juhtolv-fontsamples

Juhapekka Tolvanen

Contents

1 Copyright and contact information **3**

2 Introduction **5**

3 How to use these PDFs **6**

3.1 Decompressing 6

3.1.1 bzip2 and tar 8

3.1.2 7zip 11

3.2 Finding right PDFs 12

4 How to edit these PDFs **16**

4.1 What each script do 16

4.2 About shell environment 19

4.3 How to ensure each PDF take just one page 20

4.4 Programs used 20

4.5 Fonts used 23

4.5.1 Proportional Gothic 24

4.5.2 Proportional Mincho 26

4.5.3 Monospace (Gothic and Mincho) 27

4.5.4 Handwriting 28

4.5.5 Fonts that can not be used 29

5 Thanks **30**

1 Copyright and contact information

Author of all these files is Juhapekka Tolvanen. Author's E-Mail address is:

`juhtolv (at) iki (dot) fi`

This publication has included material from these dictionary files in accordance with the licence provisions of the Electronic Dictionaries Research Group:

- `kanjd212`
- `kanjidic`
- `kanjidic2.xml`
- `kradfile`
- `kradfile2`

See these WWW-pages:

- http://www.edrdg.org/
- http://www.edrdg.org/edrdg/licence.html
- http://www.csse.monash.edu.au/~jwb/edict.html
- http://www.csse.monash.edu.au/~jwb/kanjidic_doc.html
- http://www.csse.monash.edu.au/~jwb/kanjd212_doc.html
- http://www.csse.monash.edu.au/~jwb/kanjidic2/
- http://www.csse.monash.edu.au/~jwb/kradinf.html

All generated PDF- and TEX-files of each kanji-character use exactly the same license as `kanjidic2.xml` uses; Name of that license is Creative Commons Attribution-ShareAlike Licence (V3.0). That License text can be read here:

http://creativecommons.org/licenses/by-sa/3.0/

Full license text is here:

http://creativecommons.org/licenses/by-sa/3.0/legalcode

The SKIP (System of Kanji Indexing by Patterns) system for ordering kanji was developed by Jack Halpern (chief editor of Kanji Dictionary Publishing Society at http://www.kanji.org/) and is used as the main index for kanji in the New Japanese-English Character Dictionary (Kenkyusha/NTC, 1990) and The Kodansha Kanji Learner's Dictionary (Kodansha, 1999) and other dictionaries published by Kanji Dictionary Publishing Society.

The SKIP codes for the kanji not in the NJECD were compiled by Jack Halpern and Martin Duerst (the remaining kanji in JIS X 0208) and Jim Breen (the 5,801 kanji in JIS X 0212).

See that section "APPENDIX F. CONDITIONS FOR USING SKIP DATA by Jack Halpern (`jack@kanji.org`)" as seen in documentation of KANJIDIC and "SKIP SYSTEM OF KANJI INDEXING BY PATTERNS. Conditions Of Use":

- http://www.csse.monash.edu.au/~jwb/kanjidic_doc.html
- http://www.kanji.org/kanji/dictionaries/skip_permission.htm

SKIP-codes are extracted from `kanjidic2.xml` . If you really like SKIP-system, it is good idea to buy some wonderful kanji-dictionaries by Jack Halpern and KDPS. On the other hand, if you really like `kanjidic2.xml` and its friends, it is good idea to donate some money to EDRDG.

Unless otherwise noted, all other files are public domain, including generated PDF- and TEX-files for other than kanji-characters (kanas (hiraganas and katakanas)) and this documentation file. There is no warranty.

New version of all this may be released here:

http://iki.fi/juhtolv/japanese/fontsamples/

Some bigger parts may be available here:

http://www.elisanet.fi/juhtolv/japanese/fontsamples/

If you actually use these files, it would be nice to know; Please, send me some E-Mail about it.

2 Introduction

こんにちは。My name is Juhapekka Tolvanen (ユハペッカ・トルワネン). When I started learning Japanese language, I quickly realized that it is not easy to figure out, which parts of each character are really relevant and which parts are just decoration, that is seen only in Mincho-fonts and calligraphy. On the other hand, some kanas may look very different depending on fonts, especially き, さ and ふ. Even some kanjis look a little bit different in different fonts, for example 入 (U+5165 CJK UNIFIED IDEOGRAPH-5165). Therefore I created this solution: A bunch of shell-scripts that generate one-page LaTeX-files that are compiled to PDF-format with XeLaTeX. Each PDF-file shows one Japanese character in as many open-source Japanese font as possible.

When you are about to learn some new Japanese character, feel free to check out all those styles it is written in. Then just take some sheets of genkouyoushi and start practicing writing that character. I hope you will find these files useful, when trying to find your own style to write those characters by hand and when learning to read them. The main point of these files is this: To find the essential nature of Japanese characters, both kanjis and kanas. Of course, you need also something else than just these files.

On the other hand, if you are creating a brand new font or extending some existing font, these files may be useful for you in this way: You can check out, how certain glyph has been implemented in other fonts. After that you can just start your font-editor and add that glyph to your font.

Every character is first shown in a font called “KanjiStrokeOrders” in very big size (180 pt). As the name suggests, it is a font that shows stroke order of each character. Then come normal fonts in a smaller size (28 pt).

If you actually use these files, it would be nice to know; Please, send me some E-Mail about it.

