# Introduction to Udisks

**Finnbarr P. Murphy**
*(fpm@fpmurphy.com)*

*Udisks* (formerly called *DeviceKit-disks*) is a replacement for part of the functionality which used be provided by the now deprecated HAL (Hardware Abstraction Layer). Essentially it is an abstraction for enumerating disk and storage devices and performing operations on them.

*Udisks* provides:

- A daemon (*udisks-daemon*) that implements well-defined D-Bus interfaces that can be used to query and manipulate disk and storage devices.
- A command-line tool (*udisks*), that can be used to query and use the daemon.

Actions that a user can perform using *udisks* can be restricted using *PolicyKit*. *Udisks* relies on the kernel and *udev* where possible but does poll devices which do not publish their own details. Typically a DVD device is polled.

Some history is useful for understanding where *udisks* fits into the scheme of things on present-day Linux platforms. In the days of yore (circa 2003!), applications and desktops had no way of obtaining knowledge of hardware events except by polling devices. A young engineer, Robert Love, proposed that the kernel should be responsible for providing this information to userspace. Love spearheaded an effort, named Project Utopia, to better integrate hardware management into the desktop. The final solution was a combination of technologies: a kernel events layer to publish everything from device status ("your processor fan is failing") to system events ("a new USB stick has been mounted"), netlink (which is a simple, low overhead, fast communication channel from the kernel out to user-space) and D-Bus. If you want more background information, the seminal paper to read is On a Kernel Events Layer and User-space Message Bus System.

Contemporaneously Havoc Pennington developed a compelling view of hardware management on the Linux desktop which he discussed in his paper Making Hardware Just Work. As a result, David Zeuthen (currently at Red Hat) began the HAL project to implement Havoc's concept. Over time HAL evolved from being a hardware abstraction layer to being a system-level daemon that tied together kernel hotplug events, *sysfs* and *udev*, in order to provide applications with a single, comprehensive view of hardware, accessible via a standardized interface. The first release of HAL was in 2003. Over the next few years HAL was embraced by all the major Linux distributions. In March 2006, PolicyKit was split out as an independent project also maintained by David Zeuthen.

By 2008, it was claimed that HAL had "become a large monolithic unmaintainable mess." As a result, David Zeuthen decided to produce a modular hardware abstraction layer called DeviceKit to replace the monolithic HAL architecture. The initial release of DeviceKit was in 2008 and included *DeviceKit-disks* which was the daemon that provided interfaces to obtain information and perform operations on storage devices. DeviceKit was the death knoll for HAL and, as a result, the last release of HAL was *v0.5.15* in 2009. Interestingly, *DeviceKit* itself was a very short lived project because in December 2009 it was announced that *DeviceKit-disks* was renamed to *udisks*. Other DeviceKit modules were also renamed and ended up under other projects and, in fact, a lot of the HAL functionality was subsequently merged into *udev*. HAL is now deprecated on all the major Linux distributions and has mostly been removed from their codebases. *Udisks* is still actively maintained by David Zeuthen with the latest version (1.0.3) being released in June 2011.

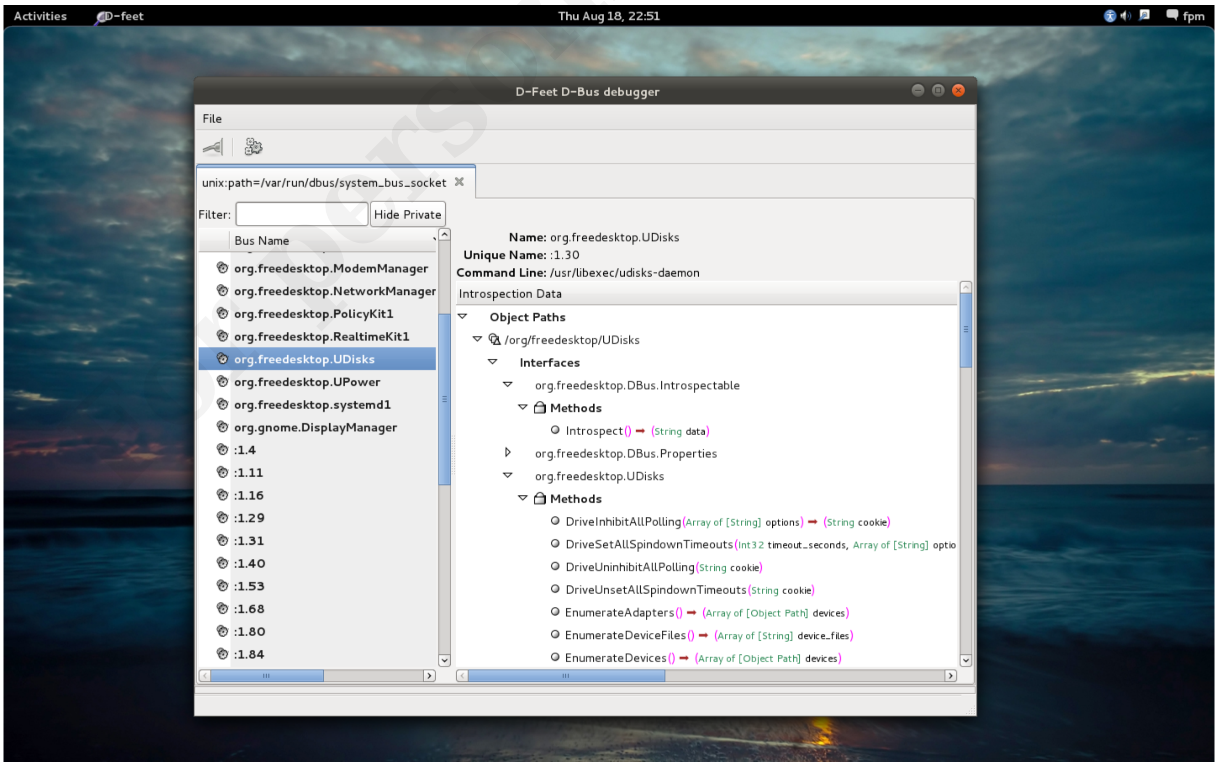
To understand *udisks*, you also need to have some understanding of D-Bus which is an Inter Process Communication (*IPC*) mechanism for applications running on a single machine to talk to each other. The core D-Bus protocol is a 2-way, asynchronous, binary protocol. While direct application to application communication is possible, the most common usage of the protocol is via a D-Bus message bus server. The server is connected to each client using D-Bus, and routes each message from the application which sent to its intended recipient. Each application which exposes its capabilities on the bus is known as a service. Services on a D-Bus expose a tree of objects to interact with. Each service has a root path; from the root come branches, one for each object provided by that service. Each object on a service tree is addressable by an object path. Object paths are typically in a reverse-domain name format, e.g. *org.freedesktop.UDisks*.

