yt-dlp 2023.9.24

Последняя версия

```
pip install yt-dlp
```

Выпущен: about 18 hours ago

A youtube-dl fork with additional features and patches

Навигация

Описание проекта

История выпусков

Загрузка файлов

Ссылки проекта

Homepage

Documentation

Funding

Source

Tracker

Статистика

Статистика GitHub:

Звёзд: 56506

Форков: 4618

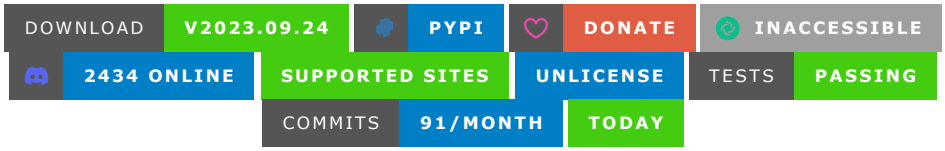
Open issues: 1121

Описание проекта

Official repository: https://github.com/yt-dlp/yt-dlp

PS: Some links in this document will not work since this is a copy of the README.md from Github

YT-DLP *A youtube-dl fork with additional features and fixes*



yt-dlp is a youtube-dl fork based on the now inactive youtube-dlc. The main focus of this project is adding new features and patches while also keeping up to date with the original project

Метаданные

Лицензия: Public Domain

Сопровождающий: pukkandan

Требует: Python >=3.7

Сопровождающие

pukkandan

Классификаторы

Development Status
- 5 - Production/Stable

Environment
- Console

License
- Public Domain

Operating System
- OS Independent

Programming Language
- Python
- Python :: 3.7
- Python :: 3.8
- Python :: 3.9
- Python :: 3.10
- Python :: 3.11
- Python :: Implementation
- Python :: Implementation :: CPython
- Python :: Implementation :: PyPy

Topic

NEW FEATURES

- Multimedia :: Video

- Forked from **yt-dlc@f9401f2** and merged with **youtube-dl@66ab08** (exceptions)
- **SponsorBlock Integration**: You can mark/remove sponsor sections in YouTube videos by utilizing the SponsorBlock API
- **Format Sorting**: The default format sorting options have been changed so that higher resolution and better codecs will be now preferred instead of simply using larger bitrate. Furthermore, you can now specify the sort order using `-S`. This allows for much easier format selection than what is possible by simply using `--format` (examples)
- **Merged with animelover1984/youtube-dl**: You get most of the features and improvements from animelover1984/youtube-dl including `--write-comments`, `BiliBiliSearch`, `BilibiliChannel`, Embedding thumbnail in mp4/ogg/opus, playlist infojson etc. Note that NicoNico livestreams are not available. See #31 for details.
- **YouTube improvements**:
  - Supports Clips, Stories (`ytstories:<channel UCID>`), Search (including filters)*, YouTube Music Search, Channel-specific search, Search prefixes (`ytsearch:`, `ytsearchdate:`)*, Mixes, and Feeds (`:ytfav`, `:ytwatchlater`, `:ytsubs`, `:ythistory`, `:ytrec`, `:ytnotif`)
  - Fix for n-sig based throttling *
  - Supports some (but not all) age-gated content without cookies
  - Download livestreams from the start using `--live-from-start` (*experimental*)
  - `255kbps` audio is extracted (if available) from YouTube Music when premium cookies are given
  - Channel URLs download all uploads of the channel, including shorts and live
- **Cookies from browser**: Cookies can be automatically extracted from all major web browsers using `--cookies-from-browser BROWSER[+KEYRING][:PROFILE][::CONTAINER]`
- **Download time range**: Videos can be downloaded partially based on either timestamps or chapters using `--download-sections`
- **Split video by chapters**: Videos can be split into multiple files based on chapters using `--split-chapters`
- **Multi-threaded fragment downloads**: Download multiple fragments of m3u8/mpd videos in parallel. Use `--concurrent-fragments` (`-N`) option to set the number of threads used
- **Aria2c with HLS/DASH**: You can use `aria2c` as the external downloader for DASH(mpd) and HLS(m3u8) formats
- **New and fixed extractors**: Many new extractors have been added and a lot of existing ones have been fixed. See the changelog or the list of supported sites
- **New MSOs**: Philo, Spectrum, SlingTV, Cablevision, RCN etc.
- **Subtitle extraction from manifests**: Subtitles can be extracted from streaming media manifests. See commit/be6202f for details
- **Multiple paths and output templates**: You can give different output templates and download paths for different types of files. You can also set a temporary path where intermediary files are downloaded to using `--paths` (`-P`)

- **Portable Configuration**: Configuration files are automatically loaded from the home and root directories. See CONFIGURATION for details
- **Output template improvements**: Output templates can now have date-time formatting, numeric offsets, object traversal etc. See output template for details. Even more advanced operations can also be done with the help of `--parse-metadata` and `--replace-in-metadata`
- **Other new options**: Many new options have been added such as `--alias`, `--print`, `--concat-playlist`, `--wait-for-video`, `--retry-sleep`, `--sleep-requests`, `--convert-thumbnails`, `--force-download-archive`, `--force-overwrites`, `--break-match-filter` etc
- **Improvements**: Regex and other operators in `--format`/`--match-filter`, multiple `--postprocessor-args` and `--downloader-args`, faster archive checking, more format selection options, merge multi-video/audio, multiple `--config-locations`, `--exec` at different stages, etc
- **Plugins**: Extractors and PostProcessors can be loaded from an external file. See plugins for details
- **Self updater**: The releases can be updated using `yt-dlp -U`, and downgraded using `--update-to` if required
- **Nightly builds**: Automated nightly builds can be used with `--update-to nightly`

See changelog or commits for the full list of changes

Features marked with a * have been back-ported to youtube-dl

Differences in default behavior

Some of yt-dlp's default options are different from that of youtube-dl and youtube-dlc:

- yt-dlp supports only Python 3.7+, and *may* remove support for more versions as they become EOL; while youtube-dl still supports Python 2.6+ and 3.2+
- The options `--auto-number` (`-A`), `--title` (`-t`) and `--literal` (`-l`), no longer work. See removed options for details
- `avconv` is not supported as an alternative to `ffmpeg`
- yt-dlp stores config files in slightly different locations to youtube-dl. See CONFIGURATION for a list of correct locations
- The default output template is `%(title)s [%(id)s].%(ext)s`. There is no real reason for this change. This was changed before yt-dlp was ever made public and now there are no plans to change it back to `%(title)s-%(id)s.%(ext)s`. Instead, you may use `--compat-options filename`
- The default format sorting is different from youtube-dl and prefers higher resolution and better codecs rather than higher bitrates. You can use the `--format-sort` option to change this to any order you prefer, or use `--compat-options format-sort` to use youtube-dl's sorting order
- The default format selector is `bv*+ba/b`. This means that if a combined video + audio format that is better than the best video-only format is found, the former will be preferred. Use `-f bv+ba/b` or `--compat-options format-spec` to revert this

- Unlike youtube-dlc, yt-dlp does not allow merging multiple audio/video streams into one file by default (since this conflicts with the use of `-f bv*+ba`). If needed, this feature must be enabled using `--audio-multistreams` and `--video-multistreams`. You can also use `--compat-options multistreams` to enable both
- `--no-abort-on-error` is enabled by default. Use `--abort-on-error` or `--compat-options abort-on-error` to abort on errors instead
- When writing metadata files such as thumbnails, description or infojson, the same information (if available) is also written for playlists. Use `--no-write-playlist-metafiles` or `--compat-options no-playlist-metafiles` to not write these files
- `--add-metadata` attaches the `infojson` to `mkv` files in addition to writing the metadata when used with `--write-info-json`. Use `--no-embed-info-json` or `--compat-options no-attach-info-json` to revert this
- Some metadata are embedded into different fields when using `--add-metadata` as compared to youtube-dl. Most notably, `comment` field contains the `webpage_url` and `synopsis` contains the `description`. You can use `--parse-metadata` to modify this to your liking or use `--compat-options embed-metadata` to revert this
- `playlist_index` behaves differently when used with options like `--playlist-reverse` and `--playlist-items`. See #302 for details. You can use `--compat-options playlist-index` if you want to keep the earlier behavior
- The output of `-F` is listed in a new format. Use `--compat-options list-formats` to revert this
- Live chats (if available) are considered as subtitles. Use `--sub-langs all,-live_chat` to download all subtitles except live chat. You can also use `--compat-options no-live-chat` to prevent any live chat/danmaku from downloading
- YouTube channel URLs download all uploads of the channel. To download only the videos in a specific tab, pass the tab's URL. If the channel does not show the requested tab, an error will be raised. Also, `/live` URLs raise an error if there are no live videos instead of silently downloading the entire channel. You may use `--compat-options no-youtube-channel-redirect` to revert all these redirections
- Unavailable videos are also listed for YouTube playlists. Use `--compat-options no-youtube-unavailable-videos` to remove this
- The upload dates extracted from YouTube are in UTC when available. Use `--compat-options no-youtube-prefer-utc-upload-date` to prefer the non-UTC upload date.
- If `ffmpeg` is used as the downloader, the downloading and merging of formats happen in a single step when possible. Use `--compat-options no-direct-merge` to revert this
- Thumbnail embedding in `mp4` is done with mutagen if possible. Use `--compat-options embed-thumbnail-atomicparsley` to force the use of AtomicParsley instead
- Some internal metadata such as filenames are removed by default from the infojson. Use `--no-clean-infojson` or `--compat-options no-clean-infojson` to revert this

- When `--embed-subs` and `--write-subs` are used together, the subtitles are written to disk and also embedded in the media file. You can use just `--embed-subs` to embed the subs and automatically delete the separate file. See #630 (comment) for more info. `--compat-options no-keep-subs` can be used to revert this
- `certifi` will be used for SSL root certificates, if installed. If you want to use system certificates (e.g. self-signed), use `--compat-options no-certifi`
- yt-dlp's sanitization of invalid characters in filenames is different/smarter than in youtube-dl. You can use `--compat-options filename-sanitization` to revert to youtube-dl's behavior
- yt-dlp tries to parse the external downloader outputs into the standard progress output if possible (Currently implemented: ~~aria2c~~). You can use `--compat-options no-external-downloader-progress` to get the downloader output as-is
- yt-dlp versions between 2021.09.01 and 2023.01.02 applies `--match-filter` to nested playlists. This was an unintentional side-effect of 8f18ac and is fixed in d7b460. Use `--compat-options playlist-match-filter` to revert this

For ease of use, a few more compat options are available:

- `--compat-options all`: Use all compat options (Do NOT use)
- `--compat-options youtube-dl`: Same as `--compat-options all,-multistreams,-playlist-match-filter`
- `--compat-options youtube-dlc`: Same as `--compat-options all,-no-live-chat,-no-youtube-channel-redirect,-playlist-match-filter`
- `--compat-options 2021`: Same as `--compat-options 2022,no-certifi,filename-sanitization,no-youtube-prefer-utc-upload-date`
- `--compat-options 2022`: Same as `--compat-options playlist-match-filter,no-external-downloader-progress`. Use this to enable all future compat options

INSTALLATION

You can install yt-dlp using the binaries, pip or one using a third-party package manager. See the wiki for detailed instructions

UPDATE

You can use `yt-dlp -U` to update if you are using the release binaries

If you installed with pip, simply re-run the same command that was used to install the program

For other third-party package managers, see the wiki or refer their documentation

There are currently two release channels for binaries, `stable` and `nightly`. `stable` is the default channel, and many of its changes have been tested by users of the nightly channel. The `nightly` channel has releases built after each push to the master branch, and will have the most recent fixes and additions, but also have more risk of regressions. They are available in their own repo.