3 How to use these PDFs

3.1 Decompressing

You need to know, how to decompress `.tar.bz2`, `.bz2` and `.7z`-files. Each directory has been archived with `tar` and then those `tar`-archives have been compressed with `bzip2`. `.bz2`-files can be decompressed with `bunzip2`. Some bigger directories and directories of seldom-needed files have been compressed with `7zip`.

`bzip2` (and `gzip`) can not compress more than one file at a time. Therefore it is very common to archive directories to one `tar`-file before compressing them with `bzip2`.

I really have to use `bzip2` instead of `gzip`, because otherwise I would break quota of WWW-servers. For some bigger directories that is not enough and I need `7zip`. Amount of disk space these files take is just so tremendous. And I do not want to use some lousy file compression formats like `ZIP`, because they take even more disk space.

Here are some test results: File sizes of one directory packed with different methods:

```
Megabytes:
 18M     kanji-G1-pdf.tar
 12M     kanji-G1-pdf.zip
 12M     kanji-G1-pdf.arj
 12M     kanji-G1-pdf.tar.gz
 12M     kanji-G1-pdf.lzh
 11M     kanji-G1-pdf.pmd
 11M     kanji-G1-pdf.tar.dct
 10M     kanji-G1-pdf.tar.bz2
8,2M     kanji-G1-pdf.tar.7z
8,2M     kanji-G1-pdf.7z

Bytes:
```

```
18186240 kanji-G1-pdf.tar
11641045 kanji-G1-pdf.zip
11608559 kanji-G1-pdf.arj
11597307 kanji-G1-pdf.tar.gz
11563175 kanji-G1-pdf.lzh
11032510 kanji-G1-pdf.pmd
10526128 kanji-G1-pdf.tar.dct
10485140 kanji-G1-pdf.tar.bz2
 8573121 kanji-G1-pdf.tar.7z
 8552322 kanji-G1-pdf.7z
```

That file with ending `.pmd` was compressed with `ppmd`. That file with ending `.tar.dct` is `.tar`-archive compressed with `dact`. As you can see, `7zip` wins with clear margin. I used best compression available in all programs, but for `7zip` I used these settings (both in aforementioned test and when creating most of those compressed `7zip` archives):

```
-t7z -m0=lzma -mx=9 -mfb=64 -md=32m -ms=on
```

It means this:

- -t7z : 7z archive
- -m0=lzma : lzma method
- -mx=9 : level of compression = 9 (Ultra)
- -mfb=64 : number of fast bytes for LZMA = 64
- -md=32m : dictionary size = 32 megabytes
- -ms=on : solid archive = on

With other options it is possible to achieve even better compression with `7zip`, but it puts my computer on its knees. But even those aforementioned options are enough to make `7zip` a clear winner. All `.tex` -files and smaller directories are compressed with tighter options, though.

On the other hand, I do not want to use even more modern compression method, like `PAQ`, `PEA` or `KGB`, because `bzip2` and `7zip` have been around much longer time and therefore they are well supported by many file archiver programs that run on many operating systems. It is enough, if I can avoid breaking quota.

Also availability of open-source implementations of compression formats has been important factor when choosing compression methods. Therefore compression formats like RAR and ACE are strictly out of question.

I also hope that my compression program choices are good for you, too: They should shorten your download times.

3.1.1 bzip2 and tar

3.1.1.1 GNU/Linux and other Unix-like operating systems

Syntax of bzip2 (and bunzip2) is very familiar for users of gzip (and gunzip). Hence, if you already know, how to decompress with combination of tar and gzip, all you need to do is to use bzip2 instead of gzip (or bunzip2 instead gunzip). It is so simple.

On any Unix-like operating system you can decompress .tar.bz2-files this way, if both of those commands are available:

```
bzip2 -cd file.tar.bz2 | tar xf -
```

If your tar is actually star , bsdtar or GNU tar, you can decompress .tar.bz2-files this way:

```
tar -x -j -v -f file.tar.bz2
```

They all also work without that option "-j" , because they can auto-detect bzip2-compression.

If file is compressed with just bzip2 and is not archived with tar before that, you can decompress it this way:

```
bunzip2 file.bz2
```

Or this way:

```
bzip2 -d file.bz2
```

If you decompress .tar.bz2-file with plain bunzip2 or bzip2 and without tar , you get some plain .tar-file. It can be "decompressed" this way:

```
tar -xvf file.tar
```

You can also use pax this way:

```
pax -r -v -f file.tar
```

Some implementations of `pax` can decompress `tar.bz2`-files, too. Read its `man`-page. In Debian GNU/Linux `pax` was downloaded from FTP-directory of OpenBSD and some of it comes from FreeBSD and rest was donated. It can decompress `.tar.bz2`-files. Also `pax`-command of Heirloom Toolchest can decompress `.tar.bz2`-files. `pax`-command of AT&T AST-utils can not handle `.tar.bz2`-files. `star` comes with `pax`-implementation called `spax` . In Debian GNU/Linux syntax is this:

```
pax -r -v -j -f file.tar.bz2
```

If you use `pax` from Heirloom Toolchest, syntax is this:

```
pax -r -v -f file.tar.bz2
```

That syntax works also with `spax` but this is better syntax for it:

```
spax -r -bz -v -f file.tar.bz2
```

Nowadays graphical desktops (especially GNOME and KDE) of GNU/Linux and other Unix-like operating systems have their own graphical programs for decompressing. I do not bother list them. There are also many programs that are not at all related to those desktop systems. Just see, what happens. if you keep right button of mouse pressed, when mouse cursor is over some compressed file. Probably some context menu opens and there is some entry that promise to do decompressing. Also double-clicking files may work.

