The make4ht build system

Michal Hoftich*

Version v0.1c
2017-04-26

Contents

1 Introduction

make4ht is a simple build system for tex4ht. It is both executable which simplifies tex4ht execution and a library which can be used to create customized conversion programs. An example of such conversion program is tex4ebook

1.1 License

*michal.h21@gmail.com

1.2 How it works

Default compilation script for tex4ht, named htlatex compiles LaTeX files to HTML with this command sequence:

```
latex $latex_options 'code for loading tex4ht.sty \input{filename}'
latex $latex_options 'code for loading tex4ht.sty \input{filename}'
latex $latex_options 'code for loading tex4ht.sty \input{filename}'
tex4ht options filename
t4ht options filename
```

The problem is that this is inefficient when you need to run program which interact with LaTeX, such as Makeindex or Bibtex. In that case, you need to create new script based on the default one, or run htlatex twice, which means six LaTeX runs.

Another problem is with t4ht application. It reads file filename.lg, generated by tex4ht, where are instructions about generated files, CSS instructions, calls to external applications, instructions for image conversions etc. It can be instructed to copy generated files to some output directory, but it doesn't preserve directory structure, so when you have images in some subdirectory, they will be copied to the output directory, but links will be pointing to non existing subdirectory.

Image conversion is directed with the env file, with really strange syntax based on whitespace and os dependent. With make4ht build files, we have simple mean to fix these issues. We can change image conversion parameters without need to modify the env file, and call actions on the output files. These actions can be either external programs such as xslt processors or HTML tidy or Lua functions.

The idea is to make system controlled by a build file. Because Lua interpreter is included in modern TeX distributions and Lua is ideal language for such task, it was chosen as language in which build script are written.

2 Build files

2.1 Commands

By default, build file is saved in file named filename + .mk4 extension. You can choose different build file with -e or --build-file command line option.

Sample:

```
Make:htlatex()
Make:match("html$", "tidy -m -xml -utf8 -q -i ${filename}")
```

Make:htlatex() is preconfigured command for calling LaTeX with tex4ht loaded on the input file. In this case it will be called one time. After compilation, tidy command is executed on the output html file.

Note that you don't have to call `tex4ht` and `t4ht` commands explicitly in the build file, they are called automatically.

You can add more commands like `Make:htlatex` with

```
Make:add("name", "command", {parameters}, repetition)
```

it can be called with

```
Make:name()
```

`command` can be text template, or function:

```
Make:add("text", "hello, input file: ${input}")
Make:add("function", function(params)
  for k, v in pairs(params) do
    print(k..": "..v)
  end
)
```

`parameters` is a table or `nil` value.

Default parameters are:

htlatex used compiler

input it is output file name in fact

tex_file input TeX file

latex_par parameters to `latex`

packages insert additionl LaTeX code which is inserted before `\documentclass`. Useful for passing options to packages or additional packages loading

tex4ht_sty_par parameters to `tex4ht.sty`

tex4ht_par parameters to `tex4ht` application

t4ht_par parameters to `t4ht` application

outdir output directory

repetition limit number of command execution.

correct_exit expected `exit code` from the command. The compilation will be terminted when the command `exit code` is different.

You may add your own parameters, they will be accessible in templates and functions.

With `repetition`, you can limit number of command executions. Its value should be number or `nil`. This is used in the case of `tex4ht` and `t4ht` commands, as they should be executed only once and they would be executed multiple

times if you include them in the build file, because they would be called also by `make4ht`. With `repetition`, second execution is blocked.

You can set expected exit code from a command with `correct_exit`. Compilation is stopped when command returns different exit code. The situation is different for LaTeX (for all TeX engines and formats, in fact), because it doesn't differentiate between fatal and non fatal errors, and it returns the same exit code in all cases. Log parsing is used because of that, error code `1` is returned in the case of fatal error, `0` is used otherwise.

2.2 File matches

Other type of action which can be specified in the build file are matches which may be called on the generated files:

```
Make:match("html$", "tidy -m -xml -utf8 -q -i ${filename}")
```

It tests filenames with `Lua` pattern matching and on matched items will execute command or function, specified in the second argument. For functions, two arguments are passed, first one is the current filename, the second one table with parameters. These parameters are the same as in previous section, except for `filename`, which contains generated output name.

2.2.1 Filters

Some default match actions which you can use are in the `filter` module. It contains some functions which ares useful for fixing some `tex4ht` bugs or shortcomings.

Example:

```
local filter = require "make4ht-filter"
local process = filter{"cleanspan", "fixligatures", "hruletohr"}
Make:htlatex()
Make:htlatex()
Make:match("html$",process)
```

Filter module is located in `make4ht-filter`. Function is returned, which is used for building filter chains then.

Built-in filters are:

cleanspan clean spurious span elements when accented characters are used

cleanspan-nat alternative clean span filter, provided by Nat Kuhn

fixligatures decompose ligatures to base characters

hruletohr `\hrule` commands are translated to series of underscore characters by `tex4ht`, this filter translate these underscores to `<hr>` elements

entites convert prohibited named entities to numeric entities (currently, only ` `, as it causes validation errors, and `tex4ht` is producing it sometimes

fix-links replace colons in local links and `id` attributes with underscores. Some cross-reference commands may produce colons in internal links, which results in validation error.

