Podcatching without an iPod

AUDIO DELIVERY

You don't need an iPod to participate in the recent phenomenon known as Podcatching. We'll show you how to receive and play back podcasts in Linux. **BY TIM HARDY**

Podcasting is the latest buzz in the world of weblogs. Also known as time shifting audio, podcasting is a method of delivering audio streams broadcast in mp3 format via rss feeds, so that they can be downloaded automatically and played at leisure. By subscribing to an RSS feed with mp3 enclosures, you can receive new podcasts automatically for playback on your favorite mp3 player. You can think of podcasting as TiVO for radio. The name podcasting derives from Apple's iPod, but the name is inaccurate, because you do not need an iPod to listen to these files. A podcast is the broadcast program, and the act of receiving a feed is called podcatching.

Although many podcasts, like many blogs, are trivial, high quality material is also available. Doug Kate at IT Conversations [1], for instance, has made over 300 recordings of interviews and keynote speeches from IT conferences. Podcasts contain interviews with key players in the IT industry, old and new, as well as highlights from conferences, speeches, and readings. Big media corporations like the BBC in the UK are also showing interest in the genre and releasing some of their radio broadcasts as podcasts.

The buzz about podcasts is mainly attributable to former MTV VJ Adam Curry[2]. Following a discussion with Adam Curry in 2001, Dave Winer decided to change the multimedia delivery model over the Internet by adding the <enclosure> sub-element to rss feeds [3]. Curry's own website ipodder.org [4] now details hundreds of podcasts.

A podcatching client functions as an aggregator that can read these RSS 2.0 feeds with enclosures. The client automatically polls your subscribed feeds and downloads the enclosed audio files, moving them to your music library or your portable mp3 player. Users of iPods and iTunes on Macs have the easiest ride. Linux users with different mp3 players have to do a little more work to get that "automagic" transfer functioning smoothly. This article describes how to configure your Linux computer for podcatching.

Getting Podcasts onto your Computer

Certain players (like the iPod) require special drivers to sync ([5] , [6]) but this article assumes you are using a type of mp3 player that (like most flash-based storage devices) can be mounted as an external drive. I'll begin by describing some Linux podcatching applications, then I'll describe some syncing techniques. If your device needs special software under Linux to sync, you can follow the first half of the tutorial to subscribe to podcasts and adapt the second half accordingly.

iPodder [7] is a GUI application written in Python for subscribing to and receiving podcasts. At the time of this writing, the latest version for Linux was iPodder-linux-2.0-RC2.tar.bz2. There are a couple of tiny little glitches that will hopefully be smoothed out before the final version is released. iPodder requires Python2.3 or higher (but has no support for Python2.5). It also requires the fol-

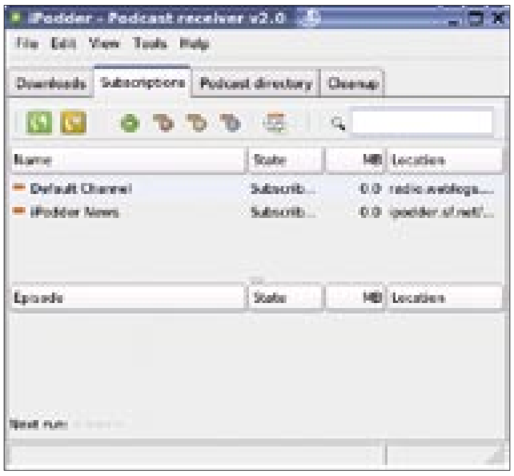

iPodder - Podcast receiver v2.0
File Edit View Tools Help
Downloads Subscriptions Podcast directory Cleanup
Name State MB Location
Default Channel Subscrib... 0.0 radio.weblogs....
iPodder News Subscrib... 0.0 ipodder.sf.net/...
Episode State MB Location
Next run:

Figure 1: iPodder's Subscriptions tab displays the feeds to which you are currently subscribed.

lowing: python-gtk, wxPythonGTK, libwxPythonGTK2.5_2, pythonlib, libpython2.3, libxml2-python and xmms-python (for xmms player support).

