Introducció a D-Bus

D-Bus (https://www.freedesktop.org/wiki/Software/dbus) és una API que es pot fer servir en el desenvolupament de qualsevol programa d'usuari per tal d'aconseguir comunicar el seu procés amb altres procesos també compatibles amb D-Bus (bé de forma punt-a-punt o bé multicast) però, i això és important, executant-se tots a la mateixa màquina. És a dir, D-Bus és un mecanisme IPC (Inter-Process Communication) -com podria ser, a un nivell molt més simplificat, una canonada-. O dit d'una altra manera: D-Bus és una manera que tenen els processos per parlar entre sí intercanviant-se ordres, notificacions d'events i missatges en general (tot plegat escrits d'una determinada forma) a través de determinats canals (els "busos") que són oferts per un servei anomenat "dbus".

Generalment hi ha dos busos principals, un anomenat "system" (l'estat del qual es pot consultar amb *systemctl status dbus*) i un altre anomenat "session" (l'estat del qual es pot consultar amb *systemctl status --user dbus*). Tots dos mantinguts pel binari *dbus-daemon* però amb paràmetres diferents.

*El bus "system" és un únic canal per on es poden comunicar totes les aplicacions/serveis del sistema entre sí indistintament de la sessió d'usuari amb la què hagin sigut iniciades. Per exemple, se sol fer servir per monitoritzar quan les tarjes de xarxa es (des)activen, quan els discos externs es (des)enxufen, quan la bateria del portàtil està a nivell baix, etc.

*De busos "session" poden haver-hi més d'un creats a la vegada perquè per cada sessió d'usuari diferent iniciada al sistema se'n crea un bus "session" diferent (però només és visible el bus "session" pertanyent a la sessió d'usuari actual). Serveix per comunicar entre sí les aplicacions posades en marxa per l'usuari en qüestió. Per exemple, reproductors de vídeo poden enviar un missatge D-Bus evitant que s'activi el salvapantalles mentre l'usuari està veient una película.

Per tal d'enviar un missatge a un bus D-Bus caldrà primer saber l'adreça de destí. Aquesta està estructurada d'una forma jeràrquica amb els següents elements:

*El nom del **servei** D-Bus (integrat dins de l'aplicació que l'implementi) que rebrà el missatge. Per tal de no solapar noms de diferents serveis, el que se sol fer és anomenar-los simplement revertint el FQDN del domini propietat del desenvolupador i afegint-hi llavors el nom de servei desitjat; per ex: *org.maemo.Alert* o *org.freedesktop.Notifications* D'aquesta manera, diferents desenvolupadors podran anomenar al servei D-Bus de la seva aplicació "Alert" o "Notifications" sense que hi hagi problemes

*Cada servei D-Bus pot contenir un o més "**objectes**", cadascun dels quals proporcionarà una funcionalitat diferent. Per identificar un objecte respecte d'un altre s'usen els "object paths", els quals són simples cadenes que tenen el mateix aspecte que les rutes absolutes de fitxers. En general, si el servei D-Bus només conté un objecte, el seu "path" sol ser el mateix nom del servei però escrit en forma de ruta i, opcionalment, afegint-hi al final un nom específic per l'objecte (és a dir, en el nostre exemple, "/org/maemo/Alert/Alerter" o /org/freedesktop/Notifications/Manager", per exemple).

*Cada objecte D-Bus implementa un conjunt "d'**interfícies**". Cada interfície representa un subconjunt determinat de les diferents ordres concretes que l'objecte pot executar, juntament amb els seus paràmetres i les possibles senyals a les què pot respondre. És possible reutilitzar la mateixa interfície en diferents objectes. El nom d'una interfície segueix les mateixes normes que els noms de serveis; és per això que per serveis senzills se sol repetir el seu nom pel nom de la seva interfície més rellevant.

*La darrera part de la direcció d'un missatge D-Bus és el nom del "**membre**" (en el cas de IPCs això se sol anomenar nom del "mètode" i en el cas de senyals, nom "de la senyal"; també hi poden haver "propietats" que són simples parelles clau<->valor editables o no). Aquest nom selecciona quin procediment executar o quina senyal a emetre. Només necessita ser únic dins de la interfície on estigui implementat.

