D-Bus – Sistema de comunicaciones entre procesos.

- El objetivo es mostrar, utilizando Python, un ejemplo de como realizar un test básico de comunicación D-Bus.

Introducción

D-Bus (*Desktop Bus*) es un sistema de comunicación de procesos. D-Bus es desarrollado como parte del proyecto freedesktop.org y en entornos linux es ampliamente utilizado.

D-Bus consiste en tres capas:

- La biblioteca *libdbus*, que permite a dos aplicaciones conectarse e intercambiar mensajes.
- Un demonio ejecutable, construido sobre libdbus, al cual pueden conectarse varias aplicaciones. El demonio expande el mensaje enviándolo a una o más aplicaciones.
- *Wrappers* para su uso.

D-Bus es utilizado en:

- La comunicación entre aplicaciones de escritorio en la misma sesión.
- La comunicación entre el S.O. y la sesión de escritorio.

Ejemplo

Introducción

El ejemplo consta de dos ficheros. Sender.py y Receptor.py los dos alojados en </home/ruben/Documents/dbus>, en cada caso se debe adecuar a la ruta en la cual guardemos los ficheros python.

Receptor.py:

```
#!/usr/bin/env python
#--encoding: UTF-8--

"""
entra en un loop esperando senales emitidas a:
dbus_interface = ruben.prova
object_path = "/home/ruben/Documents/dbus"
con el nombre de senal: 'estat'
cuando se recibe la senal la mostramos
"""

import gobject, dbus, dbus.mainloop.glib

def mostra(m):
 print m

dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)
bus = dbus.SessionBus()
bus.add_signal_receiver(
 mostra,
 path="/home/ruben/Documents/dbus",
 dbus_interface="ruben.prova",
 signal_name = "estat"
 )

loop = gobject.MainLoop()

loop.run()
```

Sender.py:

```
#!/usr/bin/env python
#--encoding: UTF-8--

"""
Emite una senal a dbus, al bus 'session' al destino:
 dbus_interface = ruben.prova
 object_path = "/home/ruben/Documents/dbus"
con el nombre de senal: 'estat'
"""
```

```
import gobject
import dbus
from dbus.service import signal,Object
import dbus.mainloop.glib

class EmetSenyal(Object):

 def __init__(self, conn, object_path='/'):
 Object.__init__(self, conn, object_path)


 @signal('ruben.prova')
 def estat(self,m):
 global loop
 print("sended signal: %s" % m)
 gobject.timeout_add(2000, loop.quit)


dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)
loop = gobject.MainLoop()
bus = dbus.SessionBus()
o = EmetSenyal(bus,object_path='/home/ruben/Documents/dbus')
o.estat('Tux')
loop.run()
```

Monitoring

En un terminal ejecutamos:

```
 python receptor.py
```

Y en otro vamos ejecutando:

```
python sender.py
```

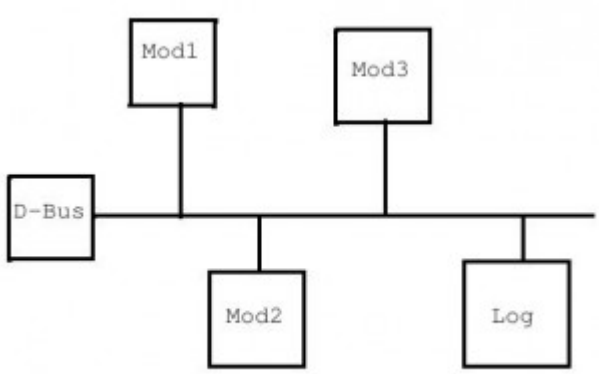
Finalmente en un terminal aparte ejecutaremos:

```
dbus-monitor
```

de esta manera podremos observar como sender.py envia la señal cada vez que lo ejecutamos.

Observación

En si el ejemplo carece de utilidad, solo pretende ser un ejemplo de iniciación. Un uso mas adecuado podría ser el paso de información, por ejemplo, usarlo para el almacenamiento de log de módulos de un aplicativo. Es decir, imaginemos que tenemos tres módulos Mod1, Mod2 y Mod3 conectados por D-Bus y un cuarto modulo de log. Cuando cualquiera de los tres módulos envía una señal de log de info o debug los demás lo ignoran y tansolo el modulo log se encarga de procesar el log y hacer lo que deba con el (almacenamiento en disco duro, envio por email…), pero en caso de que sea una senyal de log_error(moduele_name,error_name) los demas modulos recogen la senyal y sabran que han perdido una funcionalidad de dicho módulo, hasta que no se reciba una nueva señal de inicio de Mod1. De esta manera tendremos integrado un sistema de log para los diferentes módulos del aplicativo y a su vez un sistema de alertas entre estos.



Fuentes

- http://es.wikipedia.org/wiki/D-Bus

Código en GitHub

DBus code example

Ruben.