To install, unpack the archive, cd to the newly created folder, and run the installer as root:

```
tar -xvjf iPodder-➘
linux-2.0-RC2.tar.bz2
cd iPodder-linux/
sudo run ./install.sh
```

The package installs in */opt/iPodder/* and creates a symbolic link from */opt/iPodder/ipodder.sh to /usr/bin/iPodder*. If you want to add a desktop icon, there's an image file *iPodder.png* installed in */opt/iPodder*. Link this to */usr/bin/iPodder* and click it to launch, or just type *iPodder&* at a terminal prompt.

IPodder opens by default at the *Subscriptions* tab (Figure 1), which shows you the feeds to which you are currently subscribed. Clicking on any of the shows listed retrieves a list of podcasts available from that channel in the Episode window at the bottom. By default, only the last podcast on that feed is checked for download, but you can select any or all of the others if you want to catch up on earlier shows you have missed.

Hitting the Podcast directory tab (Figure 2) gives you a choice of podcasts to subscribe to and allows you to manually add feed addresses. Open a folder from the list by clicking on it; then double click on any feeds contained within to add the feeds to your subscriptions. When you are ready, return to the *Subscriptions* tab and hit the green button with the two arrows to set the downloads in motion. The downloads can be monitored in the *Download* tab (Figure 3).

Downloads can easily be scheduled using the scheduler option in *Tools/Scheduler*. You have an option of checking for new feeds at a defined regular interval or at a fixed time. Closing iPodder minimizes the application and leaves it running in the background so you can get on with something else.

iPodder creates a folder *~/iPodderData/downloads* where it stores the mp3s. Due to a minor glitch, I couldn't adjust this via the GUI because of the position of the display borders, but it can be changed by manually editing the configuration file *~/iPodderData/ipodder.cfg*. For the purposes of this article, you're going to want to create a directory called *~/podin* and set it as the download directory.

iPodder is a very elegant, well designed application with an intuitive interface. It only supports the iTunes and Windows Media player desktop apps at the moment for playing, but there is a very useful option *File | Preferences | Advanced* to automatically run a script on successful download; we will use the script later in this article. There are a couple of small omissions, such as the lack of a working routine to check free space on disk, but this and other minor glitches are on the developers' todo list and should be smoothed out shortly.

An alternative GUI application is jPodder [8]. jPodder is a cross-platform client written in Java. The Linux beta at this time of writing is 0.9, and it still has a number of rough edges. However, the client comes bundled with Azureus [9] and offers good support for podcasts delivered via bittorrent, so you may find this solution worth investigating.

Command Line Alternative

BashPodder [10] by Linc Fessenden is just 44 lines of bash code and it requires only bash, wget, and sed, all of which are installed by default on most Linux distributions.

Linc notes "BashPodder was written to be small and fast, and most importantly, to conform to the KISS rule (Keep It Simple Stupid). That way, anyone can add to and detract from the script to suit their own needs (and they are welcome to do so)."

To use BashPodder, download the main program *bashpodder.shell* and the sample configuration file *bp.conf* and place them in the directory in which you wish to keep your podcasts. There is nothing to unzip or install. Just *chmod +x bashpodder.shell* to make it executable.

bp.conf is a simple text file listing the urls of feeds, one per line. You can edit *bp.conf* by hand, but if you do, remember to hit return at the end of the last entry because the script will not parse a line if it does not end in a newline character.

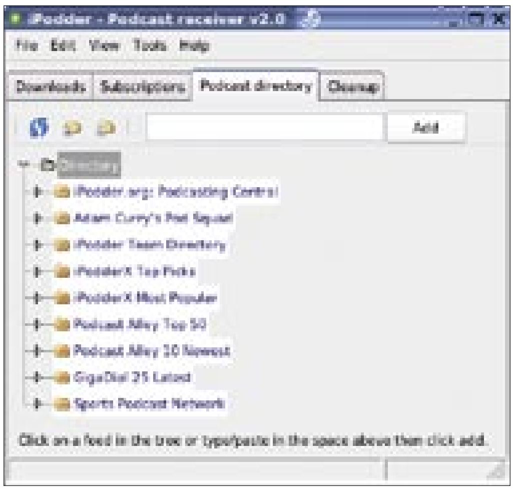

iPodder - Podcast receiver v2.0
File Edit View Tools Help
Downloads Subscriptions Podcast directory Cleanup
Add
Directory
iPodder.org: Podcasting Central
Adam Curry's Pod Squad
iPodder Team Directory
iPodderX Top Picks
iPodderX Most Popular
Podcast Alley Top 50
Podcast Alley 10 Newest
GigaDial 25 Latest
Sports Podcast Network
Click on a feed in the tree or type/paste in the space above then click add.

