A printable version of JMdict

Vincent ZOONEKYND
<zoonek@math.jussieu.fr>

Abstract

We explain how to convert the JMdict Japanese–English dictionary (in XML) into a printable format (PostScript), using tools such as XSLT, DOM, Perl or LaTeX.

Introduction

The JMdict dictionary [1] is a Japanese dictionary, that contains translations in English (but also in French, in German, or in Russian, for some entries). It comes in the form of an XML file; in other words, an entry may look as follows (it is a simplified example).

```
<entry>
 <k_ele>
  <keb>車</keb>
 </k_ele>
 <r_ele>
  <reb>くるま</reb>
 </r_ele>
 <sense>
  <gloss>car</gloss>
  <gloss>vehicle</gloss>
  <gloss>wheel</gloss>
 </sense>
</entry>
```

The structure is very easy to understand: first comes the various writings (a single one, in this case), then the readings (again, a single one), then the translations. There can be further information, such as grammatical or usage notes; restrictions (not all readings/writings/translations correspond to each reading/writing/translation); or attributes indicating the language of a translation.

Pictorially, an entry may be seen as a tree (figure 1).

We shall see how to turn this dictionary into something printable. While printing the whole dictionary may not be a very good idea, one can adapt

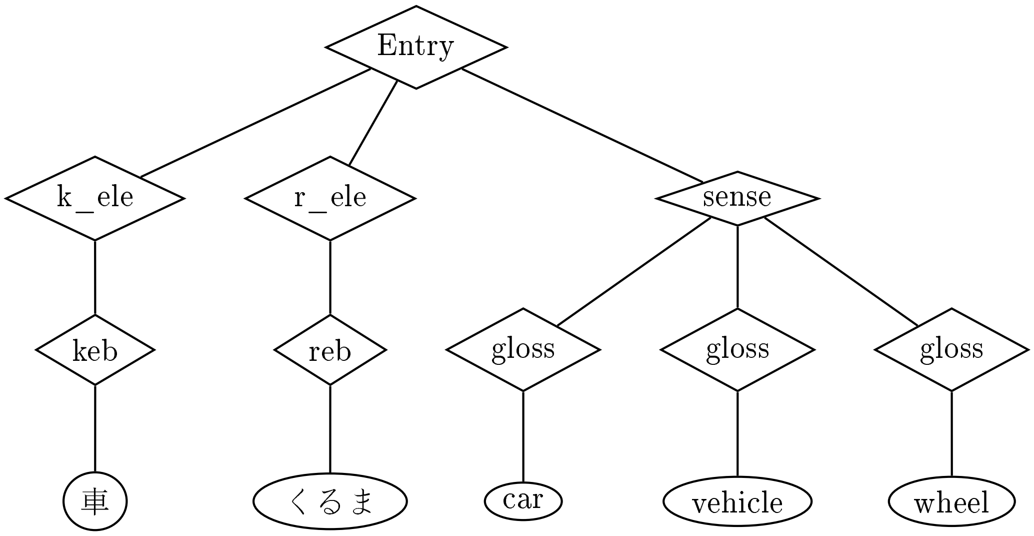

Figure 1: The tree representation of a small XML document

the methods presented here to extract a small part of it: it then becomes a vocabulary list (for instance, it could contain the wourds appearing in a text to be studied in a language class), whose printed version can have pedagogical value.

I have tried to make this article as little technical as possible: all the the details have been moved into the appendices. Before the appendices, you will find a sample page from the dictionary.

I whish to thank J.-M. Desperrier and M. Mangeot-Lerebours, who provided the initial problem, suggested that I write this article and made some constructive remarks.

1 Desired results

We now give a list of all the information we would like to see in the printable version, and how we would like it to appear.

- There should be several printable versions, with varying target language (Japanese–English, Japanese–French, etc.) ;
- There should be several versions, with more or less vocabulary (the JMdict format contains a field indicating the "priority" of a word) ;

- There should be a kana version and a romaji version ;
- The entries should be sorted (in kana or romaji order) ;
- It should look like a dictionary: the headers of each page should indicate the first and last word of the page, and, for readers not yet accustomed to kana order, it would be pleasant to have the list of all kana on the side of the page, with the current one highlighted ;
- The entries should reflect the information present in JMdict, namely:
  - Entries may have several writings

    かんずめ [缶詰, 缶詰め, 罐詰め*ok*] *n* Conserve ; boite de conserve.

  - Entries may have several pronunciations

    はいさら [灰皿] Voir はいざら.

    はいざら [灰皿] *n* Cendrier. Autre prononciation : はいさら.

  - These pronunciations may may have slightly different meanings or writings

    いちりょう [一両, 一輛] *n-adv* 1. One vehicle.
    2. [一両] One ryou (an old coin).

    おつ [乙] *adj-na n* 1. Strange; quaint; stylish; chic; spicy; queer; witty; tasty; romantic. 2. 2nd in a rank; second sign of the Chinese calendar. Alternate pronunciation: きのと.

    きのと [乙] *adj-na n* Strange; quaint; stylish; chic; spicy; queer; witty; tasty; romantic.