3.1.1.2 MacOS X

In all versions of MacOS X you can run aforementioned Unix-commands in a program called `Terminal.app` or other terminal emulator, for example `iTerm` or `xterm`. If you just double-click `.tar.bz2`-file, then so called Archive Utility[1] should decompress it. It is the default archive file handler in MacOS X. Also these graphical programs can handle `.bz2`-files:

- iArchiver: http://iarchiver.com/
- StuffIt: http://my.smithmicro.com/
- The Unarchiver: http://en.wikipedia.org/wiki/The_Unarchiver
- Zipeg: http://www.zipeg.com/

[1] http://en.wikipedia.org/wiki/Archive_Utility

3.1.1.3 Windows

In many kinds of Windows -operating systems you can run those aforementioned Unix-commands in Cygwin-environment or you may find some straight Windows ports of `tar` and `bzip2`. You can run them in `COMMAND.COM` or `CMD.EXE` or other terminal window. In addition at least these graphical programs can at least decompress `.bz2`- and `.tar.bz2`-files:

- 7-Zip: http://www.7-zip.org/
- ALZip: http://www.altools.com/ALTools/ALZip/
- IZArc: http://www.izarc.org/
- PeaZip: http://www.peazip.org/
- PKZIP: http://www.pkware.com/
- PowerArchiver: http://www.powerarchiver.com/
- Simplyzip: http://www.paehl.de/english.php
- StuffIt: http://my.smithmicro.com/
- Squeez: http://www.speedproject.de/enu/squeez/
- TugZIP: http://www.tugzip.com/
- UltimateZip: http://www.ultimatezip.com/
- Universal Extractor: http://legroom.net/software/uniextract/
- WinRAR: http://www.win-rar.com/
- WinTarBall: http://aegisknight.org/wintarball
- WinZip: http://www.winzip.com/
- Zipeg: http://www.zipeg.com/
- ZipZag: http://www.zipzag.com/

You'd better have the freshest version of program, no matter which one you choose; If you already have some of those aforementioned programs and it can not decompress `.bz2`-files, it is time to upgrade it. That WinZip is a little bit weird case: It claims to be able to decompress .bz2-files but I have been told it really can not do it.

3.1.2 7zip

All `7zip`-files are compressed to so called solid archive in order to achieve better compression. It has this drawback: You can not extract just one file from such archive; You always have to decompress the whole archive.

3.1.2.1 GNU/Linux and other Unix-like operating systems

In GNU/Linux and other Unix-like operating systems you can do decompression with a command-line program called `p7zip`. It is used this way:

```
7z x file.7z
```

Default compression programs of graphical desktops (especially KDE and GNOME) can handle `.7z` same way as they can handle other compression formats. I do not bother list them.

3.1.2.2 MacOS X

In all versions of MacOS X you can run aforementioned Unix-commands in a program called `Terminal.app` or other terminal emulator, for example `iTerm` or `xterm`. Also these graphical programs can handle `.7z`-files:

- 7zX: http://sixtyfive.xmghosting.com/products/7zx/
- iArchiver: http://iarchiver.com/
- The Unarchiver: http://en.wikipedia.org/wiki/The_Unarchiver
- Zipeg: http://www.zipeg.com/

3.1.2.3 Windows

In many kinds of Windows -operating systems you may be able to run those aforementioned Unix-commands in Cygwin-environment or you may find some straight Windows ports of `p7zip`. You can run them in `COMMAND.COM` or `CMD.EXE` or other terminal window. In addition at least these graphical programs can at least decompress `7zip`-files:

- 7-Zip: http://www.7-zip.org/
- ALZip: http://www.altools.com/ALTools/ALZip/

- IZArc: http://www.izarc.org/
- PeaZip: http://www.peazip.org/
- PowerArchiver: http://www.powerarchiver.com/
- QuickZip: http://www.quickzip.org/
- Simplyzip: http://www.paehl.de/english.php
- Squeez: http://www.speedproject.de/enu/squeez/
- TugZIP: http://www.tugzip.com/
- UltimateZip: http://www.ultimatezip.com/
- Universal Extractor: http://legroom.net/software/uniextract/
- WinRAR: http://www.win-rar.com/
- Zipeg: http://www.zipeg.com/
- ZipGenius http://www.zipgenius.it/
- ZipZag: http://www.zipzag.com/

You'd better have the freshest version of program, no matter which one you choose; If you already have some of those aforementioned programs and it can not decompress `.7z`-files, it is time to upgrade it.

3.2 Finding right PDFs

After unpacking you can view all those PDF-files with almost any PDF-viewer you happen to have. You need to figure out the right Unicode-value for that character you are looking for. For example あ is "U+3042 (HIRAGANA LETTER A)". It is basic hiragana. Therefore you must go to those PDF-files for basic hiraganas and take a file called "`U+3042.pdf`". Files in a directory called `index` should help you to find what directory has that certain PDF-file you are looking for. Files in a directory called `du_files` tells you how much hard drive space each subgroup take when they are uncompressed; You'd better consult it in order to avoid filling up all your hard drive space.

All those (sub)directories and files in directories called “hiragana” and “katakana” should be self-explanatory at least for those who know the terminology of Japanese kana-characters. As you can see, in some of those subdirectories PDF-files are almost empty; For example support for Ainu-language extensions in katakana is very poor in many fonts.