When using `--update`/`-U`, a release binary will only update to its current channel. `--update-to CHANNEL` can be used to switch to a different channel when a newer version is available. `--update-to [CHANNEL@]TAG` can also be used to upgrade or downgrade to specific tags from a channel.

You may also use `--update-to <repository>` (`<owner>/<repository>`) to update to a channel on a completely different repository. Be careful with what repository you are updating to though, there is no verification done for binaries from different repositories.

Example usage:

- `yt-dlp --update-to nightly` change to `nightly` channel and update to its latest release
- `yt-dlp --update-to stable@2023.02.17` upgrade/downgrade to release to `stable` channel tag `2023.02.17`
- `yt-dlp --update-to 2023.01.06` upgrade/downgrade to tag `2023.01.06` if it exists on the current channel
- `yt-dlp --update-to example/yt-dlp@2023.03.01` upgrade/downgrade to the release from the `example/yt-dlp` repository, tag `2023.03.01`

RELEASE FILES

Recommended

File	Description
yt-dlp	Platform-independent zipimport binary. Needs Python (recommended for **Linux/BSD**)
yt-dlp.exe	Windows (Win7 SP1+) standalone x64 binary (recommended for **Windows**)
yt-dlp_macos	Universal MacOS (10.15+) standalone executable (recommended for **MacOS**)

Alternatives

File	Description
yt-dlp_x86.exe	Windows (Vista SP2+) standalone x86 (32-bit) binary

File	Description
yt-dlp_min.exe	Windows (Win7 SP1+) standalone x64 binary built with `py2exe` (Not recommended)
yt-dlp_linux	Linux standalone x64 binary
yt-dlp_linux.zip	Unpackaged Linux executable (no auto-update)
yt-dlp_linux_armv7l	Linux standalone armv7l (32-bit) binary
yt-dlp_linux_aarch64	Linux standalone aarch64 (64-bit) binary
yt-dlp_win.zip	Unpackaged Windows executable (no auto-update)
yt-dlp_macos.zip	Unpackaged MacOS (10.15+) executable (no auto-update)
yt-dlp_macos_legacy	MacOS (10.9+) standalone x64 executable

Misc

File	Description
yt-dlp.tar.gz	Source tarball
SHA2-512SUMS	GNU-style SHA512 sums
SHA2-512SUMS.sig	GPG signature file for SHA512 sums
SHA2-256SUMS	GNU-style SHA256 sums
SHA2-256SUMS.sig	GPG signature file for SHA256 sums

The public key that can be used to verify the GPG signatures is available here
Example usage:

```
curl -L https://github.com/yt-dlp/yt-dlp/raw/master/public.key | g
gpg --verify SHA2-256SUMS.sig SHA2-256SUMS
gpg --verify SHA2-512SUMS.sig SHA2-512SUMS
```

Note: The manpages, shell completion (autocomplete) files etc. are available inside the source tarball

DEPENDENCIES

Python versions 3.7+ (CPython and PyPy) are supported. Other versions and implementations may or may not work correctly.

While all the other dependencies are optional, `ffmpeg` and `ffprobe` are highly recommended

Strongly recommended

- **ffmpeg** and **ffprobe** - Required for merging separate video and audio files as well as for various post-processing tasks. License depends on the build

  There are bugs in ffmpeg that causes various issues when used alongside yt-dlp. Since ffmpeg is such an important dependency, we provide custom builds with patches for some of these issues at yt-dlp/FFmpeg-Builds. See the readme for details on the specific issues solved by these builds

  Important: What you need is ffmpeg *binary*, **NOT** the python package of the same name

Networking

- **certifi*** - Provides Mozilla's root certificate bundle. Licensed under MPLv2
- **brotli*** or **brotlicffi** - Brotli content encoding support. Both licensed under MIT [1] [2]
- **websockets*** - For downloading over websocket. Licensed under BSD-3-Clause

Metadata

- **mutagen*** - For `--embed-thumbnail` in certain formats. Licensed under GPLv2+
- **AtomicParsley** - For `--embed-thumbnail` in `mp4`/`m4a` files when `mutagen`/`ffmpeg` cannot. Licensed under GPLv2+
- **xattr**, **pyxattr** or **setfattr** - For writing xattr metadata (`--xattr`) on **Linux**. Licensed under MIT, LGPL2.1 and GPLv2+ respectively

Misc

- **pycryptodomex*** - For decrypting AES-128 HLS streams and various other data. Licensed under BSD-2-Clause
- **phantomjs** - Used in extractors where javascript needs to be run. Licensed under BSD-3-Clause
- **secretstorage** - For `--cookies-from-browser` to access the **Gnome** keyring while decrypting cookies of **Chromium**-based browsers on **Linux**. Licensed under BSD-3-Clause
- Any external downloader that you want to use with `--downloader`

Deprecated

- **avconv** and **avprobe** - Now **deprecated** alternative to ffmpeg. License depends on the build

- **sponskrub** - For using the now **deprecated** sponskrub options. Licensed under GPLv3+
- **rtmpdump** - For downloading `rtmp` streams. ffmpeg can be used instead with `--downloader ffmpeg`. Licensed under GPLv2+
- **mplayer** or **mpv** - For downloading `rstp`/`mms` streams. ffmpeg can be used instead with `--downloader ffmpeg`. Licensed under GPLv2+

To use or redistribute the dependencies, you must agree to their respective licensing terms.

The standalone release binaries are built with the Python interpreter and the packages marked with * included.

If you do not have the necessary dependencies for a task you are attempting, yt-dlp will warn you. All the currently available dependencies are visible at the top of the `--verbose` output

COMPILE

Standalone PyInstaller Builds

To build the standalone executable, you must have Python and `pyinstaller` (plus any of yt-dlp's optional dependencies if needed). Once you have all the necessary dependencies installed, simply run `pyinst.py`. The executable will be built for the same architecture (x86/ARM, 32/64 bit) as the Python used.

```
python3 -m pip install -U pyinstaller -r requirements.txt
python3 devscripts/make_lazy_extractors.py
python3 pyinst.py
```

On some systems, you may need to use `py` or `python` instead of `python3`.

`pyinst.py` accepts any arguments that can be passed to `pyinstaller`, such as `--onefile/-F` or `--onedir/-D`, which is further documented here.

Note: Pyinstaller versions below 4.4 do not support Python installed from the Windows store without using a virtual environment.

Important: Running `pyinstaller` directly **without** using `pyinst.py` is **not** officially supported. This may or may not work correctly.

Platform-independent Binary (UNIX)

You will need the build tools `python` (3.7+), `zip`, `make` (GNU), `pandoc`* and `pytest`*.

After installing these, simply run `make`.

You can also run `make yt-dlp` instead to compile only the binary without updating any of the additional files. (The build tools marked with * are not needed for this)

Standalone Py2Exe Builds (Windows)

While we provide the option to build with py2exe, it is recommended to build using PyInstaller instead since the py2exe builds **cannot contain** `pycryptodomex`/`certifi` **and needs VC++14** on the target computer to run.

If you wish to build it anyway, install Python and py2exe, and then simply run `setup.py py2exe`

```
py -m pip install -U py2exe -r requirements.txt
py devscripts/make_lazy_extractors.py
py setup.py py2exe
```

Related scripts

- `devscripts/update-version.py` - Update the version number based on current date.
- `devscripts/set-variant.py` - Set the build variant of the executable.
- `devscripts/make_changelog.py` - Create a markdown changelog using short commit messages and update `CONTRIBUTORS` file.
- `devscripts/make_lazy_extractors.py` - Create lazy extractors. Running this before building the binaries (any variant) will improve their startup performance. Set the environment variable `YTDLP_NO_LAZY_EXTRACTORS=1` if you wish to forcefully disable lazy extractor loading.

Note: See their `--help` for more info.

Forking the project

If you fork the project on GitHub, you can run your fork's build workflow to automatically build the selected version(s) as artifacts. Alternatively, you can run the release workflow or enable the nightly workflow to create full (pre-)releases.

USAGE AND OPTIONS

```
yt-dlp [OPTIONS] [--] URL [URL...]
```

`Ctrl+F` is your friend :D

General Options:

```
--mark-watched                  Mark videos watched (even with --s
--no-mark-watched               Do not mark videos watched (defaul
--color [STREAM:]POLICY         Whether to emit color codes in out
                                optionally prefixed by the STREAM
                                stderr) to apply the setting to. C
                                of "always", "auto" (default), "ne
                                "no_color" (use non color terminal
                                sequences). Can be used multiple t
--compat-options OPTS           Options that can help keep compati
                                with youtube-dl or youtube-dlc
                                configurations by reverting some o
                                changes made in yt-dlp. See "Diffe
                                default behavior" for details
--alias ALIASES OPTIONS         Create aliases for an option strin
                                an alias starts with a dash "-", i
                                prefixed with "--". Arguments are
                                according to the Python string for
                                mini-language. E.g. --alias get-au
                                "-S=aext:{0},abr -x --audio-format
                                creates options "--get-audio" and
                                takes an argument (ARG0) and expan
                                "-S=aext:ARG0,abr -x --audio-forma
                                All defined aliases are listed in
                                output. Alias options can trigger
                                aliases; so be careful to avoid de
                                recursive options. As a safety mea
                                alias may be triggered a maximum o
                                times. This option can be used mul
```

Network Options:

```
--proxy URL                     Use the specified HTTP/HTTPS/SOCKS
                                enable SOCKS proxy, specify a prop
                                e.g. socks5://user:pass@127.0.0.1:
                                Pass in an empty string (--proxy "
                                direct connection
--socket-timeout SECONDS        Time to wait before giving up, in
--source-address IP             Client-side IP address to bind to
-4, --force-ipv4                Make all connections via IPv4
-6, --force-ipv6                Make all connections via IPv6
--enable-file-urls              Enable file:// URLs. This is disab
                                default for security reasons.
```

Geo-restriction:

```
                                "default" (only when known to be u
                                "never", an IP block in CIDR notat
                                two-letter ISO 3166-2 country code
```