Eines D-Bus:

D-Bus proporciona una comanda per experimentar amb l'enviament de diferents missatges (ja siguin membres de tipus IPCs o de tipus senyal): *dbus-send* . Els seus paràmetres més importants són:

--system : Indica que el missatge s'enviarà al bus "system".
Per defecte (o amb el paràmetre *--session*) s'envia al bus "session"
--type=method_call : Indica que es farà una IPC en comptes d'enviar una senyal
(que seria l'acció per defecte)
--print-reply : S'espera a rebre una resposta de l'execució de la IPC i mostra aquesta (si n'hi ha)

--dest=nom.servei : Obvi

/ruta/objecte : Obvi

nom.Interficie.nomMembre : Obvi

Notar com el nom del membre s'ha "d'adjuntar" amb un punt (i sense espais) al final del nom de la interfície.

Una altra comanda que forma part del paquet oficial "dbus-tools" (igual que *dbus-send*) és *dbus-monitor*, la qual mostra al terminal en temps real tots els missatges D-Bus transferits al bus "*system*" (si s'afegeix el paràmetre *--system*) o bé "*session*" actual (si s'afegeix el paràmetre *--session*).

Una altra eina alternativa per treballar amb serveis D-Bus d'una manera una mica més còmoda que amb *dbus-send* és la comanda *gdbus*, proporcionada pel projecte GTK/Gnome. A continuació es presenta un exemple d'execució RPC, on es poden veure els seus paràmetres més comuns (*--session* , *--dest*, *--object-path* i *--method*):

gdbus call --session --dest org.freedesktop.Notifications --object-path /org/freedesktop/Notifications --method org.freedesktop.Notifications.Notify miApp 42 gtk-dialog-info "Resum", "Texte de la notificació" [] {} 5000

L'execució anterior correspon concretament a la del mètode Notify, pertanyent a la interfície org.freedesktop.Notificacion, el qual, tal com es pot veure, ha de rebre vuit paràmetres: nom de qui fa la RPC (cadena), un identificador opcional (número) , nom de la icona dins de "/usr/share/icons/gnome" a mostrar (cadena), títol mostrat a la notificació (cadena), texte mostrat a la notificació (cadena), accions a realitzar un cop generada la notificació (no se sol usar; array de cadenes), característiques extra de la notificació (posició, so a emetre, etc; diccionari de cadenes) i el temps en milisegons durant el qual es mostra la notificació (número sencer). Més informació a https://developer.gnome.org/notification-spec/

NOTA: Fixeu-vos que podríem haver aconseguit el mateix simplement amb la comanda *notify-send "Texte de la notificació"*; de fet, en realitat, l'únic que fa aquesta comanda és realitzar la crida D-Bus a un servei->objecte->interfície->mètode concret (amb uns valors de paràmetres concrets) sense que l'usuari hagi de saber res de com funcionen els canals D-Bus interiorment. Fent servir *gdbus* directament el que estem fent és obviar la comanda *notify-send* i fer la mateixa feina "manualment".

També es poden enviar (a tots els processos o bé a un en concret) senyals (que han d'estar reconegudes per una determinada interfície implementada al destí per a què es tinguin en compte) mitjançant *gdbus emit* . Les senyals serveixen per notificat als programes subscrits de què ha ocorregut cert event.

D'altra banda, també és molt interessant la comanda *gdbus introspect*, que permet estudiar la composició d'un objecte de forma similar al que fa l'aplicació D-Feet (https://wiki.gnome.org/Apps/DFeet) gràficament, la qual estudiarem als exercicis.

NOTA: Si no poguéssim usar D-Feet, per conèixer la llista de serveis D-Bus disponibles podríem executar per exemple la següent comanda: *dbus-send --session --type=method_call –print-reply --dest=org.freedesktop.Dbus /org/freedesktop/DBus org.freedesktop.Dbus.ListNames*

