本稿は Linux Japan 誌 2000 年 8 月号に掲載された記事に補筆修正したものです.

Linux の辞書ツール

春に新 4 年生を迎えると，研究室では英語の論文や教科書を，担当を決めて順番に読み進める輪講が始まります．すると，学生は専門用語の知識が少ないので辞書をこまめにひかなければなりません．筆者も記憶力が減退してきたので，確認が必要でして，中学生から愛用の大修館のポケット英和，詳しく調べたいときには研究社の英和大辞典を開きます．ポケット英和はもうボロボロですが，大辞典は新品同様で，やはり重い辞書は敬遠していることがはっきりします．常時コンピュータに向かっているならば，ポケット英和よりもずっと簡便に字引を使うことができます．そこで今回は，電子辞書やそのネットワーク利用について現状を簡単に眺めてみます.

ところで，電子辞書に関しては 日本 Linux 協会のソフトウェア関連リンクページの『オフィスツール・編集・印刷』[1]W[3]に “辞書検索ツール” という項があって，大変役立ちます．紹介するツールの詳しいインストール方法や最新の情報はここから辿っていって確認してください.

辞書

最初に辞書 (つまりデータ) についてお話しします．市販の電子辞書は EPWING あるいは EXB/EXBA(電子ブック) などの規格に準拠したバイナリファイルとなっており，less など中身を見ることはできません．もちろん，辞書のための規格になっているので専用のツールを用いれば高速検索や内部リンクが活用できます．また，従来からある紙に印刷された辞書と同じ内容ですからデータの信頼性も高いですし，その量も多く，大体数 10 万の語句が収録されています.

一方，ネットワーク上で配付されている辞書は平テキストとなっており less や grep でそのまま利用できます．ただし，共通の規格がなく専用のツールに合わせたデータ構成 (フィールドや区切り) なので，見た目はてんでバラバラです (それでも平テキストなので awk や perl で相互変換は簡単)．こちらは個人で作成されている場合がほとんどで量的には限界があり収録語句は少ないです．それでも，英和辞書としては Nifty Serve の FWINF フォーラムのライブラリにある，Kurumi 氏が作成された GENE95[2]W[3] は収録語数約 5 万 3750 ですし，和英辞書としてオーストラリアの Monash 大学の Jim Breen 教授が作成された EDICT[3]W[3] は収録語数約 6 万 7200 もあります.

そんな中で，今回メインに扱う『英辞郎』[4]W[3] は，フリーではないのですが安価に入手できる辞書データでして，グループでメンテナンスされていることもあり 80 万強の収録語句を誇ります．80 万もの語句とその説明となるとファイルサイズが 50MB を越えます．ところが，Celeron 466 マシンですと，このファイルに grep をかけても数秒で終了してしまうのです．いやーハード様々ですね.

CUI

基本的には CUI が好みなので，その辺りの標準的なツールを紹介します.

look

コマンド look をご存知でしたか？ jman に依れば

```
名称
     look - 指定した文字列で始まる行を表示する
書式
     look [-df] [-t termchar] string [file]
```

となっていて，行頭に検索語を置き後ろに説明を付けてあるデータは，簡単な辞書として使えることになります．grep とどこが違うかというと，look はソートされたデータを想定しており，二分探索法 (binary search) を用いて検索速度を上げているのです．file 指定がないと /usr/share/dict/words(または/usr/dict/words) が使用されます．新しいバージョンではオプション -a で辞書ファイルを/usr/dict/web2 に切替えられます．早速お遊び，単語帳として使うために辞書を更新してみましょう．データ形式は行頭に登録語を置いておけば良いだけすからとても簡単です．例えば gene95 を使ってみましょう．解凍したばかりの gene95.txt は，SJIS/MS-DOS ですから，まず EUCJ/UNIX に変換します．色々方法がありますが，筆者は nkf と tr を使って

```
nkf -e gene95.txt | tr -d '\r' > gene95.euc
```

としています．改めて，less で内容を確認すると

```
abalone
【貝】(魚) アワビ
abandon
1. 断念する, 捨て去る, 見捨てる, 放棄する, 2. 奔放, 気まま
```

