Using the hpodder podcast aggregator

John Goerzen

Using the hpodder podcast aggregator

by John Goerzen

Table of Contents

I. hpodder Manual

hpodder

<jgoerzen@complete.org>

John Goerzen

Name

`hpodder` — Scan and download podcasts

Synopsis

hpodder [-d] [*`command`*] [*`command_args`*]

Description

Podcasting is a method of publishing radio-like programs on the Internet. Through podcasting, almost anyone can produce their own audio program, and publish episodes of it as often or as rarely as they like.

To listen to podcasts, you need a program to download the podcast's episodes from the Internet. Such a program is called a podcatcher (or sometimes a podcast aggregator). hpodder is this program.

If you'd like to get going RIGHT NOW, skip on down to the Quick Start section. Otherwise, let's take a look at the features of hpodder.

Feature List

- Convenient, easy to learn, and fast command-line interface (it's simple to do simple things, and advanced things are possible)
- Automatic discovery of feed metadata such as title
- Full history database for accurate prevention of duplicate downloads and tracking of new episodes
- Conversion tools to convert your existing feed list and history from other applications to hpodder. Supported applications and formats include: castpodder and ipodder.
- Most operations can work fully automatically across your entire podcast database, or they can work manually as well.
- Automatic updating of ID3 (v1 and v2) tags based on metadata in the podcast itself. This important feature is available through iTunes but is often missed by other podcatchers.
- hpodder operations can be easily scripted or scheduled using regular operating system tools.
- Fully customizable naming scheme for downloaded episodes, including a name collision detection and workaround algorithm.
- Automatic support for appending .mp3 extensions to MP3 files that lack it.

- Numerous database and history inquiry tools
- Small, minimalist footprint
- Power users and developers can interact directly with the embedded Sqlite3 database used by hpodder. The database has a simple schema that is developer-friendly.
- Support for resuming interrupted downloads of podcasts
- hpodder is SAFE and is designed with data integrity in mind from the beginning. It should be exceedingly difficult to lose a podcast episode, even in the event of a power failure.

Method of Operation

The basic pattern of operation with hpodder is to set up each podcast you want to receive. Each day (or hour, or whatever), hpodder will go out and update its database by pulling in the latest episode lists from the podcast feed. Then, hpodder will proceed to download any episodes that you haven't already downloaded. After each episode is downloaded, hpodder will note that fact so it isn't ever downloaded again.

Let's look at this in a bit more detail.

hpodder maintains two tables in a database. One table lists all the podcasts you know about, as well as where the podcast's feed is to be downloaded from. The feed is a file that the podcast's author publishes. It lists all the current episodes of the podcast, and some information about them. Data is added to this table with the **hpodder add** command.

The second table lists each episode for a given podcast, along with the location from which the episode can be downloaded and some other information about the episode (such as its title). Information in this table is added by **hpodder update** and updated by **hpodder download** or **hpodder catchup**.

When you first fire up hpodder, it will read its configuration file from `~/.hpodder/hpodder.conf`. What happens next depends on the command.

For **hpodder update**, the program will read information about all your podcasts. It will download each feed. Once it has the feed, it will look at each episode and compare them to the database. If a given episode is already in the database, it is ignored. Any new episodes are recorded in the database, and set to Pending so they will be downloaded on the next download run.

For **hpodder download**, the program will read information about all your episodes. For each episode marked Pending, the program will download the episode. It will then update the episode's ID3 tags based on the podcast feed. Finally, it will move the episode in-place atomically. Only after all that has been done will hpodder mark the episode as Downloaded in the database. In this way, no episode is visible to outside tools until it is completely downloaded in its final form, so you can safely play any visible program in your download directory even as downloads are happening.

Quick Start

This section will describe how a first-time hpodder user can get up and running quickly. It assumes you already have hpodder compiled or installed on your system. If not, please follow the instructions in the `INSTALL` file in the source distribution.

To get started, simply run **hpodder** at your shell prompt. hpodder will lead you through the first-time configuration -- which is only two questions and completely self-explanatory!

After this, whenever you want to download the latest episodes for your podcast, just run **hpodder** again.

At some point, you'll want to add more podcasts to hpodder. To do that, just run a command such as:

hpodder add ***`http://www.example.com/feed.xml`***

Just replace the example.com URL here with the real URL of the feed you want to add. Then run **hpodder update**. If the podcast you've just added has a whole bunch of episodes, you may not want to download them all. In that case, run **hpodder catchup** ***`id`***, where *`id`* is the podcast number that hpodder gave your new podcast when you added it.

