Logiciel FURCH, guide détaillé de l'utilisateur.

Version 2.011

1. Table

Guide détaillé, introduction.

FURCH est un logiciel pour afficher et analyser des textes. Si vous désirez des informations sur un mot, vous pointerez et cliquerez simplement sur ce mot. Le logiciel est capable de chercher dans des dictionnaires. Si le mot est trouvé, FURCH affichera l'information. FURCH est une application graphique, il utilise des fenêtres, et de façon plus générale des windgets, qui sont ces sortes de « window gadgets » (attributs de fenêtres) que l'on est maintenant habitué à voir dans les applications graphiques et les interfaces utilisateurs. FURCH est fondé sur GTK2 et est développé sous Linux. Cependant, le logiciel a été porté sur Win 98 SE, Win 2000 et Win XP.

Le logiciel FURCH a dans son état actuel deux fichiers de configuration. Le premier est « furch.conf » et le second est « gtkb.rc » . Le second fichier est cependant plus un fichier de configuration de GTK qu'un fichier FURCH. Les deux fichiers doivent être dans le même dossier que l'exécutable et ils sont utilisés pour ajuster les tailles, les couleurs et d'autres caractéristiques des fenêtres et des boutons. Ils sont aussi utilisés pour dire au logiciel quels textes lire et quels dictionnaires sont disponibles. Les deux fichiers seront analysés ci-dessous. Nous devons aussi donner des détails sur les deux normes de dictionnaire actuellement utilisées par FURCH. Des informations sur l'installation du logiciel FURCH dans Linux et les Windows OSs ont aussi été ajoutés.

Les fichiers textes.

Le logiciel FURCH ne peut analyser que des fichiers écrits en iso-latin1, c'est à dire des textes ordinaires non codés. Chaque fichier est considéré comme étant un livre comportant de nombreuses pages ou même parfois une seule. Ces fichiers peuvent contenir des balises spéciales pour indiquer les séparations en pages et aussi les entêtes et pieds de pages. Cependant cela n'est pas une obligation et s'il n'y a pas de balise, le logiciel fera de son mieux pour afficher le fichier en le découpant en pages et paragraphes lisibles. Toutefois, le fichier d'origine restera inchangé.

Vous pouvez afficher des textes générés par un traitement de texte tel que MS-Word ou OpenOffice qui contiennent de très longues lignes, aussi bien que des textes provenant de simples éditeurs tels que Notepad ou Vim. Ces derniers ont des lignes

de longueur plus raisonnables. Nous allons voir maintenant le cas des fichiers contenant des textes à balises.

Fichiers avec textes à balises.

Le fichier peut contenir des balises qui indiqueront au logiciel comment formater le texte. En son état actuel, le logiciel ne comprend que 8 balises. Ces balises sont des lignes commençant par : .NPO , .NHO , .NTO , .NFO ou .NP , .NH , .NT , .NF

Fichiers textes avec un format strict.

Vous pouvez désirer afficher sur votre ordinateur des pages d'un livre de façon à avoir sur votre écran quelque chose d'aussi proche que possible de l'original. Vous devez alors utiliser les balises .NPO , .NHO , .NTO , .NFO. La lettre « O » est mise ici pour « Original », « .NPO » signifie New Page Original Format, « .NHO » signifie New Header Original Format, « .NTO » signifie New Text Original et « .NFO » a été mis ici pour New Footer Original Format. Les deux ordres « .NPO » et « .NHO » induisent un saut à la page suivante. Ainsi, lorsqu'il y a une entête de page, vous avez seulement besoin de l'ordre « .NHO » et vous pouvez omettre « .NPO ». Avec ces ordres, la longueur de vos lignes sera la même sur votre écran et dans votre fichier.

Fichiers textes pré-découpés en pages.

Vous pouvez désirer afficher sur votre ordinateur des fichiers déjà découpés en pages mais de telle manière que les lignes sur votre écran auront au plus un nombre déterminé de caractères. Pour cela vous devez utiliser les balises .NP , .NH , .NT ,.NF. L'ordre « .NP » signifie New Page, « .NH » signifie New Header, « .NT » signifie New Text et « .NF » a été mis ici pour New Footer . Les deux ordres « .NP » et « .NH » induisent un saut à la page suivante. Ainsi, lorsque la page commence par une entête, vous aurez seulement besoin de l'ordre « .NH » et vous pouvez omettre « .NP ». Avec ces ordres, la longueur de vos lignes sera proche de la longueur indiquée dans votre fichier « furch.conf ». Avec ce format, vous devez indiquer le commencement ou la fin d'un paragraphe par une ligne blanche.

Le fichier « furch.conf ».

Le fichier « furch.conf » est en fait un fichier script qui utilise son propre langage. Ne paniquez pas, tout est expliqué par des commentaires dans le fichier. Un commentaire est une ligne commençant par le signe « # », ces lignes sont simplement ignorées par le logiciel. Le langage de script est composé d'ordres tels que : **Books, Which_dico, Highlight** etc. Généralement, chaque ordre a des attributs dont la liste est insérée entre les signes « **:>** » et « **<:** ».

L'ordre : Books.

Commençons par l'ordre **Books**. Cet ordre est utilisé pour dire au logiciel quels fichiers il doit analyser. Chaque fichier est considéré comme un livre comportant beaucoup de pages ou peut-être une seule. Le fichier ne doit contenir que du texte en iso-latin1. Lorsque le texte n'est pas formaté, le logiciel l'affichera au mieux. En tout cas, chaque page affichée aura moins de 100 lignes et mille mots. Vous devez aussi dire au logiciel où il peut trouver ces fichiers, ainsi vous devez lui donner le chemin absolu ou relatif de chaque fichier. Comme vous pouvez avoir une collection de livres, le logiciel comprend des notations telles que file5->15, qui lui disent qu'il doit prendre en compte les fichiers file5, file6, ..., file14, file15. Quoiqu'il ne soit pas nécessaire de leur donner les extensions « .txt », chaque fichier doit être un fichier texte. Voici un exemple extrait d'un fichier « furch.conf »

```
#  Liste de livres à lire (chaque fichier est
#  un livre qui peut contenir beaucoup de pages.
#  Vous pouvez utiliser le signe -> pour simplifier
#  vos entrées.
#   exemple : qr_4->6.txt signifie :
#             qr_4.txt  qr_5.txt qr_6.txt
#
Books :>
#  essai_h.txt
   ./gour.txt
   ./kentskrid-1.txt
   ./kentskrid.txt
   ./sarmoniou1-quere.txt
   ./Bremaik-09-juillet-2007.txt
   ./ALES/ales-1->14.txt
  <:
#
```

Dans l'exemple ci-dessus, le fichier « essai_h.txt » sera ignoré. La première page à être analysée sera « gour.txt ». C'est en fait une page isolée considérée comme un livre à une seule page. Elle est dans le même dossier que l'exécutable de FURCH. Il y a ensuite d'autres livres. Dans ce dossier il y a aussi un autre dossier dont le nom est «ALES» qui contient quatorze fichiers de « ales-1.txt » à « ales-14.txt ». Ce sont en fait des fichiers d'une seule page qui étaient utilisés avec les versions précédentes de FURCH. Ils seront considérés comme des livres à une seule page et affichés

comme d'habitude.

