Translation(s): English - Français - Castellano - Italiano

This page describes how to connect an iOS device (e.g. iPhone, iPod Touch, iPad, Apple TV) to a Debian Squeeze (DebianSqueeze) or Wheezy (DebianWheezy) system, using libimobiledevice. This enables the transfer of music and other files between an iPhone and a Debian computer, as well as some other functionality (see "What is Covered and Not Covered by This Document", below).

The terms "iPod" and "iPhone" are used interchangeably in this document, and "iPad" and "Apple TV" may also be appropriate surrogates.

Inhaltsverzeichnis

What is Covered and Not Covered by This Document

Older iPods

Older model iPods that do not run iOS are well-supported by a simple installation of DebianPkg: gtkpod -- you need not go through all this hassle to connect an older-model iPod and transfer music.

Debian OS on my iPhone?

* Nope. Not here. This page is for connecting an iPhone running Apple's iOS to a Debian computer.

Tethering?

* Tethering is not covered here; however if you want to tether via USB, setting up libimobiledevice may be required, and thus some of these instructions may be useful for you. For bluetooth tethering, libimobiledevice is not helpful, and thus neither is this document.

* The Arch Linux Wiki has a great article on iPhone Tethering. It would be great to debianize and integrate those instructions onto this page, or perhaps split this page into several pages dedicated to Debian and iPhone/iOS.

Details of libimobiledevice

* From the libimobiledevice webpage:

```
libimobiledevice is a cross-platform software library that talks the protocols to support iPhone, iPod Touch, iPad and App
```

* The author(s) of this document mostly just wanted to copy music between an iPhone and Debian computer. Though libimobiledevice enables much more functionality than that, such functionality is not covered in this document. However, this document can help you get libimobiledevice working and talking to your iPhone.

Updates Needed?

* This document had its last major update in Feb 2012, for libimobiledevice 1.1.1-3 and iOS 5. It has been some time since then, and that version of libimobiledevice may or may not work for more recent versions of iOS. However, these instructions may still be useful for installing later versions of libimobiledevice and other software. (In fact, the backporting instructions may be useful as a reference point for configuring completely unrelated software)

Getting Started - Backup your data

* These instructions explain how to set up your iPhone so that you can sync (copy music) from Debian Squeeze or Wheezy using a program such as gtkpod, Rhythmbox, amarok or banshee. Doing so should not disrupt your iTunes sync-- even after adding music, you should still be able to access the device using iTunes. That said, keep in mind: 1. this is free software; 2. it comes with no guarantee, and 3. it is possible that a bug could exist that could delete your music and other files. Most of us haven't had such issues, but we advise that you back up your data before starting.

Configure the iPhone, part 1

* Once configured, your iPhone should work on any GNU/Linux machine, without needing to rerun this configuration. (Provided the GNU/Linux machine has the necessary packages installed.)

* If the iphone already has music on it, you may skip to the next step

1. Create an initial audio database, if one does not exist already.
   1. Plug the iPhone into an MS Windows or Mac OS X computer
   2. Fire up iTunes
   3. Add at least one song or free podcast to the iPhone.

* This initializes a database on the iPhone which is needed for the rest of the instructions to work.

[There may be a way to copy an initial database file using only Debian; if you know how, please edit this wiki page.
The libimobiledevice homepage mentions a project called ideviceactivate.]

Connect the iPhone

* If your iPhone has a recent version of iOS (5 or later? Maybe 4.3? 4.2.1?) then you will need libimobiledevice 1.1.1 or later, available in Wheezy. Squeeze has libimobiledevice 1.0.2, which will not work with recent versions of iOS. There is a rather involved backport, described below. It's probably a good idea to first proceed with the normal installation, before attempting the backport. If the normal installation doesn't work (nowadays usually due to an "unhandled lockdown error") then you can try the backport.