In a directory called “kanji” files and sub-directories are named according to information fields in three dictionary-files created by Jim Breen et al. In the other words, they mean this:

- G1–G6 : Jouyou Kanjis taught in Elementary School during grades from first to sixth (Also known as “Gakunen-betsu kanji haitouhyou” or “Kyouiku Kanji”).
- G8 : Jouyou Kanjis taught in Secondary School
- G9 : Jinmeiyou kanjis that are NOT variants of a Jouyou kanjis
- G10 : Jinmeiyou kanjis that are variants of a Jouyou kanjis.
- other-common : Kanjis that do not belong to Jouyou or Jinmeiyou kanjis (Also known as “hyougaiji”, “hyougai kanji” or “jouyougai kanji”), BUT are still so common that they have the frequency-of-use ranking (i.e. a field called “F”).
- other-uncommon1 : Hyougaiji, that DO NOT have frequency-of-use ranking at all (in file kanjidic).
- other-uncommon2 : Hyougaiji, that is only in file kanjd212.
- other-uncommon3 : Hyougaiji, that is only in file kanjidic2.xml and is not in files kanjidic and kanjd212. Most of these PDF-files are more or less empty, because fonts support those kanjis so badly.
- other-uncommon4 : Hyougaiji, that is only seen in Unihan database of Unicode. Most of these PDF-files are more or less empty, because fonts support those kanjis so badly.

Kanjis of groups G1–G6, G8–G10, other-common and other-uncommon1 are only taken from the file kanjidic but few kanjis of group G9 are taken from both kanjidic and kanjd212.

kanjd212 has some kanjis that belong to group G9 but the rest of that file is just Hyougaiji. As you can see, a directory called “G7” do not exist on

purpose. `kanjidic.xml` has same kanjis that two other files have, and many additional kanjis.

Unihan database has many characters that are not used in Japanese language at all. I took only those characters that have Japanese On-Yomi, Kun-Yomi or both.

Those dictionary-files also have Unicode-values of each kanji. Therefore it may be good idea to use some dictionary-software that uses those files as its dictionaries. There are many of them. Jim Breen's WWW-page called WWWJDIC is a good web-interface for all his dictionaries, including all his kanji-dictionaries:

http://www.csse.monash.edu.au/~jwb/cgi-bin/wwwjdic.cgi

Exact syntax of `kanjidic`, `kanjd212` and their fields is explained here:

- http://www.csse.monash.edu.au/~jwb/kanjidic_doc.html
- http://www.csse.monash.edu.au/~jwb/kanjd212_doc.html

Information about `kanjidic2.xml` is here:

http://www.csse.monash.edu.au/~jwb/kanjidic2/

See also this:

http://www.edrdg.org/

PDFs that show kanjis has all kind of information about each kanji, for example On-Yomi, Kun-Yomi, Nanori and strokecount. If some piece of information in PDF-file is extracted from Unihan, it is indicated with text "(Unihan)". In other case information is extracted from dictionaries of Jim Breen et. al. Some information fields are shown always, even when they are empty, but some of them are shown only if they have some value.

If meaning is taken from Unihan-database, please remember this information from documentation of Unihan:

> "Definitions are for modern written Chinese and are usually (but not always) the same as the definition in other Chinese dialects or non-Chinese languages. In some cases, synonyms are indicated."
>
> "Definitions specific to non-Chinese languages or Chinese dialects other than modern Mandarin are marked, e.g., (Cant.) or (J)"

"Major definitions are separated by semicolons, and minor definitions by commas. Any valid Unicode character (except for tab, double-quote, and any line break character) may be used within the definition field."

This have following implications:

- If meaning is taken from modern written Chinese, it can not be trusted, unless synonyms are indicated.
- If meaning has some valid Unicode character that is not included in that font that is used for writing that meaning, you just can not see it.

Here are explanations for some not-so-clear information fields:

- **Frequency:** "A frequency-of-use ranking. The 2,500 most-used characters have a ranking; those characters that lack this field are not ranked. The frequency is a number from 1 to 2,500 that expresses the relative frequency of occurrence of a character in modern Japanese. This is based on a survey in newspapers, so it is biassed towards kanji used in newspaper articles. The discrimination between the less frequently used kanji is not strong." That is all that documentation of `kanjidic2.xml` said. Unfortunately it did not tell, if bigger number means more frequent kanji or vice versa. Tough luck.
- **JLPT:** The level of the Japanese Language Proficiency Test (JLPT) in which the kanji occurs. (1-4). Note: This information is becoming outdated, because levels of JLPT will be expanded: There will be five levels instead of four.
- **SKIP-code:** See "Copyright and contact information".
- **Radicals (kradfile(2)):** radicals according to `kradfile` and `kradfile2`. It tells all radicals of each kanji.
- **Radical (KangXi Zidian):** Main radical according to KangXi Zidian.
- **Radical (Classic Nelson):** Main radical as used in the Nelson's "Modern Japanese-English Character Dictionary" (i.e. the Classic, not the New Nelson). This will only be used where Nelson reclassified the kanji.
- **Radical (Dai Kan-Wa Jiten) (Unihan):** Radical according to Dai Kan-Wa Jiten.

4 How to edit these PDFs

If you are not completely satisfied with these files, you need to edit some files and then run shell scripts in order to generate PDF-files. Those scripts have some environment variables, so you may need to adjust them carefully. All scripts are put to the root of working directory. Sub-directories called “`pdf`” , “`tex`” , “`pdf-tmp`” and “`tex-tmp`” are automatically created to working directory.