Remarque 1 : Si le fichier n'existe pas ou si nous avons donné un mauvais emplacement, le fichier sera ignoré.

L'ordre : Which_dico.

Une des caractéristiques les plus importantes du logiciel FURCH est qu'il peut gérer des dictionnaires multiples. Vous devez lui dire quels dictionnaires utiliser et où les trouver. Cela est fait comme pour les fichiers textes mais cette fois l'ordre est « Which_dico ». Voici un exemple :

```
Which_dico :>
   ./dico1->3_quere
    ./roll1_kltg
    ./splf1_ex
  <:
```

Dans cet exemple nous demandons les dictionnaires « dico1_quere », « dico2_quere », « dico3_quere », « roll1_kltg » et « splf1_ex ». Ils sont situés dans le même dossier que l'exécutable de FURCH. Les dictionnaires « dico_quere » et « roll1_kltg » sont des dictionnaires en norme intégrale (norme ordinaire). Le dictionnaire « splf1_ex » est un dictionnaire en norme simplifiée. Rien ne distingue ces fichiers dans l'ordre « Which_dico »

Remarque 1 : Vous pouvez utiliser la notation « **->** ».

Remarque 2 : Le dictionnaire « dico3_quere » n'a pas encore été écrit. Il sera simplement ignoré.

Remarque 3 : L'ordre de la liste « Which_dico » est important. C'est en fait votre liste de priorité pour les dictionnaires. Lorsque vous cherchez un mot, les dictionnaires où ce mot peuvent être trouvés sont classés selon cette liste. Parmi ces dictionnaires, le contenu de celui qui a le meilleur rang dans la liste de priorité sera affiché. Vous avez cependant accès à toutes les informations dans tous les dictionnaires où ce mot a été trouvé. Ceci a déjà été expliqué lors de la présentation du logiciel.

L'ordre : Priority_search_dico.

```
Priority_search_dico :>  i  <:
```

Vous devez remplacer **i** par 0 ou 1. Si vous mettez 0, alors l'ordre de priorité pour vos dictionnaires est l'ordre que vous avez implicitement défini avec votre liste « **Which_dico** ». Si vous remplacez **i** par 1 , la priorité est le poids, cela signifie que le dictionnaire avec le plus d'information est à la meilleure place. La priorité peut varier d'un mot à l'autre, le même dictionnaire n'a pas toujours le plus d'information pour chaque mot. Ici aussi, l'information présente dans n'importe quel dictionnaire peut être atteinte.

L'ordre : Order_dico.

Example :

```
Order_dico :>
       F        B        E
<:
```

Vous pouvez choisir l'ordre d'affichage des dictionnaires dans la fenêtre dictionnaire. Par exemple, j'ai dans toutes les figures choisi d'afficher la partie bretonne au milieu avec le français à sa gauche et l'anglais à sa droite. C'est plus facile pour la correction des dictionnaires. Vous pouvez modifier ce choix en changeant les arguments de l'ordre « **Order_dico** ». Par exemple vous pouvez mettre le breton en premier , ensuite l'anglais et enfin le français en donnant les arguments :

```
   :>    B        E        F      <:
```

À présent, il y a seulement 3 langues dans le logiciel FURCH et ces langues doivent être le breton, le français et l'anglais. C'est une sérieuse limitation, j'espère que cela sera amélioré dans une future version.

L'ordre : Level_dico.

Par exemple :

```
Level_dico :>     3    3     3    <:
```

Comme il a déjà été dit, la norme ordinaire des dictionnaires possède des niveaux d'affichage. Vous pouvez sélectionner le niveau maximum d'affichage si vous le désirez. L'ordre «Level_dico» vous permet d'ajuster le volume des détails écrits dans la fenêtre dictionnaire pour chacune des trois langues.

Le niveau -1 supprimera simplement la sortie pour la langue donnée.

Le niveau 0 donne seulement le plus important. Le mot, sa transcription en KLTG pour la partie bretonne, la nature grammaticale du mot. Pour les parties en français et en anglais, vous avez la traduction du mot breton.
Le niveau 1 explique la signification du mot.
Le niveau 2 donne plus d'explications, généralement grammaticales.
Le niveau 3 existe quelquefois et c'est souvent plus du bavardage que des choses utiles.

Vous pouvez donner un niveau de sortie pour chaque langue. Donc, vous devez donner 3 nombres, mais dans quel ordre ? Bien sûr, dans l'ordre que vous avez déjà défini par l'ordre « **Order_dico** ». Vous n'avez pas obligatoirement le même niveau de sortie pour chaque langue. Vous pouvez par exemple choisir de montrer le breton complètement et de ne rien afficher pour les autre langues, vous écrirez alors :

```
Order_dico :>    B        E        F     <:
Level_dico :>    3    -1      -1    <:
```

Remarque : Le niveau d'affichage n'a pas de sens pour un dictionnaire en norme simplifiée. Il sera simplement ignoré.

L'ordre : Highlight.

Lorsque vous affichez un document HTML ou un texte avec un traitement de texte, il est souvent utile de mettre en valeur des mot particuliers ou des expressions comportant plusieurs mots. Cela serait bien de pouvoir faire de même dans la partie livre. L'ordre « **Highlight** » nous donne cette possibilité. Cependant, les mots varient, un nom peut être au singulier ou au pluriel, un verbe est conjugué. FURCH fera de son mieux pour tenir compte de toutes ces variations. Je suis conscient que FURCH ne fait pas à présent un travail parfait. Soyez patient.

Comment utiliser l'ordre « **Highlight** » ? Prenons un exemple, vous désirez mettre en évidence dans vos textes le mot « pleg » (plis), le verbe « kozeal » (parler) et l'expression composée de deux mots : « en em » (le pronom réfléchi). Vous écrirez donc :

```
Highlight :>
     pleg
     kozeal
     koze
     en em
          <:
```

Vous ne devez mettre qu'un mot ou une expression par ligne autrement le logiciel prendra ce que vous avez écrit pour une expression composée et tentera de reconnaître l'expression complète dans vos textes.
Avec ce qui est écrit ci-dessus, le logiciel reconnaîtra le mot « pleg », le pluriel « plegoù » et le mot avec une mutation « daou bleg » (deux plis). Cependant, le participe passé « pleget » (plié, courbé) et les conjugaisons comme « plegomp » ([nous] plions, [nous] courbons) seront aussi reconnus. C'est parce que « pleg » est aussi le radical du verbe « plegañ » (plier, courber). Vous pouvez cependant ne pas désirer avoir les conjugaisons du verbe et vous pouvez l'éviter en spécifiant que le « pleg » mis en valeur doit être un nom. Cela se fait en écrivant « (ag) » sur la même ligne : « pleg (ag) ». Les lettres « (ag) » signifient « anv gourel » (nom masculin). Tous les ordres disponibles comme (ag) peuvent être trouvés, par exemple, au commencement du dictionnaire « dico1_quere ». La possibilité de préciser la valeur grammaticale des mots n'a cependant été que peu testée.
Continuons avec notre liste de mots mis en valeur. Nous voulons le verbe « kozeal » et toutes ses conjugaisons. Pour cela nous devons non seulement donner « kozeal » mais aussi son radical « koze » autrement un mot comme « kozeomp » ([nous] parlons) ne sera pas reconnu. L'expression « en em » sera reconnue partout sans difficulté parce qu'elle ne varie pas. En l'état actuel, le logiciel n'est pas capable de reconnaître les expressions avec des parties variables (verbes conjugués, différents pronoms etc ...)