のように，語句と説明が 2 行に分かれています．それ

を 1 行に合成します．例えば awk なら

```
NR % 2 != 0 { printf "%s: ", $0 }
NR % 2 == 0 { print $0 }
```

という内容で genelook.awk を作成して，

```
awk -f genelook.awk < gene95.euc > word.gene95
```

とすれば良いでしょう．オリジナルの辞書 words は words.org に変更しておいてから両者をソートにかけ words に書き出します．

```
cat words.org words.gene95 | sort -df > words
```

sort のオプション -f (大文字小文字の区別を無くす) と -d (記号を無視する) は重要です．こうしておいて，look を実行してみると

```
$ look abalone
abalone: 【貝】(魚) アワビ
```

が素早く表示されます．

『英辞郎』のデータファイルも簡単に look 用に変換できます．eijirou.txt は

```
■ abalone :  【発音】ae'b ə lo'uni、【レベル】9
■ abalone 名 :  《貝》アワビ
■ Abalone Alliance :  アバローニ同盟
■ abalone steeped in sake and steamed :  アワビの酒蒸し {さかむし}
```

のように，行頭が記号 ■ で始まっています．この中塗四角の記号を取り除いて look に合わせるためのソートをかけましょう．

```
nkf -e eijirou.txt |sed 's/^■//' |
                    sort -fd > words.eijirou
```

後は words としてこのまま使うもよし，他の辞書とマージ+ソートしてもよしです．

grep

ご存知 grep は，パターンに一致した部分を含む行を表示するツールです．筆者自作のツールで使うオプション -w と -h を説明しておきます．-w はパターンマッチを単語単位で行う指定です．つまりパターンを abc と指定した場合，通常ならばそれを含む abcd wabc にもマッチするのですが，-w 指定すると abc が独立の単語となっている場合のみマッチすることになります．-h は，複数の検索ファイルを指定したときの表示を変化させます．通常，マッチした行の前に，ファイル名も表示されるのですが，これを抑制します．実例を示しましょう．

```
$ grep A-b /usr/dict/words /usr/dict/web2
/usr/dict/words:A-bomb: 《俗語》原子爆弾
/usr/dict/web2:A-bomb: 《俗語》原子爆弾
$ grep -h A-b /usr/dict/words /usr/dict/web2
A-bomb: 《俗語》原子爆弾
A-bomb: 《俗語》原子爆弾
```

xjdic

Monash 大学日本語のホームページでは，EDICT 用の xjdic というコンソールベースのネットワーク対応検索ツールが配付されています．とても軽快なので筆者は一目惚れしてしまいました．スタンドアロンで動く xjdic_sa，サーバー xjdserver，クライアント xjdic_cl などのバイナリは make 一発で作成できます．ただし，検索高速化のため，edict と (単) 漢字ファイル kanjidic の index ファイルを作成しなければなりませんし，いくつかの補助ファイルを/usr/local/share/xjdic に置かないといけません．作業ディレクトリやネットワーク越しに使う場合のサーバーホスト名などはドットファイル .xjdicrc で設定します．作業量は多めですが，辞書 xjdic23.install に従えば間違い無くインストールできます．図 1 に実行例を示します．語句 (英語，日本語どちらも可) の入力を促してきますからそれに答えるだけです．基本的に和英なので，英語を入力すると，語句の説明のフィールドを検索した結果を示してくれます．バージョン 2.3 では反転を使ってマッチした部分を強調するようになりましたが，五月蝿いと感じるならキーコマンド "}" で停止できます (トグル切替です)．他にも機能は盛り沢山ですから，キーコマンド "?" でヘルプを読んでみてください．外国人が日本語，特に漢字を勉強するために作られているのでおやと思う機能があったりして面白いですよ．