1. Install several packages:

   ```
   aptitude install libimobiledevice-utils gvfs-backends gvfs-bin gvfs-fuse
   ```

2. As root, edit the file **/etc/fuse.conf**:
   Uncomment (remove the # symbol from) **`#user_allow_other`** at the end of the file.
   If you can't find the line then simply make a new line at the end of the file that reads **`user_allow_other`**.
   Save the file! If you are not root then you will not be able to save it, and you will have to do it again, as root.
3. Ensure that your user is a member of the fuse group: as normal user, type **`groups`** and look for "fuse". If not, then:
   1. Add your user to the fuse group. I'll use "sheila" as an example; replace sheila with your username.

      ```
      usermod -aG fuse sheila
      ```

      [You should do this for each user on your system that should be able to connect iPhones to the computer]
      [Alternately you can use a GUI tool like Gnome's **System → Administration → Users and Groups**]
   2. After adding your user to the fuse group, log out and log back in. If your GNOME session (or other desktop session) was started before your account was added to the "fuse" group, then your environment does not yet include the fuse group; thus you will not be able to run programs that require group "fuse" permissions.
4. Pray. :-) If the next step does not work then there may be some significant challenges ahead.
5. Connect the iPhone.
6. Hit *Cancel* if you are asked to open it as a camera to import pictures;
7. Alternately, if you get an option to do an AFC or music mount, then you can do that.
8. If you are lucky, the iPhone will appear on the desktop, with its name (e.g. "Sheila's iPhone" or "iPhone of Sheila").
9. By default, it is mounted in ~/.gvfs (e.g. for sheila, **`/home/sheila/.gvfs/iPhone of Sheila`**)
10. If you are not so lucky -- that is, you don't see anything on the desktop, and **`~/.gvfs/iPhone`** does not exist -- then you should do a little troubleshooting.
    It may be that you are simply not running an automount daemon, or the problem may be more difficult. Go to the Mount Troubleshooting section.
    Tip: if you prefer to mount via the command line rather than with auto-mount daemons, see the mount script.

Configure the iPhone, part 2

* If you are running Wheezy or later, the following happens automatically, and you can skip to the next section.

* Start up a terminal: this section will be performed on the command line. You should be a normal user-- not root or sudo

1. First download and set up the mount script and then come back here.
2. Assign the serial number of your iPhone to a variable called $serial. Note serial= is correct, $serial= will not work

   ```
   serial=$(./mount-iphone.sh echo_serial)
   ```

3. Verify that the variable was properly assigned

   ```
   echo $serial
   ```

   If this doesn't show the serial number, go back up-- the next steps will not work.
4. cd to your home folder. (if you type cd by itself, it will bring you to your home folder)

   ```
   cd
   ```

5. cd into the iphone's mount point folder

   ```
   cd .gvfs
   cd "iPhone of Sheila"
   ```

   note: the double-quotes are important, if the name has spaces or apostrophes or other special characters in it
   *Tip: tab-completion is a handy terminal tool. Type **`cd iP`** and then press the **TAB** key.*
6. moving along...

   ```
   cd iTunes_Control
   ```

ls

7. Does the **ls** above show a directory called **Device**?
   1. If not, then

      mkdir Device
8. Return to the parent directory (e.g. "~/.gvfs/iPhone of Sheila")

   cd ..
9. Make sure you're in the base directory of the iPhone

   pwd

   The above should output, for example:

   `/home/sheila/.gvfs/iPhone of Sheila`

   If not, then get into that directory. The next step depends on it.
10. Use the following command (from DebianPkg: libgpod-common) to create `iTunes_Control/Device/SysInfoExtended`:

    ipod-read-sysinfo-extended $serial $PWD
11. Check for the new file

    ls -l iTunes_Control/Device/SysInfoExtended

* Your iPhone is now ready to interface with libgpod and all the tools that use it, like gtkpod, rhythmbox, amarok, banshee, etc.! You can use these tools on any modern linux computer to add and delete music from your iPhone, in addition to your iTunes sync computer, without destroying the playlist.

Using Rhythmbox

1. First, install it:

   aptitude install rhythmbox rhythmbox-plugins
2. Log out and log back in again.

   (*It's not clear why this is necessary, but may fix the problem if you get an error when attempting to mount the along the lines of:* *"`Error: DBus error org.freedesktop.DBus.Error.ServiceUnknown: The name :1.41 was not provided by any .service files`".*)
3. Launch Rhythmbox:

   Applications → Sound and Video → Rhythmbox
4. Find the iPhone on the left panel
5. Click on the iPhone to see what music is on it
6. You can now Drag-and-Drop files between the iPhone and your rhythmbox music collection, similar to the way you would in iTunes and other programs.