A més de la implementació "oficial" i la que ofereix Gnome, existeix una tercera alternativa per poder treballar amb D-Bus des del terminal: la proporcionada per Systemd: *busctl*. Concretament, les possibilitats bàsiques que proporciona aquesta darrera comanda són aquestes (per més informació podeu llegir l'article http://0pointer.net/blog/the-new-sd-bus-api-of-systemd.html) :

busctl {--system | --user } call nom.Servei /nom/Objecte nom.Interfície mètode [signatures param1 param2 ...]
busctl {--system | --user } get-property nom.Servei /nom/Objecte nom.Interfície propietat
busctl {--system | --user } set-property nom.Servei /nom/Objecte nom.Interfície propietat signatura valor
busctl {--system | --user } introspect nom.Servei /nom/Objecte [nom.Interfície]

on "signatura" vol dir el tipus de paràmetre indicats, que pot indicar-se amb una "i" si és de tipus sencer amb signe, una "u" si és de tipus sencer sense signe, una "s" si és de tipus "string", "b" si és de tipus booleà, "v" si és de tipus "variant" (és a dir, calaix de sastre, útil per números decimals entre altres), "ai" si és un array de sencers, "as" si ho és de cadenes, etc. Per saber tots els tipus possibles, es pot consultar https://dbus.freedesktop.org/doc/dbus-specification.html#type-system

> NOTA: En el cas de tenir una signatura amb un array, abans de la llista dels valors dels seus elements caldrà indicar el número d'aquests. És a dir, així: *as 3 hola adeu pepe*
>
> NOTA: En el cas de tenir una signatura amb un valor "variant", això vol dir que el tipus del valor forma part del propi valor. Això a la pràctica significa que abans del valor en sí caldrà indicar el tipus específic d'aquest ("i", "u", "s", "b"...i, aquí sí, "d" si és un número decimal). És a dir, així: *v d 0.7*
>
> NOTA: Si s'executa sense paràmetres (*busctl*) , mostrarà la llista de processos i serveis que fan ús de D-Bus
>
> NOTA: Es pot obtenir la llista d'objectes continguts en un determinat servei D-Bus amb la comanda *busctl tree nom.Servei*, i, a partir d'aquí, fer una instrospecció
>
> NOTA: Si es vol saber com desenvolupar un programa que publiqui un determinat servei D-Bus personalitzat fent servir la convenció Systemd, es pot llegir http://0pointer.net/blog/the-new-sd-bus-api-of-systemd.html o també https://zignar.net/2014/09/08/getting-started-with-dbus-python-systemd o https://leonardoce.wordpress.com/2015/03/11/dbus-tutorial-using-the-low-level-api

---

En qualsevol cas, sigui quina sigui l'eina de terminal que fem servir, als diferents serveis D-Bus els trobaran definits en arxius "*.service" dins de la carpeta "/usr/share/dbus-1/services" o bé dins de "/usr/share/dbus-1/system-services". Les interfícies, d'altra banda, es troben definides en arxius "*.xml" dins de la carpeta "/usr/share/dbus-1/interfaces". Per més informació podeu consultar https://dbus.freedesktop.org/doc/dbus-daemon.1.html De totes maneres, la manera més fàcil de conèixer tots els serveis D-Bus disponibles al sistema o a la sessió juntament amb els objectes, interfícies i membres que implementen és fent ús, tal com ja hem dit, del programa gràfic D-Feet.

EXERCICIS:

1.-a) Instal.la a una màquina virtual qualsevol de tipus Ubuntu Desktop el programa D-Feet i utilitza'l per esbrinar, un cop posis en marxa el reproductor VLC, quin és el nom del servei D-Bus que aquesta aplicació obre, quins són els objectes que implementa aquest servei, quines interfícies contenen aquests objectes i quins mètodes, propietats i senyals estan incloses en cadascuna de les interfícies.

aII) ¿Quina informació obtens executant la comanda *busctl --user introspect org.mpris.MediaPlayer2.vlc /org/mpris/MediaPlayer2* ? ¿ I la comanda *busctl --user introspect org.mpris.MediaPlayer2.vlc /org/mpris/MediaPlayer2 org.freedesktop.Dbus.Properties* ?

b) Amb el VLC obert, digues què fa aquesta comanda: *busctl --user call org.mpris.MediaPlayer2.vlc /org/mpris/MediaPlayer2 org.freedesktop.DBus.Properties Set ssv "org.mpris.MediaPlayer2.Player" "Volume" d 0.7*

NOTA: En el cas d'haver fet servir la comanda gdbus, l'últim valor s'ha d'escriure entre < i > perquè les dades "Variant" s'indiquen sempre amb < i > (i no caldrà indicar el tipus "d")

c) Reprodueix un vídeo qualsevol amb el VLC i a continuació omple els buits de la següent comanda per tal d'aconseguir pausar la reproducció: *busctl --user call ... ...*

d) ¿Quina comanda *busctl* similar a l'anterior hauries de fer servir ara per reanudar la reproducció pausada?