Video Selection:

```
--yes-playlist                  Download the playlist, if the URL
                                a video and a playlist
--age-limit YEARS               Download only videos suitable for
                                age
--download-archive FILE         Download only videos not listed in
                                archive file. Record the IDs of al
                                downloaded videos in it
--no-download-archive           Do not use archive file (default)
--max-downloads NUMBER          Abort after downloading NUMBER fil
--break-on-existing             Stop the download process when enc
                                a file that is in the archive
--break-per-input               Alters --max-downloads, --break-on
                                --break-match-filter, and autonumb
                                reset per input URL
--no-break-per-input            --break-on-existing and similar op
                                terminates the entire download que
--skip-playlist-after-errors N  Number of allowed failures until t
                                the playlist is skipped
```

Download Options:

```
--resize-buffer                  The buffer size is automatically r
                                 from an initial value of --buffer-
                                 (default)
--no-resize-buffer               Do not automatically adjust the bu
--http-chunk-size SIZE           Size of a chunk for chunk-based HT
                                 downloading, e.g. 10485760 or 10M
                                 is disabled). May be useful for by
                                 bandwidth throttling imposed by a
                                 (experimental)
--playlist-random                Download playlist videos in random
--lazy-playlist                  Process entries in the playlist as
                                 received. This disables n_entries,
                                 --playlist-random and --playlist-r
--no-lazy-playlist               Process videos in the playlist onl
                                 the entire playlist is parsed (def
--xattr-set-filesize             Set file xattribute ytdl.filesize
                                 expected file size
--hls-use-mpegts                 Use the mpegts container for HLS v
                                 allowing some players to play the
                                 while downloading, and reducing th
                                 of file corruption if download is
                                 interrupted. This is enabled by de
                                 live streams
--no-hls-use-mpegts              Do not use the mpegts container fo
                                 videos. This is default when not d
                                 live streams
--download-sections REGEX        Download only chapters that match
                                 regular expression. A "*" prefix d
                                 time-range instead of chapter. Neg
                                 timestamps are calculated from the
                                 "*from-url" can be used to downloa
                                 the "start_time" and "end_time" ex
                                 from the URL. Needs ffmpeg. This o
                                 be used multiple times to download
                                 sections, e.g. --download-sections
                                 "*10:15-inf" --download-sections "
--downloader [PROTO:]NAME        Name or path of the external downl
                                 use (optionally) prefixed by the p
                                 (http, ftp, m3u8, dash, rstp, rtmp
                                 use it for. Currently supports nat
                                 aria2c, avconv, axel, curl, ffmpeg
                                 wget. You can use this option mult
                                 to set different downloaders for d
                                 protocols. E.g. --downloader aria2
                                 --downloader "dash,m3u8:native" wi
                                 aria2c for http/ftp downloads, and
                                 native downloader for dash/m3u8 do
                                 (Alias: --external-downloader)
--downloader-args NAME:ARGS      Give these arguments to the extern
                                 downloader. Specify the downloader
                                 the arguments separated by a colon
                                 ffmpeg, arguments can be passed to
                                 positions using the same syntax as
                                 --postprocessor-args. You can use
                                 option multiple times to give diff
                                 arguments to different downloaders
                                 --external-downloader-args)
```

Filesystem Options:

```
--no-write-playlist-metafiles   Do not write playlist metadata whe
                                --write-info-json, --write-descrip
--clean-info-json               Remove some internal metadata such
                                filenames from the infojson (defau
--no-clean-info-json            Write all fields to the infojson
--write-comments                Retrieve video comments to be plac
                                infojson. The comments are fetched
                                without this option if the extract
                                known to be quick (Alias: --get-co
--no-write-comments             Do not retrieve video comments unl
                                extraction is known to be quick (A
                                --no-get-comments)
--load-info-json FILE           JSON file containing the video inf
                                (created with the "--write-info-js
--cookies FILE                  Netscape formatted file to read co
                                and dump cookie jar in
--no-cookies                    Do not read/dump cookies from/to f
                                (default)
--cookies-from-browser BROWSER[+KEYRING][:PROFILE][::CONTAINER]
                                The name of the browser to load co
                                from. Currently supported browsers
                                brave, chrome, chromium, edge, fir
                                opera, safari, vivaldi. Optionally
                                KEYRING used for decrypting Chromi
                                on Linux, the name/path of the PRO
                                load cookies from, and the CONTAIN
                                (if Firefox) ("none" for no contai
                                be given with their respective sep
                                By default, all containers of the
                                recently accessed profile are used
                                Currently supported keyrings are:
                                gnomekeyring, kwallet, kwallet5, k
--no-cookies-from-browser       Do not load cookies from browser (
--cache-dir DIR                 Location in the filesystem where y
                                store some downloaded information
                                client ids and signatures) permane
                                default ${XDG_CACHE_HOME}/yt-dlp
--no-cache-dir                  Disable filesystem caching
--rm-cache-dir                  Delete all filesystem cache files
```

Thumbnail Options:

```
--write-thumbnail               Write thumbnail image to disk
--no-write-thumbnail            Do not write thumbnail image to di
--write-all-thumbnails          Write all thumbnail image formats
--list-thumbnails               List available thumbnails of each
                                Simulate unless --no-simulate is u
```

Internet Shortcut Options:

```
                                .desktop). The URL may be cached b
    --write-url-link            Write a .url Windows internet shor
                                OS caches the URL based on the fil
    --write-webloc-link         Write a .webloc macOS internet sho
    --write-desktop-link        Write a .desktop Linux internet sh
```

Verbosity and Simulation Options:

```
                                Template for progress outputs, opt
                                prefixed with one of "download:" (
                                "download-title:" (the console tit
                                "postprocess:",  or "postprocess-t
                                The video's fields are accessible
                                "info" key and the progress attrib
                                accessible under "progress" key. E
                                --console-title --progress-templat
                                "download-title:%(info.id)s-%(prog
-v, --verbose                   Print various debugging informatio
--dump-pages                    Print downloaded pages encoded usi
                                to debug problems (very verbose)
--write-pages                   Write downloaded intermediary page
                                in the current directory to debug
--print-traffic                 Display sent and read HTTP traffic
```

Workarounds:

```
--encoding ENCODING             Force the specified encoding (expe
--legacy-server-connect         Explicitly allow HTTPS connection
                                that do not support RFC 5746 secur
                                renegotiation
--no-check-certificates         Suppress HTTPS certificate validat
--prefer-insecure               Use an unencrypted connection to r
                                information about the video (Curre
                                supported only for YouTube)
--add-headers FIELD:VALUE       Specify a custom HTTP header and i
                                separated by a colon ":". You can
                                option multiple times
--bidi-workaround               Work around terminals that lack
                                bidirectional text support. Requir
                                or fribidi executable in PATH
--sleep-requests SECONDS        Number of seconds to sleep between
                                during data extraction
--sleep-interval SECONDS        Number of seconds to sleep before
                                download. This is the minimum time
                                when used along with --max-sleep-i
                                (Alias: --min-sleep-interval)
--max-sleep-interval SECONDS    Maximum number of seconds to sleep
                                be used along with --min-sleep-int
--sleep-subtitles SECONDS       Number of seconds to sleep before
                                subtitle download
```

Video Format Options:

```
--no-format-sort-force          Some fields have precedence over t
                                specified sort order (default)
--video-multistreams            Allow multiple video streams to be
                                into a single file
--no-video-multistreams         Only one video stream is downloade
                                output file (default)
--audio-multistreams            Allow multiple audio streams to be
                                into a single file
--no-audio-multistreams         Only one audio stream is downloade
                                output file (default)
--prefer-free-formats           Prefer video formats with free con
                                over non-free ones of same quality
                                "-S ext" to strictly prefer free c
                                irrespective of quality
--no-prefer-free-formats        Don't give any special preference
                                containers (default)
--check-formats                 Make sure formats are selected onl
                                those that are actually downloadab
--check-all-formats             Check all formats for whether they
                                actually downloadable
--no-check-formats              Do not check that the formats are
                                downloadable
-F, --list-formats              List available formats of each vid
                                Simulate unless --no-simulate is u
--merge-output-format FORMAT    Containers that may be used when m
                                formats, separated by "/", e.g. "m
                                Ignored if no merge is required. (
                                supported: avi, flv, mkv, mov, mp4
```

Subtitle Options:

```
--write-subs                    Write subtitle file
--no-write-subs                 Do not write subtitle file (defaul
--write-auto-subs               Write automatically generated subt
                                (Alias: --write-automatic-subs)
--no-write-auto-subs            Do not write auto-generated subtit
                                (default) (Alias: --no-write-autom
--list-subs                     List available subtitles of each v
                                Simulate unless --no-simulate is u
--sub-format FORMAT             Subtitle format; accepts formats p
                                e.g. "srt" or "ass/srt/best"
--sub-langs LANGS               Languages of the subtitles to down
                                be regex) or "all" separated by co
                                --sub-langs "en.*,ja". You can pre
                                language code with a "-" to exclud
                                the requested languages, e.g. --su
                                all,-live_chat. Use --list-subs fo
                                of available language tags
```

Authentication Options:

```
-u, --username USERNAME          Login with this account ID
-p, --password PASSWORD          Account password. If this option i
                                 out, yt-dlp will ask interactively
-2, --twofactor TWOFACTOR        Two-factor authentication code
-n, --netrc                      Use .netrc authentication data
--netrc-location PATH            Location of .netrc authentication
                                 either the path or its containing
                                 Defaults to ~/.netrc
--netrc-cmd NETRC_CMD            Command to execute to get the cred
                                 for an extractor.
--video-password PASSWORD        Video password (vimeo, youku)
--ap-mso MSO                     Adobe Pass multiple-system operato
                                 provider) identifier, use --ap-lis
                                 a list of available MSOs
--ap-username USERNAME           Multiple-system operator account l
--ap-password PASSWORD           Multiple-system operator account p
                                 If this option is left out, yt-dlp
                                 interactively
--ap-list-mso                    List all supported multiple-system
--client-certificate CERTFILE    Path to client certificate file in
                                 format. May include the private ke
--client-certificate-key KEYFILE
                                 Path to private key file for clien
                                 certificate
--client-certificate-password PASSWORD
                                 Password for client certificate pr
                                 if encrypted. If not provided, and
                                 is encrypted, yt-dlp will ask inte
```

Post-Processing Options:

```
                                separated by a colon ":". ARGS are
                                semicolon ";" delimited list of NA
                                The "when" argument determines whe
                                postprocessor is invoked. It can b
                                "pre_process" (after video extract
                                "after_filter" (after video passes
                                "video" (after --format; before
                                --print/--output), "before_dl" (be
                                video download), "post_process" (a
                                video download; default), "after_m
                                (after moving video file to it's f
                                locations), "after_video" (after d
                                and processing all formats of a vi
                                "playlist" (at end of playlist). T
                                can be used multiple times to add
                                postprocessors
```

SponsorBlock Options:

Make chapter entries for, or remove various segments (sponsor, introductions, etc.) from downloaded YouTube videos using the SponsorBlock API

```
--sponsorblock-mark CATS        SponsorBlock categories to create
                                for, separated by commas. Availabl
                                categories are sponsor, intro, out
                                selfpromo, preview, filler, intera
                                music_offtopic, poi_highlight, cha
                                and default (=all). You can prefix
                                category with a "-" to exclude it.
                                for description of the categories.
                                --sponsorblock-mark all,-preview
                                [1] https://wiki.sponsor.ajay.app/
--sponsorblock-remove CATS      SponsorBlock categories to be remo
                                the video file, separated by comma
                                category is present in both mark a
                                remove takes precedence. The synta
                                available categories are the same
                                --sponsorblock-mark except that "d
                                refers to "all,-filler" and poi_hi
                                chapter are not available
--sponsorblock-chapter-title TEMPLATE
                                An output template for the title o
                                SponsorBlock chapters created by
                                --sponsorblock-mark. The only avai
                                fields are start_time, end_time, c
                                categories, name, category_names.
                                to "[SponsorBlock]: %(category_nam
--no-sponsorblock               Disable both --sponsorblock-mark a
                                --sponsorblock-remove
--sponsorblock-api URL          SponsorBlock API location, default
                                https://sponsor.ajay.app
```

Extractor Options:

```
--extractor-retries RETRIES     Number of retries for known extrac
                                (default is 3), or "infinite"
--allow-dynamic-mpd             Process dynamic DASH manifests (de
                                (Alias: --no-ignore-dynamic-mpd)
--ignore-dynamic-mpd            Do not process dynamic DASH manife
                                (Alias: --no-allow-dynamic-mpd)
--hls-split-discontinuity       Split HLS playlists to different f
                                discontinuities such as ad breaks
--no-hls-split-discontinuity    Do not split HLS playlists to diff
                                formats at discontinuities such as
                                (default)
--extractor-args IE_KEY:ARGS    Pass ARGS arguments to the IE_KEY
                                See "EXTRACTOR ARGUMENTS" for deta
                                can use this option multiple times
                                arguments for different extractors
```

CONFIGURATION

You can configure yt-dlp by placing any supported command line option to a configuration file. The configuration is loaded from the following locations:

1. **Main Configuration**:
   - The file given by `--config-location`
2. **Portable Configuration**: (Recommended for portable installations)
   - If using a binary, `yt-dlp.conf` in the same directory as the binary
   - If running from source-code, `yt-dlp.conf` in the parent directory of `yt_dlp`
3. **Home Configuration**:
   - `yt-dlp.conf` in the home path given by `-P`
   - If `-P` is not given, the current directory is searched
4. **User Configuration**:
   - `${XDG_CONFIG_HOME}/yt-dlp.conf`
   - `${XDG_CONFIG_HOME}/yt-dlp/config` (recommended on Linux/macOS)
   - `${XDG_CONFIG_HOME}/yt-dlp/config.txt`
   - `${APPDATA}/yt-dlp.conf`
   - `${APPDATA}/yt-dlp/config` (recommended on Windows)
   - `${APPDATA}/yt-dlp/config.txt`
   - `~/yt-dlp.conf`
   - `~/yt-dlp.conf.txt`
   - `~/.yt-dlp/config`
   - `~/.yt-dlp/config.txt`

   See also: Notes about environment variables
5. **System Configuration**:
   - `/etc/yt-dlp.conf`
   - `/etc/yt-dlp/config`
   - `/etc/yt-dlp/config.txt`

E.g. with the following configuration file yt-dlp will always extract the audio, not copy the mtime, use a proxy and save all videos under `YouTube` directory in your home directory:

```
# Lines starting with # are comments

# Always extract audio
-x

# Do not copy the mtime
--no-mtime

# Use this proxy
--proxy 127.0.0.1:3128

# Save all videos under YouTube directory in your home directory
-o ~/YouTube/%(title)s.%(ext)s
```

Note: Options in configuration file are just the same options aka switches used in regular command line calls; thus there **must be no whitespace** after `-` or `--`, e.g. `-o` or `--proxy` but not `- o` or `-- proxy`. They must also be quoted when necessary as-if it were a UNIX shell.

You can use `--ignore-config` if you want to disable all configuration files for a particular yt-dlp run. If `--ignore-config` is found inside any configuration file, no further configuration will be loaded. For example, having the option in the portable configuration file prevents loading of home, user, and system configurations. Additionally, (for backward compatibility) if `--ignore-config` is found inside the system configuration file, the user configuration is not loaded.

Configuration file encoding

The configuration files are decoded according to the UTF BOM if present, and in the encoding from system locale otherwise.

If you want your file to be decoded differently, add `# coding: ENCODING` to the beginning of the file (e.g. `# coding: shift-jis`). There must be no characters before that, even spaces or BOM.

Authentication with netrc

You may also want to configure automatic credentials storage for extractors that support authentication (by providing login and password with `--username` and `--password`) in order not to pass credentials as command line arguments on every yt-dlp execution and prevent tracking plain text passwords in the shell command history. You can achieve this using a `.netrc` file on a per-extractor basis. For that you will need to create a `.netrc` file in `--netrc-location` and restrict permissions to read/write by only you:

```
touch ${HOME}/.netrc
chmod a-rwx,u+rw ${HOME}/.netrc
```

After that you can add credentials for an extractor in the following format, where *extractor* is the name of the extractor in lowercase:

```
machine <extractor> login <username> password <password>
```

E.g.

```
machine youtube login myaccount@gmail.com password my_youtube_pass
machine twitch login my_twitch_account_name password my_twitch_pas
```

To activate authentication with the `.netrc` file you should pass `--netrc` to yt-dlp or place it in the configuration file.

The default location of the .netrc file is `~` (see below).

As an alternative to using the `.netrc` file, which has the disadvantage of keeping your passwords in a plain text file, you can configure a custom shell command to provide the credentials for an extractor. This is done by providing the `--netrc-cmd` parameter, it shall output the credentials in the netrc format and return `0` on success, other values will be treated as an error. `{}` in the command will be replaced by the name of the extractor to make it possible to select the credentials for the right extractor.

E.g. To use an encrypted `.netrc` file stored as `.authinfo.gpg`

```
yt-dlp --netrc-cmd 'gpg --decrypt ~/.authinfo.gpg' https://www.you
```

Notes about environment variables

- Environment variables are normally specified as `${VARIABLE}`/`$VARIABLE` on UNIX and `%VARIABLE%` on Windows; but is always shown as `${VARIABLE}` in this documentation
- yt-dlp also allow using UNIX-style variables on Windows for path-like options; e.g. `--output`, `--config-location`
- If unset, `${XDG_CONFIG_HOME}` defaults to `~/.config` and `${XDG_CACHE_HOME}` to `~/.cache`
- On Windows, `~` points to `${HOME}` if present; or, `${USERPROFILE}` or `${HOMEDRIVE}${HOMEPATH}` otherwise
- On Windows, `${USERPROFILE}` generally points to `C:\Users\<user name>` and `${APPDATA}` to `${USERPROFILE}\AppData\Roaming`

OUTPUT TEMPLATE

The `-o` option is used to indicate a template for the output file names while `-P` option is used to specify the path each type of file should be saved to.

tl;dr: navigate me to examples.

The simplest usage of `-o` is not to set any template arguments when downloading a single file, like in `yt-dlp -o funny_video.flv "https://some/video"` (hard-coding file extension like this is *not* recommended and could break some post-processing).

It may however also contain special sequences that will be replaced when downloading each video. The special sequences may be formatted according to Python string formatting operations, e.g. `%(NAME)s` or `%(NAME)05d`. To clarify, that is a percent symbol followed by a name in parentheses, followed by formatting operations.

The field names themselves (the part inside the parenthesis) can also have some special formatting:

1. **Object traversal**: The dictionaries and lists available in metadata can be traversed by using a dot `.` separator; e.g. `%(tags.0)s`, `%(subtitles.en.-1.ext)s`. You can do Python slicing with colon `:`; E.g. `%(id.3:7:-1)s`, `%(formats.:.format_id)s`. Curly braces `{}` can be used to build dictionaries with only specific keys; e.g. `%(formats.:.{format_id,height})#j`. An empty field name `%()s` refers to the entire infodict; e.g. `%(.{id,title})s`. Note that all the fields that become available using this method are not listed below. Use `-j` to see such fields
2. **Addition**: Addition and subtraction of numeric fields can be done using `+` and `-` respectively. E.g. `%(playlist_index+10)03d`, `%(n_entries+1-playlist_index)d`
3. **Date/time Formatting**: Date/time fields can be formatted according to strftime formatting by specifying it separated from the field name using a `>`. E.g. `%(duration>%H-%M-%S)s`, `%(upload_date>%Y-%m-%d)s`, `%(epoch-3600>%H-%M-%S)s`
4. **Alternatives**: Alternate fields can be specified separated with a `,`. E.g. `%(release_date>%Y,upload_date>%Y|Unknown)s`
5. **Replacement**: A replacement value can be specified using a `&` separator according to the `str.format` mini-language. If the field is *not* empty, this replacement value will be used instead of the actual field content. This is done after alternate fields are considered; thus the replacement is used if *any* of the alternative fields is *not* empty. E.g. `%(chapters&has chapters|no chapters)s`, `%(title&TITLE={:>20}|NO TITLE)s`
6. **Default**: A literal default value can be specified for when the field is empty using a `|` separator. This overrides `--output-na-placeholder`. E.g. `%(uploader|Unknown)s`
7. **More Conversions**: In addition to the normal format types `diouxXeEfFgGcrs`, yt-dlp additionally supports converting to `B` = **B**ytes, `j` = **j**son (flag `#` for pretty-printing, `+` for Unicode), `h` = HTML escaping, `l` = a comma separated **l**ist (flag `#` for `\n` newline-separated), `q` = a string **q**uoted for the terminal (flag `#` to split a list into different arguments), `D` = add **D**ecimal suffixes (e.g. 10M) (flag `#` to use 1024 as factor), and `S` = **S**anitize as filename (flag `#` for restricted)

8. **Unicode normalization**: The format type `U` can be used for NFC Unicode normalization. The alternate form flag (`#`) changes the normalization to NFD and the conversion flag `+` can be used for NFKC/NFKD compatibility equivalence normalization. E.g. `%(title)+.100U` is NFKC

To summarize, the general syntax for a field is:

```
%(name[.keys][addition][>strf][,alternate][&replacement][|default]
```

Additionally, you can set different output templates for the various metadata files separately from the general output template by specifying the type of file followed by the template separated by a colon `:`. The different file types supported are `subtitle`, `thumbnail`, `description`, `annotation` (deprecated), `infojson`, `link`, `pl_thumbnail`, `pl_description`, `pl_infojson`, `chapter`, `pl_video`. E.g. `-o "%(title)s.%(ext)s" -o "thumbnail:%(title)s\%(title)s.%(ext)s"` will put the thumbnails in a folder with the same name as the video. If any of the templates is empty, that type of file will not be written. E.g. `--write-thumbnail -o "thumbnail:"` will write thumbnails only for playlists and not for video.

