The **download** package*†

Simon Sigurdhsson [sigurdhsson@gmail.com]

Version 1.0

Abstract The download package allows LaTeX to download files using cURL or wget.

1 Introduction

This package, inspired by a question on TeX.SE[1], allows LaTeX to download files using cURL or wget. Since it needs to run external commands, it requires unrestricted `\write18` access (***Note:** do not indiscriminately run pdfLaTeX with the `--shell-escape` flag; using this package it would be possible to download malicious `.tex` that may abuse the `\write18` access to harm your system*).

2 Usage

The package is very simple to use, but requires a *nix platform with either cURL or wget installed and present in the `PATH`.

2.1 Options

`engine` `auto,curl,wget` `(auto)`

The package only has one option, which controls what underlying software is used to download the file. As of version v1.0, the two engines available are cURL and wget. The default, which is used when no option

*Available on http://www.ctan.org/pkg/skbundle.

†Development version available on https://github.com/urdh/download.

[1] Klinger 2012.

is supplied, is `auto`. In this mode, download will look for wget and cURL, in that order, and use the first one available.

2.2 Macros

`\download` [⟨*filename*⟩]{⟨*url*⟩}

The only macro provided by download is `\download`. With it, you can download any file from any ⟨*url*⟩ supported by the underlying engine (wget supports `http(s)` and `ftp`, cURL supports a few more; for most cases wget should be enough). The optional argument ⟨*filename*⟩ makes the underlying engine save the file with the specified filename (***Note:*** *this also enables file existence checking; without it, wget and cURL will attempt to download the file even if it exists — wget see the existing file and do nothing, but cURL will download a new copy with a numeral suffix*).

3 Implementation

Let's have a look at the simple implementation. The package is based on LaTeX3, and should comply with the standards described i the expl3 manual. In any case, we begin by loading a few packages (expl3 for the LaTeX3 core, l3keys2e for applying l3keys functionality to LaTeX 2_ε package option parsing, pdftexcmds for the `\pdf@shellescape` macro and xparse for the public API definitions).

⟨package⟩
```
\RequirePackage{expl3,l3keys2e,pdftexcmds,xparse}
```

Then, we declare ourselves to provide the download LaTeX3 package.

⟨package⟩
```
\ProvidesExplPackage{download}
    {2012/12/31}{1.0}{download files with LaTeX}
```

3.1 Messages

We define a couple of messages using l3keys functionality.

The two first messages will be used as fatal errors, when we notice that functionality we absolutely *require* (*e.g.* either unrestricted `\write18` or the specified engine) is missing.

⟨package⟩
```
\msg_new:nnnn{download}{no-write18}{Could~not~use~\string\write18!}
    {Please~run~‘latex‘~with~the~‘--shell-escape‘~flag.}
\msg_new:nnnn{download}{no-engine}{Could~not~find~cURL~or~wget!}
    {Please~make~sure~either~cURL~or~wget~is~installed~and~in~your~PATH.}
```

We also have a message being displayed when `\download` is being used without its optional argument. This is a warning, since it may imply that cURL is creating a lot of unwanted files.

⟨package⟩
```
\msg_new:nnnn{download}{no-name}{Using~\string\download\space~with~no~filename!}
    {This~means~I~will~download~the~file~even~if~it~already~exists.}
```

The last two messages are diagnostics written to the log when `engine` is set to `auto`.

⟨package⟩
```
\msg_new:nnn{download}{use-curl}{Using~cURL.}
\msg_new:nnn{download}{use-wget}{Using~wget.}
```

3.2 The `\write18` test

We require unrestricted `\write18` and as such we must test for it. Using the `\pdf@shellescape` macro from pdftexcmds, we can define a new conditional that decides if we have unrestricted `\write18`.

`oad_if_shellescape:F` (no arguments)

⟨package⟩
```
\prg_new_conditional:Nnn\__download_if_shellescape:{F}{
    \if_cs_exist:N\pdf@shellescape
```

If the command sequence exists (it really should), we test to see if it is equal to one. The `\pdf@shellescape` macro will be zero if no `\write18` access is available, two if we have restricted access and one if access is unrestricted.

⟨package⟩
```
        \if_int_compare:w\pdf@shellescape=\c_one
            \prg_return_true:
        \else:
            \prg_return_false:
        \fi:
```

If the command sequence doesn’t exist, we assume that we have unrestricted `\write18` access (we probably don’t), and let LaTeX complain about it later.