  - There are some cross-references

    アルバイト *de: n* Part-time job (esp. high school students) (de: Arbeit); See also パートタイム.

2 XML processing

There are several standardized technologies to manipulate XML documents; the main ones go by the acronyms XSLT (well-suited for simple manipulations, such as selection or reordering), SAX (the lowest level API, but the most efficient in terms of memory and speed – we won't use it) and DOM (well-suited for complex tasks). All of these are accessible from a variety of languages (C, Java, Perl, etc.): we have chosen Perl, because it is reasonably fast, helps reduce the time spent programming, and comes with a large number of modules (here, mainly XML::LibXML to process XML files and Unicode::String to process UTF8-encoded strings).

2.1 Using XSLT to select part of the dictionary

This is rather straightforward. For instance, to get the french entries, we ask XSLT to copy the entries that have a <gloss> tag with a lang="fr" attribute (and forget the others), and in these entries, to copy everything except <gloss> blocks without a lang="fr" attribute. The same goes for the selection of any language, or for the selection of entries with a given priority.

More precisely, as XSLT is merely a language and not a program, we feed such an XSLT stylesheet to an XSLT processor, such as xsltproc, Sablotron or Xalan.

We know have the subset of JMdict we whish to typeset.

2.2 Using DOM

As a single entry of JMdict can contain different writings and different readings, with slight sense variations, it may seem tricky to extract the correct information. But as DOM is called from a real programming language (in contrast with XSLT), it proves rather easy: we just read in the entries, one by one; for each entry, we find the various pronunciations; for each pronunciation, we find the corresponding writings and translations; and we finally print all these.

Now, we have an unordered dictionary, with the entries in the desired form.

2.3 Sorting

The japanese sorting rules are as follows. To compare two words, first compare them (according to the usual kana order, あいうえおかきくけこ...), without taking into account the size of the characters (や vs ゃ) or the presence of nigori (ば or ぱ vs は), and after replacing the katakana lengthening mark by the preceeding vowel. If this is not sufficient to distinguish the words, we compare the characters one by one, referring to the following order.

ー < ぁ < ァ < あ < ア
は < ハ < ば < バ < ぱ < パ
...

It may seem easy: most programming languages can sort text, according to the rules of a language (this is part of what is called "localization"). But actually, it doesn't work, because the localization libraries consider hiragana and katakana as completely diferent characters, and does not know about small kana.

My next idea was to try and write a comparison function: unsurprisingly, that proved to be too slow. The solution, known among Perl programmers as the Schwartzian Transform [10], consists in replacing the string to sort (with complicated rules) by another string with simpler sorting rules. More explicitely, we start by replacing each character by the closest "bare" hiragana and then by its rank (in hexadecimal) in the kana syllabary.

characters	bare hiragana	rank
あ ぁ ア ァ	あ	00
い ぃ イ ィ	い	01
ふ ぶ ぷ フ ブ プ	ふ	1E

Then, we add the information we just removed, viz., the rank of the character among those that yielded the same hiragana.

characters	subrank
ー	00
あ	01
ぁ	02
ア	03
ァ	04

characters	subrank
は	01
ハ	02
ば	03
バ	04
ぱ	05
パ	06

For instance, プライバシー yields "1E29041C0E04 060204040200".

3 LaTeX woes – problems and solutions

3.1 Omega

LaTeX is a text processor that can achieve a rather hight typographic quality and that can be used for batch processing. It can typeset Japanese text (together with other languages, such as English or French) with the help of the CJK package [8], but switching from Japanese to non-Japanese text at each entry fills its memory – it won't work with the whole dictionary. Another solution would be pLaTeX, a Japanese version of LaTeX, but it is almost unknown outside Japan and I do not know how it works with non asian writings. I have finally chosen Omega, a multilingual extension of TeX.[1] The slight problem was that almost no one knew how to typeset Japanese with Omega and there was a complete lack of documentation.

3.2 Fonts

There exists several free fonts with Japanese characters, such as Cyberbit [2] or Kochi [7]. These are True Type fonts, but Omega does not directly recognize them: they have to be transformed. As this is *very* technical, I have written a Perl script that does this transformation automatically, for any True Type Unicode font [15].

3.3 Japanese typography and OTPs

Contrary to common belief, Japenese text cannot be broken into lines anywhere: there are some characters before which (closing brackets or quotes, small kana, katakana lengthening mark) or after which (opening brackets or quotes) line breaking is forbidden. If we do not say anything to Omega, it won't break the lines at all (by default, line breaking is only allowed at spaces or at hyphenation points in English words).

Omega provides us with two means of inserting these line-breaking authorizations: either by defining macros, that are expanded until there is nothing to expand (as with LaTeX) or by considering the text as a stream, that can be processed as such, in "pipelines", either by an external program or by a program written in a dedicated language (this is called an OTP – Omega Translation Process).

We can thus write an OTP that says "insert a line-breaking authorization between any two characters, provided it is forbidden neither after the first nor before the second".