For all that editing you need some Mad LaTeX Skillz and some Shell-Fu, too. You also need many fonts installed in your operating system. I created all these files in Debian GNU/Linux, but it may be possible to do it in other Linux-distribution, UNIX-like OS, UNIX-compatible OS or UNIX, too. Who knows, maybe all this can be done under Cygwin, too.

4.1 What each script do

A script `fontsamples_all.sh` runs these scripts in this order:

- `doall_files.sh`
- `compiledocs.sh`
- `release_docs.sh`

After those scripts it runs `uploadweb.sh` and `uploadweb2.sh` so that updated files in my WWW-directories are uploaded to WWW-sites. I do not provide those uploading-scripts, because they are so specific to my system. Hence, you may need to comment them away or create your own.

A script called `doall_files.sh` downloads needed dictionaries with a script called `getdics.sh` and then runs a script called `doall_subgroup.sh` for each subgroup. `doall_subgroup.sh` runs three scripts for each subgroup:

- `compile_subgroup.sh`
- `compress_subgroup.sh`

- `release_subgroup.sh`

A script called `compile_subgroup.sh` runs script `createtexfile.sh` for each character that belong to that subgroup, but in case of kanjis it runs script `createtexfile_kanji.sh` , instead, and in case of kanjis only seen in Unihan database `createtexfile_kanji_unihan.sh` is used, instead. During that process LaTeX-source code file is created for that character and XeLaTeX is used for compiling that file.

`compress_subgroup.sh` compress that subgroup according to setting in that group's rc-file. `release_subgroup.sh` copies compressed files of subgroup to right WWW-directory according to setting in that group's rc-file. It also runs scripts `uploadweb.sh` and `uploadweb2.sh` so that new files are uploaded to WWW-site immediately after each subgroup is ready.

A script called `doallbutrelease_files.sh` is just like a script called `doall_files.sh` , but it runs a script called `doallbutrelease_subgroup.sh` instead of a script called `doall_subgroup.sh` for each subgroup. A script called `doallbutrelease_subgroup.sh` is just like `doall_subgroup.sh`, but it do not release files of that subgroup. It means that subgroup is not moved to WWW-directory and WWW-uploading is not done.

A script called "`compiledocs.sh`" compiles this documentation file from LaTeX-source code to PDF. `release_docs.sh` is used for copying all documentation and scripts to WWW-directories. It also compresses them.

`release_docs.sh` also use a Perl-script called `tree.pl` to create sitemap for my other WWW-site; unlike my main WWW-site it can not create directory listings automatically. `tree-template.html` is template for `tree.pl`. Original script is here:

http://www.danielnaber.de/tree/

A script called `fontsamples.rc.sh` has all those environment variables that other scripts need; you must carefully edit it or you will be taunted. A script called `checkworkdirs.sh` makes sanity checks in home directory and work directory. A script called `checkwwwdirs.sh` makes sanity checks in WWW directory. All those three scripts are included by many scripts.

That script called `compiledocs.sh` can be run while `doall_files.sh` is still running.

If you want to compile PDFs that belong only to one or few subgroups, for example just basic hiraganas, you can run `compile_subgroup.sh` so that it compiles only that subgroup and nothing else. For example all those basic hiraganas are handled this way:

```
./compile_subgroup.sh hiragana basic
```

Then it tries to include a file called `./rc.hiragana_basic.sh` . There are other files whose name start `rc.` for subgroups. Those file names should be self-explanatory and they correspond to the subdirectories in directories called `pdf` and `tex` . Order of subgroups is quite important, because files that show output of command called `du` may look weird, if you compiled each subgroup in some weird order. After that you can run `compress_subgroup.sh` and `release_subgroup.sh` for that subgroup. It is also possible to run that script called `doall_subgroup.sh` for just one subgroup this way:

```
./doall_subgroup.sh hiragana basic
```

A script called `cleanall.sh` is good, if you want to remove useless files before running any other scripts. It especially needed, if you have to kill whole compiling process violently. There is also a script called `purgeall.sh`; it is just like a script called `cleanall.sh` , but it do not remove so many files. A script called `purgedocs.sh` removes temporary files that were created during compiling this documentation file, but a script called `cleandocs.sh` removes also compiled docs.

When you run some scripts, they create file `*_starttime.txt` when they start and `*_endttime.txt` when they end. It shows time as seconds from the beginning of UNIX-epoch. With simple calculation you can count, how many seconds that script run. Then you may want to convert those seconds to days, hours and minutes.

If you do not want to release those files in WWW-directory, I have scripts for that purpose, too: A script called `doallbutrelease_files.sh` is just like `doall_files.sh`, but it do not move files to WWW-directory. A script called `doallbutrelease_subgroup.sh` is just like `doall_subgroup.sh`, but it do not move files to WWW-directory.

And did I mention that you need to carefully edit `fontsamples.rc.sh` ? If you fail to do that, you will be taunted. You have been warned.

In that directory called `progress` are many files that can be used for monitoring progress of compilation process of current group and subgroup. You can use a script called `progress_status.sh` for that monitoring. They have this information:

- `GROUP` : Group, for example “`kanji`”.
- `SUBGROUP` : Subgroup, for example “`G1`”.

- `N` : How far we have progressed in that subgroup.
- `M` : How many characters that group has.
- `g` : Character itself, for example “音”.
- `CURRUCS` : Unicode-value and Unicode-name of current character, for example ”`U+97F3 CJK UNIFIED IDEOGRAPH-97F3`”.

But those files in that directory called `progress` are useless, if you compile more than one subgroup simultaneously. In that case you’d better comment away some lines from a script called `compile_subgroup.sh` . You can follow progress of compilation process also with command called `ps` .