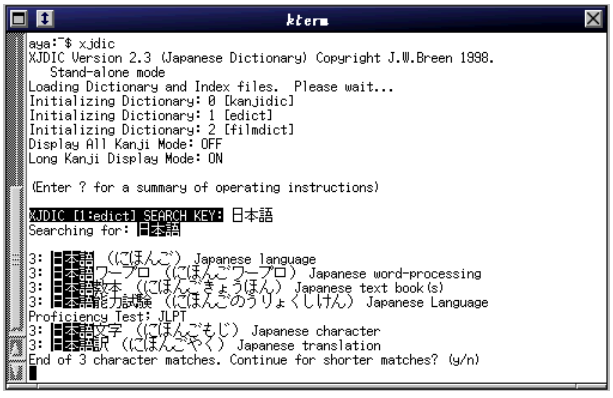

図 1　xjdic:EDICT 専用のコンソールベース検索ツール

Tcl/Tk による GUI ラッパー

さて，辞書に words.eijirou を指定して単語を look でひくと収録例が多くて、あっという間にスクロールしてしまう場合が多々あります．辞書の指定を一々するのも面倒ですし，ここは一つ GUI フロントエンドを簡単に作ってみましょう．使うツールキットは Tcl/Tk にします．ソース 1 を見てください．英和 (etoj)・和英 (jtoe)・単純検索 (search) という 3 つのモードを設けました．和英辞書には edict を指定しています (8 行)．16, 29, 49 行にある “`rsh $server`” は，速度 (CPU 及び HD アクセス) の速いホストで実行させることを目論だものです．6 行目がそのホスト指定です．localhost のままなら当然ローカルになります．11 行から始まる etoj では，look を起動してその結果を words に収めています．catch は外部コマンドがエラーを返した場合に Tcl/Tk が停止してしまうのを防ぐ常套手段です．look に

```
"\'$searchword \'"
```

を渡しているのは，grep のところで説明した -w のオプションのように単語単位の一致を実現するためです．図 2 に実行画面を示します．

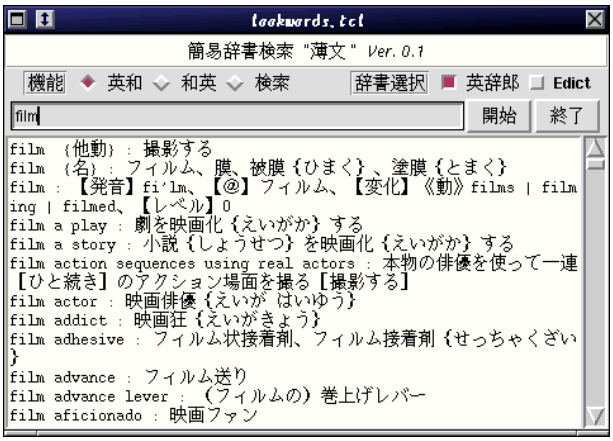

図 2　簡易辞書検索 “薄文” の実行例：英和モード

jtoe や search では単純に grep を用いています．特に search では，一致する行がかなりの数になる場合があるので，一致した部分をハイライト表示することにしました．65〜75 行の matches が文字列中の一致した場所を調べるプロシジャーです．59 行で matches を呼び出して tag 付けをし，61 行で tag 付けされた部分文字列の背景色を緑色にしています (図 3)．

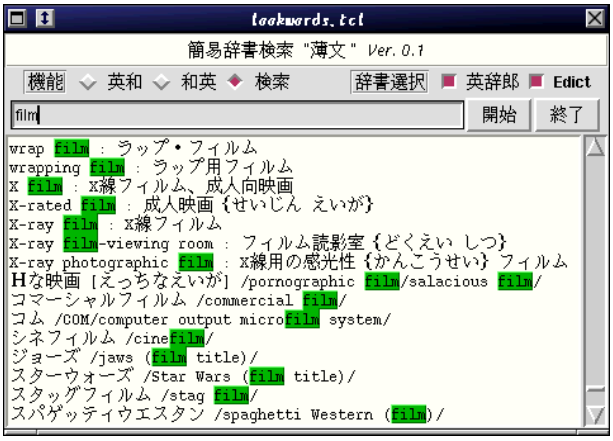

図 3　簡易辞書検索 “薄文” の実行例．単純検索モードでは一致文字をハイライトします．

標準規約電子辞書

look や grep あるいは xjdic は平テキストを操るのが大好きという筆者の偏った考えに基づいて書かずにはいられなかったのですが，現状はもっと電子辞書の機能を活用するツールが揃ってきています．その中でも，bookview/ndtpd は紹介せずにはいられない逸品です．なお bookview/ndtpd は VinePlus と Plamo2.0 には含まれています．

ndtp

NDTP(Network Dictionary Transfer Protocol) はその名の通り (電子) 辞書をネットワーク越しに使うためのプロトコルで，ndtpd は笠原基之氏 [5] によるサーバー実装例です．make 一発ですが，ndptd の設定ファイル

```
/usr/local/etc/ndtp.conf
```