Note: Due to post-processing (i.e. merging etc.), the actual output filename might differ. Use `--print after_move:filepath` to get the name after all post-processing is complete.

The available fields are:

- `id` (string): Video identifier
- `title` (string): Video title
- `fulltitle` (string): Video title ignoring live timestamp and generic title
- `ext` (string): Video filename extension
- `alt_title` (string): A secondary title of the video
- `description` (string): The description of the video
- `display_id` (string): An alternative identifier for the video
- `uploader` (string): Full name of the video uploader
- `license` (string): License name the video is licensed under
- `creator` (string): The creator of the video
- `timestamp` (numeric): UNIX timestamp of the moment the video became available
- `upload_date` (string): Video upload date in UTC (YYYYMMDD)
- `release_timestamp` (numeric): UNIX timestamp of the moment the video was released
- `release_date` (string): The date (YYYYMMDD) when the video was released in UTC
- `modified_timestamp` (numeric): UNIX timestamp of the moment the video was last modified
- `modified_date` (string): The date (YYYYMMDD) when the video was last modified in UTC
- `uploader_id` (string): Nickname or id of the video uploader

- `channel` (string): Full name of the channel the video is uploaded on
- `channel_id` (string): Id of the channel
- `channel_follower_count` (numeric): Number of followers of the channel
- `channel_is_verified` (boolean): Whether the channel is verified on the platform
- `location` (string): Physical location where the video was filmed
- `duration` (numeric): Length of the video in seconds
- `duration_string` (string): Length of the video (HH:mm:ss)
- `view_count` (numeric): How many users have watched the video on the platform
- `concurrent_view_count` (numeric): How many users are currently watching the video on the platform.
- `like_count` (numeric): Number of positive ratings of the video
- `dislike_count` (numeric): Number of negative ratings of the video
- `repost_count` (numeric): Number of reposts of the video
- `average_rating` (numeric): Average rating give by users, the scale used depends on the webpage
- `comment_count` (numeric): Number of comments on the video (For some extractors, comments are only downloaded at the end, and so this field cannot be used)
- `age_limit` (numeric): Age restriction for the video (years)
- `live_status` (string): One of "not_live", "is_live", "is_upcoming", "was_live", "post_live" (was live, but VOD is not yet processed)
- `is_live` (boolean): Whether this video is a live stream or a fixed-length video
- `was_live` (boolean): Whether this video was originally a live stream
- `playable_in_embed` (string): Whether this video is allowed to play in embedded players on other sites
- `availability` (string): Whether the video is "private", "premium_only", "subscriber_only", "needs_auth", "unlisted" or "public"
- `start_time` (numeric): Time in seconds where the reproduction should start, as specified in the URL
- `end_time` (numeric): Time in seconds where the reproduction should end, as specified in the URL
- `extractor` (string): Name of the extractor
- `extractor_key` (string): Key name of the extractor
- `epoch` (numeric): Unix epoch of when the information extraction was completed
- `autonumber` (numeric): Number that will be increased with each download, starting at `--autonumber-start`, padded with leading zeros to 5 digits
- `video_autonumber` (numeric): Number that will be increased with each video
- `n_entries` (numeric): Total number of extracted items in the playlist
- `playlist_id` (string): Identifier of the playlist that contains the video
- `playlist_title` (string): Name of the playlist that contains the video
- `playlist` (string): `playlist_id` or `playlist_title`
- `playlist_count` (numeric): Total number of items in the playlist. May not be known if entire playlist is not extracted

- `playlist_index` (numeric): Index of the video in the playlist padded with leading zeros according the final index
- `playlist_autonumber` (numeric): Position of the video in the playlist download queue padded with leading zeros according to the total length of the playlist
- `playlist_uploader` (string): Full name of the playlist uploader
- `playlist_uploader_id` (string): Nickname or id of the playlist uploader
- `webpage_url` (string): A URL to the video webpage which if given to yt-dlp should allow to get the same result again
- `webpage_url_basename` (string): The basename of the webpage URL
- `webpage_url_domain` (string): The domain of the webpage URL
- `original_url` (string): The URL given by the user (or same as `webpage_url` for playlist entries)

All the fields in Filtering Formats can also be used

Available for the video that belongs to some logical chapter or section:

- `chapter` (string): Name or title of the chapter the video belongs to
- `chapter_number` (numeric): Number of the chapter the video belongs to
- `chapter_id` (string): Id of the chapter the video belongs to

Available for the video that is an episode of some series or programme:

- `series` (string): Title of the series or programme the video episode belongs to
- `season` (string): Title of the season the video episode belongs to
- `season_number` (numeric): Number of the season the video episode belongs to
- `season_id` (string): Id of the season the video episode belongs to
- `episode` (string): Title of the video episode
- `episode_number` (numeric): Number of the video episode within a season
- `episode_id` (string): Id of the video episode

Available for the media that is a track or a part of a music album:

- `track` (string): Title of the track
- `track_number` (numeric): Number of the track within an album or a disc
- `track_id` (string): Id of the track
- `artist` (string): Artist(s) of the track
- `genre` (string): Genre(s) of the track
- `album` (string): Title of the album the track belongs to
- `album_type` (string): Type of the album
- `album_artist` (string): List of all artists appeared on the album
- `disc_number` (numeric): Number of the disc or other physical medium the track belongs to
- `release_year` (numeric): Year (YYYY) when the album was released

Available only when using `--download-sections` and for `chapter:` prefix when using `--split-chapters` for videos with internal chapters:

- `section_title` (string): Title of the chapter
- `section_number` (numeric): Number of the chapter within the file
- `section_start` (numeric): Start time of the chapter in seconds
- `section_end` (numeric): End time of the chapter in seconds

Available only when used in `--print`:

- `urls` (string): The URLs of all requested formats, one in each line
- `filename` (string): Name of the video file. Note that the actual filename may differ
- `formats_table` (table): The video format table as printed by `--list-formats`
- `thumbnails_table` (table): The thumbnail format table as printed by `--list-thumbnails`
- `subtitles_table` (table): The subtitle format table as printed by `--list-subs`
- `automatic_captions_table` (table): The automatic subtitle format table as printed by `--list-subs`

Available only after the video is downloaded (`post_process`/`after_move`):

- `filepath`: Actual path of downloaded video file

Available only in `--sponsorblock-chapter-title`:

- `start_time` (numeric): Start time of the chapter in seconds
- `end_time` (numeric): End time of the chapter in seconds
- `categories` (list): The SponsorBlock categories the chapter belongs to
- `category` (string): The smallest SponsorBlock category the chapter belongs to
- `category_names` (list): Friendly names of the categories
- `name` (string): Friendly name of the smallest category
- `type` (string): The SponsorBlock action type of the chapter

Each aforementioned sequence when referenced in an output template will be replaced by the actual value corresponding to the sequence name. E.g. for `-o %(title)s-%(id)s.%(ext)s` and an mp4 video with title `yt-dlp test video` and id `BaW_jenozKc`, this will result in a `yt-dlp test video-BaW_jenozKc.mp4` file created in the current directory.

Note: Some of the sequences are not guaranteed to be present since they depend on the metadata obtained by a particular extractor. Such sequences will be replaced with placeholder value provided with `--output-na-placeholder` (`NA` by default).

Tip: Look at the `-j` output to identify which fields are available for the particular URL

For numeric sequences you can use numeric related formatting; e.g. `%(view_count)05d` will result in a string with view count padded with zeros up to 5 characters, like in `00042`.

Output templates can also contain arbitrary hierarchical path, e.g. `-o "%(playlist)s/%(playlist_index)s - %(title)s.%(ext)s"` which will result in downloading each video in a directory corresponding to this path template. Any missing directory will be automatically created for you.

To use percent literals in an output template use `%%`. To output to stdout use `-o -`.

The current default template is `%(title)s [%(id)s].%(ext)s`.

In some cases, you don't want special characters such as 中, spaces, or &, such as when transferring the downloaded filename to a Windows system or the filename through an 8bit-unsafe channel. In these cases, add the `--restrict-filenames` flag to get a shorter title.

Output template examples

```
$ yt-dlp --print filename -o "test video.%(ext)s" BaW_jenozKc
test video.webm    # Literal name with correct extension

$ yt-dlp --print filename -o "%(title)s.%(ext)s" BaW_jenozKc
youtube-dl test video ''_ä↭𝕐.webm    # All kinds of weird charact

$ yt-dlp --print filename -o "%(title)s.%(ext)s" BaW_jenozKc --res
youtube-dl_test_video_.webm    # Restricted file name

# Download YouTube playlist videos in separate directory indexed b
$ yt-dlp -o "%(playlist)s/%(playlist_index)s - %(title)s.%(ext)s"

# Download YouTube playlist videos in separate directories accordi
$ yt-dlp -o "%(upload_date>%Y)s/%(title)s.%(ext)s" "https://www.yo

# Prefix playlist index with " - " separator, but only if it is av
$ yt-dlp -o "%(playlist_index&{} - |)s%(title)s.%(ext)s" BaW_jenoz

# Download all playlists of YouTube channel/user keeping each play
$ yt-dlp -o "%(uploader)s/%(playlist)s/%(playlist_index)s - %(titl

# Download Udemy course keeping each chapter in separate directory
$ yt-dlp -u user -p password -P "~/MyVideos" -o "%(playlist)s/%(ch

# Download entire series season keeping each series and each seaso
$ yt-dlp -P "C:/MyVideos" -o "%(series)s/%(season_number)s - %(sea

# Download video as "C:\MyVideos\uploader\title.ext", subtitles as
# and put all temporary files in "C:\MyVideos\tmp"
$ yt-dlp -P "C:/MyVideos" -P "temp:tmp" -P "subtitle:subs" -o "%(u

# Download video as "C:\MyVideos\uploader\title.ext" and subtitles
$ yt-dlp -P "C:/MyVideos" -o "%(uploader)s/%(title)s.%(ext)s" -o "

# Stream the video being downloaded to stdout
$ yt-dlp -o - BaW_jenozKc
```

FORMAT SELECTION

By default, yt-dlp tries to download the best available quality if you **don't** pass any options. This is generally equivalent to using `-f bestvideo*+bestaudio/best`. However, if multiple audiostreams is enabled (`--audio-multistreams`), the default format changes to `-f bestvideo+bestaudio/best`. Similarly, if ffmpeg is unavailable, or if you use yt-dlp to stream to `stdout` (`-o -`), the default becomes `-f best/bestvideo+bestaudio`.

Deprecation warning: Latest versions of yt-dlp can stream multiple formats to the stdout simultaneously using ffmpeg. So, in future versions, the default for this will be set to `-f bv*+ba/b` similar to normal downloads. If you want to preserve the `-f b/bv+ba` setting, it is recommended to explicitly specify it in the configuration options.