The Japenese line-breaking rules are actually more precise [9]. First, among the characters before or after which it is forbidden to break the line,

[1] Actually, it is not Omega but Omega + LaTeX, which is also known as Lambda.

this line break can be more or less strictly forbidden. Second, some punctuation characters (period, comma, closing quotes) are allowed to protrude in the right margin: it actually suffices to redefine the "line-breaking authorization" command to acchieve this.[2]

Actually, all this will only be needed when JMdict entries have sample sentences [14] – for we currently only have to typeset words, not sentences.

3.4 Headers

Typesetting a dictionary is (almost) standard LaTeX stuff [11]: we can ask LaTeX to put a *mark* (something that can contain any text, but won't be typeset) at each entry and then, when it typesets the header of a page, ask it to retrieve the first and last marks of the page – we can thus put the first and last words of the page on its header. The same goes for the first letter of the words (Perl will take care of converting it to a "bare" hiragana): we just have to put a 2-element list in the mark.

3.5 The text in the margin

We would like to have, in the margin, the list of all the kana, in the correct order, with the current kana highlighted.

One way of highlighting text is to use the PSTricks macros, that enable one to include PostScript commands invisible a LaTeX document – for instance, a grey circle under the character. To put the characters vertically, we do not write vertically (though we could) but we just put the characters in (insisible) boxes and we ask LaTeX to stack these boxes on top of each other. Finally, we write in the margin exactly as we wrote in the page headers.

Actually, with our font-size and page height, all the characters do not fit in the margin: instead, we only put the character in -A for each consonant (あ, か, さ, etc.) except the current character.

3.6 Other choices

We could have chosen other tools to typeset the dictionary.

When one wants to typeset an XML file, one immediately thinks of XSL-FO [3]. Though it is certainly possible to typeset a dictionary with it,

[2] It is also possible to do that at the font level, without OTPs [13].

it is not as versatile as LaTeX. Moreover, the style sheet has to be written in XML – it is a wonderful language for computers, but it is not really human friendly. Furthermore, it does not seem to reach LaTeX's typographic quality. For these reasons, we shall not look into XSL-FO for the moment.

Still in the TeX realm, we could have used Omega + ConTeXt + Metapost (instead of Omega + LaTeX + PSTricks) [4, 5]. ConTeXt is easier to configure than LaTeX, integrates easily with PDF (at least without Omega – but we need it), and the graphic capabilities of Metapost are more impressive than those of PSTricks. Yet, the software is rather new and its user-base rather small.

Conclusion

We have seen how to turn a dictionary from an XML format to something printable. In order to do this, we have used many different tools, each time selecting the most appropriate one: XSLT to select the part of the dictionary we want, DOM to transform the entries, LaTeX (Omega) to do the typesetting and Perl as a glue language and to do what couldn't be done easily with the other tools.

References

[1] J. Breen, *JMdict*, `http://www.csse.monash.edu.au/~jwb/j_jmdict.html`.
[2] *The Cyberbit font*, `ftp://ftp.netscape.com/pub/communicator/extras/fonts/windows`.
[3] *FOP*, `http://xml.apache.org/fop/`.
[4] H. Hagen, *ConTeXt*, `http://www.ntg.nl/context/document.htm`.
[5] ———, *The MetaFUN manual*, `http://www.ntg.nl/context/metapost.htm`, 2002.
[6] Y. Haralambous and J. Plaice, *Omega*, `http://omega.cse.unsw.edu.au:8080/index.html`.
[7] *The Kochi fonts*, `http://www.on.cs.keio.ac.jp/~yasu/jp_fonts.html`.
[8] W. Lemberg, *The CJK package for LaTeX*, `http://cjk.ffii.org/`.
[9] K. Lunde, *CJKV information processing*, O'Reilly, 1998.
[10] R. Schwartz, *Sorting*, Unix Review, January 1996.
[11] P. van Oostrum, *Fancyhdr: page layout in LaTeX*, `fancyhdr.dvi`, in most TeX distributions, 1998.
[12] T. van Zandt, *PSTricks*, `http://www.tug.org/applications/PSTricks/`.

[13] R. Wright, *Sticky characters for traditional chinese using an OVP for an Omega level 1 font*, `http://omega.cse.unsw.edu.au/pipermail/omega/2002-June/000408.html`, 2002.

[14] V. Zoonekynd, *Comment trouver des phrases d'exemple pour agrémenter un dictionnaire*, `http://www.math.jussieu.fr/~zoonek/UNIX/41_corpus/corpus.html`, June 2002.

[15] ———, *Installing Unicode True Type fonts under Omega*, `http://www.math.jussieu.fr/~zoonek/LaTeX/FontInstallUnicode/`, April 2002.

[16] ———, *OTP externes*, `http://www.math.jussieu.fr/~zoonek/UNIX/37_omega_OTP/otp.html`, May 2002.

[17] ———, *Typesetting Japanese with Omega*, `http://www.math.jussieu.fr/~zoonek/LaTeX/Omega-Japanese/doc.html`, June 2002.

[18] ———, *Version imprimable du dictionnaire JMdict (preliminary and technical version of this article, contains links to the resulting dictionaries)*, `http://www.math.jussieu.fr/~zoonek/LaTeX/Dictionnaire_japonais/main.htm%l`, May 2002.