を多少いじる必要があります．詳しくは，高林哲氏の解説記事 [6] W[3]をご覧ください．

bookview

ndtp の X クライアントである bookview[7]W[3] も笠原氏によって開発されました．起動してまず最初に Setup を行います．サーバー名とポート番号を入力すれば，すぐにサーバーと通信できます (スタンドアロンなら，サーバー名は localhost，ポート番号は 2010 とすれば良いはず)．メイン画面で Server: と Book: をプルダウンメニューから選択して準備完了します．Word:

に調べたい語句を入力して Enter するだけです．日本語入力は，Tcl/Tk ではデフォルトでは Ctrl-\ で起動しますが，設定によっては Ctrl-Space になっているかもしれません．従来の紙の辞書のような内容が表示

図 4 bookview の実行画面：辞書は三省堂の【新グローバル&ニューセンチュリー】

されます．すなわち，1 つの語句に対して発音や文法や複数の用例が整理されて一挙に示されるのです．

英辞郎の EPWING 化

bookview の紹介だけで終ってはつまりません．実はいろいろな辞書データを EPWING のサブセットである JIS X 4081 準拠のデータに変換するツール freepwing[8]W[3] が笠原氏によって開発されています．変換スクリプトも附属していて edict や gene95 を標準規約風電子辞書として使う道が開けたわけです，ただただ感謝 m(_ _)m.

ところで,『英辞郎』も変換して使ってみたいですね．残念ながら freepwing にはそのためのスクリプトが附属してません．そこで，sdic の援護を仰ぐことにしました．sdic は Emacs で辞書をひくためのツールで，土屋雅稔氏 [9] W[3]が開発・保守しています．その中に eijirou.txt を sdic 形式に変換する Perl スクリプト eijirou.perl があります．さらに，馬場肇 [10] 氏の手による，辞書データを sdic 形式から JIS X 4081 形式に変換する汎用 Perl スクリプト fpwsdic を活用しようというわけです．完全他力本願状態ですね．

具体的な手順ですが，それぞれのスクリプトに記された通りに実行するだけです．まずは，eijirou.txt から eijirou.sdic への変換：

```
nkf [-S] -e eijirou.txt |perl eijirou.perl >
eijirou.sdic
```

続いて，EPWING サブセットへの変換ですが，ちょっと手間がかかります．feijirou.sdic にはキーワードとして長すぎる文章が入っているため，途中でエラーになってしまうのです．そこで，次のような awk フィルター (cut.awk) を使って長すぎるものは無条件に削除しました (他にも良い方法があるに違いないのですが).

```
{ if (length($0) > 127) {
    flag = 0;
    n = split($0,keys,/\<H\>|\<K\>/);
    for (i=1; i<=n; i++) {
      if (match(keys[i],/\<\/K\>/) > 127)
         flag = 1;
    }
    if (flag == 0) print $0;
  } else { print $0 }
}
```

すなわち，

```
awk -f cut.awk < eijirou.sdic > eijirou_s.sdic
```

とした後，馬場氏の Perl スクリプトを実行します．

```
perl fpwsdic eijirou_s.sdic
```

この作業は Celeron 466 でも 1 時間弱かかりました．なお，Perl は バージョン 5.005 以上が必要です．この後，freewping.html にあるように，

```
perl /usr/local/libexec/freepwing/fpwsort
perl /usr/local/libexec/freepwing/fpwindex
perl /usr/local/libexec/freepwing/fpwcontrol
perl /usr/local/libexec/freepwing/fpwlink
```