L'ordre : Width_pix_dico.

Nous devons définir la taille des différentes fenêtres présentées dans la première partie de ce logiciel. Le logiciel GTK2 permet des ajustements par un étirage à la souris des fenêtres, ainsi nous pouvons ajuster les fenêtres livre et dictionnaire. Comme nous l'avons déjà vu, la fenêtre dictionnaire est divisée en trois sous-fenêtres, une pour chaque langue, français, breton, anglais (dans l'ordre que vous avez déjà choisi par **`Order_dico`**). Vous devez choisir la taille de chaque sous-fenêtre. Par exemple :

```
Width_pix_dico :>
 260     260     260   <:
```

Si, comme précédemment, la somme des largeurs des sous-fenêtres est supérieure à la taille totale de la fenêtre dictionnaire, une règle horizontale sera créée.
Remarue : Il n'est pas nécessaire d'avoir trois nombres égaux pour les paramètres de l'ordre « **`Width_pix_dico`** ».

L'ordre : Color_i_dico. (avec 0 ≤ i ≤ 4)

Comme vous l'avez vu dans le guide de présentation du logiciel, 5 couleurs de fond ont été utilisées pour l'affichage des lignes des dictionnaires. Un fond blanc est pour une ligne ordinaire. Un fond vert très clair est utilisé à chaque fois que nous avons un nouveau paragraphe principal, c'est à dire à chaque fois que nous avons un nouveau mot ou une nouvelle nature grammaticale pour un mot donné. Un fond bleu très pâle est utilisé comme séparation de moindre importance. Par insertion de lignes blanches, les lignes vertes et bleues sont au même niveau dans toutes les colonnes. Par cette astuce, il est plus facile de passer d'une langue à l'autre. Un fond rose très pâle est utilisé pour une ligne cliquable à la souris. Ces lignes sont utilisées pour naviguer dans la base de données. Les lignes roses qui se correspondent peuvent cependant ne pas être au même niveau pour les différentes langues. La dernière couleur, jaune, est utilisée lorsque vous avez des mots définis comme entrées secondaires dans les dictionnaires. Pour les distinguer des autres mots, leur première ligne est en jaune. Comme ils peuvent être n'importe où à l'intérieur de l'article affiché, les lignes où on les trouve sont aussi colorées en jaune. Généralement ces couleurs par défaut sont très facilement distinguées, cependant vous pouvez désirer les ajuster à votre propre perception des couleurs. Une fois, j'ai aussi été obligé d'assombrir ces fonds sur un ordinateur portable avec un écran de très faible contraste. Les ordres « Color_0_dico », « Color_1_dico », « Color_2_dico », « Color_3_dico » et « Color_4_dico » peuvent être utilisés pour ajuster la couleur de fond des lignes du dictionnaire.
Bien que j'aie utilisé les mots blanc, bleu, vert, rose clair ou jaune, vous pouvez donner à ces lignes la couleur de fond que vous désirez en ajustant les composantes rouges, vertes et bleues de la couleur. Ces trois nombres peuvent être donnés par leurs valeurs hexadécimales (de 0x0000 à 0xffff) ou simplement par leurs valeurs décimales ordinaires (de 0 à 65535). Pour la notation hexadécimale, le « 0x « est simplement une indication enjoignant au logiciel de prendre le nombre comme hexadécimal. Ci-dessous, vous avez un exemple de l'utilisation de ces quatre ordres pour la définition des couleurs aux valeurs par défaut, blanc, bleu clair, vert clair, rose clair et jaune.

```
#       Color_0_dico = white  (for an ordinary line)
#       Color_1_dico = very light blue  (first line of a
#                      paragraph)
#       Color_2_dico = very light green  (first line of a
#                      new entry word,
#                      first line for a new meaning of the
#                      word)
#       Color_3_dico = very light pink  (for a
#                      mouse clickable line )
```

```
#       Color_4_dico = yellow for words defined as secondary
entries.
#

#                     RED       GREEN    BLUE
Color_0_dico :>    0xffff    0xffff    0xffff  <:
Color_1_dico :>    0xdddd    0xdddd    0xffff  <:
Color_2_dico :>    0xdddd    0xffff    0xdddd  <:
Color_3_dico :>    0xffff    0xdddd    0xdddd  <:
Color_4_dico :>    0xffff    0xffff    0x5555  <:
#
```

L'ordre : Font_dico.

À vrai dire, la définition des polices de caractères est un terrain fangeux, au moins pour moi. Ce que je vais dire sera cependant suffisant pour que vous puissiez choisir une police pour la fenêtre livre et la fenêtre dictionnaire. La police pour la fenêtre dictionnaire est choisie dans le fichier « furch.conf » par l'ordre « Font_dico ». Pour la fenêtre livre, les polices (nous pouvons avoir plusieurs polices) sont déterminées dans le fichier « gtkb.rc ». Ceci sera expliqué plus tard.
Nous devons distinguer deux cas différents, pour Linux et pour un système MS Windows.
Pour Linux. Pour Linux, vous devez choisir une police qui est dans votre système. Pour cela vous avez les deux commandes :« xlsfonts » et « xfontsel ».
Dans une fenêtre console, la commande « **`xlsfonts > file_name`** » listera toutes les polices disponibles dans votre système Linux. Leurs noms seront écrits dans le fichier « file_name ». Vous pouvez lire ce fichier avec n'importe quel éditeur de texte tels que vi, vim, gvim, emacs, etc. La commande « **`xfontsel`** » fait démarrer un petit programme qui est capable d'afficher les polices. La plupart des polices dans Linux ne sont pas ajustables, ce qui veut dire que vous ne pouvez pas faire varier leur taille de façon continue. Vous avez seulement quelques tailles de disponibles, par exemple, sur mon ordinateur, pour la police « courier » j'ai les tailles suivantes :

8,10,11,12,14,17 **->** 26,34

Ainsi, les tailles 13, 15, 16, 27, 28 .... n'existent pas. Si vous demandez l'une des tailles manquantes, le programme reviendra automatiquement à la police par défaut sans le signaler. Ceci peut être assez trompeur parce que vous n'aurez pas une évolution continue de l'apparence de vos polices.
Les polices sont aussi « variable width » (largeur variable suivant les lettres) ou « fixed width » (même largeur pour chaque lettre comme avec une machine à écrire).