Interfaces are the contract between an object and its callers, much like interfaces in C++, GObject, etc. An interface, as in most object-oriented programming environments, defines the methods and properties an object exposes. In addition, D-Bus interfaces can also contain signals. Signals provide a way for objects to notify other objects about events which happen to them: for example, a Door interface might provide a signal to other objects when it is opened, perhaps called door-opened. (A Building Management object might listen out for door-opened signals, so it knows when specific doors are opened and closed.)

Typically there are multiple D-Bus message buses active at any one time. These can be divided into two categories:

- A system bus (AKA system channel) which is a platform-wide single instance bus with security restrictions on what messages it will accept. It's used for system-wide communication and communication between the user desktop and the operating system.
- A session bus (AKA private channel) is created for each user session. It allows applications within that user session to communicate with each other and with the platform.

A great way of inspecting a service on D-Bus is to use John Palmieri's D-Feet graphical D-Bus debugging tool. Here's an example of it monitoring the system bus on Fedora 15:



```
unix:path=/var/run/dbus/system_bus_socket

org.freedesktop.ModemManager
org.freedesktop.NetworkManager
org.freedesktop.PolicyKit1
org.freedesktop.RealtimeKit1
org.freedesktop.UDisks
org.freedesktop.UPower
org.freedesktop.systemd1
org.gnome.DisplayManager
:1.4
:1.11
:1.16
:1.29
:1.31
:1.40
:1.53
:1.68
:1.80
:1.84

        Name: org.freedesktop.UDisks
 Unique Name: :1.30
Command Line: /usr/libexec/udisks-daemon

Object Paths
  /org/freedesktop/UDisks
    Interfaces
      org.freedesktop.DBus.Introspectable
        Methods
          Introspect() ➡ (String data)
      org.freedesktop.DBus.Properties
      org.freedesktop.UDisks
        Methods
          DriveInhibitAllPolling(Array of [String] options) ➡ (String cookie)
          DriveSetAllSpindownTimeouts(Int32 timeout_seconds, Array of [String] optio
          DriveUninhibitAllPolling(String cookie)
          DriveUnsetAllSpindownTimeouts(String cookie)
          EnumerateAdapters() ➡ (Array of [Object Path] devices)
          EnumerateDeviceFiles() ➡ (Array of [String] device_files)
          EnumerateDevices() ➡ (Array of [Object Path] devices)
```

The system bus on Fedora 15 is located at *unix:path=/var/run/dbus/system_bus_socket*. You can find your session bus via your environment.