4.2 About shell environment

Your environment must be UTF-8-safe: You must have some locale that uses UTF-8. Shell-scripts use `bash`, because it has had good Unicode-support long time. If I remember correctly, bash-scripts really do not have any “bashisms” at all. Hence, you should be able to run those scripts with any Bourne Shell -compatible shell that is Unicode-safe. The latest versions of `zsh` have quite good support for UTF-8. If I remember correctly, the latest versions of AT&T Korn Shell have good Unicode-support, but at least in interactive use it is not seen.

That script called `compiledocs.sh` can be run under any Bourne Shell -compatible shell. It do not need Unicode-safe shell at all. I decided to use `dash` (“Debian Almquist Shell”), because it is POSIX-compatible and fast. If you do not have `dash`, you can use something else, for example `ash`, `bash` or `ksh`.

Shell scripts make some calls to `awk`, `head` and other such UNIX-utilities, but practically any UNIX have them, anyway. But you must be sure that you have a POSIX-utility called `printf`. It is builtin command in `bash`, `zsh` and `ksh`, but it is also available as a stand-alone program. I am not sure, what utilities must be Unicode-safe. At least GNU-implementations of them seems to work for me. Fortunately I do not try to use `tr` for nothing else but removing some characters, because its GNU-implementation is not Unicode-safe:

http://bugs.debian.org/431231

There is also so called “Heirloom Toolchest”, that is totally Unicode-safe. If everything else fails, just use those programs in that tool chest:

http://heirloom.sourceforge.net/

4.3 How to ensure each PDF take just one page

During editing and compiling you'd better ensure, that each character's PDF-file takes only one page. Just check out that file called `pdfpages.txt`

If it looks like this, everything is fine:

```
Pages:          1
```

But if it looks like this, it means some of those PDFs take two pages and therefore you must adjust font sizes, line spacing and marginals and then recompile them all:

```
Pages:          1
Pages:          2
```

File called `pdfpages.txt` is updated every time when compilation of each subgroup has been finished.

4.4 Programs used

I try to tell all this about each program:

- Name
- Homepage(s)
- Name of Debian package.

Names of Debian-packages are mentioned, but so called "`apt`-line" is not, because all those packages are already available in Debian GNU/Linux, at least as unstable package. I do not bother mention the most common Unix-utilities, like `awk` `sort`, `du` and so on.

First things you need are shells called "`bash`" and "`dash`":

- http://tiswww.case.edu/php/chet/bash/bashtop.html
- http://www.gnu.org/software/bash/
- `bash`
- http://gondor.apana.org.au/~herbert/dash/

- `dash`

You a need Python-program called "`unicode`":

- ftp://ftp.debian.org/debian/pool/main/u/unicode
- `unicode`

Both Jim Breen's kanji-dictionaries and Unihan database of Unicode are downloaded with a bomb-proof multi-protocol download-client called "`lftp`":

- ftp://ftp.monash.edu.au/pub/nihongo/kanjidic.gz
- ftp://ftp.monash.edu.au/pub/nihongo/kanjd212.gz
- http://www.csse.monash.edu.au/~jwb/kanjidic2/kanjidic2.xml.gz
- ftp://ftp.monash.edu.au/pub/nihongo/kradzip.zip
- ftp://ftp.unicode.org/Public/UNIDATA/Unihan.zip
- http://ftp.yar.ru/lftp/
- `lftp`

That way we have always the fresh version of those three kanji dictionaries and Unihan database. After downloading kanji dictionaries are uncompressed with `gunzip`, of course:

- http://www.gzip.org
- `gzip`

Unihan database is zipped. Therefore it is uncompressed with Info-ZIP's unzip:

- http://info-zip.org/
- `unzip`

First two dictionary-files are converted to UTF-8 with a command called `iconv`, but `kanjidic2.xml` is already in UTF-8 -format.

If those dictionary files are already in working directory, they are not downloaded again. If you know or suspect that there are newer versions available in FTP-site, just move or delete those files away from your working directory; When you will (re-)run some compiling script, it will download new dictionary files.

If you do not want to use `lftp`, you can use some other software, instead, for example `curl`. And if for some reason you can not or do not want to use `iconv`, you can use `recode`. I have provided ready commands as commented lines.

Unlike other dictionary-files `kanjidic2.xml` is in XML-format. Therefore we need `xml2` in order to parse it easily:

- http://ofb.net/~egnor/xml2/
- `xml2`

You'd better have a TEX-distribution called TEX Live, because it has almost all TEX-stuff you ever need, including XƎTEX and XƎLATEX. It is already available in Debian GNU/Linux.

http://www.tug.org/texlive/

When this documentation file is compiled from LATEX source code to PDF-file, it is then converted to PostScript with `pdftops`. It is a part of the library called "poppler":

- http://poppler.freedesktop.org/
- `poppler-utils`

I do not want to convert any other PDF-files to PostScript, because they take so much disk space, already.

In the end files are archived and/or compressed with "`7zip`", "`bzip2`" or with both "`tar`" and "`bzip2`". We try use tar-implementation called "`star`", if it is available. If it is not available, we try to find GNU Tar. If it is not available, we use whatever version of "`tar`" that we can find.

- http://cdrecord.berlios.de/old/private/star.html
- http://developer.berlios.de/projects/star
- ftp://ftp.berlios.de/pub/star/

- star
- http://www.gnu.org/software/tar/
- tar
- http://www.bzip.org/
- bzip2
- http://p7zip.sourceforge.net/
- p7zip
- p7zip-full

That "pdfinfo" belongs to package called "poppler-utils". Those commands sort and uniq belong to GNU coreutils, and they are also very common utilities in almost every UNIX.