Les polices à largeur fixes avec les tailles les plus courantes sont :

```
# * lucidatypewriter  8,10,11,12,14,17 -> 26,34
# * courier           8,10,11,12,14,17 -> 26,34
# * fixed             12 -> 16,18,24
```

Si vous voulez choisir la première police avec la taille 17, vous écrirez la ligne :

```
Font_dico :>   Lucida bald 17   <:
```

Dans Linux vous avez aussi des polices à largeur variable. En voici quelques unes très courantes :

```
# *
# * helvetica         8,10,11,12,14,17,18,20,24,25,34
# * lucida            8,10,11,12,14,17,18,19,20,24,25,26,34
#
#Example : Font_dico :>   Helvetica bald 18   <:
```

Pour utiliser la police helvetica, vous pouvez insérer la ligne suivante dans le fichier « furch.conf » :

```
#
Font_dico :>   Helvetica bald 20   <:
```

Vous pouvez essayer : medium, normal, bold, italic.

Pour Windows. Maintenant, il n'y a plus de différence entre Linux et Windows en ce qui concerne le logiciel FURCH.

Les ordres : Horiz_pix_total, Verti_pix_total.

Comme il a été dit plus haut, la version 2.011 du logiciel FURCH a des fenêtres ajustables. Cependant, la taille par défaut de la fenêtre principale est fixée dans le fichier furch.conf par les ordres Horiz_pix_total et Verti_pix_total.
Par exemple :

```
#
  Horiz_pix_total :>   1200  <:
  Verti_pix_total :>   850  <:
#
```

Bien entendu, vous devez choisir les polices dans votre fenêtre livre en accord avec la taille que vous avez imposée à votre fenêtre. Si les lignes sont trop longues ou si vous avez trop de lignes vous ne pourrez pas afficher la page tout entière mais des règles-ascenseurs apparaîtront automatiquement et elles vous permettront de

positionner selon votre désir la page que vous affichez.
Les polices utilisées dans la fenêtre livre sont déterminées dans le fichier de configuration « gtkb.rc ».

L'ordre : Width_pix_book.

Une fois que vous avez défini la taille de la fenêtre principale, vous devrez signifier au logiciel, quelle part attribuer à la fenêtre livre et à la fenêtre dictionnaire. Cela peut être fait en donnant la largeur en pixels de la fenêtre livre. La différence entre la largeur de la fenêtre principale et celle de la fenêtre livre sera automatiquement attribuée à la fenêtre dictionnaire.

La taille part défaut de la fenêtre livre peut être modifiée par l'ordre : "Width_pix_book" :

```
#
#  Number of pixels for the width of the window
# displaying the book part
#        example : Horiz_pix_total/2
#
    Width_pix_book :> 580 <:
#
```

Comme il a été dit dans le guide de présentation, ce qui est donné dans le fichier furch.conf est seulement des tailles par défaut. Vous pouvez ajuster ces tailles en draguant les coins et les côtés des fenêtres. Si vous quittez l'application en pressant le bouton « Quit », votre configuration sera enregistrée dans le fichier « furch_last_state ». Ce fichier est un simple fichier texte, vous pouvez l'ouvrir, vous y trouverez des lignes comme :

```
1566   934    Size of main window width height
688    Width of book window
```

Ces deux lignes sont très explicites, elles vous donnent la largeur en pixels de la fenêtre principale (ici 1566), la hauteur (ici 934) et la largeur de la fenêtre livre (ici 688) au moment où vous avez quitté l'application pour la dernière fois. Ces valeurs peuvent être très utiles parce qu'il vous suffit d'ajuster vos fenêtres une seule fois, puis de quitter l'application en pressant le bouton « Quit ». Enfin vous prendrez ces valeurs dans le fichier « furch_last_state » et modifierez avec ces valeurs le fichier « furch.conf ». Ainsi, même si vous quittez l'application en détruisant la fenêtre principale, vous aurez de bonnes valeurs par défaut pour les tailles de vos fenêtres.

Les ordres : Nb_char_line, Nb_line_by_page.

Ces ordres sont utilisés pour donner des valeurs par défaut au nombre de caractères

par ligne et au nombre de lignes par page. Ces valeurs seront utilisées à chaque fois que cela sera possible. Bien entendu, elles ne peuvent pas être utilisées lorsque l'on a un livre dans le format original, ainsi, elles sont ignorées avec les balises « NPO , .NHO , .NTO , .NFO ». Lorsque nous avons un texte formaté avec les balises « .NP , .NH , .NT , .NF », le nombre de caractères par ligne sera pris en compte. Lorsque nous avons un texte non formaté, les deux valeurs par défaut seront utilisées.
Voici un exemple pour l'utilisation de ces ordres :

```
#
#    Nb_char_by_line = Number of characters by line :
# This parameter is used to display texte
# whenever possible
      Nb_char_line  :>  53  <:
#
#    Nb_line_by_page = Number of lines by page :
#  This parameter is used to display texte
#  whenever possible
      Nb_line_by_page  :>  32  <:
```

Remarque : Il aurait mieux valu laisser le logiciel déterminer lui-même le nombre de caractères par ligne et le nombre de lignes par page mais la programmation pour ce faire eût été difficile. Cependant quelque chose du même genre a été fait pour les fenêtres dictionnaires.

Le fichier « gtkb.rc ».

Le fichier « gtkb.rc » est en fait un fichier de configuration Gtk, donc nous devons utiliser les ordres Gtk.

Dans la fenêtre livre, les mots sont en fait des boutons avec labels. La couleur de la police peut être noire pour les noms ordinaires, bleue pour les expressions ou rouge pour les noms qui ne sont pas reconnus. Il y a aussi une couleur de police verte qui n'est plus utilisée mais qui a été gardée pour des raisons de compatibilité avec la version 1 de FURCH. Chaque mot, en fait chaque bouton, est cliquable et sa couleur de fond change lorsque le curseur de la souris est sur lui. C'est pour indiquer qu'il est prêt à être cliqué. Nous pouvons aussi choisir de mettre en valeur un mot de façon permanente pour qu'il soit facilement distinguable dans une page. Cela se fait en donnant un fond vert clair au mot. Toutes ces possibilités s'obtiennent en définissant des boutons avec différents styles. Par exemple pour la famille rouge, nous avons :

```
style "button_red"
style "button_red_up"
style "button_red_hlight"
style "button_red_hlight_up"
#
```

Et nous avons les mêmes 4 styles pour les familles bleue, verte et black (noire). Pour chaque bouton nous devons aussi définir la police. Il est plus simple de prendre la même police partout mais ce n'est pas obligatoire. Vous pouvez choisir, dans la partie livre, une taille de lettres plus grande que celle utilisée dans la fenêtre dictionnaire. Pour chaque style, les couleurs de fond (bg) et de premier plan (fg) seront aussi définies. Les couleurs sont données par leurs composantes rouges, vertes et bleues. Chaque composante est un nombre variant de 0.0 à 1.0. Par exemple, nous avons pour le style «button_red» :

```
#
style "button_red"
{
   bg[NORMAL]= {0.8 ,0.8 ,1.0  }
   fg[NORMAL]= {1.0 ,0.0 ,0.0  }
     font_name = "Helvetica Bald 17"
}
#
```

Le bouton black (noir) est spécial car il est utilisé, non seulement pour les mots dans

la page, mais aussi pour les boutons de commande au bas de la page livre. C'est pourquoi il a non seulement un état NORMAL mais aussi des états PRELIGHT, ACTIVE et INSENSITIVE.