Again, from here on, you can just run **hpodder** to download all your new episodes.

Options

hpodder always is invoked with the name of a specific operation, such as `update` or `add`. In hpodder, these operations are called *commands*. Each command has its own options, which are given after the command on the hpodder command line. A full summary of each command's options is given later in this manual.

You may obtain a list of all commands with **hpodder lscommands**. Help is available for any individual command with **hpodder** ***`command`*** **--help**. Global help is available with **hpodder --help**.

Global Option

This option may be specified *before* any command.

-d
--debug

Enables debugging output. This verbose output helps you learn what hpodder is doing every step of the way and diagnose any problems you may encounter.

Commands in hpodder

hpodder has many different commands. If you do not specify a command, the **fetch** command is automatically selected for you. This section will discuss each command in detail. Note that all commands are case-sensitive and should be *given in lowercase*.

All commands support the command `--help`. Running **hpodder** ***`command`*** **--help** will display information about the command and its options. Since all commands support this, it won't be explicitly listed for each command below.

add

hpodder add *`URL`*

This command is used to add a new podcast to hpodder. You can must provide the URL (link) to the podcast you want to add to this command. For example:

hpodder add http://soundofhistory.complete.org/files_soh/podcast.xml

A podcast can be later removed with **hpodder rm**. You can adjust its URL later with **hpodder mv**.

catchup

hpodder catchup [-n `number`] [`castid`...]

Running **catchup** will cause hpodder to mark all but the most recent episodes as Skipped. This will prevent hpodder from automatically downloading such episodes.

-n *NUM*
--number-eps=*NUM*

> By default, only the single most recent episode is exempted from being "caught up". If you want to exclude more episodes from being "caught up" -- and thus allow more to be downloaded -- use this option to allow more episodes to remain downloadable.

castid ...

> By default, this command will operate on all podcasts. You can limit the podcasts on which it operates with this option. See specifying podcast IDs later in this manual for more information.

disable

hpodder disable castid...

This command will flag podcasts as disabled. Podcasts flagged disabled will be skipped during an **update**, **download**, or **fetch**. They will still participate with all other commands. **hpodder lscasts** will notify you of which podcasts are disabled.

This can be useful if you want to stop following a podcast for awhile, but think you may want to come back to it in the future. The podcast URL and your download history will remain in the hpodder database, unlike with **hpodder rm**.

Disabled podcasts can be re-enabled with **hpodder enable**.

One or more podcast IDs are required; see the section below on specifying podcast IDs for more details.

download

hpodder download [`castid`...]

The **download** command is used to actually perform the download of podcasts to your system. By default, **download** will download all available episodes. You can, however, specify only certain podcasts to process; if you do, all available episodes for only those podcasts will be downloaded.

castid ...

By default, this command will operate on all podcasts. You can limit the podcasts on which it operates with this option. See specifying podcast IDs later in this manual for more information.

enable

hpodder enable castid...

This command will flag podcasts as enabled. This is the default state. See **hpodder disable** for information on manually disabling podcasts and what it means to be disabled.

One or more podcast IDs are required; see the section below on specifying podcast IDs for more details.

fetch

hpodder fetch [*castid*...]

The **fetch** is the main worker command for hpodder. It is simply equivolent to **hpodder update** followed by **hpodder download**. That is, it will scan all podcasts for new episodes, then download any pending episodes.

This command is the default command if no command is given on the hpodder command line.

As a special feature, the first time that **fetch** is invoked, it will execute the new user setup procedure.

castid ...

By default, this command will operate on all podcasts. You can limit the podcasts on which it operates with this option. See specifying podcast IDs later in this manual for more information.

import-ipodder

hpodder import-ipodder [--from=*PATH*]

With this command, hpodder can import both your podcast list and your download history from ipodder or CastPodder. hpodder will import all podcasts referenced there, with the exception that any podcasts that are already in hpodder's database will be entirely untouched.

--from=*PATH*

By default, hpodder will look for the ipodder database in the `.ipodder` directory in the user's home directory. This may not always be correct: for instance, on non-Unix platforms or when using CastPodder, this directory will be different. With this option, you can tell hpodder where to find the ipodder/CastPodder database.

lscasts

hpodder lscasts [-l]

This command will display all podcasts that are configured within hpodder. For each podcast, you will see the podcast ID, the number of pending downloads, the total number of episodes ever seen by hpodder, and the title of the podcast.