A command called du is used so that GNU-extension "-h" is used. du-command of Heirloom Toolchest has it, too, but du-command of AT&T AST-tools do not. I do not know about other implementations of it. If you can not find du-command with that extension, just comment those impossible du-commands away from script called compile_subgroup.sh.

4.5 Fonts used

I try to tell all this about each font:

- Name
- Homepage(s)
- Name of Debian package.
- "apt-line" or line to be added to sources.list of apt.

If name of Debian-package is mentioned, but "apt-line" is not, it means that package is already available in Debian GNU/Linux, at least as unstable package.

Please, tell me, if some of those fonts are really not open-source or free in the sense of freedom. I want to use Debian Free Software Guidelines as a definition of freeness; it is basically the same thing as Open Source Definition.

If you know some other wonderful Japanese fonts I should use, please tell me about them. And please do not suggest any non-free fonts. Unfortunately some fonts are free, but they cause some problems; Such fonts are listed below.

First you need a font called “KanjiStrokeOrders”. It is in a Debian-package called `ttf-kanjistrokeorders` and its homepage is here:

- http://sites.google.com/site/nihilistorguk/

When compiling any of those PDF-files, you must have DejaVu -font family installed in your operating system:

- http://dejavu.sourceforge.net/
- `ttf-dejavu`
- `ttf-dejavu-core`
- `ttf-dejavu-extra`

4.5.1 Proportional Gothic

- “Droid Sans Fallback” and “Droid Sans Japanese”
  - http://en.wikipedia.org/wiki/Droid_(font) (Click that link saying: “Androids repository containing fonts in TrueType format”)
  - http://android.git.kernel.org/ (gitweb repository. You must go to this sub-directory: ”platform / frameworks / base.git / data / fonts /”)
- IPAPGothic
  - http://ossipedia.ipa.go.jp/ipafont/
  - `otf-ipafont`
- KonatuPlus Light
  - http://www.geocities.jp/ep3797/modified_fonts_01.html
- Konatu
  - http://www.masuseki.com/rnote.php?u=be/konatu.htm
  - `ttf-konatu`
- Kochi Gothic

  - http://sourceforge.jp/projects/efont/files/
  - http://openlab.ring.gr.jp/efont/
  - ttf-kochi-gothic

- “M+ 1c thin”, “M+ 1c light”, “M+ 1c regular”, “M+ 1c medium”, “M+ 1c bold”, “M+ 1c heavy”, “M+ 1c black”, “M+ 1p thin”, “M+ 1p light”, “M+ 1p regular”, “M+ 1p medium”, “M+ 1p bold”, “M+ 1p heavy”, “M+ 1p black”, “M+ 2c thin”, “M+ 2c light”, “M+ 2c regular”, “M+ 2c medium”, “M+ 2c bold”, “M+ 2c heavy”, “M+ 2c black”, “M+ 2p thin”, “M+ 2p light”, “M+ 2p regular”, “M+ 2p medium”, “M+ 2p bold”, “M+ 2p heavy” and “M+ 2p black” (These fonts are in otherwise alphabetical order, but in addition I have put them in weight-order).
  - http://mplus-fonts.sourceforge.jp/

- Mona
  - http://monafont.sourceforge.net/index-e.html
  - ttf-mona

- Sawarabi Gothic
  - http://sourceforge.jp/projects/sawarabi-fonts/
  - ttf-sawarabi-gothic

- Sazanami Gothic
  - http://sourceforge.jp/projects/efont/files/
  - http://openlab.ring.gr.jp/efont/
  - ttf-sazanami-gothic

- “Togoshi Gothic” and “Togoshi Mona Gothic”
  - http://sourceforge.jp/projects/togoshi-font/
  - ttf-togoshi-gothic
  - deb http://www.mithril-linux.org/~henrich/debian/package ./

- “Ume P Gothic”, “Ume P Gothic C4”, “Ume P Gothic C5”, ”Ume P Gothic O5“, “Ume P Gothic S4” , “Ume P Gothic S5”
  - http://ume-font.sourceforge.jp/
  - http://sourceforge.jp/projects/ume-font/wiki/FrontPage

  - http://sourceforge.jp/projects/ume-font/
  - ttf-umefont
- “UmePlus CLP Gothic” and “UmePlus P Gothic”
  - http://www.geocities.jp/ep3797/modified_fonts_01.html
  - ttf-umeplus
- VL PGothic
  - http://dicey.org/vlgothic/
  - ttf-vlgothic

4.5.2 Proportional Mincho

This time those fonts are not used in exact alphabetical order; Hanamin is moved to the end, because it lacks kanas. It has only kanjis.

- Dejima
  - http://code.google.com/p/dejima-fonts/
  - ttf-dejima-mincho
- IPAPMincho
  - See “IPAPGothic”
- Kochi Mincho
  - http://sourceforge.jp/projects/efont/files/
  - http://openlab.ring.gr.jp/efont/
  - ttf-kochi-mincho
- Sawarabi Mincho
  - See Sawarabi Gothic, but Debian-package do not exist, yet.
- Sazanami Mincho
  - http://sourceforge.jp/projects/efont/files/
  - http://openlab.ring.gr.jp/efont/
  - ttf-sazanami-mincho
- Togoshi Mincho

  - http://sourceforge.jp/projects/togoshi-font/
  - ttf-togoshi-mincho
  - deb http://www.mithril-linux.org/~henrich/debian/package ./

- “Ume P Mincho”
  - See “Ume P Gothic”

- HanaMin
  - http://fonts.jp/hanazono/
  - ttf-hanazono

4.5.3 Monospace (Gothic and Mincho)

First we use monospace gothic fonts, except a font called GNU Unifont. Then we use monospace mincho fonts. Then comes GNU Unifont. GNU Unifont is originally a bitmap-font, but it has been converted to TrueType-format so that each pixel is bunch of drawing commands that create filled square.