Les noms tels que "button_red" sont seulement des références pour le logiciel, vous pouvez bien entendu prendre la couleur que vous voulez et vous pouvez choisir des lettres bleues pour votre bouton rouge si vous préférez.Ceci est vrai pour les autres familles de boutons telle la famille des boutons bleus.

Vous pouvez avoir toutes vos écritures en noire si vous préférez ne pas distinguer les expressions des autres mots. Une différence de couleur entre les mots simples et les expressions composées n'est plus nécessaire dans la version 2 de FURCH parce que, si vous avez le pointeur de votre souris sur l'un des mots formant l'expression, celle-ci sera mise en valeur toute entière. Cela n'était pas vrai dans la version 1.

Les normes pour les dictionnaires.

Un problème important pour FURCH est l'affichage de l'information dans la fenêtre dictionnaire sous la forme la plus plaisante et agréable possible. Comme nous l'avons vu, nous pouvons avoir, dans la présente version du logiciel, jusqu'à trois colonnes, une pour chaque langue reconnue. L'utilisateur est libre de spécifier pour chaque colonne la largeur en pixels et le nombre de caractères sera déterminé par le logiciel. Vous devez aussi diviser votre texte en paragraphes. Cette situation est très semblable à celle que vous pouvez rencontrer en affichant un document du web. Pour le web la solution fût l'invention du langage HTML et l'écriture de logiciels pour ce langage. Nous suivrons exactement la même approche, inventer des langages (que j'appelle des normes de dictionnaires) et concevoir des logiciels pour afficher les dictionnaires écrits avec ces langages. Ces langages sont comme le HTML des langages à balises, mais ils sont très simplifiés parce que le problème est plus simple et aussi parce que nous ne pouvons pas espérer que l'utilisateur de base apprenne un langage difficile supplémentaire. Telles sont les raisons de l'établissement de la « norme régulière » pour l'écriture des dictionnaires. Les dictionnaires sont en texte ordinaire avec des balises et il n'y a pas de classement alphabétique des entrées. Les balises indiquent les langues, les séparations entre les paragraphes, le niveau de lecture, etc. Cependant, il est vite devenu évident que même s'il n'est pas trop difficile d'écrire un dictionnaire en norme ordinaire en partant de zero, c'est tout de même beaucoup de travail pour transformer une liste de mots déjà existante en dictionnaire lisible par le logiciel FURCH. Il semble, au hasard des conversations, que beaucoup de ceux qui étudient le breton aient fait pour leur usage personnel des listes de mots lisibles par les moyens informatiques. Ce serait bien s'ils pouvaient utiliser leurs listes dans FURCH sans trop d'efforts et même mieux, si ils pouvaient autoriser les autres à utiliser ces listes. C'est pourquoi la « norme simplifiée » a été

créée. En gros, un auteur de dictionnaire aura simplement à introduire un mot ou une expression par un caractère « # » en début de ligne et à mettre après le mot ou l'expression un signe de ponctuation ou un saut à la ligne. Après cela, il mettra les explications jusqu'au signe « # » suivant. Avec la « norme simplifiée », tout dictionnaire déjà existant peut ainsi devenir utilisable avec un minimum de modifications. Des explications détaillées sur les deux normes seront données plus loin mais nous devons d'abord parler des caractères accentués ou spéciaux.

Le problème des caractères accentués ou spéciaux.

Le français et le breton ont tous les deux des caractères accentués ou spéciaux. Par exemple :

« àñïçæùœéèë ... » et les majuscules correspondantes « ÀÑÏÇÆÙŒÉÈË ... ».

Vous devez être capable d'indiquer au logiciel un caractère de ce genre mais le caractère spécifié doit aussi exister dans la police utilisée pour l'affichage des textes. En fait, tous les caractères dont nous avons besoin en français et en breton sont dans toutes les polices latines standard excepté le « œ » français et sa majuscule correspondante « Œ ». Souvent vous pouvez sélectionner de tels caractères sur votre clavier en pressant une seule touche ou une combinaison de touches. Cela dépend des paramètres de configuration de votre OS et du traitement de texte que vous utilisez. Toutes ces lettres accentuées seront reconnues par FURCH et affichées correctement dans la fenêtre livre ou la fenêtre dictionnaire. Cependant, selon votre logiciel, il est possible que vous ne puissiez obtenir certains caractères en frappant une combinaison de touches. Vous utiliserez alors des séquences de touches commençant par un caractère d'échappement « \ » suivi par un accent tel que « ' » et une lettre telle que « u ». Le logiciel comprendra et écrira « ù ». Vous pouvez mélanger les deux notations si vous le désirez. Ainsi le nom « Éloïse Salaün » peut aussi être écrit :

« Élo\:ise Sala\:un » , « \'Eloïse Salaün » , « \'Elo\:ise Sala\:un » ou toute autre combinaison que vous imaginerez.

Remarque. Ici j'ai supposé que vous suivez la tendance moderne et que vous accentuez aussi les lettres majuscules.

Généralement vous ne pouvez pas afficher correctement le caractère spécial « œ ». Il est cependant mieux de le mettre dans vos dictionnaires sous la forme composée « \oe ». Lorsque les polices seront écrites pour la norme UTF8, vous n'aurez pas à corriger vos dictionnaires.

Vous pouvez également utiliser les caractères accentués ou spéciaux dans la ligne de commande située au haut de la fenêtre dictionnaire. Ceci, cependant, n'est pas obligatoire. Vous pouvez tout aussi bien ignorer tous les accents. Dans le logiciel

FURCH, la recherche des mots se fait sans les accents. Il est plus simple de ne pas les écrire quand vous utilisez la ligne de commande.

Ici, vous trouverez toutes les séquences de caractères reconnues actuellement par FURCH.