Figure 2: The Podcast directory tab offers a choice of subscription options and also lets you manually enter an address.

RSS Enclosures

The following example of a section of an RSS feed shows an <enclosure> tag with its three attributes: the url of the file to download, its size in bytes (length), and what its type is.

```
<enclosure ➘
url="http://downloads.bbc.co.uk➘
/rmhttp/radio4/history/➘
inourtime/➘
inourtime19_pod.mp3"➘
 length="12582787" ➘
 type="audio/mpeg"/>
```

Dave Winer's original specification defined a maximum of one enclosure per item, but as with so many aspects of RSS, this is still subject to fierce discussion.

By default, BashPodder places podcasts into a datestamped folder in the folder from which you run it. For the purposes of this article, we are going to change it to *~/podin*. To change this, open the script in your favorite text editor, find the *datadir* definition near the top, and change it from *datadir=$(date +%Y-%m-%d)* to *datadir=/path/to/podin*

To run the script automatically, use crontab. Crontab uses vi by default; to edit it using your favorite editor, export this as the variable VISUAL beforehand. Personally I like joe.

```
export VISUAL=joe
crontab -e
```

To set bashpodder.shell to run at 9am every day for example add a line:

```
0 9 * * * ➲
/path/to/bashpodder.shell
```

The site also lists plenty of user-suggested revisions to the script and add-ons, such as Leon Pennington's BPConf, a simple KDE configuration frontend [11]

Preparing to Sync

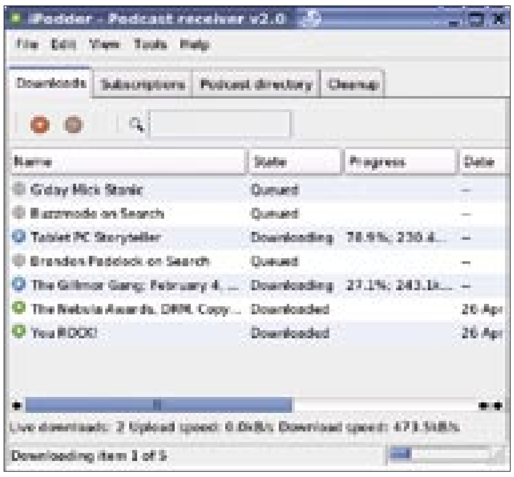
What happens when you plug in your device depends on your distribution. Suse 9.2, for example, uses *subfs/submount* and will automatically mount it with a name derived from the serial number of the device. Another quirk of the Suse distributions is that they use */media/* where others use */mnt/*. Plugging in my PowerMusic mp3 player on a machine running Suse, for instance, automatically creates a mount point */media/usb-035211010152:0:0:0p1*

(Other distributions will handle hotplug events differently, and you may have to adapt this to suit. From now on, I will refer to the mp3player mount point as */path/to/mp3player*: remember to adapt this mount point for your own system.)

For ease of use, we can create a symbolic link that is easier to remember:

```
ln -s /media/usb-➲
035211010152:0:0:0p1 ➲
/path/to/mp3player
```

We already have a folder for our new podcasts that we've called *~/podin*. We



```
iPodder - Podcast receiver v2.0
File  Edit  View  Tools  Help
Downloads | Subscriptions | Podcast directory | Cleanup

Name                                State        Progress          Date
G'day Mick Stanic                   Queued                         –
Buzzmode on Search                  Queued                         –
Tablet PC Storyteller               Downloading  78.9%; 230.4...   –
Brandon Paddock on Search           Queued                         –
The Gillmor Gang: February 4, ...   Downloading  27.1%; 243.1k...  –
The Nebula Awards. DRM. Copy ...    Downloaded                     26 Apr
You ROCK!                           Downloaded                     26 Apr

Live downloads: 2 Upload speed: 0.0kB/s Download speed: 473.5kB/s
Downloading item 1 of 5
```

Figure 3: Track and manage downloads in the Downloads tab.

now need to create another called *~/podmirror*, which will hold a copy of the files on the mp3player.