* NOTE: Older versions of rhythmbox may have a difficult time syncing the iPhone properly. Some of us have had real difficulties with music we put on the device not ever showing up. Gtkpod seems to sync more reliably.

* The following may help:

1. Use rhythmbox's rightclick menu to eject the iPhone, and wait for some time while it syncs
2. Connect the iPhone to your iTunes computer and cancel the "cancel sync" operation
3. Install gstreamer1.0-plugins-* (all of them... this seems to help some people)
4. ??? not sure what else

mount-iphone.sh script

* Some people do not have an automount daemon in their DE; some others simply prefer to mount things manually.

* This script is also very useful for the section, "configure the iPhone, part 2"

1. Script is here: mount-iphone.sh
   (clicking the link takes you to an "Attachment page" where you can "Download" the script)
2. Open a terminal
3. Navigate to the directory where you saved the script
4. Set the execute bit:

   chmod +x mount-iphone.sh

* For those who prefer to simply copy a few commands, the script basically does this:

```
idVendor=$(lsusb -v 2>/dev/null | awk '/idVendor.*Apple/{print $2; exit}')
iSerial=$(lsusb -v -d $idVendor: 2>/dev/null | awk '/iSerial/{print $3; exit}')
gvfs-mount afc://$iSerial/
```

or

gvfs-mount -u afc://$iSerial/

or

gvfs-mount -l

or

echo $iSerial

* Now it is time to try running the script.

1. First run it with no parameters, to see if it detects the iPhone

   `./mount-iphone.sh`

   If you see something like,

   `fb9961044533cd317cb6f2bce3424c2771ae16d6 is mounted`

   or

   `fb9961044533cd317cb6f2bce3424c2771ae16d6 is not mounted`

   ...then that's good!
2. If it is not yet mounted, you can mount it like so:

   `./mount-iphone.sh mount`
3. If all is well, you will see a message like this:

   `mounted iphone with serial fb9961044533cd317cb6f2bce3424c2771ae16d6`

   (note, your serial number will be different)
4. Alternately, you might see an unwelcome error message such as the following:

   `Unhandled lockdown error (-5)`

   If you get an error, go to the Mount Troubleshooting section.

* If it worked correctly, you may want to go back to Configure the iPhone, part 2.

Here is the script inline, for reference convenience: (skip to the Next Section)

```
# debian's wiki system seems to prohibit a code section starting with #! - so delete the two lines before #!/bin/bash ;)
#
#!/bin/sh
#
# mount-iphone.sh
# This script attempts to mount or unmount the first connected ipod/iphone.
# Usage: ./mount-iphone.sh [mount | umount | echo_serial]
# It should be dash-friendly
#
# Written by Mohamed Ahmed, Dec 2010
#
# Refactored and extended by David Emerson, Feb 2012
#
# You can configure send_msg to use either a console echo, or notify-send.
# notify-send is part of the debian package, libnotify-bin
# The apple pictures in /usr/share/pixmaps are part of the gnome-desktop-data package
#
# uncomment the following if you want to see the mount command used:
# show_mount_cmd=1

# you can uncomment this line to see all the commands sh executes:
# set -x

show_msg ()
{
  # notify-send -t 4000 -u normal "mount-iphone" "$1" -i "/usr/share/pixmaps/apple-$2.png"
  echo "$1" >&2
}

get_device_ids ()
{
  # get the Apple vendor id (idVendor) from lsusb
  idVendor=$(lsusb -v 2>/dev/null | awk '/idVendor.*Apple/{print $2; exit}')
  [ -z "$idVendor" ] && { show_msg "Cannot find any Apple device" "red"; exit 1; }
  # get the device serial number (iSerial)
  iSerial=$(lsusb -v -d $idVendor: 2>/dev/null | awk '/iSerial/{print $3; exit}')
  [ -z "$iSerial" ] && { show_msg "Cannot find serial number of Apple device $idVendor" "red"; exit 1; }
}

is_mounted ()
{
  gvfs-mount -l | grep -i "mount.*$1" >/dev/null
}

mount_iphone ()
```

```
{
  [ -z $show_mount_cmd ] || echo gvfs-mount afc://$1/ >&2
  if gvfs-mount afc://$1/; then
    show_msg "mounted iphone with serial $1" "green"
  else
    show_msg "iphone mount failed" "red"
    exit 1
  fi
}

unmount_iphone ()
{
  ## now gvfs unmount the device
  [ -z $show_mount_cmd ] || echo gvfs-mount -u afc://$1/ >&2
  if gvfs-mount -u afc://$1/; then
    show_msg "unmounted iphone with serial $1" "red"
  else
    show_msg "iphone umount failed" "red"
    exit 1
  fi
}

case $1 in
  mount)
    get_device_ids
    is_mounted && { show_msg "$iSerial is already mounted" 'green'; exit; }
    mount_iphone $iSerial
    ;;
  umount|unmount)
    get_device_ids
    is_mounted || { show_msg "$iSerial is not mounted" 'red'; exit; }
    unmount_iphone $iSerial
    ;;
  echo_serial)
    get_device_ids
    echo $iSerial
    ;;
  '')
    get_device_ids
    is_mounted && show_msg "$iSerial is mounted" 'green' || show_msg "$iSerial is not mounted" 'red'
    ;;
  *)
    echo "Usage: $0 [mount | umount | echo_serial]"
    exit 1
    ;;
esac

exit 0
```