« \oe » pour « œ » et « \OE » pour « Œ »

(elles seront, cependant, écrites oe et OE)

« \ae » pour « æ » et « \AE » pour « Æ »

« \~n » pour « ñ » et « \~N » pour « Ñ »

« \,c » pour « ç » et « \,C » pour « Ç »

« \`a » pour « à » et « \`A » pour « À »

« \^a » pour « â » et « \^A » pour « Â »

« \`e » pour « è » et « \`E » pour « È »

« \'e » pour « é » et « \'E » pour « É »

« \:e » pour « ë » et « \:E » pour « Ë »

« \^e » pour « ê » et « \^E » pour « Ê »

« \^i » pour « î » et « \^I » pour « Î »

« \:i » pour « ï » et « \:I » pour « Ï »

« \:u » pour « ü » et « \:U » pour « Ü »

« \^u » pour « û » et « \^U » pour « Û »

« \`u » pour « ù » et « \`U » pour « Ù »

« \:o » pour « ö » et « \:O » pour « Ö »

« \^o » pour « ô » et « \^O » pour « Ô »

D'autre séquences du même type peuvent être ajoutées dans le futur, si c'est nécessaire.

Remarque : l'accent aigu peut avoir différents aspects suivant la police qui est utilisée.

La norme simplifiée.

Comme il a déjà été dit, la norme simplifiée a été conçue pour faciliter l'utilisation des lexiques existants. Je supposerai cependant que ces lexiques sont en texte brut, sans codage ou compression. Si ce n'est pas vrai, il faudra d'abord transcrire le lexique en texte brut. Nous devons dire ensuite au logiciel FURCH que nous avons un dictionnaire en norme simplifiée. Cela se fait en mettant la séquence de caractères « !S » au début du fichier. Examinons un exemple de dictionnaire en norme simplifiée tel que le fichier « splf1.ex » qui est distribué avec FURCH. Les premières lignes du fichier ont été listées ci-dessous :

```
!
!S
!# Spe1
!
!   Ceci est un exemple de dictionnaire en norme
!    simplifiée.
!   Notez : Il faut aussi mettre le radical des verbes
!           pour que FURCH puisse détecter les formes
!           conjuguées.
!   This is an example of simplified norm dictionary.
!   Note : It is necessary to have the stem of the verbs
!        as separate entries in the dictionary for FURCH
!        to recognize conjugated forms.
!
!   Dictionary splf1_ex  done January 2005
!
!
#legestr
   legestr (nom masculin)= homard, pluriel : ligistri
   . lobster
#boue. (nom masculin)= bou\'ee, pluriel : boueio\`u
   . buoy
```

Nous remarquons immédiatement que des lignes commencent par le caractère « ! ». Ce sont des commentaires et elles sont généralement ignorées par le logiciel. Cependant les lignes commençant par « !S » ou par « !# » sont spéciales. Le « !S » indique que nous avons un dictionnaire en norme simplifiée et le « !# » introduit l'alias du dictionnaire.

Les alias des dictionnaires.

Chaque dictionnaire doit recevoir un alias. C'est une courte séquence de caractères qui peut être utilisée dans un autre dictionnaire pour faire référence au vôtre. Par exemple, ici le dictionnaire simplifié a reçu l'alias « Spe1 ». Vous devez noter que FURCH tient compte de la casse, il différencie les majuscules des minuscules. Si un autre auteur désire citer le mot « legestr » (homard) dans le dictionnaire « splf1.ex », il peut écrire par exemple :

```
voyez le mot << legestr >Spe1> dans l'excellent
 dictionnaire écrit par Mr ...
```

La ligne de ce dictionnaire qui contient << legestr >Spe1> sera affichée avec une couleur de fond rose. Le lecteur peut voir les explications données pour ce mot dans le dictionnaire « splf1.ex » par un simple clic sur la ligne. Vous pouvez vous-même faire référence à d'autre dictionnaires de cette façon. Vous pouvez même faire référence à un mot dans votre propre dictionnaire en écrivant simplement << quelque chose >>. Où « quelque chose » est dans votre dictionnaire un mot unique ou une expression composée de plusieurs mots.

Le choix de l'alias est important. Il doit être court, on doit le retenir facilement et il ne doit pas avoir été utilisé pour un autre dictionnaire.

Les entrées de dictionnaires.

Chaque mot ou expression que vous expliquez dans votre dictionnaire est une entrée pour ce dictionnaire. Vous devez indiquer que vous avez une entrée et on le fait en commençant une ligne par le caractère « # ». Ce caractère sera suivi du mot ou de l'expression puis ensuite par un caractère de ponctuation ou de saut à la ligne suivante (deux caractères de saut pour Windows mais c'est automatique). Après l'entrée de dictionnaire, vous pouvez mettre toutes vos explications et traductions jusqu'à l'entrée suivante, c'est à dire jusqu'à la première ligne commençant par un « # ». Vos lignes ne doivent pas avoir plus de 100 caractères et il vaut mieux en avoir beaucoup moins. Limitez vous à 70 caractères par exemple. Vous pouvez utiliser autant de lignes que vous voulez, FURCH ne s'en soucie pas. Pour l'affichage dans la fenêtre dictionnaire, FURCH coupe les lignes après les mots (pas de coupure dans les mots pour l'instant) lorsque le nombre maximum de caractères a été atteint ou s'il serait dépassé par l'adjonction d'un autre mot. Vous avez par exemple en figure 1 la sortie des deux premières entrées du dictionnaire « splf1.ex ». Quoique ces sorties soient parfaitement lisibles, elles ne sont pas parfaites parce que les traductions en anglais et en français ne sont pas séparées.

