiva *DefaultInstance=* ? (consulta *man systemd.unit*)

*[Unit]*
*Description=iperf3 server on port %i*
*[Service]*
*ExecStart=/usr/bin/iperf3 -s -1 -p %i*
*[Install]*
*WantedBy=multi-user.target*
*DefaultInstance=5201*

**b)** Després d'executar *sudo systemctl daemon-reload*, executa la línia següent al terminal i digues què fa (comprova-ho observant la sortida de *ss -tln* , o bé provant de connectar-te amb sengles clients *iperf3* a cada port disponible): *for port in {9200..9210}; do sudo systemctl start iperf@$port ; done*

**c)** A l'exemple oficial (https://iperf.fr/iperf-servers.php#host-iperf3) apareixen dins de la secció *[Service]* de l'arxiu "iperf.service" les dues línies addicionals indicades a continuació. ¿Per a què serveix la directiva *RuntimeMaxSec=* (consulta *man systemd.service*) i quin efecte té el valor de la directiva *Restart=* indicada? ¿Quina és el raonament per afegir aquestes dues línies?

*RuntimeMaxSec=3600*
*Restart=always*

**3.-a)** Busca a la seva pàgina del manual per a què serveix la comanda *agetty* i per a què serveix el seu paràmetre *-a nomusuari*

**b)** Llegeix el següent paràgraf i, a partir d'aquí, digues un valor possible que pot ser assignat a l'identificador %I (d'un significat molt similar al conegut %i) que apareix escrit a la plantilla "autovt@.service" :

*"When the user switches consoles using Ctrl+Alt+F2, Ctrl+Alt+F3,and so on, a new terminal then is spawned. In this case systemd calls a service named autovt@.service providing the appropriate argument such as tty2 or tty3 to the unit file. The %i identifier provides this argument value to the agetty binary so the terminal starts on that new console (as it can seen in ExecStart= line from template file)."*

**c)** Modifica ara la invocació a la comanda *agetty* escrita en la línia *ExecStart=* que apareix dins de l'arxiu "autovt@.service" així: afegint el paràmetre *-a nomusuari* i esborrant el paràmetre *-o "..."* que hi ha allà actualment (perquè és incompatible amb el nou paràmetre *-a*). ¡O millor, executa la comanda *sudo systemctl edit autovt@.service* per tal de generar un arxiu "override" amb el següent contingut, el qual no modificarà l'arxiu "autovt@.service" original i, per tant, serà més elegant!:

*[Service]*
*ExecStart=*
*ExecStart=agetty -a nomusuari -J %I $TERM*

¿Què passa ara quan pulses Ctrl+Alt+F2, etc ? ¿Què mostra la comanda *systemctl status autovt@** ?

NOTA: Si no vols un inici de sessió automàtic complet però tampoc no vols escriure el teu nom d'usuari, pots escriure aquesta línia alternativa: *ExecStart=-/usr/bin/agetty* ***-n -o*** *nomusuari -J %I $TERM*
NOTA: L'opció *Type = Idle* que es troba al fitxer predeterminat "getty @ .service" retarda l'inici del servei fins que es completin tots els treballs (sol·licituds de canvi d'estat a unitats) per evitar contaminar la sol·licitud d'inici de sessió amb missatges d'arrencada. Si vols començar a utilitzar-lo immediatament, afegeix *Type=simple* a un fitxer ".override"

**d)** Enmascara la instància concreta tty5 de la plantilla "autovt@.service". ¿Què passa ara si fas Ctrl+Alt+F5 ? ¿Què et mostra la comanda *systemctl status autovt@tty5* ? ¿Què et mostra aquesta mateixa comanda si indiques el nom d'un terminal virtual on has entrat en l'apartat anterior?

**4.-a)** Crea un script anomenat "mon-url.sh" dins de la teva carpeta personal amb el següent contingut (i dóna-li permisos d'execució!). Prova'l i esbrina per a què serveix:

```
#!/bin/bash
URL=$(systemd-escape -u "$1")
ans=$(curl -s -o /dev/null -w "%{http_code}" "$URL")
echo $ans
if [[ "$ans" == "200" ]] ;  then
      systemd-cat -p notice -t monitorURL echo "S'ha recuperat ${URL}. Status: ${ans}"
elif [[ "$ans" != "200" ]] ;  then
      systemd-cat -p err -t monitorURL echo "S'ha perdut ${URL}. Status: ${ans}"
fi
```

**b)** Crea un fitxer anomenat "mon-url.service" dins de ~/.config/system/user" amb el següent contingut...:

```
[Unit]
Description="Comprovant web %i"
[Service]
Type=oneshot
ExecStart=bash %h/mon-url.sh %i
```

...i respon: ¿quin valor té "%h" sempre a l'arxiu anterior? ¿I quin valor podria tenir "%i" (a partir d'haver entès el codi de l'script)?