- “IPAGothic” and “IPAMincho”
  - See “IPAPGothic”

- KonatuPlus Mono Light
  - See “KonatuPlus Light”

- KonatuTohaba
  - See “Konatu”

- “M+ 1mn thin”, “M+ 1mn light”, “M+ 1mn regular”, “M+ 1mn medium”, “M+ 1mn bold”, “M+ 1m thin”, “M+ 1m light”, “M+ 1m regular”, “M+ 1m medium”, “M+ 1m bold”, “M+ 2m thin”, “M+ 2m light”, “M+ 2m regular”, “M+ 2m medium” and “M+ 2m bold” (These fonts are in otherwise alphabetical order, but in addition I have put them in weight-order).
  - See “M+ 1c thin” etc.

- Togoshi Mono
  - http://sourceforge.jp/projects/togoshi-font/
  - ttf-togoshi-gothic
  - deb http://www.mithril-linux.org/~henrich/debian/package ./

- “Ume Gothic”, “Ume Gothic C4”, “Ume Gothic C5”, “Ume Gothic O5”, “Ume Gothic S4”, “Ume Gothic S5”, “Ume UI Gothic”, “Ume UI Gothic O5”
  - See “Ume P Gothic”
  - “UmePlus CL Gothic” and “UmePlus Gothic”

    See “UmePlus CLP Gothic” and “UmePlus Gothic”
- VL Gothic
  - See “VL PGothic”
- Ume Mincho
  - See “Ume P Gothic”
- GNU Unifont
  - http://unifoundry.com/
  - `ttf-unifont`

4.5.4 Handwriting

- Choumei
  - See KanjiStrokeOrders (but Debian-package do not exist, yet.)
- “kiloji”, “kiloji - B”, “kiloji - D” and “kiloji - P”
  - http://www.ez0.net/distribution/font/kiloji/
  - `ttf-kiloji`
- “AoyagiKouzanFont2OTF”, “AoyagiKouzanFont2”, “AoyagiKouzanFont-Gyousyo”, “AoyagiKouzanFontSousyo2” and “AoyagiKouzanFontT”.
  - http://www.geocities.jp/ep3797/japanese_fonts.html
  - http://www7a.biglobe.ne.jp/~kouzan/
  - http://musashi.or.tv/aoyagikouzanfontt.htm
  - `ttf-aoyagi-kouzan-t`
- AoyagiSosekiFont2
  - http://musashi.or.tv/aoyagikouzanfontt.htm
  - `ttf-aoyagi-soseki`

- “KouzanBrushFontGyousyo”, “KouzanBrushFontSousyo” and “KouzanBrushFont”.
  - http://www.geocities.jp/ep3797/japanese_fonts.html
  - http://www7a.biglobe.ne.jp/~kouzan/
  - http://musashi.or.tv/kouzanmouhitufont.htm
  - `ttf-kouzan-mouhitsu`
- “YOzFont”, “YOzFont90”, “YOzFontA”, “YOzFontA90”, “YOzFontAF”, “YOzFontAF90”, “YOzFontC”, “YOzFontC90”, “YOzFontCF”, “YOzFontCF90”, “YOzFontE”, “YOzFontE90”, “YOzFontEF”, “YOzFontEF90”, “YOzFontF”, “YOzFontF90”, “YOzFontN” and “YOzFontN90”.
  - http://yozvox.web.infoseek.co.jp/
  - `otf-yozvox-yozfont`
  - `otf-yozvox-yozfont-antique`
  - `otf-yozvox-yozfont-cute`
  - `otf-yozvox-yozfont-edu`
  - `otf-yozvox-yozfont-new-kana`
  - `otf-yozvox-yozfont-standard-kana`

4.5.5 Fonts that can not be used

- “Misaki Gothic” and “Misaki Mincho”
  - Shows empty space
  - http://www.geocities.jp/littlimi/misaki.htm
  - `ttf-misaki`
- “Ume P Mincho S3” and “Ume Mincho S3”
  - Error message: Not loadable.
  - See “Ume P Gothic”
- “YOzFontNF” and “YOzFontNF90”
  - Error message: Not loadable.
  - See “YOzFont” etc.

5 Thanks

I like to thank authors of all those fonts, but especially I want to thank Tim Eyre for his KanjiStrokeOrders-font.

I like to thank Jim Breen for his `kanjidic`-file and other dictionary files. Because there are many other contributors of those files, I want to thank them, too.

I also like to thank all authors of those wonderful software programs and TEX -related macros, packages and other extensions, that make these files possible. They are just too numerous to mention. But Donald Knuth deserves special thanks, because without him TEX would not exist and without TEX all those TEX -related things would not exist; Donald Knuth started it all.

On the other hand, modern TEX-engine called XETEX is so wonderful invention that I just can not help mentioning it and its authors. So, thank you, Jonathan Kew.

If you actually use these files, it would be nice to know; Please, send me some E-Mail about it.