The general syntax for format selection is `-f FORMAT` (or `--format FORMAT`) where `FORMAT` is a *selector expression*, i.e. an expression that describes format or formats you would like to download.

tl;dr: navigate me to examples.

The simplest case is requesting a specific format; e.g. with `-f 22` you can download the format with format code equal to 22. You can get the list of available format codes for particular video using `--list-formats` or `-F`. Note that these format codes are extractor specific.

You can also use a file extension (currently `3gp`, `aac`, `flv`, `m4a`, `mp3`, `mp4`, `ogg`, `wav`, `webm` are supported) to download the best quality format of a particular file extension served as a single file, e.g. `-f webm` will download the best quality format with the `webm` extension served as a single file.

You can use `-f -` to interactively provide the format selector *for each video*

You can also use special names to select particular edge case formats:

- `all`: Select **all formats** separately
- `mergeall`: Select and **merge all formats** (Must be used with `--audio-multistreams`, `--video-multistreams` or both)
- `b*`, `best*`: Select the best quality format that **contains either** a video or an audio or both (ie; `vcodec!=none or acodec!=none`)
- `b`, `best`: Select the best quality format that **contains both** video and audio. Equivalent to `best*[vcodec!=none][acodec!=none]`
- `bv`, `bestvideo`: Select the best quality **video-only** format. Equivalent to `best*[acodec=none]`
- `bv*`, `bestvideo*`: Select the best quality format that **contains video**. It may also contain audio. Equivalent to `best*[vcodec!=none]`
- `ba`, `bestaudio`: Select the best quality **audio-only** format. Equivalent to `best*[vcodec=none]`
- `ba*`, `bestaudio*`: Select the best quality format that **contains audio**. It may also contain video. Equivalent to `best*[acodec!=none]` (Do not use!)
- `w*`, `worst*`: Select the worst quality format that contains either a video or an audio

- `w`, `worst`: Select the worst quality format that contains both video and audio. Equivalent to `worst*[vcodec!=none][acodec!=none]`
- `wv`, `worstvideo`: Select the worst quality video-only format. Equivalent to `worst*[acodec=none]`
- `wv*`, `worstvideo*`: Select the worst quality format that contains video. It may also contain audio. Equivalent to `worst*[vcodec!=none]`
- `wa`, `worstaudio`: Select the worst quality audio-only format. Equivalent to `worst*[vcodec=none]`
- `wa*`, `worstaudio*`: Select the worst quality format that contains audio. It may also contain video. Equivalent to `worst*[acodec!=none]`

For example, to download the worst quality video-only format you can use `-f worstvideo`. It is however recommended not to use `worst` and related options. When your format selector is `worst`, the format which is worst in all respects is selected. Most of the time, what you actually want is the video with the smallest filesize instead. So it is generally better to use `-S +size` or more rigorously, `-S +size,+br,+res,+fps` instead of `-f worst`. See Sorting Formats for more details.

You can select the n'th best format of a type by using `best<type>.<n>`. For example, `best.2` will select the 2nd best combined format. Similarly, `bv*.3` will select the 3rd best format that contains a video stream.

If you want to download multiple videos, and they don't have the same formats available, you can specify the order of preference using slashes. Note that formats on the left hand side are preferred; e.g. `-f 22/17/18` will download format 22 if it's available, otherwise it will download format 17 if it's available, otherwise it will download format 18 if it's available, otherwise it will complain that no suitable formats are available for download.

If you want to download several formats of the same video use a comma as a separator, e.g. `-f 22,17,18` will download all these three formats, of course if they are available. Or a more sophisticated example combined with the precedence feature: `-f 136/137/mp4/bestvideo,140/m4a/bestaudio`.

You can merge the video and audio of multiple formats into a single file using `-f <format1>+<format2>+...` (requires ffmpeg installed); e.g. `-f bestvideo+bestaudio` will download the best video-only format, the best audio-only format and mux them together with ffmpeg.

Deprecation warning: Since the *below* described behavior is complex and counter-intuitive, this will be removed and multistreams will be enabled by default in the future. A new operator will be instead added to limit formats to single audio/video

Unless `--video-multistreams` is used, all formats with a video stream except the first one are ignored. Similarly, unless `--audio-multistreams` is used, all formats with an audio stream except the first one are ignored. E.g. `-f bestvideo+best+bestaudio --video-multistreams --audio-multistreams` will download and merge all 3 given formats. The resulting file will have 2 video streams and 2 audio streams. But `-f bestvideo+best+bestaudio --no-video-multistreams` will download and merge only `bestvideo` and `bestaudio`. `best` is ignored since another format containing a video stream (`bestvideo`) has already been selected. The order of the formats is therefore important. `-f`

`best+bestaudio --no-audio-multistreams` will download only `best` while `-f bestaudio+best --no-audio-multistreams` will ignore `best` and download only `bestaudio`.

Filtering Formats

You can also filter the video formats by putting a condition in brackets, as in `-f "best[height=720]"` (or `-f "[filesize>10M]"` since filters without a selector are interpreted as `best`).

The following numeric meta fields can be used with comparisons `<`, `<=`, `>`, `>=`, `=` (equals), `!=` (not equals):

- `filesize`: The number of bytes, if known in advance
- `filesize_approx`: An estimate for the number of bytes
- `width`: Width of the video, if known
- `height`: Height of the video, if known
- `aspect_ratio`: Aspect ratio of the video, if known
- `tbr`: Average bitrate of audio and video in KBit/s
- `abr`: Average audio bitrate in KBit/s
- `vbr`: Average video bitrate in KBit/s
- `asr`: Audio sampling rate in Hertz
- `fps`: Frame rate
- `audio_channels`: The number of audio channels
- `stretched_ratio`: `width:height` of the video's pixels, if not square

Also filtering work for comparisons `=` (equals), `^=` (starts with), `$=` (ends with), `*=` (contains), `~=` (matches regex) and following string meta fields:

- `url`: Video URL
- `ext`: File extension
- `acodec`: Name of the audio codec in use
- `vcodec`: Name of the video codec in use
- `container`: Name of the container format
- `protocol`: The protocol that will be used for the actual download, lower-case (`http`, `https`, `rtsp`, `rtmp`, `rtmpe`, `mms`, `f4m`, `ism`, `http_dash_segments`, `m3u8`, or `m3u8_native`)
- `language`: Language code
- `dynamic_range`: The dynamic range of the video
- `format_id`: A short description of the format
- `format`: A human-readable description of the format
- `format_note`: Additional info about the format
- `resolution`: Textual description of width and height

Any string comparison may be prefixed with negation `!` in order to produce an opposite comparison, e.g. `!*=` (does not contain). The comparand of a string comparison needs to be quoted with either double or single quotes if it contains spaces or special characters other than `._-`.

Note: None of the aforementioned meta fields are guaranteed to be present since this solely depends on the metadata obtained by particular extractor, i.e. the metadata offered by the website. Any other field made available by the extractor can also be used for filtering.

Formats for which the value is not known are excluded unless you put a question mark (`?`) after the operator. You can combine format filters, so `-f "bv[height<=?720][tbr>500]"` selects up to 720p videos (or videos where the height is not known) with a bitrate of at least 500 KBit/s. You can also use the filters with `all` to download all formats that satisfy the filter, e.g. `-f "all[vcodec=none]"` selects all audio-only formats.

Format selectors can also be grouped using parentheses; e.g. `-f "(mp4,webm)[height<480]"` will download the best pre-merged mp4 and webm formats with a height lower than 480.

Sorting Formats

You can change the criteria for being considered the `best` by using `-S` (`--format-sort`). The general format for this is `--format-sort field1,field2...`.

The available fields are:

- `hasvid`: Gives priority to formats that have a video stream
- `hasaud`: Gives priority to formats that have an audio stream
- `ie_pref`: The format preference
- `lang`: The language preference
- `quality`: The quality of the format
- `source`: The preference of the source
- `proto`: Protocol used for download (`https`/`ftps` > `http`/`ftp` > `m3u8_native`/`m3u8` > `http_dash_segments` > `websocket_frag` > `mms`/`rtsp` > `f4f`/`f4m`)
- `vcodec`: Video Codec (`av01` > `vp9.2` > `vp9` > `h265` > `h264` > `vp8` > `h263` > `theora` > other)
- `acodec`: Audio Codec (`flac`/`alac` > `wav`/`aiff` > `opus` > `vorbis` > `aac` > `mp4a` > `mp3` > `ac4` > `eac3` > `ac3` > `dts` > other)
- `codec`: Equivalent to `vcodec,acodec`
- `vext`: Video Extension (`mp4` > `mov` > `webm` > `flv` > other). If `--prefer-free-formats` is used, `webm` is preferred.
- `aext`: Audio Extension (`m4a` > `aac` > `mp3` > `ogg` > `opus` > `webm` > other). If `--prefer-free-formats` is used, the order changes to `ogg` > `opus` > `webm` > `mp3` > `m4a` > `aac`
- `ext`: Equivalent to `vext,aext`
- `filesize`: Exact filesize, if known in advance
- `fs_approx`: Approximate filesize
- `size`: Exact filesize if available, otherwise approximate filesize
- `height`: Height of video
- `width`: Width of video

- `res`: Video resolution, calculated as the smallest dimension.
- `fps`: Framerate of video
- `hdr`: The dynamic range of the video (`DV` > `HDR12` > `HDR10+` > `HDR10` > `HLG` > `SDR`)
- `channels`: The number of audio channels
- `tbr`: Total average bitrate in KBit/s
- `vbr`: Average video bitrate in KBit/s
- `abr`: Average audio bitrate in KBit/s
- `br`: Average bitrate in KBit/s, `tbr`/`vbr`/`abr`
- `asr`: Audio sample rate in Hz

Deprecation warning: Many of these fields have (currently undocumented) aliases, that may be removed in a future version. It is recommended to use only the documented field names.

All fields, unless specified otherwise, are sorted in descending order. To reverse this, prefix the field with a `+`. E.g. `+res` prefers format with the smallest resolution. Additionally, you can suffix a preferred value for the fields, separated by a `:`. E.g. `res:720` prefers larger videos, but no larger than 720p and the smallest video if there are no videos less than 720p. For `codec` and `ext`, you can provide two preferred values, the first for video and the second for audio. E.g. `+codec:avc:m4a` (equivalent to `+vcodec:avc,+acodec:m4a`) sets the video codec preference to `h264` > `h265` > `vp9` > `vp9.2` > `av01` > `vp8` > `h263` > `theora` and audio codec preference to `mp4a` > `aac` > `vorbis` > `opus` > `mp3` > `ac3` > `dts`. You can also make the sorting prefer the nearest values to the provided by using `~` as the delimiter. E.g. `filesize~1G` prefers the format with filesize closest to 1 GiB.