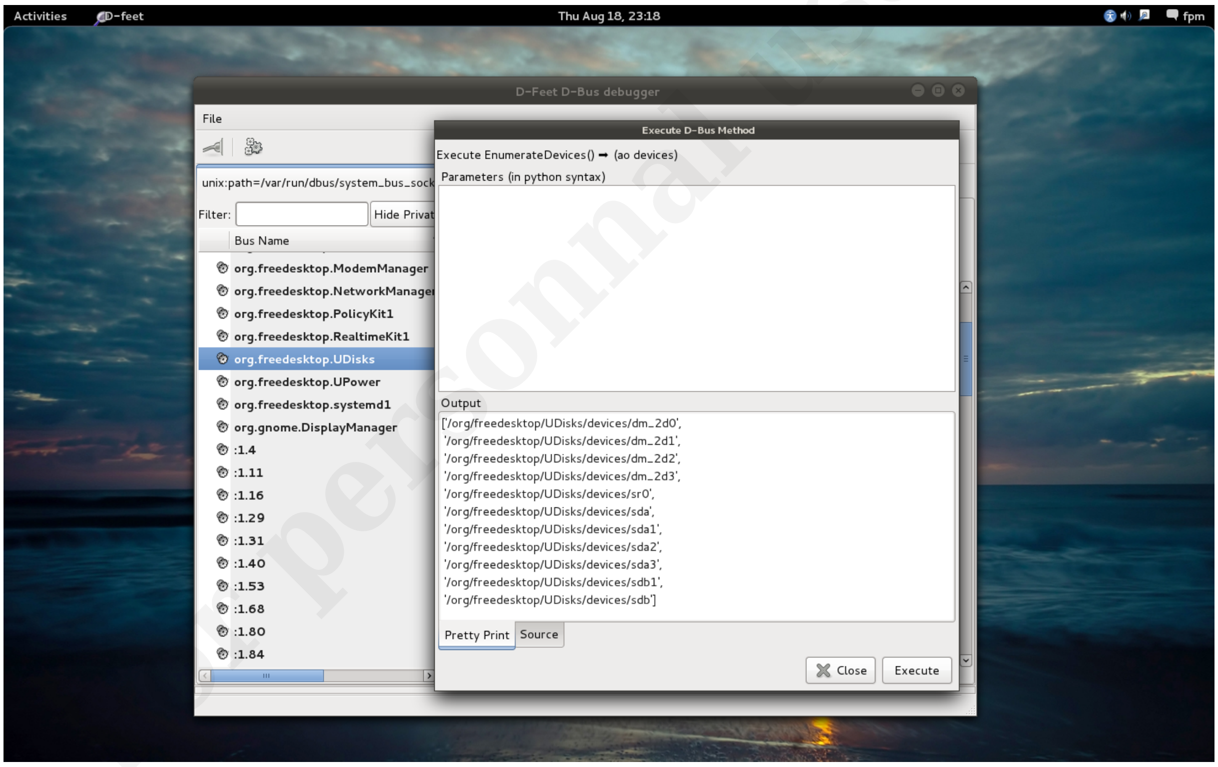
```
$ env | grep DBUS_SESSION_BUS_ADDRESS
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-58iznPau2Z,guid=32abe0fedb01bd5a92019a1e0
0000048
```

In this particular case you would enter the following string into D-Feet: *unix:abstract=/tmp/dbus-58iznPau2Z,guid=32abe0fedb01bd5a92019a1e00000048*.

Notice the reference to a command line called */usr/libexec/udisks-daemon* in the above image. This daemon provides the *org.freedesktop.UDisks* service on the system message bus. You should never have to start this daemon as it will be automatically started by the D-Bus message bus daemon (*dbus-daemon*) whenever an application calls into the *org.freedesktop.UDisks* service.

```
$ ps -ef | grep udisk
root      1585     1  0 Aug16 ?        00:00:02 /usr/libexec/udisks-daemon
root      1587  1585  0 Aug16 ?        00:00:03 udisks-daemon: polling /dev/sdb /dev/sr0
```

You can also double click on any method on the services you discover to invoke the method. For example, here is the output of the *EnumerateDevices* method of the *org.freedesktop.UDisks* service:



unix:path=/var/run/dbus/system_bus_sock

org.freedesktop.ModemManager
org.freedesktop.NetworkManager
org.freedesktop.PolicyKit1
org.freedesktop.RealtimeKit1
org.freedesktop.UDisks
org.freedesktop.UPower
org.freedesktop.systemd1
org.gnome.DisplayManager
:1.4
:1.11
:1.16
:1.29
:1.31
:1.40
:1.53
:1.68
:1.80
:1.84

Execute EnumerateDevices() ➡ (ao devices)

Parameters (in python syntax)

Output

```
['/org/freedesktop/UDisks/devices/dm_2d0',
 '/org/freedesktop/UDisks/devices/dm_2d1',
 '/org/freedesktop/UDisks/devices/dm_2d2',
 '/org/freedesktop/UDisks/devices/dm_2d3',
 '/org/freedesktop/UDisks/devices/sr0',
 '/org/freedesktop/UDisks/devices/sda',
 '/org/freedesktop/UDisks/devices/sda1',
 '/org/freedesktop/UDisks/devices/sda2',
 '/org/freedesktop/UDisks/devices/sda3',
 '/org/freedesktop/UDisks/devices/sdb1',
 '/org/freedesktop/UDisks/devices/sdb']
```

A useful command line tool is *dbus-monitor* which displays signals sent by and messages sent to a D-Bus bus. Invoke it with:

```
dbus-monitor --system
```

for the system bus or

```
dbus-monitor --session
```

for your session bus.