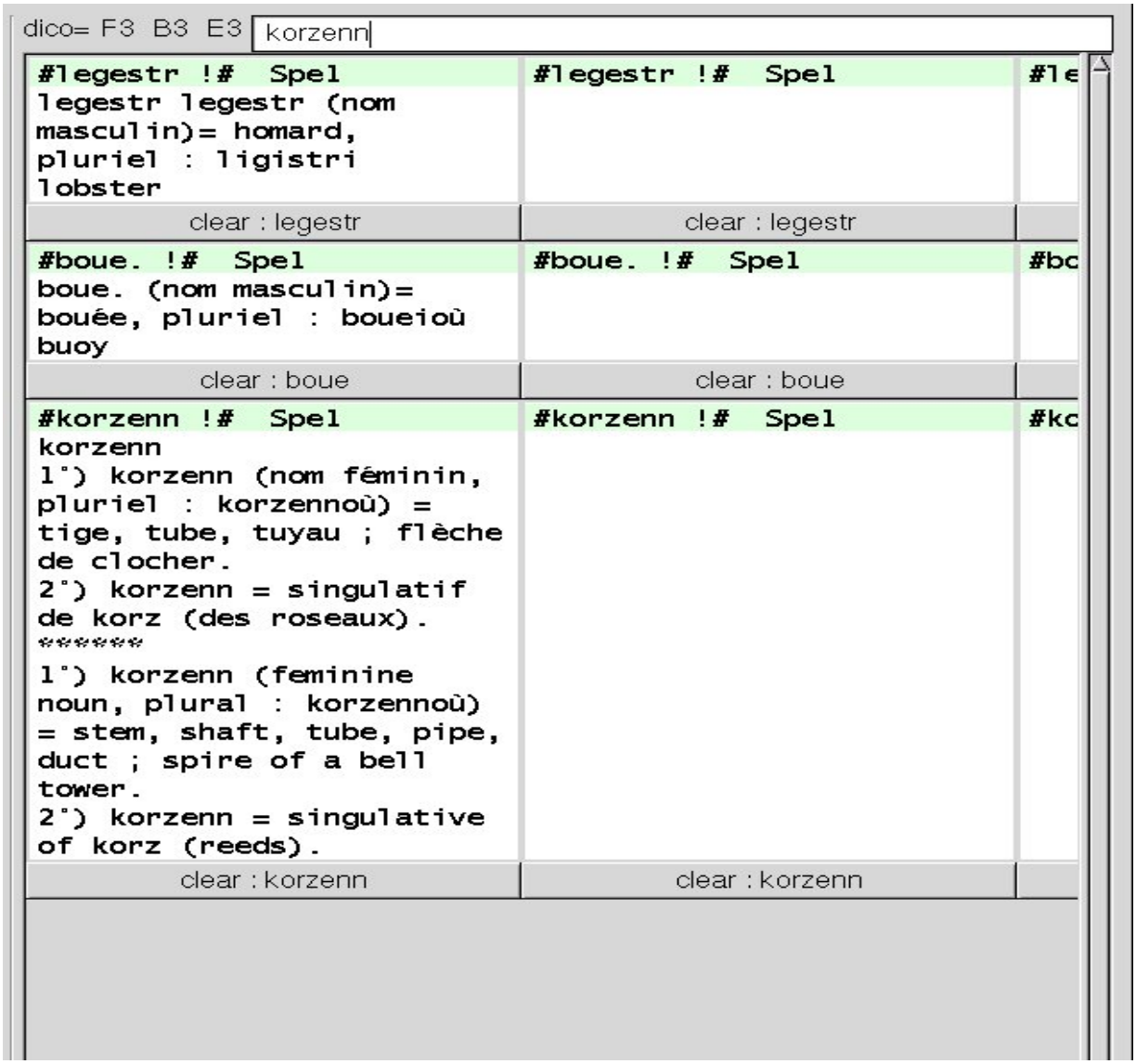

Figure 1. (splf1.jpg) *Sorties pour différents mots du dictionnaire en norme simplifiée « splf1_ex ».*

Nous allons voir maintenant deux artifices simples qui ont été rajoutés à la norme simplifiée et qui peuvent améliorer vos sorties. Vous n'êtes pas obligés de les utiliser !

Quand une ligne commence par un point « . » ou par un chiffre 0 → 9.

En fait, il serait plus correct de dire que cela s'applique au cas où le premier caractère qui n'est pas un espace est un point ou un chiffre. Pour ces deux cas, à l'écran, nous commencerons une nouvelle ligne à cet endroit. Cela peut être utilisé pour séparer le français de l'anglais et pour distinguer différents cas ou significations. Voyons, par exemple, le mot « korzenn » dans le fichier « splf1_ex », nous avons :

```
#korzenn
```

```
1°) korzenn (nom féminin, pluriel : korzenno\`u) =
    tige, tube, tuyau ; flèche de clocher.
2°) korzenn = singulatif de korz (des roseaux).
. ******
1°) korzenn (feminine noun, plural : korzenno\`u) =
    stem, shaft, tube, pipe, duct ;
    spire of a bell tower.
2°) korzenn = singulative of korz (reeds).
```

Avec ce texte, la ligne « ****** » séparera la traduction en français de celle en anglais. Le point lui-même ne sera pas écrit. Le « 1°) » et le « 2°) » commenceront de nouvelles lignes à l'écran. Vous pouvez voir en figure 1 la sortie obtenue.

La norme ordinaire.

Comme nous l'avons vu ci-dessus, avec la norme simplifiée toutes les sorties sont sur la même colonne et la séparation entre les langues est maladroite. Le principal désavantage avec la norme simplifiée est cependant que nous ne pouvons pas introduire des informations grammaticales sous une forme compréhensible par le logiciel. En l'état actuel, FURCH ne fait pas grand chose des informations grammaticales mais elles sont déjà utiles pour les verbes conjugués ou les noms au pluriel. L'utilisation des informations grammaticales s'accroîtra probablement dans les prochaines versions et ces informations font partie de la norme ordinaire. Nous allons voir maintenant en détail comment écrire un dictionnaire en norme ordinaire. Prenons, par exemple, le dictionnaire « dico1_quere ».

L'alias de dictionnaire.

Comme nous avons vu pour la norme simplifiée, une ligne de commentaire spéciale donne l'alias choisie pour le dictionnaire. Cet alias sera utilisé pour faire rapidement référence au dictionnaire. Ici, comme l'alias est « Qe1 », une des premières lignes du fichier sera :

```
!# Qe1
```

Aucun indication de norme est nécessaire, celle par défaut est la « norme ordinaire » et après quelques commentaires nous pouvons commencer avec la première entrée. Cependant, voyons tout d'abord les abréviations grammaticales.

Les abréviations grammaticales.

Voici une liste des abréviations utilisées pour donner des indications grammaticales au logiciel. Les indications doivent être dupliquées en texte ordinaire pour le lecteur humain car elles n'apparaissent pas dans la fenêtre de sortie du dictionnaire. On donne seulement les abréviations les plus courantes, elles sont considérées comme plus ou moins fixées. D'autres seront ajoutées si c'est nécessaire.

(ag) anv gourel : nom masculin.

(agl) anv gourel lies : nom masculin pluriel.

(aw) anv gwregel : nom féminin.

(awl) anv gwregel lies : nom féminin pluriel.

(adgl) anv denel gourel lies : nom de personnes masculin pluriel.

(agn) anv-gwan : adjectif.

(arg) araogenn : préposition.

(agv) anv-gwan-verb : participe passé (pour un verbe).

(es) estlamadenn : exclamation.

(gms) ger-mell strizh : article défini.

(gma) ger-mell amstrizh : article indéfini.

(gs) ger-stagañ : particule de liaison.

(r) rannig : particule.

(rv) rannig-verb : particule verbale.

(rga) raganv : pronom.

(rgaw) raganv gwregel : pronom féminin.

(rgaw) raganv gourel : pronom masculin.

(rgg) raganv gour : pronom personnel.

(rkg) rakger : préfixe.

(rkv) rakverb : adverbe.

(stg) stagell : conjonction.

(stgu) stagell genurzhiañ, conjonction de coordination.

(tl) tro-lavar : expression.

(vd) verb displeget : verbe conjugué.

(v) verb : verbe, les verbes conjugués peuvent être indiqués par vnn où nn sont deux chiffres.

Le logiciel FURCH utilise certaines des abréviations données plus haut pour faire des vérifications grammaticales avant de suggérer une signification de mot. C'est en particulier vrai pour les verbes conjugués.

Les lignes d'entrées d'un dictionnaire.