[19] Zvon, *XSLT tutorial*, `http://www.zvon.org/xxl/XSLTreference/Output/index.html`.

いまけんしんしつ［居間兼寝室］Salon et chambre.

いましめる［戒める］*v1* Donner un conseil ; donner un avertissement.

いまわしい ［忌まわしい］ *adj* Abominable ; exécrable ; odieux ; répugnant.

いみ［意味］*n vs* Sens ; signification.

いみん［移民］*n* Émigré ; immigrant ; émigration ; immigration.

いむ［忌む］*v5m* Détester ; haïr ; éviter.

いも ［芋］ *n col* 1. Pomme de terre ; patate. 2. Lourdaud ; rustre.

いもの ［鋳物］ *n* Pièce de fonte ; pièce fondue ; pièce moulée.

いもほり［芋掘り］*n* Champ de pommes de terre.

いもん［慰問］*n* Condoléances.

いらい［依頼］*n vs* Demande ; requête.

いらい［以来］*n-adv n-t* Depuis ; dorénavant.

いりょう［医療］*n* Soins médicaux ; traitement médical.

いる［射る］*v1* Abattre ; tirer.

いれいさい［慰霊祭］*n* Service funèbre.

いれる ［入れる］ *v1* 1. Faire entrer ; introduire ; ajouter ; insérer. 2. Comprendre ; inclure. 3. Écouter ; tolérer.

いろ［色］*n* 1. Couleur ; teinte. 2. Amour ; sensualité.

いろうなく［遺漏なく］Complet ; sans omission.

いろか［色香］*n* Teinte et parfum ; beauté ; charme.

いろがみ［色紙］*n* Papier coloré.

いろどる ［彩る，色取る］ *v5r* Colorer ; peindre ; maquiller ; colorier.

いわ［岩，磐］*n* Roc ; rocher.

いわう［祝う］*v5u* Fêter ; célébrer.

いわのぼり［岩登り］*n* Escalade ; varappe.

いわや［岩屋］*n* Caverne ; grotte.

いん［員］*n n-suf* Membre.

いん［韻］*n* Rime.

いんが［因果］*adj-na n* Cause et effet ; destin.

いんかん［印鑑］*n* Tampon ; sceau.

いんき ［陰気］ *adj-na n* Sombre ; obscure ; mélancolique.

いんきょ［隠居］*n* 1. Retraite. 2. Retraité.

いんけい［陰茎］*n* Pénis.

いんご［隠語，韻語］*n* Langage secret ; langage codé ; jargon.

いんさつ［印刷］*n vs* Impression ; imprimer.

いんさつぶつ［印刷物］*n* Un imprimé.

いんし［印紙］*n* Tampon.

いんじゃ［隠者］*n* Ermite.

いんしょう［印象］*n* Impression.

いんしょく［飲食］*n* Manger et boire.

いんせい［陰性］*adj-na n* Négatif (attitude, réaction).

いんぜい［印税］*n* Royalties ; dividendes.

いんせき［引責］*n vs* Prendre la responsabilité de.

いんぞく［姻族］*n* Parents par alliance.

いんたい［引退］*n vs* Retraite.

いんとく［隠匿］*n* Dissimulation ; cacher.

いんとくしゃ ［隠匿者］ Dissimulateur ; personne cachée.

いんとくぶっし［隠匿物資］Matériel, biens dissimulés.

いんにく［印肉］*n* Tampon encreur pour sceau (ex : tampon au nom d'une société).

いんねん ［因縁］ *n* 1. Destin ; fatalité. 2. Origine ; cause ; rapport ; lien. 3. Prétexte.

いんぶん［韻文］*n* Vers ; poème.

いんぼう［陰謀］*n* Complot ; conspiration.

いんよう［引用］*n* Citation.

いんようく［引用句］Citation.

いんりつ［韻律］*n* Rythme ; mètre (en poésie).

いんりょう［飲料］*n* Boisson.

いんりょうすい ［飲料水］ *n* Eau potable ; eau (à boire).

いんりょく［引力］*n* Gravité.

う

うえ［上］*adj-no n-adv n n-suf* Dessus ; sur ; au-dessus ; supérieur ; autorité supérieure.

うえきばち［植木鉢］*n* Pot de fleur.

うえる［植える］*v1* Planter ; faire pousser.

うえる［飢える］*v1* Être affamé ; mourir de faim.

うおいちば［魚市場］*n* Marché aux poissons ; criée.

うおうさおう［右往左往］*n* Perdre le nord (figuré).

うかがう［伺う］*v5u vi vt hon* Visiter ; demander ; s'informer ; s'entendre dire ; implorer (une divinité pour une révélation).

うかぬかお ［浮かぬ顔］ Mine maussade ; air découragé ; faire la tête.

うかぶ［浮かぶ，浮ぶ*io*］*v5b* 1. Flotter ; faire surface. 2. Venir à l'esprit.