We'll leave this folder empty to begin with for reasons that will become apparent. To prevent the podcasts getting muddled up with other music tracks on your mp3 player, you can also create a folder on the device, which we will call */path/to/mp3player/podcasts/*.

That Syncing Feeling

Synchronizing a device with a folder is not as straightforward as it first appears, especially if files can be deleted directly on the mp3 player as well as on the computer. We're going to make two key assumptions: that there is nothing on the podcast folder of the device when we begin, and that there is no other means of adding an mp3 file to the device apart from syncing with the computer.

The process needs to be divided into three stages.

Stage one synchronizes the mp3player podcast folder */path/to/mp3player/podcasts/* with the *~/podmirror* folder on the computer. It deletes any file in *~/podmirror* that is not on the player based on the assumption that it has been deliberately deleted from the player because it is no longer wanted. Secondly, if a file exists on the player but does not exist in *~/podmirror*, the process ignores the file and does not create a new copy in *~/podmirror*, based on the assumption that you've deliberately deleted it from the folder on your computer.

Stage two is a simple transfer of new mp3s from *~/podin* to *~/podmirror*. This transfer cannot be done until the first synchronization has finished, otherwise the new files would be deleted, since they are not on the mp3 player.

Stage three synchronizes *~/podmirror* with */path/to/mp3player/podcasts/*. It deletes any file in */path/to/mp3player/podcasts/* that is not in *~/podmirror* based on the assumption that you have deliberately deleted it from your computer. It then creates a new copy on the mp3 player of any file in *~/podmirror* that is not on the device: these will be the new files you have transferred from *~/podin*.

Be careful that you understand the processes – and back up your device before the first sync. A mistake could be disastrous for the files on your mp3 player, just as syncing after accidentally deleting files from your player could wipeout the contents of *~/podmirror*.

To sync the device, we're going to use the command line application rsync. Most distributions should come with their own version; alternatively, download and compile the source from [12].

As detailed above, the first step is to delete any files from *podmirror/* that you've deleted on the player:

```
rsync -r --progress --delete ➲
--existing --size-only ➲
/path/to/mp3player/podcasts/ ➲
/path/to/podmirror/
```

Rewriting Filenames for Small Displays

Many mp3 players have tiny displays on which it is hard to read longer file names. Consider rewriting file names to make them shorter using sed.

IT Conversations podcasts have very long file names. To reduce them to just the name of the key speaker, run the following in *~/podin*

```
for i in ITC.*; do
    mv $i $(basename $i |
sed -e "sITC\.[^-]*-\([^-]*\)➲
[\.-].*/\1.mp3/g");
      done
```

This will convert ITC.AC2004-Richard-Marks-2004.11.07.mp3 to just Richard-Marks.mp3.

Most flash-based devices use the FAT32 filesystem that doesn't preserve unix-style time stamps.

The *--size-only* option allows you to skip updating files that are identical, but whose time stamps appear to have changed because of this.

The *-existing* flag prevents rsync from reinstalling any files you may have deleted from *~/podmirror*. The *-progress* flag will give us a clear indication of what the command is doing, which can be helpful when running manually.

Stage two, copying the new podcasts from *~/podin* to *~/podmirror*, is straightforward:

```
mv /path/to/podin/* ↩
/path/to/podmirror/
```

Stage three uses rsync again but in the opposite direction, and without the *--existing* flag this time, because we want to copy new files onto the device:

```
rsync -r --progress ↩
--delete --size-only ↩
/path/to/podmirror/ ↩
/path/to/mp3player/podcasts
```

When you are finished, don't forget to unmount the player before removing it to prevent possible damage to the drive. Unmount by giving the *eject /path/to/mp3player* command, and you'll be ready to listen to your podcasts at leisure.