**c)** Executa les següents comandes, una rera l'altra:

```
systemctl --user start mon-url@https:--github.com
systemctl --user start mon-url@https:--redhat.com
systemctl --user start mon-url@https:--hola.com
```

NOTA: La URL indica a les comandes anteriors té el símbol "-" en comptes de "/" perquè aquest darrer no és un símbol vàlid per una instància. Per poder tonar a "escriure" el símbol correcte i així poder fer servir la URL correctament escrita, a l'script es fa servir la comanda especial *systemd-escape*, la qual, mitjançant el seu paràmetre *-u* reconeix "intel·ligentment" quin ha de ser el caràcter substituït per tornar a tenir el "/".

**d)** Per no haver d'escriure totes les comandes anteriors cada cop que es vulguin monitoritzar les URLs, crea un nou fitxer anomenat "mon-urls.target" dins de ~/.config/system/user" amb el següent contingut:

```
[Unit]
Description="Comprovant totes les web"
Wants=mon-url@https:--github.com mon-url@https:--redhat.com  mon-url@https:--hola.com
```

...i tot seguit executa *systemctl –user start mon-urls.target* Comprova al Journal què ha passat.

NOTA: Observa com s'ha triat l'acció "start" en comptes de "isolate" per "anar al target" "mon-urls". Això és perquè "start" posa en marxa totes les dependències indicades a les línies Wants=/Requires= mentre que "isolate", tot i fer això mateix, també atura tota la resta d'units (cosa que en aquest cas no ens convé!)

**5.-** ¿Què expliquen aquest paràgraf?

*In addition to the main instance "systemd-journald.service" there's also a template unit "systemd-journald@.service". A new unit file setting LogNamespace= has been added, taking an instance name, that assigns services to the specified log namespaces. As each log namespace is serviced by its ownindependent journal daemon, this functionality may be used to improve performance and increase isolation of applications, at the price of losing global message ordering. Each instance of journald has a separate set of configuration files, with possibly different disk usage limitations and other settings. Moreover,  systemd-journald also gained the ability to exit on idle, which is useful in the context of log namespaces, as this means log daemons for log namespaces can be activated automatically on demand and will stop automatically when no longer used, minimizing resource usage. Finally, journalctl now takes a new option --namespace= to show logs from a specific log namespace."*

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*). 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 "/usr/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
NOTA: Un aspecte molt interessant dels sockets és que es poden vincular a ports privilegiats del sistema (és a dir menors de 1024) encara que el servei corresponent s'estigui executant com un usuari diferent de "root" (via les directives User= i Group=). Això és possible perquè l'escolta al port feta pel socket és independent totalment de la feina del servei pròpiament dita, cosa que fa, doncs, que poguem executar serveis sense privilegis però rebent peticions des d'un port privilegiat

**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.

NOTA: Existeixen altres tipus de sockets que es poden definir mitjançant directives específiques (com ara **ListenSequentialPacket=/ruta/arxiu.socket**, la qual indica un socket de tipus UNIX -que només serveix per comunicacions entre processos de la mateixa màquina-) però no les estudiarem. Per més, *man systemd.socket*

**Accept=yes**

Genera una instància del servei diferent per cada connexió. Això implica que el socket es farà responsable d'apagar automàticament aquesta instància quan la connexió en qüestió finalitzi. Cal, però, que el servei s'hagi definit com a plantilla per a què això funcioni. Si el seu valor és *no* (per defecte) només una instància del servei gestiona totes les connexions i, llavors, el socket no es fa pas responsable d'apagar el servei; d'això se n'hauria d'encarregar el propi servei (normalment passat un determinat temps sense activitat global) fent servir algun mecanisme propi intern.

**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. No compatible amb *Accept=yes*

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

Com qualsevol altra unit, es poden veure la llista de sockets amb la comanda *systemctl list-units -t socket* però a més disposem de la comanda específica *systemctl list-sockets*, la qual informa de quin servei corresponent activen i en quin port/socket UNIX escolten.