If you want to list the available services from the command line, you can do so using *dbus-send*. Here is some of the output for Fedora 15:

```
# dbus-send --system --print-reply --reply-timeout=2000   --type=method_call --dest=org.fr
eedesktop.DBus    /org/freedesktop/DBus  org.freedesktop.DBus.ListNames
method return sender=org.freedesktop.DBus -> dest=:1.618 reply_serial=2
   array [
      string "org.freedesktop.DBus"
      string "org.freedesktop.systemd1"
      string "org.freedesktop.NetworkManager"
      string "com.redhat.ifcfgrh1"
      string "org.freedesktop.ModemManager"
      string "org.freedesktop.RealtimeKit1"
      string "org.freedesktop.Accounts"
      string "org.freedesktop.PolicyKit1"
      string "org.bluez"
      string ":1.592"
      string "org.freedesktop.PackageKit"
      string "com.redhat.NewPrinterNotification"
      string "com.redhat.PrinterDriversInstaller"
      string ":1.615"
      string "org.gnome.DisplayManager"
      string ":1.616"
      string "org.freedesktop.Avahi"
      string ":1.618"
      string "org.freedesktop.ConsoleKit"
      string "org.freedesktop.UDisks"
      string "org.freedesktop.UPower"
      .....
   ]
```

Note the *org.freedesktop.UDisks* service.

Try monitoring your session bus while you insert a USB stick. Quite a lot of information is outputted. Here is the initial message flow from the time the USB stick is inserted to just before Nautilus is invoked:

```
signal sender=org.freedesktop.DBus -> dest=:1.156 serial=2 path=/org/freedesktop/DBus; int
erface=org.freedesktop.DBus; member=NameAcquired
   string ":1.156"
method call sender=:1.156 -> dest=org.freedesktop.DBus serial=3 path=/org/freedesktop/DBus;
 interface=org.freedesktop.DBus; member=AddMatch
   string "type='method_call'"
method call sender=:1.156 -> dest=org.freedesktop.DBus serial=4 path=/org/freedesktop/DBus;
 interface=org.freedesktop.DBus; member=AddMatch
   string "type='method_return'"
method call sender=:1.156 -> dest=org.freedesktop.DBus serial=5 path=/org/freedesktop/DBus;
 interface=org.freedesktop.DBus; member=AddMatch
   string "type='error'"
signal sender=:1.36 -> dest=(null destination) serial=104 path=/org/gtk/Private/RemoteVolu
meMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=DriveChanged
   string "org.gtk.Private.GduVolumeMonitor"
   string "0xb56410"
   struct {
      string "0xb56410"
      string "Kingston DataTraveler 2.0"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removabl
e drive"
      boolean true
      boolean true
      boolean true
      boolean true
      boolean true
      boolean false
      boolean false
      boolean true
      uint32 1
      array [
         string "0xb66c40"
      ]
      array [
         dict entry(
            string "unix-device"
            string "/dev/sdc"
         )
      ]
   }
signal sender=:1.36 -> dest=(null destination) serial=105 path=/org/gtk/Private/RemoteVolu
meMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=DriveConnected
   string "org.gtk.Private.GduVolumeMonitor"
   string "0xb56410"
   struct {
      string "0xb56410"
      string "Kingston DataTraveler 2.0"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removabl
e drive"
      boolean true
      boolean true
      boolean true
      boolean true
      boolean true
      boolean false
      boolean false
      boolean true
      uint32 1
      array [
         string "0xb66c40"
      ]
      array [
         dict entry(
            string "unix-device"
            string "/dev/sdc"
         )
      ]
   }
signal sender=:1.36 -> dest=(null destination) serial=106 path=/org/gtk/Private/RemoteVolu
meMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=VolumeAdded
   string "org.gtk.Private.GduVolumeMonitor"
   string "0xb66c40"
   struct {
      string "0xb66c40"
      string "1littlegeek"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removabl
e drive"
      string ""
      string ""
      boolean true
      boolean true
      string "0xb56410"
      string ""
      array [
         dict entry(
            string "unix-device"
            string "/dev/sdc1"
         )
         dict entry(
            string "label"
            string "1littlegeek"
         )
         dict entry(
            string "uuid"
            string "1711-0DC4"
         )
      ]
   }
method call sender=:1.34 -> dest=org.gtk.Private.GduVolumeMonitor serial=13 path=/org/gtk/
Private/RemoteVolumeMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=VolumeM
ount
   string "0xb66c40"
   string ""
   uint32 0
   string "2297:4"
method call sender=:1.42 -> dest=org.freedesktop.DBus serial=23 path=/org/freedesktop/DBus;
 interface=org.freedesktop.DBus; member=AddMatch
   string "type='signal',interface='ca.desrt.dconf.Writer',path='/ca/desrt/dconf/Writer/us
er',arg0path='/org/gnome/deja-dup/'"
method call sender=:1.42 -> dest=org.freedesktop.DBus serial=24 path=/org/freedesktop/DBus;
 interface=org.freedesktop.DBus; member=RemoveMatch
   string "type='signal',interface='ca.desrt.dconf.Writer',path='/ca/desrt/dconf/Writer/us
er',arg0path='/org/gnome/deja-dup/'"
signal sender=:1.36 -> dest=(null destination) serial=107 path=/org/gtk/Private/RemoteVolu
meMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=VolumeChanged
   string "org.gtk.Private.GduVolumeMonitor"
   string "0xb66c40"
   struct {
      string "0xb66c40"
      string "1littlegeek"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removabl
e drive"
      string ""
      string ""
      boolean true
      boolean true
      string "0xb56410"
      string "0xb7a240"
      array [
         dict entry(
            string "unix-device"
            string "/dev/sdc1"
         )
         dict entry(
            string "label"
            string "1littlegeek"
         )
         dict entry(
            string "uuid"
            string "1711-0DC4"
         )
      ]
   }
signal sender=:1.36 -> dest=(null destination) serial=108 path=/org/gtk/Private/RemoteVolu
meMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=MountAdded
   string "org.gtk.Private.GduVolumeMonitor"
   string "0xb7a240"
   struct {
      string "0xb7a240"
      string "1littlegeek"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removabl
e drive"
      string ""
      string "file:///media/1littlegeek"
      boolean true
      string "0xb66c40"
      array [
      ]
   }
```

```
signal sender=:1.29 -> dest=(null destination) serial=234 path=/org/gtk/gio/DesktopAppInfo;
 interface=org.gtk.gio.DesktopAppInfo; member=Launched
   array of bytes [
      2f 75 73 72 2f 73 68 61 72 65 2f 61 70 70 6c 69 63 61 74 69 6f 6e 73 2f
      6e 61 75 74 69 6c 75 73 2e 64 65 73 6b 74 6f 70 00
   ]
   string ":0"
   int64 28949
   array [
      string "file:///media/1littlegeek"
   ]
   array [
      dict entry(
         string "startup-id"
         variant             string "gnome-settings-daemon-2297-ultra.fpmurphy.xcom-nautil
us-3_TIME0"
      )
      dict entry(
         string "origin-prgname"
         variant             array of bytes [
               67 6e 6f 6d 65 2d 73 65 74 74 69 6e 67 73 2d 64 61 65 6d 6f 6e
               00
            ]
      )
      dict entry(
         string "origin-pid"
         variant             int64 2297
      )
   ]
```