い か さ た な は ま や ら わ

A How to do it yourself

We explain how to install all the software required to process the dictionary. I assume you are on a Unix machine (for instance, Linux on a PC). In the sequel, a character % or > at the beginning of a line is a prompt (you do not have to type it).

A.1 Perl

I assume that Perl is already installed. If not, you may install it as follows (use the -Dprefix option only if you want to install it in a non standard location).

```
% ncftp ftp.lip6.fr
ncftp / > cd pub/perl/CPAN/src
ncftp /pub/perl/CPAN/src > get stable.tar.gz

% tar zxvf stable.tar.gz
% cd perl-5.6.1/
% sh Configure -Dprefix=$HOME/gnu/`uname` -Uinstallusrbinperl -des
% make
% make test && make install
% cd ../
```

Install the latest version of the libxml library (use the -prefix option only if you whish to install it in a nin-standard location).

```
% wget ftp://fr.rpmfind.net/pub/libxml/libxml2-2.4.23.tar.gz
% wget ftp://fr.rpmfind.net/pub/libxml/libxslt-1.0.19.tar.gz
% tar zxvf libxml2-2.4.23.tar.gz
% tar zxvf libxslt-1.0.19.tar.gz
% cd libxml2-2.4.23/
% ./configure --prefix=$HOME/gnu/`uname`
% make
% make install
% cd ../libxslt-1.0.19/
% make
% make install
% cd ../
```

Install the various Perl modules

```
% perl -MCPAN -e shell}
```

You have to answer many questions, but the only important one is the location of the nearest CPAN archives (the only thing you have to type is the figures, after the quare brackets).

```
First, pick a nearby continent and country (you can pick several of
each, separated by spaces, or none if you just want to keep your
existing selections). Then, you will be presented with a list of URLs
of CPAN mirrors in the countries you selected, along with previously
selected URLs. Select some of those URLs, or just keep the old list.
Finally, you will be prompted for any extra URLs -- file:, ftp:, or
http: -- that host a CPAN mirror.

(1) Africa
(2) Asia
(3) Central America
(4) Europe
(5) North America
(6) Oceania
(7) South America
Select your continent (or several nearby continents) [] 4

(1) Austria
(2) Belgium
(3) Bulgaria
(4) Croatia
(5) Czech Republic
(6) Denmark
(7) Estonia
(8) Finland
(9) France
(10) Germany
(11) Greece
...
Select your country (or several nearby countries) [] 9

(1) ftp://cpan.cict.fr/pub/CPAN/
(2) ftp://cpan.mirrors.easynet.fr/pub/ftp.cpan.org/
(3) ftp://ftp.club-internet.fr/pub/perl/CPAN/
(4) ftp://ftp.lip6.fr/pub/perl/CPAN/
(5) ftp://ftp.oleane.net/pub/mirrors/CPAN/
(6) ftp://ftp.pasteur.fr/pub/computing/CPAN/
(7) ftp://ftp.u-paris10.fr/perl/CPAN
(8) ftp://ftp.u-strasbg.fr/CPAN
(9) ftp://ftp.uvsq.fr/pub/perl/CPAN/
(10) ftp://mir1.ovh.net/ftp.cpan.org
```

```
Select as many URLs as you like [] 1 2 3 4 5 6 7 8 9 10

cpan> install Unicode::String
```

Unfortunately, it crashes (at least for me): it says that some *.gz files are not what they seem to be. Actually, these were compressed text files that were uncompressed but whose names have not been changed.

```
% wget http://www.math.jussieu.fr/~zoonek/LaTeX/Dictionnaire_japonais/rename.pl
% perl rename.pl 's/\.gz$//' `file ~/.cpan/**/*.gz | grep ASCII | sed 's/:.*//'`
% gzip ~/.cpan/**/*.(txt|data)
```

We can now resume the installation process.

```
% perl -MCPAN -e shell
cpan> install XML::LibXML
cpan> install Unicode::String
cpan> install XML::LibXSLT
```

As it doesn't work, we insist.

```
cpan> force install XML::LibXSLT
```

Actually, the following modules were automatically installed.

```
authors/id/G/GA/GAAS/MIME-Base64-2.12.tar.gz
authors/id/G/GA/GAAS/Unicode-String-2.06.tar.gz
authors/id/M/MS/MSERGEANT/XML-LibXSLT-1.50.tar.gz
authors/id/M/MS/MSERGEANT/XML-SAX-0.10.tar.gz
authors/id/P/PH/PHISH/XML-LibXML-1.52.tar.gz
authors/id/R/RB/RBERJON/XML-NamespaceSupport-1.07.tar.gz
authors/id/R/RB/RBS/File-Spec-0.82.tar.gz
authors/id/T/TJ/TJENNESS/File-Temp-0.12.tar.gz
```

You can read the documentation of these modules as follows.

```
% perldoc Unicode::String
```

A.2 Omega

I assume that LaTeX and Omega 1.15 are already installed. You can check that Omega is installed by typing

```
omega --version
```

If they aren't there, you may install from the TEXlive CD.

```
ftp://ftp.loria.fr/pub/ctan/systems/texlive/Images/texlive7-20020604-cd1.iso.bz2
```

