Replacing YouTube & Invidious

Amolith

2020-08-03T05:43:37-04:00

Contents

Preventing duplicates 1
Selecting quality 2
Embedding subtitles 2
Limiting downloads 2
Naming format 3
Getting notifications 3
Writing the script 3
Finished script 5
Automation 6
Edits 6

Omar Roth, the developer of Invidious, recently wrote a blog post about *Stepping away from open source.* While I never used the official instance, I thought this was a good opportunity to create a tool that downloads videos from YouTubers I'm subscribed to so I can watch them offline in whatever manner I prefer.

To that end, youtube-dl is by far the most reliable and versatile option. Having been around since before 2008[1], I don't think the project is going anywhere. MPV is my media player of choice and it relies on youtube-dl for watching online content from Twitch to YouTube to PornHub to much more.

Conveniently, youtube-dl comes with all of the tools and flags needed for exactly this purpose so scripting and automating it is incredibly easy. Taking a look at `man youtube-dl` reveals a *plethora* of options but only a few are necessary.

Preventing duplicates

The main thing when downloading an entire channel is ensuring the same video isn't downloaded more than once. `--download-archive` will save the IDs of all the videos downloaded to prevent them from being downloaded again. All that's required is a file path at which to store the list.

[1] The first commit for the current youtube-dl repository was made on 21 July 2008 and it references "the new youtube-dl", which suggests versions prior.

```
--download-archive .archives/<channel>.txt
```

Selecting quality

By default, youtube-dl downloads the *single* file with the best quality audio and video so it doesn't need to mux[2] them together after. However, I prefer to have the highest possibly quality and don't mind waiting a few seconds longer for FFmpeg to combine audio and video files. `--format` lets the user decide which format they prefer and whether they want to focus on storage efficiency or quality. The basic options are `best`, `worst`, `bestvideo`, and `bestaudio`. These will download a *single* file that is the highest or lowest quality of both video/audio or video-only/audio-only. There are a lot of other options for more fine-grained control over what to look for but, as I mentioned, I want the highest video and the highest quality audio so I use `bestvideo+bestaudio`. After downloading both of those files, they will be muxed together.

```
--format bestvideo+bestaudio
```

Embedding subtitles

I enjoy subtitles but I know many people don't so this section can certainly be ignored.

There are a few options for fetching and embedding subtitles. To get "real" subtitles that someone transcribed manually, use `--write-sub`. For YouTube's auto-generated subtitles that may or may not be horribly inaccurate, use `--write-auto-sub`. For selecting the language, `--sub-lang <language-code>`, for embedding them, `--embed-subs`, and for selecting the format, `--sub-format <desired-format>`. I like using SRT files so I have that first with `best` beside it like so: `--sub-format srt/best`. SRT is preferred but, if unavailable, whatever other highest-quality format will be used and embedded instead.

```
--write-sub --write-auto-sub --sub-format srt/best --sub-lang en
    --embed-subs
```

Limiting downloads

When switching to this method, the initial download will pull *all* of a channel's videos. I certainly don't want this so they should be limited in some way. `--dateafter` and `--playlist-end` serve very nicely. The former will only download videos published *after* a certain date and the latter will only download X number of videos.

```
--dateafter 20200801 --playlist-end 5
```

[2] In the digital media world, muxing is process of combining two or more files into one. This might be a video track, multiple audio tracks, and/or subtitle tracks.

EDIT: A reader sent me an email with this improvement to the archive functionality. Rather than checking the last five days of videos to see if they've already been downloaded, this snippet will check when the archive file was last edited and use that as the `--dateafter` parameter, making the script a bit more efficient.

```
AFTER=$(date -r .archives/"$1".txt +%Y%m%d 2>/dev/null || date
    +%Y%m%d)
--dateafter $AFTER --playlist-end 5
```

Naming format

The final parameter to look at is how to name the files once they're downloaded. `--output` provides templating functionality and there are a *lot* of options. For this use, an acceptable template might be something like `Channel/Title.ext`. In youtube-dl's templating format, that's `"%(uploader)s/%(title)s.%(ext)s"`.

```
--output "%(uploader)s/%(title)s.%(ext)s"
```

Getting notifications

I don't yet have a good method for getting notifications when there are *new* videos but there is a simple way to get notified when the script is finished running. `notify-send` is one of the easiest and has pretty simple syntax as well: the first string is the notification summary and the second is a longer description. You can optionally pass an icon name to make it look a little better.

```
notify-send -i video-x-generic "Downloads finished" "Check the
    YouTube folder for new videos"
```

For some reason, I don't get icons when the generic name is specified but I know the command will work on most systems. On mine, I have to pass the path to the icon file I want: `-i /usr/share/icons/Suru++-Dark/apps/64/video.svg`

Writing the script

I want to store the videos in `~/Videos/YouTube` and I want the archive records stored in `.archives` so the first line (after the shebang[3]) creates those directories if they don't already exist and the second enters the `YouTube` folder.

```
mkdir -p "$HOME/Videos/YouTube/.archives"
cd "$HOME/Videos/YouTube"
```

[3] A *shebang* is a sequence of characters that tell your program loader what interpreter to use. `#!/bin/bash` is what you would use if it's a bash script. To use a Python interpreter, `#!/usr/bin/env python` will use the program search path to find the appropriate executable.