⟨package⟩
```
    \else:
        \prg_return_true:
    \fi:
}
```

3.3 Utility functions

The existence tests for cURL and wget, explained later, create a temporary file. We might as well clean that up at once, since the user probably won't do that after each run of LaTeX, and an old file may break the check.

`\__download_rm:n` #1: The file to be removed

⟨package⟩
```
\cs_new:Npn\__download_rm:n#1{
    \immediate\write18{rm~#1}
}
```

3.4 Testing for cURL and wget

Testing for the existence of wget and cURL is done by calling the standard *nix `which` command. We define one conditional for every engine (starting with cURL):

`load_if_curl_test:TF` (no arguments)

`nload_if_curl_test:T` (no arguments)

`nload_if_curl_test:F` (no arguments)

`load_if_curl_test_p:` (no arguments)

⟨package⟩
```
\prg_new_conditional:Nnn\__download_if_curl_test:{TF,T,F,p}{
```

First, run `which` to create the temporary file.

⟨package⟩
```
  \immediate\write18{which~curl~&&~touch~\jobname.aex}
```

A temporary boolean is defined to hold the result of the test (this is a bit of carco cult programming, any clues as to why placing the result inside `\file_if_exists:nTF` doesn't work are welcome), and if the test is successful we remove the temporary file.

⟨package⟩
```
    \bool_set:Nn\l_tmpa_bool{\c_false_bool}
    \file_if_exist:nTF{\jobname.aex}{
        \__download_rm:n{\jobname.aex}
        \bool_set:Nn\l_tmpa_bool{\c_true_bool}
    }{}
```

Finally, the temporary boolean is used to return a result.

⟨package⟩
```
    \if_bool:N\l_tmpa_bool
        \prg_return_true:
    \else:
        \prg_return_false:
    \fi:
}
```

The conditional for wget is defined analogously to that of cURL (again, a bit of cargo cult programming; ideally we'd be DRY and have one does-this-executable-exist-conditional, but for some reason that wouldn't work out for me — hints appreciated).

`load_if_wget_test:TF` (no arguments)

`nload_if_wget_test:T` (no arguments)

`nload_if_wget_test:F` (no arguments)

`load_if_wget_test_p:` (no arguments)

⟨package⟩
```
\prg_new_conditional:Nnn\__download_if_wget_test:{TF,T,F,p}{
    \immediate\write18{which~wget~&&~touch~\jobname.aex}
    \bool_set:Nn\l_tmpa_bool{\c_false_bool}
    \file_if_exist:nTF{\jobname.aex}{
        \__download_rm:n{\jobname.aex}
        \bool_set:Nn\l_tmpa_bool{\c_true_bool}
    }{}
    \if_bool:N\l_tmpa_bool
        \prg_return_true:
    \else:
        \prg_return_false:
    \fi:
}
```

3.5 Using cURL and wget

Actually using cURL and wget for downloading is simple, issuing two different commands depending on wether we have the optional argument or not (i.e. it is `\NoValue`).

`_download_curl_do:nn` #1: Filename to save file to, or `\NoValue`
#2: URL to fetch the file from

```
⟨package⟩ \cs_new:Npn\__download_curl_do:nn#1#2{
              \IfNoValueTF{#1}{
```

When no optional argument is given, we just invoke cURL with the `-s` (silent) flag.

```
⟨package⟩         \immediate\write18{curl~-s~#2}
              }{
```

When we *do* have an optional argument, we add the `-o` flag to specify the output file.

```
⟨package⟩         \immediate\write18{curl~-s~-o~#1~#2}
              }
          }
```

`_download_wget_do:nn` #1: Filename to save file to, or `\NoValue`
#2: URL to fetch the file from

```
⟨package⟩ \cs_new:Npn\__download_wget_do:nn#1#2{
              \IfNoValueTF{#1}{
```

With wget, we pass the `-q` (quiet) flag as well as the `-nc` (no clobber) file, to avoid downloading files that already exist.

```
⟨package⟩         \immediate\write18{wget~-q~-nc~#2}
              }{
```

Again, when we have an optional argument we add the `-O` flag to specify the output file.

```
⟨package⟩         \immediate\write18{wget~-q~-nc~-O~#1~#2}
              }
          }
```

3.6 The `auto` engine

The automatic engine uses the tests and macros of the other engines to provide functionality without selecting an engine. We first define a conditional that, in essence, steps through the available engines testing for their existence. If any of them exist, we're in business.