Function `filter` accepts also function arguments, in this case this function takes file contents as parameter and modified contents are returned.

Example:

```
local filter  = require "make4ht-filter"
local changea = function(s) return s:gsub("a","z") end
local process = filter{"cleanspan", "fixligatures", changea}
Make:htlatex()
Make:htlatex()
Make:match("html$",process)
```

In this case, spurious span elements are joined, ligatures are decomposed, and then all letters 'a' are replaced with 'z' letters.

2.3 Image conversion

It is possible to convert parts of LaTeX input to pictures, it is used for example for math or diagrams in `tex4ht`.

These pictures are stored in special `dvi` file, on which `dvi to image` commands are called.

This conversion is normally configured in the `env file`, which is system dependent and which has a bit unintuitive syntax. This configuration is processed by `t4ht` application and conversion commands are called for all pictures.

It is possible to disable `t4ht` image processing and configure image conversion in the make file:

```
Make:image("png$",
"dvipng -bg Transparent -T tight -o ${output}  -pp ${page} ${source}")
```

`Make:image` takes two parameters, pattern to match image name and action. Action can be either string template with conversion command, or function which takes table with parameters as argument.

There are three parameters:

- output - output image file name
- source - `dvi` file with the pictures
- page - page number of the converted image

2.4 The `mode` variable

The `mode` variable contains contents of `--mode` command line option. It can be used to run some commands conditionally. For example:

```
if mode == "draft" then
  Make:htlatex{}
else
  Make:htlatex{}
  Make:htlatex{}
  Make:htlatex{}
end
```

In this example (which is the default configuration used by `make4ht`), LaTeX is called only once when `make4ht` is called with `draft` mode:

```
make4ht -m draft filename
```

2.5 The `settings` table

You may want to access to the parameters also outside commands, file matches and image conversion functions. For example, if you want to convert your file to the `OpenDocument Format`, you can use the following settings, based on `oolatex` command:

```
settings.tex4ht_sty_par = settings.tex4ht_sty_par ..",ooffice"
settings.tex4ht_par = settings.tex4ht_par .. " ooffice/! -cmozhtf"
settings.t4ht_par = settings.t4ht_par .. " -cooxtpipes -coo "
```

3 Command line options

```
make4ht - build system for tex4ht
Usage:
make4ht [options] filename ["tex4ht.sty op." "tex4ht op."
     "t4ht op" "latex op"]
-b,--backend (default tex4ht) Backend used for xml generation.
     possible values: tex4ht or lua4ht
-c,--config (default xhtml) Custom config file
-d,--output-dir (default "")  Output directory
-e,--build-file (default nil)  If build file name is different
     than `filename`.mk4
-l,--lua  Use lualatex for document compilation
-m,--mode (default default) Switch which can be used in the makefile
-n,--no-tex4ht  Disable dvi file processing with tex4ht command
-s,--shell-escape Enables running external programs from LaTeX
-u,--utf8  For output documents in utf8 encoding
-v,--version  Print version number
-x,--xetex Use xelatex for document compilation
<filename> (string) Input file name
```

You can still use `make4ht` in same way as `htlatex`

```
make4ht filename "customcfg, charset=utf-8" " -cunihtf -utf8" " -dfoo"
```

Note that this will not use `make4ht` routines for output dir making and copying If you want to use them, change the line above to:

```
make4ht filename "customcfg, charset=utf-8" " -cunihtf -utf8"  -d foo
```

This call has the same effect as following:

```
make4ht -u -c customcfg -d foo filename
```

Output directory doesn't have to exist, it will be created automatically. Specified path can be relative to current directory, or absolute:

```
make4ht -d use/current/dir/ filename
make4ht -d ../gotoparrentdir filename
make4ht -d ~/gotohomedir filename
make4ht -d c:\documents\windowspathsareworkingtoo filename
```

4 Changelog

- 2017/04/26
  - Released version `v0.1c`
- 2017/03/16
  - check for `TeX capacity exceeded` error in the LaTeX run.
- 2016/12/19
  - use full input name in `tex_file` variable. This should enable use of files without `.tex` extension.
- 2016/10/22
  - new command available in the build file: `Make:add_file(filename)`. This enables filters and commands to register files to the output.
  - use ipairs instead of pairs for traversing files and executing filters. This should ensure correct order of executions.
- 2016/10/18
  - new filter: replace colons in `id` and `href` attributes with underscores
- 2016/01/11

  - fixed bug in loading documents with full path specified

- 2015/12/06 version 0.1b
  - modifed lapp library to recognize `--version` and
  - added `--help` and `--version` command line options

- 2015/11/30
  - use `kpse` library for build file locating

- 2015/11/17
  - better `-jobname` handling

- 2015/09/23 version 0.1a
  - various documentation updates

- `mozhtf` profile for unicode output is used, this should prevent ligatures in the output files

- 2015/06/29 version 0.1
  - major README file update

- 2015/06/26
  - added Makefile
  - moved INSTALL instructions from README to INSTALL