The fields `hasvid` and `ie_pref` are always given highest priority in sorting, irrespective of the user-defined order. This behaviour can be changed by using `--format-sort-force`. Apart from these, the default order used is: `lang,quality,res,fps,hdr:12,vcodec:vp9.2,channels,acodec,size,br,asr,proto,ext,hasaud,source,id`. The extractors may override this default order, but they cannot override the user-provided order.

Note that the default has `vcodec:vp9.2`; i.e. `av1` is not preferred. Similarly, the default for hdr is `hdr:12`; i.e. dolby vision is not preferred. These choices are made since DV and AV1 formats are not yet fully compatible with most devices. This may be changed in the future as more devices become capable of smoothly playing back these formats.

If your format selector is `worst`, the last item is selected after sorting. This means it will select the format that is worst in all respects. Most of the time, what you actually want is the video with the smallest filesize instead. So it is generally better to use `-f best -S +size,+br,+res,+fps`.

Tip: You can use the `-v -F` to see how the formats have been sorted (worst to best).

Format Selection examples

```
# preferring better codec and then larger total bitrate for the sa
$ yt-dlp -S "+res:480,codec,br"
```

MODIFYING METADATA

The metadata obtained by the extractors can be modified by using `--parse-metadata` and `--replace-in-metadata`

`--replace-in-metadata FIELDS REGEX REPLACE` is used to replace text in any metadata field using python regular expression. Backreferences can be used in the replace string for advanced use.

The general syntax of `--parse-metadata FROM:TO` is to give the name of a field or an output template to extract data from, and the format to interpret it as, separated by a colon `:`. Either a python regular expression with named capture groups, a single field name, or a similar syntax to the output template (only `%(field)s` formatting is supported) can be used for `TO`. The option can be used multiple times to parse and modify various fields.

Note that these options preserve their relative order, allowing replacements to be made in parsed fields and viceversa. Also, any field thus created can be used in the output template and will also affect the media file's metadata added when using `--embed-metadata`.

This option also has a few special uses:

- You can download an additional URL based on the metadata of the currently downloaded video. To do this, set the field `additional_urls` to the URL that you want to download. E.g. `--parse-metadata "description:(?P<additional_urls>https?://www\.vimeo\.com/\d+)"` will download the first vimeo video found in the description
- You can use this to change the metadata that is embedded in the media file. To do this, set the value of the corresponding field with a `meta_` prefix. For example, any value you set to `meta_description` field will be added to the `description` field in the file - you can use this to set a different "description" and "synopsis". To modify the metadata of individual streams, use the `meta<n>_` prefix (e.g. `meta1_language`). Any value set to the `meta_` field will overwrite all default values.

Note: Metadata modification happens before format selection, post-extraction and other post-processing operations. Some fields may be added or changed during these steps, overriding your changes.

For reference, these are the fields yt-dlp adds by default to the file metadata:

Metadata fields	From
`title`	`track` or `title`
`date`	`upload_date`

Metadata fields	From
`description`, `synopsis`	`description`
`purl`, `comment`	`webpage_url`
`track`	`track_number`
`artist`	`artist`, `creator`, `uploader` or `uploader_id`
`genre`	`genre`
`album`	`album`
`album_artist`	`album_artist`
`disc`	`disc_number`
`show`	`series`
`season_number`	`season_number`
`episode_id`	`episode` or `episode_id`
`episode_sort`	`episode_number`
`language` of each stream	the format's `language`

Note: The file format may not support some of these fields

Modifying metadata examples

```
# Replace all spaces and "_" in title and uploader with a `-`
$ yt-dlp --replace-in-metadata "title,uploader" "[ _]" "-"
```

EXTRACTOR ARGUMENTS

Some extractors accept additional arguments which can be passed using `--extractor-args KEY:ARGS`. `ARGS` is a `;` (semicolon) separated string of `ARG=VAL1,VAL2`. E.g. `--extractor-args "youtube:player-client=android_embedded,web;include_live_dash" --extractor-args "funimation:version=uncut"`

Note: In CLI, `ARG` can use `-` instead of `_`; e.g. `youtube:player-client"` becomes `youtube:player_client"`

The following extractors use this feature:

youtube

- `lang`: Prefer translated metadata (`title`, `description` etc) of this language code (case-sensitive). By default, the video primary language metadata is preferred, with a fallback to `en` translated. See youtube.py for list of supported content language codes
- `skip`: One or more of `hls`, `dash` or `translated_subs` to skip extraction of the m3u8 manifests, dash manifests and auto-translated subtitles respectively
- `player_client`: Clients to extract video data from. The main clients are `web`, `android` and `ios` with variants `_music`, `_embedded`, `_embedscreen`, `_creator` (e.g. `web_embedded`); and `mweb`, `mweb_embedscreen` and `tv_embedded` (agegate bypass) with no variants. By default, `ios,android,web` is used, but `tv_embedded` and `creator` variants are added as required for age-gated videos. Similarly, the music variants are added for `music.youtube.com` urls. You can use `all` to use all the clients, and `default` for the default clients.
- `player_skip`: Skip some network requests that are generally needed for robust extraction. One or more of `configs` (skip client configs), `webpage` (skip initial webpage), `js` (skip js player). While these options can help reduce the number of requests needed or avoid some rate-limiting, they could cause some issues. See #860 for more details
- `player_params`: YouTube player parameters to use for player requests. Will overwrite any default ones set by yt-dlp.
- `comment_sort`: `top` or `new` (default) - choose comment sorting mode (on YouTube's side)
- `max_comments`: Limit the amount of comments to gather. Comma-separated list of integers representing `max-comments,max-parents,max-replies,max-replies-per-thread`. Default is `all,all,all,all`
  - E.g. `all,all,1000,10` will get a maximum of 1000 replies total, with up to 10 replies per thread. `1000,all,100` will get a maximum of 1000 comments, with a maximum of 100 replies total

- `formats`: Change the types of formats to return. `dashy` (convert HTTP to DASH), `duplicate` (identical content but different URLs or protocol; includes `dashy`), `incomplete` (cannot be downloaded completely - live dash and post-live m3u8)
- `innertube_host`: Innertube API host to use for all API requests; e.g. `studio.youtube.com`, `youtubei.googleapis.com`. Note that cookies exported from one subdomain will not work on others
- `innertube_key`: Innertube API key to use for all API requests

youtubetab (YouTube playlists, channels, feeds, etc.)

- `skip`: One or more of `webpage` (skip initial webpage download), `authcheck` (allow the download of playlists requiring authentication when no initial webpage is downloaded. This may cause unwanted behavior, see #1122 for more details)
- `approximate_date`: Extract approximate `upload_date` and `timestamp` in flat-playlist. This may cause date-based filters to be slightly off

generic

- `fragment_query`: Passthrough any query in mpd/m3u8 manifest URLs to their fragments if no value is provided, or else apply the query string given as `fragment_query=VALUE`. Does not apply to ffmpeg
- `variant_query`: Passthrough the master m3u8 URL query to its variant playlist URLs if no value is provided, or else apply the query string given as `variant_query=VALUE`
- `hls_key`: An HLS AES-128 key URI *or* key (as hex), and optionally the IV (as hex), in the form of `(URI|KEY)[,IV]`; e.g. `generic:hls_key=ABCDEF1234567980,0xFEDCBA0987654321`. Passing any of these values will force usage of the native HLS downloader and override the corresponding values found in the m3u8 playlist
- `is_live`: Bypass live HLS detection and manually set `live_status` - a value of `false` will set `not_live`, any other value (or no value) will set `is_live`

funimation

- `language`: Audio languages to extract, e.g. `funimation:language=english,japanese`
- `version`: The video version to extract - `uncut` or `simulcast`

crunchyrollbeta (Crunchyroll)

- `format`: Which stream type(s) to extract (default: `adaptive_hls`). Potentially useful values include `adaptive_hls`, `adaptive_dash`, `vo_adaptive_hls`, `vo_adaptive_dash`, `download_hls`, `download_dash`, `multitrack_adaptive_hls_v2`
- `hardsub`: Preference order for which hardsub versions to extract, or `all` (default: `None` = no hardsubs), e.g. `crunchyrollbeta:hardsub=en-US,None`

vikichannel

- `video_types`: Types of videos to download - one or more of `episodes`, `movies`, `clips`, `trailers`

niconico

- `segment_duration`: Segment duration in milliseconds for HLS-DMC formats. Use it at your own risk since this feature **may result in your account termination.**

youtubewebarchive

- `check_all`: Try to check more at the cost of more requests. One or more of `thumbnails`, `captures`

gamejolt

- `comment_sort`: `hot` (default), `you` (cookies needed), `top`, `new` - choose comment sorting mode (on GameJolt's side)

hotstar

- `res`: resolution to ignore - one or more of `sd`, `hd`, `fhd`
- `vcodec`: vcodec to ignore - one or more of `h264`, `h265`, `dvh265`
- `dr`: dynamic range to ignore - one or more of `sdr`, `hdr10`, `dv`

niconicochannelplus

- `max_comments`: Maximum number of comments to extract - default is `120`

tiktok

- `api_hostname`: Hostname to use for mobile API requests, e.g. `api-h2.tiktokv.com`
- `app_version`: App version to call mobile APIs with - should be set along with `manifest_app_version`, e.g. `20.2.1`
- `manifest_app_version`: Numeric app version to call mobile APIs with, e.g. `221`

rokfinchannel

- `tab`: Which tab to download - one of `new`, `top`, `videos`, `podcasts`, `streams`, `stacks`

twitter

- `api`: Select one of `graphql` (default), `legacy` or `syndication` as the API for tweet extraction. Has no effect if logged in

stacommu, wrestleuniverse

- `device_id`: UUID value assigned by the website and used to enforce device limits for paid livestream content. Can be found in browser local storage

twitch

- `client_id`: Client ID value to be sent with GraphQL requests, e.g. `twitch:client_id=kimne78kx3ncx6brgo4mv6wki5h1ko`

nhkradirulive (NHK らじる★らじる LIVE)

- `area`: Which regional variation to extract. Valid areas are: `sapporo`, `sendai`, `tokyo`, `nagoya`, `osaka`, `hiroshima`, `matsuyama`, `fukuoka`. Defaults to `tokyo`

nflplusreplay

- `type`: Type(s) of game replays to extract. Valid types are: `full_game`, `full_game_spanish`, `condensed_game` and `all_22`. You can use `all` to extract all available replay types, which is the default

Note: These options may be changed/removed in the future without concern for backward compatibility

PLUGINS

Note that **all** plugins are imported even if not invoked, and that **there are no checks** performed on plugin code. **Use plugins at your own risk and only if you trust the code!**

Plugins can be of `<type>`s `extractor` or `postprocessor`.

- Extractor plugins do not need to be enabled from the CLI and are automatically invoked when the input URL is suitable for it.
- Extractor plugins take priority over builtin extractors.
- Postprocessor plugins can be invoked using `--use-postprocessor NAME`.