This on a Fedora 15 GNOME desktop. Note that there is no reference to *udisks* anywhere. Instead the work is done by the *MountAdded* member of *org.gtk.Private.GduVolumeMonitor* which is part of GVFS (GNOME Virtual File System). KDE, XFCE, LXDE and other desktops all provide their own mechanisms and bypass *udisks* to a greater or less extent. That does not mean that you cannot use *udisks*; it just means that you need to be aware of the other mechanisms. *Udisks* is probably most useful in environments which do not have a desktop.

The main command line utility for interacting with the *udisks-daemon* is *udisks*. It has a large number of options.

```
$ udisks --help
Usage:
  udisks [OPTION...] udisks commandline tool

Help Options:
  -h, --help                     Show help options

Application Options:
  --enumerate                    Enumerate objects paths for devices
  --enumerate-device-files       Enumerate device files for devices
  --dump                         Dump all information about all devices
  --monitor                      Monitor activity from the disk daemon
  --monitor-detail               Monitor with detail
  --show-info                    Show information about a device file
  --inhibit-polling              Inhibit polling
  --inhibit-all-polling          Inhibit all polling
  --poll-for-media               Poll for media
  --set-spindown                 Set spindown timeout for drive
  --set-spindown-all             Set spindown timeout for all drives
  --spindown-timeout             Spindown timeout in seconds
  --inhibit                      Inhibit the daemon
  --mount                        Mount the given device
  --mount-fstype                 Specify file system type
  --mount-options                Mount options separated by comma
  --unmount                     Unmount the given device
  --unmount-options             Unmount options separated by comma
  --detach                      Detach the given device
  --detach-options              Detach options separated by comma
  --eject                       Eject the given device
  --eject-options               Eject options separated by comma
  --ata-smart-refresh           Refresh ATA SMART data
  --ata-smart-wakeup            Wake up the disk if it is not awake
  --ata-smart-simulate          Inject libatasmart BLOB for testing
```

Use the *emunerate* option to list the devices that the daemon is monitoring:

```
# udisks --enumerate
/org/freedesktop/UDisks/devices/dm_2d0
/org/freedesktop/UDisks/devices/dm_2d1
/org/freedesktop/UDisks/devices/dm_2d2
/org/freedesktop/UDisks/devices/dm_2d3
/org/freedesktop/UDisks/devices/sdb
/org/freedesktop/UDisks/devices/sdb1
/org/freedesktop/UDisks/devices/sr0
/org/freedesktop/UDisks/devices/sda
/org/freedesktop/UDisks/devices/sda1
/org/freedesktop/UDisks/devices/sda2
/org/freedesktop/UDisks/devices/sda3
```

Use the *monitor* option to watch activity in real time.

```
# udisks --monitor
Monitoring activity from the disks daemon. Press Ctrl+C to cancel.
added:     /org/freedesktop/UDisks/devices/sdc
added:     /org/freedesktop/UDisks/devices/sdc1
job-changed: /org/freedesktop/UDisks/devices/sdc1
changed:     /org/freedesktop/UDisks/devices/sdc1
job-changed: /org/freedesktop/UDisks/devices/sdc1
```