There is a slight problem with ovp2ovf (a program, part of the Omega distribution, that is used to install fonts): it crashes. I have put an older version (a binary for Intel 80386) for that works on

```
http://www.math.jussieu.fr/~zoonek/LaTeX/FontInstallUnicode/ovp2ovf
```

We install some Unicode fonts with Japanese characters.

```
wget http://www.on.ics.keio.ac.jp/~yasu/linux/fonts/kochi-gothic-0.2.20020503.tar.bz2
wget http://www.on.ics.keio.ac.jp/~yasu/linux/fonts/kochi-mincho-0.2.20020503.tar.bz2
wget http://www.math.jussieu.fr/~zoonek/LaTeX/FontInstallUnicode/font_install_unicode.pl
wget ftp://ftp.netscape.com/pub/communicator/extras/fonts/windows/Cyberbit.ZIP

bzip2 -dc kochi-gothic*.tar.bz2 | tar xvf -
bzip2 -dc kochi-mincho*.tar.bz2 | tar xvf -
unzip Cyberbit.ZIP
mv Cyberbit.ttf cyberbit.ttf
mv kochi*/fonts/*.ttf .

perl font_install_unicode.pl
```

It takes a *long* time. Now, we can check that it actually works. (The cyberbit font is rather complete, but the kochi fonts do not contain accented characters.)

```
for i in 0_*/*.ps
do
  gv $i
done
```

You may wish to install the fonts by running `make install` in the `0_tmp_*/` directories. You may also specify a non standard installation directory (if you know what you do):

```
make install TEXMF=$HOME/gnu/`uname`/lib/texmf
```

A.3 The dictionary

We can now build the dictionary.

```
% cd 0_tmp_kochi-mincho/
% mv Makefile Makefile.old
% wget ftp://ftp.cc.monash.edu.au/pub/nihongo/JMdict.gz
% mv JMdict.gz JMdict.xml.gz
% wget -r -l1 -R gz,bz2 http://www.math.jussieu.fr/~zoonek/LaTeX/Dictionnaire_japonais/
% mv `find www.math.jussieu.fr -type f -print` ./
% rm -rf www.math.jussieu.fr/
% make
```

B XSLT style sheet

The following stylesheet extracts the French entries from JMdict.

```
<xsl:stylesheet version = '1.0'
   xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

 <xsl:output encoding="utf-8"/>

 <xsl:template match="/">
  <JMdict>
   <xsl:apply-templates select="*"/>
  </JMdict>
 </xsl:template>

 <xsl:template match="entry">
  <xsl:if test="sense/gloss[@g_lang='fr']">
   <entry>
    <xsl:apply-templates select="*"/>
   </entry>
  </xsl:if>
 </xsl:template>

 <xsl:template match="ent_seq|k_ele|r_ele|info">
  <xsl:copy-of select="."/>
 </xsl:template>

 <xsl:template match="sense">
  <sense>
   <xsl:apply-templates select="*"/>
  </sense>
```

```
  </xsl:template>

  <xsl:template match="gloss[@g_lang='fr']">
    <xsl:copy-of select="."/>
  </xsl:template>
  <xsl:template match="gloss">
  </xsl:template>


  <xsl:template match="stagk|stagr|gram|xref|ant|field|misc|example|s_inf">
    <xsl:copy-of select="."/>
  </xsl:template>

</xsl:stylesheet>
```

C DOM parser

Writing a parser with DOM amounts to using the following commands.

- getElementsByTagName: gives the list of child (or grandchild, etc.) nodes with a given name.
- getFirstChild: gives the first child node; premier noeud fils.
- getData: gives the contents of a text node.