Tying It All Together

We'll call the finished script, syncplayer (Listing 1). The script will fail to run if the mp3 player is not mounted or if the destination folder does not exist. Adjust */path/to/* in lines 3-5 to fit your personal set-up.

To automatically run this script when new podcasts arrive, you can set iPodder to call it by specifying syncplayer in the option to run scripts on download (File/Preferences/Advanced).

If you are using bashpodder, you just need to add a line to to the end of bashpodder.shell, either by editing it or by simply typing *echo "/path/to/syncplayer" >> bashpodder.shell*. (Be careful to type >> not > in order to append rather than overwrite the file.)

If you want to automatically run the script on insertion of your player, you can take advantage of hotplug to trigger it.

Find the vendor id and product id for the device by plugging it in and checking the output of *cat /proc/bus/usb/devices* for the relevant details of your mp3 player.

For my PowerMusic player, the following section is relevant:

```
P:  Vendor=0d7d ProdID=0153 ↩
Rev= 1.00
S:  Product=PowerMusic
```

Now create */etc/hotplug/usb/powermusic.usermap* to trigger a script we'll call *on_plug_powermusic* to run when the device is plugged in:

```
# /etc/hotplug/usb↩
/powermusic.usermap
# powermusic mp3 player
on_plug_powermusic ↩
0x0003 0x0d7d 0x0153 0x0 0x0 ↩
0x0 0x0 0x0 0x0 0x0 0x0 0x0
```

Listing 1: syncplayer

```
01 #!/bin/bash
02 # syncplayer
03 # The location of the folders.
   Adjust for your personal
   set-up.
04 mp3player=/path/to/mp3player
   # the mount point for your
   player
05 podin=/path/to/podin
06 podmirror=/path/to/podmirror
07 # Test player is mounted and
   folder podcasts exists
08 [ ! -d $mp3player/ ] && echo
   "mp3 player not mounted" &&
   exit 1
09 [ ! -d $mp3player/podcasts ]
   && echo "Device folder
   podcasts not present" && exit
   1
10 # Perform the synchronisation
   process
11 rsync -r --progress --delete
   --existing --size-only
   $mp3player/podcasts/
   $podmirror/
12 mv $podin/* $podmirror/
13 rsync -r --progress --delete
   --size-only $podmirror/
   $mp3player/podcasts
```

The first field labels which flags to match. This field should be left as 0x0003. The second and third field are the vendor id and product id, respectively, that we obtained from */proc/usb/devices*. The other fields are not used so they can be left as 0x0.

Next we need to create *on_plug_powermusic* in */etc/hotplug/usb/*

```
# /etc/hotplug/usb/↩
on_plug_powermusic
# on_plug_powermusic
until ↩
[ -e /path/to/mp3player ];↩
 do sleep 1; done
su user -c /path/to/syncplayer↩
>& /var/log/messages
exit 0
```

Change user in line 3 to your user name and fill in both cases of */path/to/*. Make the file executable, then restart hotplug:

```
chmod +x ↩
on_plug_powermusic ↩
&& /etc/init.d/boot.hotplug ↩
restart
```

Now, whenever you plug the device in, the syncplayer script is triggered, transferring any new podcasts to your mp3 player. Just remember to unmount it when it is done. ■

INFO

[1] IT Conversations: *http://www.itconversations.com/*

[2] Adam Curry's Website: *http://live.curry.com/*

[3] RSS Payloads: *http://www.thetwowayweb.com/payloadsforrss*

[4] iPodder: *http://ipodder.org/*

[5] GNUpod: *http://www.gnu.org/software/gnupod/gnupod.html*

[6] gtkpod: *http://gtkpod.sourceforge.net/*

[7] iPodder: *http://ipodder.sourceforge.net/index.php*

[8] jPodder: *http://jpodder.com/*

[9] Azureus: *http://azureus.sourceforge.net/*

[10] BashPodder: *http://linc.homeunix.org:8080/scripts/bashpodder/*

[11] BFConf: *http://www.leonscape.co.uk/linux/bpconf/*

[12] rsync homepage: *http://freshmeat.net/projects/rsync/*