Here is what is displayed when a USB stick is inserted:

```
added:     /org/freedesktop/UDisks/devices/sdc
Showing information for /org/freedesktop/UDisks/devices/sdc
  native-path:                  /sys/devices/pci0000:00/0000:00:1d.7/usb2/2-2/2-2:1.0/host1
3/target13:0:0/13:0:0:0/block/sdc
  device:                       8:32
  device-file:                  /dev/sdc
    presentation:               /dev/sdc
    by-id:                      /dev/disk/by-id/usb-Kingston_DataTraveler_2.0_5B7A1000ABCA-
0:0
    by-path:                    /dev/disk/by-path/pci-0000:00:1d.7-usb-0:2:1.0-scsi-0:0:0:0
  detected at:                  Fri 12 Aug 2011 09:57:59 PM EDT
  system internal:              0
  removable:                    1
  has media:                    1 (detected at Fri 12 Aug 2011 09:57:59 PM EDT)
    detects change:             1
    detection by polling:       1
    detection inhibitable:      1
    detection inhibited:        0
  is read only:                 0
  is mounted:                   0
  mount paths:
  mounted by uid:               0
  presentation hide:            0
  presentation nopolicy:       0
  presentation name:
  presentation icon:
  size:                        4127195136
  block size:                  512
  job underway:                no
  usage:
  type:
  version:
  uuid:
  label:
  partition table:
    scheme:                    mbr
    count:                     1
  drive:
    vendor:                    Kingston
    model:                     DataTraveler 2.0
    revision:                  PMAP
    serial:                    5B7A1000ABCA
    WWN:
    detachable:                1
    can spindown:              0
    rotational media:          Yes, unknown rate
    write-cache:               unknown
    ejectable:                 0
    adapter:                   Unknown
    ports:
    similar devices:
    media:
      compat:
    interface:                 usb
    if speed:                  480000000 bits/s
    ATA SMART:                 not available

added:     /org/freedesktop/UDisks/devices/sdc1
Showing information for /org/freedesktop/UDisks/devices/sdc1
  native-path:                 /sys/devices/pci0000:00/0000:00:1d.7/usb2/2-2/2-2:1.0/host1
3/target13:0:0/13:0:0:0/block/sdc/sdc1
  device:                      8:33
  device-file:                 /dev/sdc1
    presentation:              /dev/sdc1
    by-id:                     /dev/disk/by-id/usb-Kingston_DataTraveler_2.0_5B7A1000ABCA-
0:0-part1
    by-id:                     /dev/disk/by-uuid/3520-FE20
    by-path:                   /dev/disk/by-path/pci-0000:00:1d.7-usb-0:2:1.0-scsi-0:0:0:0
-part1
  detected at:                 Fri 12 Aug 2011 09:57:59 PM EDT
  system internal:             0
  removable:                   0
  has media:                   1 (detected at Fri 12 Aug 2011 09:57:59 PM EDT)
    detects change:            0
    detection by polling:      0
    detection inhibitable:     0
    detection inhibited:       0
  is read only:                0
  is mounted:                  0
  mount paths:
  mounted by uid:              0
  presentation hide:           0
  presentation nopolicy:       0
  presentation name:
  presentation icon:
  size:                        4124180480
  block size:                  512
  job underway:                no
  usage:                       filesystem
  type:                        vfat
  version:                     FAT32
  uuid:                        3520-FE20
  label:                       1LITTLEGEEK
  partition:
    part of:                   /org/freedesktop/UDisks/devices/sdc
    scheme:                    mbr
    number:                    1
    type:                      0x0c
    flags:                     boot
    offset:                    31744
    alignment offset:          0
    size:                      4124180480
    label:
    uuid:

job-changed: /org/freedesktop/UDisks/devices/sdc1
  job underway:                FilesystemMount, initiated by uid 502
changed:     /org/freedesktop/UDisks/devices/sdc1
Showing information for /org/freedesktop/UDisks/devices/sdc1
  native-path:                 /sys/devices/pci0000:00/0000:00:1d.7/usb2/2-2/2-2:1.0/host1
3/target13:0:0/13:0:0:0/block/sdc/sdc1
  device:                      8:33
  device-file:                 /dev/sdc1
    presentation:              /dev/sdc1
    by-id:                     /dev/disk/by-id/usb-Kingston_DataTraveler_2.0_5B7A1000ABCA-
0:0-part1
    by-id:                     /dev/disk/by-uuid/3520-FE20
    by-path:                   /dev/disk/by-path/pci-0000:00:1d.7-usb-0:2:1.0-scsi-0:0:0:0
-part1
  detected at:                 Fri 12 Aug 2011 09:57:59 PM EDT
  system internal:             0
  removable:                   0
  has media:                   1 (detected at Fri 12 Aug 2011 09:57:59 PM EDT)
    detects change:            0
    detection by polling:      0
    detection inhibitable:     0
    detection inhibited:       0
  is read only:                0
  is mounted:                  1
  mount paths:             /media/1LITTLEGEEK
  mounted by uid:              502
  presentation hide:           0
  presentation nopolicy:       0
  presentation name:
  presentation icon:
  size:                        4124180480
  block size:                  512
  job underway:                no
  usage:                       filesystem
  type:                        vfat
  version:                     FAT32
  uuid:                        3520-FE20
  label:                       1LITTLEGEEK
  partition:
    part of:                   /org/freedesktop/UDisks/devices/sdc
    scheme:                    mbr
    number:                    1
    type:                      0x0c
    flags:                     boot
    offset:                    31744
    alignment offset:          0
    size:                      4124180480
    label:
    uuid:

job-changed: /org/freedesktop/UDisks/devices/sdc1
  job underway:                no
```

Remember that *udisks* is an abstraction for, among other things, performing operations on block devices. So you can do things like retrieve information about a device or mount or unmount the device.