From here, a way to reuse the youtube-dl command is necessary so parameters can be changed in one place and they'll apply to all channels. Functions are intended for exactly this purpose and are formatted like so:

```
functionName () {
    # Code here
}
```

I've named the function `dl` so mine looks like this:

```
dl () {
    AFTER=$(date -r .archives/"$1".txt +%Y%m%d 2>/dev/null ||   \
        date +%Y%m%d)

    youtube-dl --download-archive .archives/"$1".txt -f         \
        bestvideo+bestaudio --dateafter 20200801 --write-sub    \
        --write-auto-sub --sub-format srt/best --sub-lang en    \
        --embed-subs -o "%(uploader)s/%(title)s.%(ext)s"        \
        --playlist-end 5 "$2"
    sleep 5
}
```

Because it's in a function that's called repeatedly, the `AFTER` variable will be reevaluated each time using a different archive file to ensure no videos are missed. The backslashes at the end (\) tell bash that it's a single command spanning multiple lines. At the bottom, `sleep` just waits 5 seconds before downloading the next channel. It's unlikely that YouTube will ratelimit a residential address for this but it is still possible. Waiting a bit before continuing reduces the likelihood further.

Note the use of `"$1"` and `"$2"` in the archive path and at the very end of the youtube-dl command. This lets the user define what the archive file should be named and what channel to download videos from. A line using the function would be something like:

```
dl linustechtips https://www.youtube.com/user/LinusTechTips
```

The result would be this directory structure:

```
YouTube
|-- .archives
|   \-- linustechtips.txt
\-- Linus Tech Tips
    \-- They still make MP3 players.mkv
```

The last line is the notification command:

```
notify-send -i video-x-generic "Downloads finished" "Check the
    YouTube folder for new videos"
```

Finished script

This the script I have in use right now.

```
#!/bin/bash

mkdir -p "$HOME/Videos/YouTube/.archives"
cd "$HOME/Videos/YouTube"


dl () {
    AFTER=$(date -r .archives/"$1".txt +%Y%m%d 2>/dev/null ||   \
        date +%Y%m%d)

    youtube-dl --download-archive .archives/"$1".txt -f         \
        bestvideo+bestaudio --dateafter $AFTER --write-sub      \
        --write-auto-sub --sub-format srt/best --sub-lang en    \
        --embed-subs -o "%(uploader)s/%(title)s.%(ext)s"        \
        --playlist-end 5 "$2"
    sleep 5
}

dl vsauce               https://www.youtube.com/user/Vsauce
dl pewdiepie            https://www.youtube.com/user/PewDiePie
dl techaltar
    https://www.youtube.com/channel/UCtZO3K2p8mqFwiKWb9k7fXA
dl avikaplan            https://www.youtube.com/user/AviKaplanMusic
dl lukesmith
    https://www.youtube.com/channel/UC2eYFnH61tmytImy1mTYvhA
dl techlinked           https://www.youtube.com/c/techlinked/
dl robscallon           https://www.youtube.com/user/robs70986987
dl logosbynick
    https://www.youtube.com/channel/UCEQXp_fcqwPcqrzNtWJ1w9w
dl techquickie          https://www.youtube.com/user/Techquickie
dl andrewhuang
    https://www.youtube.com/user/songstowearpantsto
dl jamesveitch          https://www.youtube.com/user/james948
dl brandonacker         https://www.youtube.com/user/brandonacker
dl linustechtips        https://www.youtube.com/user/LinusTechTips
dl roomieofficial       https://www.youtube.com/user/RoomieOfficial
dl fridaycheckout
    https://www.youtube.com/channel/UCRG_N2uO405WO4P3Ruef9NA
dl lastweektonight      https://www.youtube.com/user/LastWeekTonight
dl bingingwithbabish    https://www.youtube.com/user/bgfilms

notify-send --icon /usr/share/icons/Suru++-Dark/apps/64/video.svg
    "Downloads finished" "Check the YouTube folder for new videos"
```

Automation

This is a very simple process. 1. Store your script wherever you want but take note of the directory. 2. Run `crontab -e` * If you don't already have a cron utility installed, try `cronie`. It should be in most repos. 4. Paste and edit: `0 */6 * * * /home/user/path/to/script.sh` 5. Save 6. Exit 7. ??? 8. Profit

The pasted line runs the script every 6th hour of every day, every week, every month, and every year. To change the frequency just run `crontab -e`, edit the line, and save. Crontab Generator or Crontab Guru might be useful if the syntax is confusing.

Have fun!

Edits

1. Synchronised `--dateafter` parameter with last modified timestamp of the archive file — thank you Dominic!