-l

If you add the `-l` option, then **lscasts** will also display the feed URL for each podcast.

lscommands

hpodder lscommands

This command will display a list of all available hpodder commands along with a brief description of each.

lsepisodes / lseps

hpodder lsepisodes [-l] [*`castid`*...]

hpodder lseps [-l] [*`castid`*...]

The **lsepisodes** command will display a list of every episode known to hpodder. The output will include the ID of the podcast to which the episode belongs, the episode ID, the status of the episode, and the title of the episode.

lseps is simply an alias for **lsepisodes** and performs in the same manner.

-l

If you add the `-l` option, then **lsepisodes** includes the download URL for each episode in its output.

`castid ...`

By default, this command will operate on all podcasts. You can limit the podcasts on which it operates with this option. See specifying podcast IDs later in this manual for more information.

rm

hpodder rm castid...

This command will remove all knowledge about a given podcast from hpodder, including all entries about that podcast in the episode database.

One or more podcast IDs are required; see the section below on specifying podcast IDs for more details. Unlike most other hpodder commands that accept an empty podcast ID list to mean all podcasts, **rm** does not because of the destructive potential of such a request.

setstatus

hpodder setstatus --castid=`ID` --status=`STATUS` epid...

The **setstatus** command is used to manually adjust the status flags on individual episodes. You can use it to flag individual episodes for downloading (or not).

You must specify at least one episode ID. *Note that the plain IDs given to this command are episode IDs*, and not podcast IDs like other commands.

Statuses are case-sensitive and must be given with a leading uppercase letter and trailing lowercase letters. Available status are given later in this manual.

settitle

hpodder settitle --castid=`ID` --title=`TITLE`

The **settitle** is used to manually set the title of a given podcast. Normally, hpodder will automatically get the title from the podcast's XML feed. Sometimes the XML feed for the podcast may not provide a useful title. In those situations, you can use **settitle** to manually override the title.

Please note that if you want to set the title to a name that contains spaces, you will need to quote it for the shell.

update

hpodder update [`castid`...]

The update command will cause hpodder to look at each podcast feed. It will download the latest copy of the feed and compare the episodes mentioned in the feed to its internal database of episodes. For any episode mentioned in the feed that is not already in the internal database of episodes, hpodder will add it to its database and set its status to Pending.

`castid ...`

By default, this command will operate on all podcasts. You can limit the podcasts on which it operates with this option. See specifying podcast IDs later in this manual for more information.

Specifying podcast IDs

Each podcast in hpodder gets a numeric ID. This ID is automatically assigned by hpodder and is not changable. The ID is given out when a podcast is added with the **add** command, or with the **lscasts** or **lsepisodes** commands.

The ID is designed as a constant way to refer to a particular podcast. A podcast's title may change, or even its feed URL, but the ID of a podcast will never change. It also is short and easy to type on the command line.

Several commands can take a list of podcast IDs. If no IDs are given, the commands will default to operating on all podcasts. One or more IDs can be given, separated by spaces. If IDs are given, then the commands will operate only on the podcasts with the given IDs.

The special keyword `all` may be given, which tells the system to operate on all podcasts. This yields the same result as giving no IDs at all.

Status Flags in hpodder

Several places in this manual, you've seen hpodder statuses mentioned. Each episode in hpodder has an associated status. The statuses are:

Pending

> The given episode is ready to download

Downloaded

> The given episode has already been downloaded by hpodder

Error

> An error occured while downloading this episode. It will not be downloaded again unless the flag is set back to Pending.

Skipped

> The user has requested that this episode not be downloaded. Commands such as **catchup** or **import-ipodder** could cause this.

Automatic Error Handling

For whatever reason, podcast feeds or individual episodes sometimes fail to download. The reasons for this range from the podcast being taken down by its author to the network being disconnected from the local computer.

People that track many podcasts over a long time will probably find it annoying to have hpodder attempt to download invalid feeds or episodes over and over again. For that reason, hpodder 1.0.0 introduced automatic error handling.

Once a podcast feed or episode has failed at least 15 times, it's been at least 21 days since the first download attempt (episodes) or last update (feeds), hpodder will automatically mark the item to be skipped in future runs. For podcast feeds, hpodder disabled the podcast; this status will appear in **hpodder lscasts**. For episodes, hpodder sets the status to Error; this will appear in **hpodder lseps**. Both can be changed later, with **hpodder enable** or **hpodder setstatus**, respectively.