を実行し honmon というファイルが出来上がれば成功です．あともう 1 つ作業があります catalogs という辞書の名前や形式や配置ディレクトリを記述するファイルを作成しないといけません．これは，太田純氏 [11] W[3]の開発された EPWING 支援ツール群 epwutil から catdump というコマンドを使います．catlogs.txt の内容を次のように適当に編集して

```
[Catalog]
FileName    = catalogs
Type        = EPWING1
Books       = 1

[Book]
Title       = "EIJIROU"
BookType    = 6001
Directory   = "eijirou"
```

catalogs ファイルを作成します．

```
catdump -u catalogs.txt catalogs
```

最後にファイルを移動して構成を

```
EIJIROU/catlogs
EIJIROU/eijirou/data/honmon
```

となるようにします．ntdp.conf に EIJIROU を登録して ndtpd を起動し直してやっと準備終了．英辞郎を bookview で使っている様子を図 5 に示します．うーん，圧倒的な語数の迫力が感じられなくなってしまいました．原因ははっきりしています．単語に一致する範囲が厳密になったからです．例えば “film” 自身と “film a play” 等の用例が違うキーワードとして収録されてしまうからなのです．厳密なのもいいのですが，筆者は辞書を “読む” のが好きなので，ちょっぴり寂しいです．しかし，応答は速く，これで良いかなと納得もしています．

図 5　bookview：辞書は『英辞郎』

ソース 1 lookwords.tcl

```
 1: #!/bin/sh
 2: # the next line restarts using wish \
 3: exec wish "$0" "$@"
 4:
 5: tk_setPalette background "gray85"
 6: set server "localhost"
 7: set ejdict "/mnt/DIC/eijirou/words.eijirou"
 8: set jedict  "/mnt/DIC/edict/edict"
 9: set words "見つかりませんでした．"
10:
11: proc etoj {} {
12:   global searchword dict words ejdict server
13:   if { "$searchword" == "" } then {
14:     set words "語句を指定してください"
15:   } else {
16:     catch {set words [eval exec rsh $server look -df \
17:                             "\'$searchword \'" $ejdict]}
18:   }
19:   .f2.m delete 1.0 end
20:   .f2.m insert 1.0 $words
21: }
22:
23: proc jtoe {} {
24:   global searchword dict words jedict server
25:   if { "$searchword" == "" } then {
26:     set words "語句を指定してください"
27:   } else {
28:     set fail [catch \
29:         {set words [eval exec rsh $server \
30:                 grep -h -w "\'$searchword \'" $jedict ]}]
31:     if { $fail == 1} then {set words \
32:                             "見つかりませんでした．"}
33:   }
34:   .f2.m delete 1.0 end
35:   .f2.m insert 1.0 $words
36: }
37:
38: proc search {} {
39:   global searchword dict words server
40:   set flag 0
41:   if { "$searchword" == "" } then {
42:     set words "語句を指定してください"
43:     set flag 1
44:   } elseif { "$dict" == " " } then {
45:     set words "辞書を指定してください"
46:     set flag 1
47:   } else {
48:     set fail [catch \
49:         {set words [eval exec rsh $server grep -h \
50:                               "\'$searchword \'" $dict ]}]
51:     if { $fail == 1} then {set words \
52:                             "見つかりませんでした．"}
53:   }
54:   .f2.m delete 1.0 end
55:   .f2.m insert 1.0 $words
56:   if { $flag == 1 } return
57:
58:   matches .f2.m $searchword {
59:     .f2.m tag add greening first last
60:   }
61:   .f2.m tag configure greening -background "#00b000" \
62:             -borderwidth 1 -relief raised
63: }
64:
65: proc matches {w pattern script} {
66:   set len [string length $pattern]
67:   set ptr 1
68:   scan [$w index end] %d numlines
69:   for {set i 1} {$i < $numlines} {incr i} {
70:     $w mark set last $i.0
71:     while { 1 } {
72:       set ptr [$w search $pattern last "last lineend"]
73:       if {$ptr == ""} then break
74:       $w mark set first $ptr
75:       $w mark set last "first +$len chars"
76:       uplevel $script
77:     }
78:   }
79: }
80:
81: proc detdict {} {
82:   global jedict ejdict d1 d2 dict
83:     if {$d1 == 1} then {
84:       set dict1 $ejdict
85:     } else {
86:       set dict1 ""
87:     }
88:     if {$d2 == 1} then {
89:       set dict2 $jedict
90:     } else {
91:       set dict2 ""
92:     }
93:   set dict "$dict1 $dict2"
94: }
95:
96: proc detmode {} {
97:   global func
98:   if {$func == "eiwa" } then {
99:     .f0.cb1 select
100:    .f0.cb2 deselect
101:  } elseif {$func == "waei" } then {
102:    .f0.cb1 deselect
103:    .f0.cb2 select
104:  }
105: }
106:
107: proc start {} {
108:   global func
109:   detmode
```

```
110 :   detdict
111 :   update
112 :   if {$func == "eiwa" } then {
113 :     etoj
114 :   } elseif {$func == "waei" } then {
115 :     jtoe
116 :   } else {
117 :     search
118 :   }
119 : }
120 :
121 : label .l -text "簡易辞書検索  \"薄文 \"  Ver. 0.1" \
122 :         -relief groove -bg "ivory"
123 : frame .f0
124 : label .f0.l1 -text "機能" -relief groove
125 : radiobutton .f0.r1 -text "英和" -variable func \
126 :                    -value "eiwa" -command {detmode}
127 : radiobutton .f0.r2 -text "和英" -variable func \
128 :                    -value "waei" -command {detmode}
129 : radiobutton .f0.r3 -text "検索" -variable func \
130 :                    -value "kensaku"
131 : label .f0.space -width 4
132 : label .f0.l2 -text "辞書選択" -relief groove
133 : checkbutton .f0.cb1 -text "英辞郎" -variable d1
134 : checkbutton .f0.cb2 -text "Edict" -variable d2
135 :
136 : pack .f0.l1 -side left -padx 6
137 : pack .f0.r1 .f0.r2 .f0.r3 .f0.space -side left
138 : pack .f0.l2 -side left -padx 6
139 : pack .f0.cb1 .f0.cb2 -side left
140 :
141 : frame .f1
142 : button .f1.j -text "開始" -command "start"
143 : button .f1.q -text "終了" -command "exit"
144 : entry .f1.e -width 50 -textvariable searchword
145 : pack .f1.e .f1.j .f1.q -side left
146 :
147 : frame .f2
148 : text .f2.m -width 64 -height 15  -relief sunken \
149 :            -yscrollcommand ".f2.y set" -bg "ivory"
150 : scrollbar .f2.y -command ".f2.m yview"
151 : pack .f2.m .f2.y -side left -fill y
152 :
153 : pack .l  -ipady 4 -fill x -expand yes
154 : pack .f0 .f1 .f2
155 : .f0.cb1 select
156 : .f0.r1 select
157 : bind .f1.e <Key-Return> {.f1.j flash; .f1.j invoke}
```