`nload_if_auto_test:F` (no arguments)

⟨package⟩
```
\prg_new_conditional:Nnn\__download_if_auto_test:{F}{
```

To avoid excessive nesting of conditional statements, we define a boolean (initialized to `\c_false_bool`) which we then set to `\c_true_bool` every time we find an engine that exists.

⟨package⟩
```
    \bool_set:Nn\l_tmpb_bool{\c_false_bool}
    \__download_if_curl_test:T{\bool_set:Nn\l_tmpb_bool{\c_true_bool}}
    \__download_if_wget_test:T{\bool_set:Nn\l_tmpb_bool{\c_true_bool}}
```

We use that boolean to return a result from the conditional.

⟨package⟩
```
    \if_bool:N\l_tmpb_bool
        \prg_return_true:
    \else:
        \prg_return_false:
    \fi:
}
```

We also define an automatic equivalent of the engine `_do` macros, which selects wget if possible and cURL as a second choice.

`_download_auto_do:nn` #1: Filename to save file to, or `\NoValue`
#2: URL to fetch the file from

⟨package⟩
```
\cs_new:Npn\__download_auto_do:nn#1#2{
    \__download_if_wget_test:TF{
        \msg_info:nn{download}{use-wget}
        \__download_wget_do:nn{#1}{#2}
    }{
        \msg_info:nn{download}{use-curl}
        \__download_curl_do:nn{#1}{#2}
    }
}
```

3.7 Package options

As detailed earlier in the documentation, the only option of the package is `engine`. Here, we use the l3keys functionality to define a key-value system which we later use to read the package options.

⟨package⟩
```
\keys_define:nn{download}{
    engine .choice:,
```

`\__download_do:nn` #1: Filename to save file to, or `\NoValue`
#2: URL to fetch the file from

`nload_if_auto_test:F` (no arguments)

First, the `auto` value. We globally define two macros as aliases to the underlying `_do` and `_if_test` macros.

⟨package⟩
```
    engine / auto .code:n =
        {\cs_gset_eq:NN\__download_do:nn\__download_auto_do:nn
         \prg_new_eq_conditional:NNn\__download_if_test:\__download_if_auto_test:{F}},
```

We do the same for the cURL and wget options.

⟨package⟩
```
    engine / curl .code:n =
        {\cs_gset_eq:NN\__download_do:nn\__download_curl_do:nn
         \prg_new_eq_conditional:NNn\__download_if_test:\__download_if_curl_test:{F}},
    engine / wget .code:n =
        {\cs_gset_eq:NN\__download_do:nn\__download_wget_do:nn
         \prg_new_eq_conditional:NNn\__download_if_test:\__download_if_wget_test:{F}},
```

Lastly, we initialize the option with the `auto` value.

⟨package⟩
```
    engine .initial:n = auto,
    engine .default:n = auto,
}
```

Now, given the key-value system, we parse the options.

⟨package⟩
```
\ProcessKeysPackageOptions{download}
```

3.8 Performing the tests

Now that we know what engine we will be using, we can check for `\write18` support and engine existence.

⟨package⟩
```
\__download_if_shellescape:F{\msg_fatal:nn{download}{no-write18}}
\__download_if_test:F{\msg_fatal:nn{download}{no-engine}}
```

3.9 Public API

The public API consists of only one macro, `\download`. It simply calls the backend macro, unless the optional argument is given and the file exists. If the file doesn't exist, it also emits a friendly warning.

`\download` `#1`: Filename to save file to, or `\NoValue`
`#2`: URL to fetch the file from

⟨package⟩
```
\DeclareDocumentCommand\download{om}{
    \IfNoValueTF{#1}{
        \msg_warning:nn{download}{no-name}
        \__download_do:nn{#1}{#2}
    }{
        \file_if_exist:nTF{#1}{}{\__download_do:nn{#1}{#2}}
    }
}
```

And we're done.

⟨package⟩
```
\endinput
```

4 Changes

v1.0

General: Initial version.

5 Index

Numbers written in boldface refer to the page where the corresponding entry is described; numbers underlined refer to the page were the implementation of the corresponding entry is discussed. Numbers in roman refer to other mentions of the entry.

6 Bibliography

Klinger, Max (2012). *Creating a URL downloading command to be used with e.g. \includegraphics*. URL: http://tex.stackexchange.com/questions/88430/creating-a-url-downloading-command-to-be-used-with-e-g-includegraphics.