e) Executa el mètode adient amb *busctl* per tal de tancar el reproductor VLC

Systemd ofereix un canal D-Bus (de sistema) per tal de gestionar tots els serveis. De fet, la comanda *systemctl* el que fa realment és realitzar les crides pertinents a aquest canal, no més que això (d'una forma molt convenient per l'usuari, això sí). L'exercici següent emula la funcionalitat d'aquesta comanda llençant de forma directa crides a D-Bus tal com ho faria ella.

2.-a) A la mateixa màquina virtual de l'exercici anterior, executa aquesta comanda: *sudo busctl call org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager StopUnit ss "cups.service" "replace"* ¿Què fa? (comprova-ho)

NOTA: No facis cas del segon paràmetre del mètode StopUnit...és un valor intern que sempre valdrà "replace"

b) Amb l'ajuda de D-Feet i, sobre tot, després de llegir https://www.freedesktop.org/wiki/Software/systemd/dbus , digues què fa la comanda *sudo busctl call org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager EnableUnitFiles asbb 1 "cups.service" false true* i quin és el significa de cadascun dels seus paràmetres.

El dimoni *systemd-timedated* se sol emprar per atendre sota demanda peticions fetes des de la comanda *timedatectl*, les quals solen consistir en canviar manualment l'hora del sistema (o també, depenent del paràmetre emprat, la zona horària preestablerta) en absència d'un client NTP. Tots els paràmetres de la comanda *timedatectl* en realitat l'únic que fan és una crida D-Bus a un determinat mètode d'un servei->objecte-interfície determinat. L'exercici següent emula la funcionalitat de la comanda *timedatectl* fent la crida al D-Bus directament. Més informació: https://www.freedesktop.org/wiki/Software/systemd/timedated

3.- a) Busca a D-Feet el servidor D-Bus oferit pel dimoni *systemd-timedated* i digues com es diu

b) Sabent que el servei "org.freedesktop.timedate1" inclou l'objecte "/org/freedesktop/timedate1" ofereix la interfície "org.freedesktop.timedate1" que conté la propietat "Timezone", escriu la comanda *busctl* adient per tal d'obtenir el seu valor al terminal

c) Sabent que el servei "org.freedesktop.timedate1" inclou l'objecte "/org/freedesktop/timedate1" ofereix la interfície "org.freedesktop.timedate1" que conté el mètode "SetTimezone()" per canviar la zona horària del sistema, escriu la comanda *busctl* adient per tal de realitzar aquest canvi (i comprova seguidament que s'hagi fet)

Un altre dimoni que proporciona un servei D-Bus és *systemd-hostnamed* . Aquest dimoni se sol emprar per atendre sota demanda peticions (via D-Bus) fetes des de la comanda *hostnamectl*, les quals solen consistir bàsicament en demanar el nom actual de la màquina o bé en voler canviar-ho. En altres paraules: la comanda *hostnamectl* en realitat no és més que un client D-Bus que realitza les peticions mitjançant aquesta via (ja siguin de consulta o d'actualització de dades) al servei *systemd-hostnamed*, el qual es posa en marxa només quan és rep la petició (no està tota l'estona funcionant). Això significa que la mateixa funció que podem fer amb la comanda *hostnamectl* es pot realitzar mitjançant qualsevol eina que permeti fer crides al canal D-Bus, com ara *dbus-send*, *gdbus* o la pròpia de Systemd, *busctl*. Més informació: https://www.freedesktop.org/wiki/Software/systemd/hostnamed

4.- Observa amb D-Feet amb quin objecte, interfície i mètode necessitaries comunicar-te per canviar el nom actual de la màquina i utilitza la comanda *busctl* per construir la crida adient per fer-ho. Comprova que funcioni

5.- Amb l'ajuda de D-Feet i, sobre tot, després de llegir https://upower.freedesktop.org/docs/ref-dbus.html , escriu una comanda qualsevol (a escollir) que faci ús de l'API D-Bus proporcionada pel servei UPower i digues per a què serveix aquesta comanda.