**EXERCICIS:**

**1.-a)** Crea el fitxer "/opt/dateserver.sh" amb el següent contingut (i dona-li permisos d'execució!). Tot seguit executa'l i comprova que funciona correctament en local i en primer pla:

```
#!/bin/bash
while [[ true ]]; do
        date && sleep 1
done
```

**b)** Crea el fitxer "/etc/systemd/system/dateserver**@**.service" amb el següent contingut i no facis res més:

```
[Unit]
Description=Servei de data
[Service]
Type=simple
ExecStart=/opt/dateserver.sh
StandardOutput=socket
#StandardError=journal
```

**c)** Crea ara el fitxer "/etc/systemd/system/dateserver.socket" amb el següent contingut i tot seguit inicia el socket corresponent (opcionalment el pots habilitar també, ja que té la directiva *WantedBy=*) i comprova que estigui, efectivament, iniciat i (amb *ss -tnl*) que el port 55555 estigui obert:

```
[Unit]
Description=Servei de data al port 55555
[Socket]
ListenStream=55555
Accept=true
[Install]
WantedBy=sockets.target
```

**cII)** Obre un terminal i executa la comanda *ncat ipServidor 55555* ¿Què veus? Obre un altre terminal diferent i executa la mateixa comanda. ¿Què veus?

**cIII)** ¿Què et mostra la comanda *systemctl status dateserver.socket*? ¿I la comanda *systemctl status dateserver@** ? ¿I la comanda *systemctl list-units dateserver@** ? ¿I la comanda *systemctl list-sockets*?

**cIV)** Atura tots els clients. ¿Què mostren llavors, passada una estona, de nou les comandes *systemctl status dateserver.socket* i *systemctl status dateserver@** ?

**2.-** Fes que el servidor SSH que tinguis instal.lat a la màquina (si no el tens, instal.la'l) s'inicïi només a través d'un socket. Concretament:

**a)** Crea el fitxer "/etc/systemd/system/sshMitjo.socket" amb el següent contingut:

*[Unit]*
*Description=El meu SSH Socket*
*[Socket]*
*ListenStream=2222*
*Accept=yes*
*[Install]*
*WantedBy=sockets.target*

NOTA: Posarem a escoltar el socket "sshMitjo" en un port diferent del 22 perquè així podrem tenir el servidor "estàndard" també funcionant en paral·lel, si volem.

**b)** Crea el fitxer "/etc/systemd/system/sshMitjo@.service" amb el següent contingut:

*[Unit]*
*Description=El meu SSH Server*
*[Service]*
*Type=simple*
*ExecStart=-sshd* ***-i***
*RuntimeDirectory=sshd*
***StandardInput=socket***
*StandardOutput=socket*

NOTA: Aquí la clau està en la combinació del paràmetre -i del binari *sshd* (el qual fa que habilitar la possibilitat de què pugui rebre peticions a través de sockets), i la directiva StandardInput (la qual realitza de forma efectiva aquest tipus de comunicació entre el socket i el servidor SSH). La directiva RuntimeDirectory simplement estableix el nom de la carpeta temporal sota "/run" que el procés sshd farà servir pel seu funcionament intern...és una dada que és necessària per a què s'inicïi sense errors.. Respecte el com saber el nom del binari, simplement n'hi ha prou observant el contingut de la línia *ExecStart=* de l'arxiu "sshd.service" original.
NOTA: És important el "-" davant del nom del binari. Això garanteix que *systemd* oblidi l'estat de sortida del procés sshd. Normalment, *systemd* emmagatzemarà l'estat de sortida de totes les instàncies de servei que moren de forma anormal. De vegades, SSH morirà de manera anormal amb un codi de sortida 1 o similar, i volem assegurar-nos que això no faci que systemd mantingui informació sobre nombroses connexions anteriors que van morir d'aquesta manera (fins que aquesta informació s'oblidi amb *systemctl reset-failure* ).

**c)** Executa la comanda *systemctl enable sshMitjo.socket* i reinicia la màquina. Un cop fet, comprova que el socket sshMitjo estigui funcionant però no el servei sshMitjo. Executa *ssh -p 2222 usuari@ipServidor* per entrar al servidor SSH (ho hauries d'aconseguir sense problemes) i comprova seguidament que ara sí que està funcionant una instància del servei sshMitjo Ara tanca la sessió SSH i comprova tot seguit si encara està funcionant la instància anterior o ja no.

**d)** ¿Què faria una comanda com *systemctl kill sshd@172.31.0.52:22-172.31.0.4:47779.service*?