Use the *show-info* option to retrieve information about a device.

```
# udisks --show-info /dev/sr0
Showing information for /org/freedesktop/UDisks/devices/sr0
  native-path:                 /sys/devices/pci0000:00/0000:00:1f.2/host1/target1:0:0/1:0:
0:0/block/sr0
  device:                      11:0
  device-file:                 /dev/sr0
    presentation:              /dev/sr0
    by-id:                     /dev/disk/by-id/ata-HP_DVD_Writer_1070d
    by-path:                   /dev/disk/by-path/pci-0000:00:1f.2-scsi-1:0:0:0
  detected at:                 Fri 19 Aug 2011 12:00:11 PM EDT
  system internal:             0
  removable:                   1
  has media:                   0
    detects change:            1
    detection by polling:      1
    detection inhibitable:     1
    detection inhibited:       0
  is read only:                0
  is mounted:                  0
  mount paths:
  mounted by uid:              0
  presentation hide:           0
  presentation nopolicy:       0
  presentation name:
  presentation icon:
  size:                        0
  block size:                  0
  job underway:                no
  usage:
  type:
  version:
  uuid:
  label:
  drive:
    vendor:                    HP
    model:                     HP DVD Writer 1070d
    revision:                  LH23
    serial:
    WWN:
    detachable:                0
    can spindown:              0
    rotational media:          Yes, unknown rate
    write-cache:               unknown
    ejectable:                 1
    adapter:                   /org/freedesktop/UDisks/adapters/0000_3a00_3a1f_2e2
    ports:
      /org/freedesktop/UDisks/adapters/0000_3a00_3a1f_2e2/host1
    similar devices:
    media:
      compat:                  optical_cd optical_cd_r optical_cd_rw optical_dvd optical_d
vd_plus_r optical_dvd_plus_r_dl optical_dvd_plus_rw optical_dvd_r optical_dvd_ram optical_
dvd_rw optical_mrw optical_mrw_w
    interface:                 scsi
    if speed:                  (unknown)
    ATA SMART:                 not available
```

Nowadays the preferred way of mounting or unmounting filesystems without root access seems to be *udisks*. To mount a device:

```
#  udisks --mount /dev/sdc1
Mounted /org/freedesktop/UDisks/devices/sdc1 at /media/1littlegeek
```

You can pass mount options to *udisks*. For example:

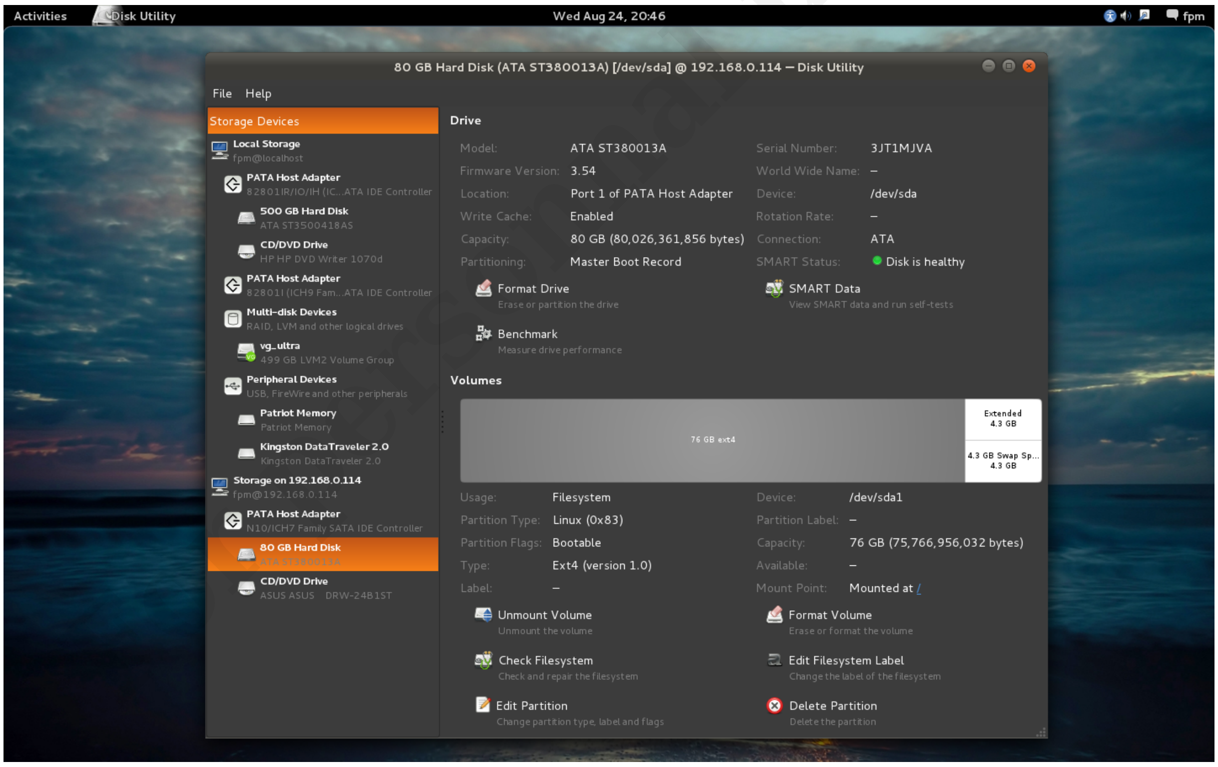
```
# udisks --mount /dev/sdc1 --mount-fstype=vfat
```