Here is a simplified version of our parser (this is not the real code, which is slightly more complicated).

```
#! perl -w
use strict;

use XML::LibXML;
my $parser = new XML::DOM::Parser;
my $doc = $parser->parsefile ($ARGV[0]);
my $root = $doc->getDocumentElement;

# We loop over the entries
foreach my $entry (@{ $root->getElementsByTagName("entry") }) {

  # We get the list of all the writings
  my $keb = $entry->getElementsByTagName("k_ele");
  my @keb = map { $_->getElementsByTagName("keb")->[0]
```

```
          ->getFirstChild->getData }
        (@$keb);

# We get all the readings
my $r_ele = $entry->getElementsByTagName("r_ele");
foreach my $r (@$r_ele) {
 my $current_reading = $r->getElementsByTagName("reb")->[0]
            ->getFirstChild->getData;

 # Beware of the restrictions on the readings
 my $re_restr = $r->getElementsByTagName("re_restr");
 my @writings;
 if( $re_restr->getLength >0 ){
  @writings = map { $_->getFirstChild->getData } (@$re_restr);
 } else {
  @writings = @keb;
 }

 # We can now print pronunciations and readings
 if( scalar @writings > 0 ){
  print "\\reading{$current_reading}";
  print "\\writing{". join(', ', @writings) ."} ";
 } else {
  print "\\reading{$current_reading} ";
 }

 # We get the grammatical notes
 my $info = $entry->getElementsByTagName("info");
 foreach my $i (@$info) {
  # On oublie links, bibl, etym, audit
  my $r;
  $r = $i->getElementsByTagName("lang");
  push @rem, map {$_->getFirstChild->getData} (@$r);
  $r = $i->getElementsByTagName("dial");
  push @rem, map {$_->getFirstChild->getData} (@$r);
 }
 my $sense = $entry->getElementsByTagName("sense");
 my @rem;
 foreach my $s (@$sense) {
  my $r;
  $r = $s->getElementsByTagName("gram");
  push @rem, map {$_->getFirstChild->getData} (@$r);
  $r = $s->getElementsByTagName("field");
  push @rem, map {$_->getFirstChild->getData} (@$r);
  $r = $s->getElementsByTagName("misc");
```

```
    push @rem, map {$_->getFirstChild->getData} (@$r);
   }

   # We print these notes
   if( scalar @rem >0 ){
    print "  ". join(" ",
       map { "\\gram{$_}" } @rem
      ) ;
   }

   # We print the translations (taking care of the restrictions)
   foreach my $s (@$sense) {
    my $gloss = $s->getElementsByTagName("gloss");

    my $stagr = $s->getElementsByTagName("stagr");
    if($stagr->getLength >0){
       my $stay_here = FALSE;
       foreach my $res (@$stagr) {
  $stay_here = TRUE
   if $res->getFirstChild->getData eq $current_reading;
       }
       next unless $stay_here;
    }

    print '  \a ';

    my $stagk = $s->getElementsByTagName("stagk");
    if($stagk->getLength >0){
       print "\\stagk{". join(', ',
       map { $_->getFirstChild->getData } (@$stagk)
      ) ."} ";
    }

    my @glosses = map { $_->getFirstChild->getData } (@$gloss);
    print "\\defi{";
    print join('; ', @glosses);
    print ".} $newline";
   }
  }
}
```

D Sorting and locales

Locale is the name of a library that takes into account "localization", i.e., that modifies the behaviour of some function according to the country or the language (for instance, a letter is not the same thing in French or in Russian; different countries use different decimal separators; even for languages using the same writing system, sorting rules may differ, etc.).
For sorting, locales may be used as follows, in Perl,

```
use POSIX qw(locale_h);
...
{
 use locale;
 setlocale(LC_COLLATE, "ja_JP.utf8");
 ... sort ...
}
```

or in shell

```
LC_COLLATE=ja_JP.utf8 sort essai.utf8
```

According to your locale library it may or it may not work as it should (here, it doesn't).

E Lambda file

A UTF8-encoded Lambda file, using the Cyberbit font, could look as follows.

```
\ocp\TexUTF=inutf8
\InputTranslation currentfile \TexUTF

\documentclass[twoside]{article}

\DeclareFontFamily{T1}{cyber}{}
\DeclareFontShape{T1}{cyber}{m}{n}{<-> ombitstreamcyberbitroman}{}
\def\cyber{\fontfamily{cyber}\selectfont}
\begin{document}
This text is in English and uses the Computer Modern fonts.

\begin{cyber}
 これはサンプル文章です。
\end{cyber}
\end{document}
```

If the text were to contain sentences and not just words, we should take care of line-breaking problems, with a few OTPs (in the main text, I only mentionned the one allowing line breaks, but one should also take care of non-significant white spaces inside Japanese text). We would change the definition of the \cyber macro as follows.

```
\ocp\OCPjpspaces=JapaneseStripUnneededSpaces
\ocp\OCPjphyph=JapaneseLineBreaking
\ocplist\JapaneseOCP=
  \addbeforeocplist 1 \OCPjpspaces
  \addbeforeocplist 1 \OCPjphyph
\nullocplist
\def\cyber{%
  \fontfamily{cyber}\selectfont
  \pushocplist\JapaneseOCP
}
```

F Sample OTP

Here is an OTP implementing the a simplified version of the Japanese line-breaking rules.

```
% This says that input and output are 2-byte characters
input: 2;
output: 2;

% We first list the different character classes,
% merely by listing the Unicode codes of the characters
% they contain.
STRICTLY_FORBIDDEN_AFTER = (@"0028|@"005B|@"007B|@"2018|
  @"201C|@"3008|@"300A|@"300C|@"300E|@"3010|@"3014|
  @"3016|@"FF08|@"FF08|@"FF3B|@"FF5B|@"FF5B|@"FF62);
% I do not reproduce the definition of the other classes.
FORBIDDEN_AFTER = ( ... );
SLIGHTLY_FORBIDDEN_AFTER = ( ... );
STRICTLY_FORBIDDEN_BEFORE = ( ... );
FORBIDDEN_BEFORE = ( ... );
SLIGHTLY_FORBIDDEN_BEFORE = ( ... );
ASIAN = (...);

ANY = .;

% We then give the rules to be applied.
```

```
% Each time, the first line should match the entry
% (it it doesn't, Omega tries the next expression).
% If it does, it writes what follows the => sign
% on the output and leaves what follows the <= sign
% in the input stream.

expressions:

{ANY}{STRICTLY_FORBIDDEN_BEFORE}
  => #(\1) "\CJKunbreakablekernone "
  <= #(\2);
{STRICTLY_FORBIDDEN_AFTER}{ANY}
  => #(\1) "\CJKunbreakablekernone "
  <= #(\2);
{ANY}{FORBIDDEN_BEFORE}
  => #(\1) "\CJKunbreakablekerntwo "
  <= #(\2);
{FORBIDDEN_AFTER}{ANY}
  => #(\1) "\CJKunbreakablekerntwo "
  <= #(\2);
{ANY}{SLIGHTLY_FORBIDDEN_BEFORE}
  => #(\1) "\CJKunbreakablekernthree "
  <= #(\2);
{SLIGHTLY_FORBIDDEN_AFTER}{ANY}
  => #(\1) "\CJKunbreakablekernthree "
  <= #(\2);
{ASIAN}{ANY}
  => #(\1) "\CJKbreakablekern "
  <= #(\2);
{ANY}{ASIAN}
  => #(\1) "\CJKbreakablekern "
  <= #(\2);
```