Mount Troubleshooting

* If you are here, then chances are you had trouble mounting the device. If not, then you need not be here...

1. The first troubleshooting action to take is to set up the Mount Script, above. Do that!
2. Try and mount
   `./mount-iphone.sh mount`

* If you get an **`Unhandled lockdown error`**, the cause for this error can be varied:

- If you are running wheezy, solving the lockdown error may be as simple as running:
  `idevicepair unpair && idevicepair pair`
- If you are running squeeze:
  - This error is caused by a change in iOS behavior, which broke libimobiledevice. (Introduced in iOS 5, late 2011.)
  - In order to fix this error in squeeze, you may need to backport libimobiledevice 1.1.1. This is not trivial, but instructions are below!

* Other errors... please help expand the wiki!

backporting libimobiledevice 1.1.1 to Squeeze

backporting caveats

* As of 2014-Mar there is no official backport. It would be great if there was! You should check for one before doing this. Really.
http://backports.debian.org/Packages/ - search for libimobiledevice in squeeze-backports and squeeze-backports-sloppy

* If someone has backported libimobiledevice 1.1.1, then please make mention of it here.

* Don't delete this section, though, as it may very well be useful to someone in the future, for reasons unknown and unexpected to you now.

* Unfortunately, after performing this backport, Squeeze's rhythmbox is still a little buggy, and is not very reliable about putting music on the iPod. Note the issues mentioned in the rhythmbox section. You can definitely use it to get music FROM the iPod, though!

* Squeeze's gtkpod doesn't seem to work at all. However, it is relatively easy to compile gtkpod 2.0.2 from snapshot.debian.org, and that seems to work pretty well with iOS 5!

* In order to proceed with building this backport, you must be comfortable with the following:

1. manipulating /etc/sources.list - making multiple changes
2. modifying source files
3. building debian packages using dpkg-buildpackage
4. installing debs
5. patiently following all the steps in order

* NOTE: this works with libimobiledevice 1.1.1-3, available 2012-Feb-16. If, by the time you try this, wheezy has moved to a later version of libimobiledevice, and it does not work, then your best bet may be to get libimobiledevice 1.1.1 from http://snapshot.debian.org - refer to the gtkpod 2.0.2 instructions, below.