The default minimums of 15 attempts and 21 days may be adjusted in the hpodder configuration file, either globally or on a per-podcast basis.

If you wish to disable checking entirely, you can put lines such as `epfaildays = 123456789` and `podcastfaildays = 123456789` in your `DEFAULT` section in `~/.hpodder/hpodder.conf`. Of course, if you have podcasts that still fail after 338,237 years, you could be in trouble.

hpodder Configuration File

hpodder has a configuration file in which you can set various options. This file normally lives under `~/.hpodder/hpodder.conf`.

The configuration file has multiple sections. Each section has a name and is introduced with the name in brackets. Each section has one or more options.

The section named DEFAULT is special in that it provides defaults that will be used whenever an option can't be found under a different section.

Let's start by looking at an example file, and then proceed to examine all the options that are available.

```
[DEFAULT]

; Most podcasts are downloaded to here
downloaddir = /home/jgoerzen/podcasts

namingpatt = %(safecasttitle)s/%(safefilename)s

; Don't disable a podcast due to errors unless it's been at least 20
; days since the last (or first) attempt
podcastfaildays = 20

[general]

; The following line tells hpodder that
; you have already gone through the intro.
showintro = no

maxthreads = 2
progressinterval = 1

[31]
; Store this particular podcast somewhere else
downloaddir = /nfs/remote/podcasts

; And we don't care as much about disabling it
podcastfaildays = 5
```

In this example, you saw some "general" options, such as `showintro`. There are two other sections represented: `31` and `DEFAULT`.

Whenever hpodder looks for information about a particular podcast, it first checks to see if it can find that option in a section for that podcast. If not, it checks the `DEFAULT` section. If it still doesn't find an answer, it consults its built-in defaults.

In this example, all podcasts share the same naming scheme. All podcasts except podcast 31 are downloaded to the same place. That podcast goes elsewhere because its `downloaddir` overrides the default.

General Options

These are specified in the `general` section.

maxthreads

> The maximum number of simultaneous download threads that will be active at any given time. hpodder can download multiple files at once, and this options says how many it can download simultaneously. It defaults to 2.

progressinterval

> How frequently to update the status bar on the screen, in seconds. It defaults to 1, which will update the status every second. Raise it if you are running hpodder over a very low-bandwidth link and are concerned about flooding it with status updates.

showintro

> The first time you run **fetch**, hpodder automatically writes a configuration file for you that sets this option to `no`. This prevents you from having to do the new user intro more than once.

Per-Podcast Options

These options may be specified in `DEFAULT` or in a per-podcast section. If placed in `DEFAULT`, they will apply to all podcasts unless overridden.

Basic Per-Podcast Options

downloaddir

> The main directory into which all podcasts should be stored. It will be created by hpodder when necessary if it does not already exist. The default is `~/podcasts`

epfailattempts

> The minimum number of attempts to download this episode before the episode will be considered to be marked Error. Default is 15.

epfaildays

> The minimum number of days that must have elapsed between the first attempt to download the episode and the present time before the episode will be considered to be marked Error. Default is 21.

namingpatt

How to name downloaded files. This pattern is relative to the `downloaddir`. The default is `%(safecasttitle)s/%(safefilename)s`

This option will be provided with several replaceable tokens. Tokens have the form `%(`*`tokname`*`)s`. That is, the percent sign, the token name in perenthesis, and then an "s" character. The tokens made available for this option are:

castid

The numeric ID for this podcast

epid

The numeric ID for this episode

safecasttitle

The title of the podcast, as specified in the feed. Special characters, such as spaces or exclamation marks, are converted to underscores.

safeeptitle

The title of this episode, as specified in the podcast's feed, with special characters converted to underscores.

safefilename

The component from the URL for this episode after the last slash in the URL, with special characters converted to underscores.

podcastfailattempts

The minimum number of attempts to download this podcast before the episode will be considered to be marked disabled. Default is 15.

podcastfaildays

The minimum number of days that must have elapsed between the last successful download of the podcast's feed and the present time before the podcast will be considered to be marked disabled. Default is 21.

Per-Podcast Command Options

These are external commands that will be run in certain situations. For each of the commands, several environment variables are set. These variables are not pre-sanitized and may contain whitespace or special characters. *Extreme caution must be exercized to properly quote these variables when using them in shell commands or scripts.* The following environment variables are set:

CASTID

The numeric ID for this podcast

CASTTITLE

The title of the podcast, verbatim

EPFILENAME

The on-disk filename where this episode has been stored

EPID

The numeric epidose ID for this episode

EPTITLE

The title of this episode, as specified in the podcast's feed.