Plugins are loaded from the namespace packages `yt_dlp_plugins.extractor` and `yt_dlp_plugins.postprocessor`.

In other words, the file structure on the disk looks something like:

```
    yt_dlp_plugins/
        extractor/
            myplugin.py
        postprocessor/
            myplugin.py
```

yt-dlp looks for these `yt_dlp_plugins` namespace folders in many locations (see below) and loads in plugins from **all** of them.

See the wiki for some known plugins

Installing Plugins

Plugins can be installed using various methods and locations.

1. **Configuration directories**: Plugin packages (containing a `yt_dlp_plugins` namespace folder) can be dropped into the following standard configuration locations:
   - **User Plugins**
     - `${XDG_CONFIG_HOME}/yt-dlp/plugins/<package name>/yt_dlp_plugins/` (recommended on Linux/macOS)
     - `${XDG_CONFIG_HOME}/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
     - `${APPDATA}/yt-dlp/plugins/<package name>/yt_dlp_plugins/` (recommended on Windows)
     - `${APPDATA}/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
     - `~/.yt-dlp/plugins/<package name>/yt_dlp_plugins/`
     - `~/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
   - **System Plugins**
     - `/etc/yt-dlp/plugins/<package name>/yt_dlp_plugins/`
     - `/etc/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
2. **Executable location**: Plugin packages can similarly be installed in a `yt-dlp-plugins` directory under the executable location (recommended for portable installations):
   - Binary: where `<root-dir>/yt-dlp.exe`, `<root-dir>/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
   - Source: where `<root-dir>/yt_dlp/__main__.py`, `<root-dir>/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
3. **pip and other locations in `PYTHONPATH`**
   - Plugin packages can be installed and managed using `pip`. See yt-dlp-sample-plugins for an example.
     - Note: plugin files between plugin packages installed with pip must have unique filenames.
   - Any path in `PYTHONPATH` is searched in for the `yt_dlp_plugins` namespace folder.
     - Note: This does not apply for Pyinstaller/py2exe builds.

`.zip`, `.egg` and `.whl` archives containing a `yt_dlp_plugins` namespace folder in their root are also supported as plugin packages.

- e.g. `${XDG_CONFIG_HOME}/yt-dlp/plugins/mypluginpkg.zip` where `mypluginpkg.zip` contains `yt_dlp_plugins/<type>/myplugin.py`

Run yt-dlp with `--verbose` to check if the plugin has been loaded.

Developing Plugins

See the yt-dlp-sample-plugins repo for a template plugin package and the Plugin Development section of the wiki for a plugin development guide.

All public classes with a name ending in `IE`/`PP` are imported from each file for extractors and postprocessors repectively. This respects underscore prefix (e.g. `_MyBasePluginIE` is private) and `__all__`. Modules can similarly be excluded by prefixing the module name with an underscore (e.g. `_myplugin.py`).

To replace an existing extractor with a subclass of one, set the `plugin_name` class keyword argument (e.g. `class MyPluginIE(ABuiltInIE, plugin_name='myplugin')` will replace `ABuiltInIE` with `MyPluginIE`). Since the extractor replaces the parent, you should exclude the subclass extractor from being imported separately by making it private using one of the methods described above.

If you are a plugin author, add yt-dlp-plugins as a topic to your repository for discoverability.

See the Developer Instructions on how to write and test an extractor.

EMBEDDING YT-DLP

yt-dlp makes the best effort to be a good command-line program, and thus should be callable from any programming language.

Your program should avoid parsing the normal stdout since they may change in future versions. Instead they should use options such as `-J`, `--print`, `--progress-template`, `--exec` etc to create console output that you can reliably reproduce and parse.

From a Python program, you can embed yt-dlp in a more powerful fashion, like this:

```
from yt_dlp import YoutubeDL

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']
with YoutubeDL() as ydl:
    ydl.download(URLS)
```

Most likely, you'll want to use various options. For a list of options available, have a look at `yt_dlp/YoutubeDL.py` or `help(yt_dlp.YoutubeDL)` in a Python shell. If you are already familiar with the CLI, you can use `devscripts/cli_to_api.py` to translate any CLI switches to `YoutubeDL` params.

Tip: If you are porting your code from youtube-dl to yt-dlp, one important point to look out for is that we do not guarantee the return value of `YoutubeDL.extract_info` to be json serializable, or even be a dictionary. It will be dictionary-like, but if you want to ensure it is a serializable dictionary, pass it through `YoutubeDL.sanitize_info` as shown in the example below

Embedding examples

Extracting information

```
import json
import yt_dlp

URL = 'https://www.youtube.com/watch?v=BaW_jenozKc'

# ℹ See help(yt_dlp.YoutubeDL) for a list of available options and
ydl_opts = {}
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    info = ydl.extract_info(URL, download=False)

    # ℹ ydl.sanitize_info makes the info json-serializable
    print(json.dumps(ydl.sanitize_info(info)))
```

Download using an info-json

```
import yt_dlp

INFO_FILE = 'path/to/video.info.json'

with yt_dlp.YoutubeDL() as ydl:
    error_code = ydl.download_with_info_file(INFO_FILE)

print('Some videos failed to download' if error_code
      else 'All videos successfully downloaded')
```

Extract audio

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

ydl_opts = {
    'format': 'm4a/bestaudio/best',
    # ℹ See help(yt_dlp.postprocessor) for a list of available Pos
    'postprocessors': [{  # Extract audio using ffmpeg
        'key': 'FFmpegExtractAudio',
        'preferredcodec': 'm4a',
    }]
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    error_code = ydl.download(URLS)
```

Filter videos

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

def longer_than_a_minute(info, *, incomplete):
    """Download only videos longer than a minute (or with unknown
    duration = info.get('duration')
    if duration and duration < 60:
        return 'The video is too short'

ydl_opts = {
    'match_filter': longer_than_a_minute,
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    error_code = ydl.download(URLS)
```

Adding logger and progress hook

```
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download(URLS)
```

Add a custom PostProcessor

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

# ℹ See help(yt_dlp.postprocessor.PostProcessor)
class MyCustomPP(yt_dlp.postprocessor.PostProcessor):
    def run(self, info):
        self.to_screen('Doing stuff')
        return [], info


with yt_dlp.YoutubeDL() as ydl:
    # ℹ "when" can take any value in yt_dlp.utils.POSTPROCESS_WHEN
    ydl.add_post_processor(MyCustomPP(), when='pre_process')
    ydl.download(URLS)
```

Use a custom format selector

```
ydl_opts = {
    'format': format_selector,
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download(URLS)
```

DEPRECATED OPTIONS

These are all the deprecated options and the current alternative to achieve the same effect

Almost redundant options

While these options are almost the same as their new counterparts, there are some differences that prevents them being redundant

```
-j, --dump-json                  --print "%()j"
-F, --list-formats               --print formats_table
--list-thumbnails                --print thumbnails_table --print
--list-subs                      --print automatic_captions_table
```

Redundant options

While these options are redundant, they are still expected to be used due to their ease of use

```
--no-playlist-reverse            Default
--no-colors                      --color no_color
```

Not recommended

While these options still work, their use is not recommended since there are other alternatives to achieve the same

```
--force-generic-extractor        --ies generic,default
--exec-before-download CMD       --exec "before_dl:CMD"
--no-exec-before-download        --no-exec
--all-formats                    -f all
--all-subs                       --sub-langs all --write-subs
--print-json                     -j --no-simulate
--autonumber-size NUMBER         Use string formatting, e.g. %(aut
--autonumber-start NUMBER        Use internal field formatting lik
--id                             -o "%(id)s.%(ext)s"
--metadata-from-title FORMAT     --parse-metadata "%(title)s:FORMA
--hls-prefer-native              --downloader "m3u8:native"
--hls-prefer-ffmpeg              --downloader "m3u8:ffmpeg"
--list-formats-old               --compat-options list-formats (Al
--list-formats-as-table          --compat-options -list-formats [D
--youtube-skip-dash-manifest     --extractor-args "youtube:skip=da
--youtube-skip-hls-manifest      --extractor-args "youtube:skip=hl
--youtube-include-dash-manifest  Default (Alias: --no-youtube-skip
--youtube-include-hls-manifest   Default (Alias: --no-youtube-skip
--geo-bypass                     --xff "default"
--no-geo-bypass                  --xff "never"
--geo-bypass-country CODE        --xff CODE
--geo-bypass-ip-block IP_BLOCK   --xff IP_BLOCK
```

Developer options

These options are not intended to be used by the end-user

```
--test                           Download only part of video for t
--load-pages                     Load pages dumped by --write-page
--youtube-print-sig-code         For testing youtube signatures
--allow-unplayable-formats       List unplayable formats also
--no-allow-unplayable-formats    Default
```

Old aliases

These are aliases that are no longer documented for various reasons

```
--avconv-location                --ffmpeg-location
--clean-infojson                 --clean-info-json
```

```
--cn-verification-proxy URL      --geo-verification-proxy URL
--dump-headers                   --print-traffic
--dump-intermediate-pages        --dump-pages
--force-write-download-archive   --force-write-archive
--load-info                      --load-info-json
--no-clean-infojson              --no-clean-info-json
--no-split-tracks                --no-split-chapters
--no-write-srt                   --no-write-subs
--prefer-unsecure                --prefer-insecure
--rate-limit RATE                --limit-rate RATE
--split-tracks                   --split-chapters
--srt-lang LANGS                 --sub-langs LANGS
--trim-file-names LENGTH         --trim-filenames LENGTH
--write-srt                      --write-subs
--yes-overwrites                 --force-overwrites
```

Sponskrub Options

Support for SponSkrub has been deprecated in favor of the `--sponsorblock` options

```
--sponskrub                      --sponsorblock-mark all
--no-sponskrub                   --no-sponsorblock
--sponskrub-cut                  --sponsorblock-remove all
--no-sponskrub-cut               --sponsorblock-remove -all
--sponskrub-force                Not applicable
--no-sponskrub-force             Not applicable
--sponskrub-location             Not applicable
--sponskrub-args                 Not applicable
```

No longer supported

These options may no longer work as intended

```
--prefer-avconv                  avconv is not officially supporte
--prefer-ffmpeg                  Default (Alias: --no-prefer-avcon
-C, --call-home                  Not implemented
--no-call-home                   Default
--include-ads                    No longer supported
--no-include-ads                 Default
--write-annotations              No supported site has annotations
--no-write-annotations           Default
--compat-options seperate-video-versions  No longer needed
```

Removed

These options were deprecated since 2014 and have now been entirely removed

```
-A, --auto-number                -o "%(autonumber)s-%(id)s.%(ext)s
-t, -l, --title, --literal       -o "%(title)s-%(id)s.%(ext)s"
```

CONTRIBUTING

See CONTRIBUTING.md for instructions on Opening an Issue and Contributing code to the project

WIKI

See the Wiki for more information