The “line-breaking authorization” macros are defined as follows.

```
\def\CJKunbreakablekernone{%
  \nobreak
  \hskip 0sp plus 2sp minus 2sp
  \nobreak
}
\def\CJKunbreakablekerntwo{%
  \penalty 500
  \hskip 0sp plus 2sp minus 2sp
  \penalty 500
}
```

```
\def\CJKunbreakablekernthree{%
 \penalty 200
 \hskip 0sp plus 2sp minus 2sp
 \penalty 200
}
\def\CJKbreakablekern{\hskip 0sp plus 2pt minus 2sp}
```

For the characters that are allowed to protrude into the right margin, we use the followong macro.

```
\def\CJKprotrude#1{%
 \discretionary{\rlap{#1}}
         {}%
         {#1}%
}
```

G Marking commands in LaTeX

Contrary to what we said in the text, we do not put a 2-element list in the marks (which is what LaTeX actually uses), but a three-element list, containing the current word, the current “letter” and the complicated stuff to put in the margin.

```
\def\markthree#1#2#3{\gdef\@themark{{#1}{#2}{#3}}{%
  \let\protect\@unexpandable@protect
  \let\label\relax \let\index\relax \let\glossary\relax
  \mark{\@themark}}\if@nobreak\ifvmode\nobreak\fi\fi}
\long\def\@firstofthree#1#2#3{#1}
\long\def\@secondofthree#1#2#3{#2}
\long\def\@thirdofthree#1#2#3{#3}
\def\firstword{\expandafter\@firstofthree\firstmark{}{}{}}
\def\lastword{\expandafter\@firstofthree\botmark{}{}{}}
\def\firstkana{\expandafter\@secondofthree\firstmark{}{}{}}
\def\firstkanabox{\expandafter\@thirdofthree\firstmark{}{}{}}
```

We use it at the beginning of each entry.
We ask the fancyhdr package to put the words in the header in the following way (this simplified code, does not take into account what appears in the margin).

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

```
\fancyhf{}
\fancyhead[RO,LE]{%
 \begingroup
 \cyber
 \firstword\myemdash\lastword
 \endgroup
}
\fancyhead[LO,RE]{%
 \begingroup
 \cyber
 \firstkana
 \endgroup
}
```

H Sample entry

The resulting LaTeX file looks as follows.

```
\sortkey{0308322005 0301010303}%
\firstletter{あ}%
\markthree{あかんぼう}{あ}{\protect\kanabox{あ}{あ}}%
\reading{\pri{あかんぼう}}
\beginwritings
 \writing{赤ん坊}%
\endwritings
\gram{n}
\endgram
\defi{Bébé; nourrisson.}
\enddefi
```

I PSTricks code for the margin

Here is the code used to highlight a character.

```
\usepackage{pstricks,calc}
\newlength\kanawidth
\newlength\kanaheight
\def\verticalcircle#1{%
 \settowidth{\kanawidth}{#1}%
 \settoheight{\kanaheight}{#1}%
 \addtolength{\kanaheight}{\depthof{#1}}%
 \vbox to \kanaheight{%
```

```
    \vss
     \hbox to \kanawidth{%
       \hss
    \hbox{\pscirclebox[framesep=0pt,fillcolor=yellow,fillstyle=solid,linecolor=yellow]{#1}}%
       \hss
    }%
    \vss
  }%
}
```

Here is the code used to print the list of characters in the margin.

```
\def\kanaboxitem#1#2#3{%
  \hbox{\ifthenelse{\equal{#1}{#3}}{\verticalcircle{\textbf{#2}}}{#3}}%
  \vskip 3pt
}
\def\kanabox#1#2{%
  \vskip 2cm%
  \kanaboxitem{#1}{#2}{あ}%
  \vfill
  \kanaboxitem{#1}{#2}{か}%
  \vfill
  \kanaboxitem{#1}{#2}{さ}%
  \vfill
  ...
  \vfill
  \kanaboxitem{#1}{#2}{わ}%
  \removelastskip
  \vskip 2cm%
}
```