EPURL

The URL of this episode.

FEEDURL

The URL of the podcast's feed.

SAFECASTTITLE

The title of the podcast, as specified in the feed. Special characters, such as spaces or exclamation marks, are converted to underscores.

SAFEEPTITLE

The title of this episode, as specified in the podcast's feed, with special characters converted to underscores.

Here are the supported commands:

gettypecommand

This command is intended to analyze the content of the file and return the true MIME type of the file, based on the on-disk content. If this command exits with an error, the MIME type given in the podcast feed will be used. If you want to always use the MIME type in the podcast feed, you can set this to `/bin/false` or the empty string.

The default value is: `file -b -i "${EPFILENAME}"`

It is expected that this program will write its result to standard output. The first token of the output is taken to be the MIME type. The remainder will be discarded. For instance, for the output `text/x-pascal; charset=us-ascii`, the type will be taken to be `text/x-pascal`. If the program exits with a nonzero exit code, its output will not be used.

postproccommand

This command provides a user-configurable post-processing hook for downloaded podcasts. It is only invoked on files whose type matches the `postproctypes` list. This command is the very last step in the downloading process.

The default value adds ID3 tags to MP3 files. It is: `mid3v2 -T "${EPID}" -A "${CASTTITLE}" -t "${EPTITLE}" --WOAF "${EPURL}" --WOAS "${FEEDURL}" "${EPFILENAME}"`

Per-Podcast Type Processing Lists

These options govern what types of files are processed in different ways. The types used here are MIME types. They will be the actual type determined by `gettypecommand`, or if that command is unable to determine a useful type, the MIME type given by the podcast's RSS feed. Items in these lists are to be separated by commas.

postproctypes

This is the comma-separated list of MIME types on which `postproccommand` will operate. The special single token `ALL` means to operate on all types. To disable post-processing entirely, you can set this to the empty string. The default is: `audio/mpeg, audio/mp3, x-audio/mp3`

renametypes

This option governs the automatic renaming of downloaded files. Some servers do not present files with proper extensions to match their file type. This can confuse various software and devices. hpodder can automatically fix up extensions on such files. Each entry in the list in a MIME type, a colon, and the desired filename suffix. Note that no whitespace is allowed around the colon.

The default is: `audio/mpeg:.mp3, audio/mp3:.mp3, x-audio/mp3:.mp3`

Curl Configuration File

Internally, hpodder uses the Curl application to perform downloads across the Internet. Curl is a remarkably flexible application, and hpodder takes advantage of that to provide you with quite a few options.

You can customize Curl as much as you like by creating a Curl configuration file in `~/.hpodder/curlrc`. Please see curl(1) for more details on the content of that file.

Some things you can do with this file include restricting the maximum download rate, suppressing or adjusting the progress meter, configuring proxies, etc.

Tips & Hints

Here are a few tips and hints to make hpodder more pleasant for you.

Going Through a Proxy

If your connections must go through a proxy, you have two options: set an environment varilable or configure the proxy in your `~/.hpodder/curlrc`. If you use an environment variable, your settings will also impact other applications -- and that's probably what you want. See the Environment section later for tips on doing that.

Limiting Your Download Speed

Sometimes, you may not want hpodder to use all of your available bandwidth. Perhaps you don't want it to slow down other activities too much. To do this, just create a `~/.hpodder/curlrc` file. Put in it something like this:

```
limit-rate=20k
```

This will limit the download rate to 20 KB/sec.

This rate limitation is imperfect and may not do well during **update**, but it should do exactly what you want during **download**.

Environment

hpodder does not read any environment variables directly. However, it does pass on the environment to the programs it calls, such as Curl. This can be useful for specifying proxies. Please see curl(1) for more details. As specified in the Per-Podcast Command Options section, hpodder will also set certain variables for post-processing of downloaded files.

Conforming To

- The Extensible Markup Language (XML)[1] standard (W3C)
- RSS 2.0[2] (Harvard Law)
- HTTP 1.1, FTP, plus SSL/TLS and any other protocols supported by Curl
- ID3 v1 and v2

Copyright

1. http://www.w3.org/XML/
2. http://blogs.law.harvard.edu/tech/rss

Author

hpodder, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

See Also

curl(1), mid3v2(1)

The hpodder homepage at http://software.complete.org/hpodder, a general description of podcasting at http://en.wikipedia.org/wiki/Podcast

Index