To unmount a device:

```
#  udisks --unmount /dev/sdc1
```

By the way, the *udisks* utility does not provide any options to increase the verbosity or to enable debug output, and does not write anything to *syslog* or any other log.

You can remotely manage disks using *udisks*, *udisks-tcp-bridge* and *ssh*. This is how the GNOME Disk Utility (*Palimpsest*) is able to manage disks on remote servers. For example here is a screenshot where *Palimpsest* is remotely accessing a disk on a system whose IPv4 address is 192.168.0.114.



Drive

Model: ATA ST380013A
Firmware Version: 3.54
Location: Port 1 of PATA Host Adapter
Write Cache: Enabled
Capacity: 80 GB (80,026,361,856 bytes)
Partitioning: Master Boot Record
Serial Number: 3JT1MJVA
World Wide Name: –
Device: /dev/sda
Rotation Rate: –
Connection: ATA
SMART Status: ● Disk is healthy

Volumes

Usage: Filesystem
Partition Type: Linux (0x83)
Partition Flags: Bootable
Type: Ext4 (version 1.0)
Label: –
Device: /dev/sda1
Partition Label: –
Capacity: 76 GB (75,766,956,032 bytes)
Available: –
Mount Point: Mounted at /

All that is needed for this to work is that the remote system has an operational ssh server and *udisks-tcp-bridge* is installed. If you are a Fedora 15 client, you also need to install the *openssh-askpass* package so that *Palimpsest* can ask you for a password if it is required. Otherwise you will see a "ssh_askpass: exec(/usr/libexec/openssh/ssh-askpass): No such file or directory" message.

Here is how *Palimpsest* internally sets up the connection:

- Client sets up a D-Bus server on port *LOCAL_PORT* (in the 9000 to 10000 range)
- Client creates a big random number *SECRET*

- Client creates a ssh connection to Server requesting a port-forward to Client:LOCAL_PORT using -R 0:localhost:LOCAL_PORT
- Client parses *REMOTE_PORT* and launches *udisks-tcp-bridge -p REMOTE* on Server and then writes *SECRET* and then a newline
- Server (through the bridge) connects to Client:LOCAL_PORT
- Server invokes the *org.freedesktop.UDisks.Client.Authorize* method with *SECRET*
- Client checks *SECRET*. If incorrect the Server is disconnected, otherwise the collection is established.

There is a very close relationship between *udisks* and *udev*. Normally, on modern desktops when a new USB stick or DVD inserted, it is automatically mounted and a file manager is then displayed because of a rule in the *udev* database. Suppose, instead, that you want that USB stick or DVD to be ignored and not mounted. How would you configure your system to do this?

An Internet search will return many pages of links that purport to show you how to do this. Guess what? 99.99% of the described methods are wrong as far as modern versions of Linux are concerned. The majority of the solutions simply did not work because the documentation refers to obsolete versions of *udev*, *HAL* (Hardware Abstraction Layer), *DBus* or solutions based on *PolicyKit* or *gsettings*

The simplest method that I have found is to create a low numbered file (I call mine *10-local.rules*) in */etc/udev/rules.d* containing the following rule:

```
ACTION=="add|change", SUBSYSTEM=="block", KERNEL=="sd*|sr", ENV{UDISKS_PRESENTATION_HIDE}="
1"
```

A twist on the above rule would apply to a kiosk-like LiveCD containing a persistent data partition to restrict mounting to /dev/sda*:

```
ACTION=="add|change", SUBSYSTEM=="block", KERNEL=="sd[b-e]*|sr", ENV{UDISKS_PRESENTATION_H
IDE}="1"
```

Device properties in the udev database that affect *udisks* operations include the following:

- *UDISKS_PRESENTATION_HIDE*: If set to 1, this is a hint to presentation level software that the device should not be shown to user.
- *UDISKS_PRESENTATION_NOPOLICY*: If set to 1, this is a hint to presentation level software that the device should not be automounted or auto-assembled.
- *UDISKS_DISABLE_POLLING*: If set to 1, disable the polling of drives for media changes, for devices which do not send out media notifications by themselves.
- *UDISKS_PRESENTATION_NAME*: The name to use for the device when presenting it to the user.
- *UDISKS_PRESENTATION_ICON_NAME*: The icon to use when presenting the device to the user.

Read the *udisks(7)* manpage for the full list and more details.

If you want more information about *udisks*, you should check out freedesktop.org. By the way, there is also a *udisks2* (Version 1.90.0) specification which uses the *org.freedesktop.UDisks2* bus name that we have not discussed here as it is work in progress.