* Alright, if you're still with me, let's get started...

step-by-step instructions for the backport

- Configure the build environment
  1. Be root
  2. First we need to configure our build environment:
     `apt-get update`
     `aptitude install build-essential`
- Prepare to compile libimobiledevice
  1. Now we need to get the wheezy or sid version of the libimobiledevice sources. In `/etc/apt/sources.list`, add:
     `deb-src http://ftp.us.debian.org/debian/ wheezy main`
     (make sure it says deb-src not deb)
     (also make sure this is the only deb-src line. You can use sid rather than wheezy, but don't have a squeeze deb-src line)
  2. update apt's list
     `apt-get update`
  3. Fetch the build dependencies
     `aptitude build-dep libimobiledevice`
  4. You might also like to install ccache, which caches compiled objects and speeds build time (after the first compilation)
     `apt-get install ccache`
  5. Be normal user
  6. Go to a directory of your choice where you would like to build these packages, and then:
     `apt-get source libimobiledevice`
- Compile libimobiledevice
  1. *Unfortunately it probably won't work without modification. There are a couple things we need to fix...*
  2. in **`tools/idevicebackup2.c`**, there is an instance of "**`GStatBuf st`**" that must be changed to "**`struct stat st`**"
     (because GStatBuf is a new type that does not exist in Squeeze's libs)
  3. There has often been an issue building the documentation. You can try to build first, or you can just skip the docs.
     In **`debian/control`** comment out (with #) the entire section for **`Package: libimobiledevice-doc`**
  4. Alright, now let's try to build!
  5. Go to the libimobiledevice-1.1.1 directory
  6. Activate ccache
     `[[ $PATH = *ccache* ]] || export PATH="/usr/lib/ccache:$PATH"`
  7. build it!
     `dpkg-buildpackage -b -us -uc`
  8. If it is successful then it will put several .deb files in the *parent* directory.
- Install the new libimobiledevice
  1. Be root
  2. Go to the directory where those shiny new debs are
  3. The ones you want to install are `libimobiledevice2`, `libimobiledevice-dev`, `libimobiledevice-utils`, and `python-imobiledevice`
     `dpkg -i libimobiledevice2... ...and the others...`
- Compile and install gvfs and libgpod
  1. gvfs and libgpod list libimobiledevice-dev as a build-dependency. So we need to make sure the 1.1.1 version of libimobiledevice-dev is

installed, so that gvfs and libgpod are linked against that! If the above steps were successful, then you should have libimobiledevice-dev 1.1.1 installed. You can check like so:

`dpkg -l libimobiledevice\*`

2. Stay as root
3. Return to /etc/apt/sources.list; remove that deb-src sid/wheezy line, and re-enable the deb-src squeeze line.
   *(From this point onward, we want to build **squeeze**'s packages against the wheezy/sid libimobiledevice-dev we compiled)*
   `apt-get update`
4. Get more build dependencies
   `aptitude build-dep gvfs libgpod gtkpod`
5. Gtkpod 2.0.2 also requires a couple other build dependencies. These are not fetched automatically because aptitude is getting build dependencies for the packages available via your deb-src lines, which refer to Squeeze's gtkpod 0.99.14. Since 2.0.2 has added a couple requirements since 0.99.14, we need to get them manually:
   `aptitude install libanjuta-dev libgdl-1-dev libgstreamer0.10-dev`
6. Be normal user
7. Go to your build folder (if you're not already there :-)
8. Get the sources
   `apt-get source gvfs`
   `apt-get source libgpod`
9. If you like, you can edit debian/changelog and increment the version. I inserted these lines on the top:

```
gvfs (1.6.4-3+ios501) unstable; urgency=low

  * Recompile, linking against libimobiledevice 1.1.1

 -- my name <my_email@gmail.com>  Thu, 16 Feb 2012 11:34:00 -0700
```

   and similar for libgpod
   *NOTE: **whitespace is very important** - if you get an **`error: unable to determine source changed by`** then maybe you have too many or too few whitespaces before * or -- NOTE: When I first did this, I installed these custom-compiled versions before starting with gtkpod. `aptitude build-dep gtkpod` complained that my libgpod4 0.7.93-0.3+ios501 was not the right version-- I had to recompile and install it without the +ios501 debian/changelog addition, in order to get aptitude build-dep to work.*
10. Activate ccache
    `[[ $PATH = *ccache* ]] || export PATH="/usr/lib/ccache:$PATH"`
11. cd into the gvfs folder and build:
    `dpkg-buildpackage -b -us -uc`
12. similar for libgpod - go into the libgpod folder and execute the same dpkg-buildpackage command
13. Be root
14. install `gvfs`, `gvfs-backends`, `gvfs-bin`, `gvfs-fuse`, `libgpod4`, `libgpod-common`, `libgpod-dev`
15. You might want to apply a ***hold*** to these packages in aptitude, so that you don't accidentally "upgrade" them to official debian ones later, wiping out your custom-compiled ones.