参考文献

[1] 日本 Linux 協会の Linux 関連リンク『オフィスツール・編集・印刷』のページ． W3
http://www.linux.or.jp/link/editor.html

[2] gene95 は下記から入手できます． W3
http://www.namazu.org/~tsuchiya/sdic/data/gene.html

[3] edict は namazu.org の SDIC のページから入手できます． W3
オリジナル『ジム・ブリーンの日本ページ』は，下記 URL です． W3
http://www.rdt.monash.edu.au/~jwb/japanese.html

[4] 『英辞郎』CD-ROM はネット申込により購入できます． W3
http://www.nifty.ne.jp/eijiro/

[5] NDTPD の公式サイト． W3
http://www.sra.co.jp/people/m-kasahr/ndtpd/index-ja.html

[6] Software Design 1999 年 9 月号掲載記事に加筆修正したもの． W3
http://openlab.ring.gr.jp/edict/unix/

[7] BookView の公式サイト． W3
http://www.sra.co.jp/people/m-kasahr/bookview/index-ja.html

[8] FreePWING の公式サイト． W3
http://www.sra.co.jp/people/m-kasahr/freepwing/

[9] SDIC のホームページ． W3
http://www.namazu.org/~tsuchiya/sdic/index.html

[10] 筆者は知らなかったのですが，同じ事を考える人はいるもので，この原稿の執筆時には既に fpweijiro が 千原さんにより作成されていたようです．調査不足を反省しています (^^;． W3
http://www02.so-net.ne.jp/~chihara/fpweijiro/

[11] EPWUTIL の FTP 配布サイト． W3
http://openlab.ring.gr.jp/edict/epwutil/