- Compile and install a newer gtkpod

  *This is rather unorthodox, **but** I found that squeeze's gtkpod simply would not work with newer iPhones, whereas wheezy's version is built against gnome3 libs, making it impractical to backport. So I looked on snapshot.debian.org for the latest available version of gtkpod prior to the gnome3 migration, which is gtkpod 2.0.2-1. This compiled with only a few complaints, and works well!*

  1. Download the gtkpod 2.0.2 source files (.debian.tar.gz, .dsc, and .orig.tar.gz) from http://snapshot.debian.org/package/gtkpod/2.0.2-1/.
  2. Put them into the build directory, where you should have lots of other such files, as well as recently-built .debs.
  3. Extract the sources:
     `dpkg-source -x gtkpod_2.0.2-1.dsc`
  4. cd into the gtkpod folder and attempt to build:
     `dpkg-buildpackage -b -us -uc`
  5. This failed for me, stating a requirement for libgpod >= 0.8.
     1. Since 0.7.93 was leading up to 0.8 (it was a jump from 0.7.2 to pre-0.8) our 0.7.93 version will work just fine, if we override the dependency checks.
     2. However, you should run the above dpkg-buildpackage to make sure there are not any ***other*** missing build dependencies.
  6. When you have verified that libgpod is the only violating dependency, you can skip checking these dependencies like so:
     `dpkg-buildpackage -b -us -uc -d`
  7. Be root
  8. Install the new gtkpod .debs:
     `dpkg -i gtkpod... gtkpod-data... libgtkpod1...`

* As you can see, this backport is not for the faint of heart. Good luck!

- After you have all this stuff installed, you will still need to:
  1. mount the device, probably using the Mount Script, above.
  2. Configure the iPhone (part 2)
  3. start gtkpod!

List of Programs

- DebianPkg: rhythmbox - music player for GNOME, which can playback and transfer content from the iPhone [SID don't seem to work with iPhone OS 4]
- DebianPkg: gthumb - an image viewer and browser for GNOME, which can import/show images from an iPod

List of Tools

Tools and programs useful to iPhone users in debian

- DebianPkg: libplist-utils - Apple property list converter
  - DebianMan: plutil(1) - A converter tool for binary or XML Apple property lists
- DebianPkg: libimobiledevice-utils - Library for communicating with iPhone and iPod Touch devices
  - DebianMan: idevicesyslog(1) - Relay syslog of a connected iPhone/iPod Touch.
  - DebianMan: idevice_id(1) - Prints device name or a list of attached iPhone/iPod Touch devices.
  - DebianMan: ideviceimagemounter(1) - Mount disk images on the iPhone/iPod Touch.
  - DebianMan: idevicebackup(1) - Create or restore backup for iPhone/iPod Touch devices.
  - DebianMan: ideviceinfo(1) - Show information about the first connected iPhone/iPod Touch.
  - DebianMan: idevicescreenshot(1) - Gets a screenshot from the connected iPhone/iPod Touch.
- DebianPkg: ipheth-utils - USB tethering driver for the iPhone [support utilities]
- DebianPkg: ifuse - FUSE module for iPhone and iPod Touch devices
  - DebianMan: ifuse(1) - Mount filesystem of an iPhone/iPod Touch.
- DebianPkg: usbmuxd - USB multiplexor daemon for iPhone and iPod Touch devices
  - DebianMan: iproxy(1) - proxy that enables tcp service access to iPhone/iPod
  - DebianMan: usbmuxd(1) - iPhone/iPod Touch USB multiplex server daemon

Orphan:

- DebianPkg: python-imobiledevice - Library for communicating with iPhone and iPod Touch devices

CategoryPhone