Nous pouvons maintenant expliquer comment introduire une nouvelle entrée dans un dictionnaire. Il serait plus exacte de parler d'une « ligne d'entrée » . C'est une ligne (de moins de 100 caractères) qui commence par un signe « # ». Ci-dessous nous avons quelques lignes extraites du dictionnaire « dico1_quere » :

```
#douar  *1 (ag-0)
    ?   il n'y a qu'une seule entree pour douar

    B>1 douar="douar (ag)" liester : "douaro\`u" ,
            "douareier (agl)"
```

Nous avons à cet endroit l'entrée « douar » (terre) dans le dictionnaire « dico1_quere ». Comme pour la norme simplifiée, nous indiquons que nous avons une entrée par le signe « # ». Nous avons alors le mot ou l'expression jusqu'au signe« * » qui lui-même précède un nombre qui est le nombre d'identités grammaticales différentes pour le mot ou l'expression. Nous avons ensuite les abréviations grammaticales. Ici nous avons seulement une seule identité grammaticale « (ag-0) » parce que « douar » est seulement un nom masculin.

Remarque : Le « -0 » dans « (ag-0) » n'a pas de signification spéciale, c'est seulement un séparateur pour le logiciel. Initialement FURCH a démarré comme un projet pour faire des statistiques sur l'utilisation des mots. Le « -0 » était supposé être remplacé par un pourcentage d'utilisation. Quoique je crois toujours que cela serait important pour l'apprentissage des langues, par manque de temps ceci a été abandonné. Le « -0 » et aussi le signe « ? » au commencement d'une ligne, pour introduire des commentaires entre la ligne d'entrée et les explications, sont tout ce qui reste de ce projet maintenant définitivement arrêté.

Un autre exemple de ligne d'entrée est donnée ci-dessous pour le mot « a » qui a 5 identités grammaticales différentes.

```
#a    *5   (arg-0)  (rv-0)  (gs-0) (v13-0) (es-0)
    ?   -1  Araogenn (eus) / Pr\'eposition (de)
    ?                           / preposition (of)
    ?   -2  Rannig-verb/ particule verbale / verbal particle
```

```
?   -3  ger-stagan (anv-verb + ober) /particule de liaison /
?                                    /linking particle
?   -4  verb mont amzer vreman trede gour unan
?             / verbe aller pres. 3i\`eme pers. du sing.
?                   / verb to go present third pers. sing.
?   -5  Estlammadenn / Exclamation / Exclamation

B>1  a="a (arg)". Araogenn.
```

Des liens peuvent être mis dans la ligne d'entrée.

Quand un mot est expliqué dans un dictionnaire, on désire parfois faire référence à un autre mot dans le même dictionnaire ou au même mot dans un autre dictionnaire. Nous avons déjà vu pour la norme simplifiée que nous pouvons utiliser la notation : << expression >>. si nous faisons référence à une entrée dans le même dictionnaire, ou à << expression >alias> si nous faisons référence à une entrée dans un autre dictionnaire donné par son alias. Un simple clic sur l'expression nous permet d'afficher l'information supplémentaire dans la fenêtre dictionnaire. La même notation peut être utilisée dans les ligne des entrées de dictionnaire, la différence étant que nous n'avons pas à cliquer, l'information supplémentaire est considérée par l'auteur du dictionnaire comme absolument nécessaire et elle est toujours affichée. L'exemple ci-dessous pris dans « dico2_quere » montre le mot « mailhurell » (maillot de bébé) qui fait référence à un synonyme, le mot « maillur », dans le même dictionnaire.

```
#mailhurell  *1   (aw-0)   << maillur >>

    B>1  mailhurell = "mailhurell (aw)". Anv gwregel,
                         liester : "mailhurello\`u (awl)".
```

On présente maintenant un autre exemple pris du même dictionnaire « dico2_quere ». En fait, le mot « koulz » (aussi bien) avait déjà été rencontré mais écrit sous la forme « kouls ». Les explications complètes pour ce mot sont données dans le dictionnaire « dico1_quere » dont l'alias est « Qe1 » et elles n'ont pas besoin d'être répétées dans « dico2_quere ». Il suffit de mettre **<< kouls >Qe1>** dans la ligne de l'entrée pour afficher toute l'information du dictionnaire « dico1_quere » après ce qui est donné pour « koulz » dans « dico2_quere ».

```
#koulz *2 (rkv-0)  (ag-0)  << kouls >Qe1>

    B>1  koulz = "koulz (rkv) (ag)". Rakverb pe anv gourel.
```

.

Remarque : Les informations données ci-dessus pour les références des entrées lignes sont la norme actuelle. Cependant, une autre norme, considérée maintenant comme obsolète, fût utilisée dans « dico1_quere ». Par exemple le pluriel « deliou » (feuilles) nous enverra à «delienn » (feuille) par la notation : « **> delienn** »

```
#deliou    *1 (awl-0)              > delienn
    B>1  deliou = "delio\`u (awl)" Liester eus < delienn >.
    F>1  delio\`u = "feuilles (nfp)" Pluriel de < delienn >.
    E>1  delio\`u = "leaves" Plural of < delienn >.

#delienn    *1 (aw-0)
    ?il n'y a qu'une seule entree pour delienn

    B>1 delienn = "delienn (aw)". Anv gwregel,
         liesterio\`u : "delienno\`u (awl)" ha "delio\`u (awl)".
```

Vous remarquerez immédiatement les difficultés avec cette notation. Vous ne pouvez pas faire référence à des expressions composées de plusieurs mots et vous ne pouvez pas envoyer un lecteur dans un autre dictionnaire. C'est pourquoi cette notation, quoique encore valide, est considérée comme obsolète et elle sera abandonnée dès que « dico1_quere » sera mis à jour.

La numérotation des paragraphes.

Il a déjà été dit dans la première partie de cette documentation que l'un des traits les plus importants du logiciel FURCH était la possibilité d'afficher l'information de manière synoptique avec des langues séparées en colonnes. L'information est aussi coupée en petits paragraphes qui commencent au même niveau. Le lecteur peut ainsi regarder en premier les explications en breton et s'il a des difficultés, il aura au même niveau une traduction en anglais ou français. Nous devons voir maintenant comment obtenir un affichage synoptique. Cela est fait par une numérotation spéciale des paragraphes. Nous savons déjà que nous devons différencier les langues, qu'un mot peut avoir différentes significations ou valeurs grammaticales indépendantes, que nous pouvons avoir différentes profondeurs de lecture. Toutes les informations nécessaires seront données par une numérotation telle que :

B>1[2.0.1]

qui veut dire paragraphe breton, première signification, commentaires de niveau 3 (niveau non essentiel) numéro 0.1. Le mieux est de prendre un exemple. Prenons « word1 », un mot ou une expression écrit sur la page d'un vieux livre dans une

quelconque vieille orthographe fantaisiste. Ce mot peut avoir deux significations différentes correspondant à deux valeurs grammaticales différentes. Nous aurons dans le dictionnaire quelque chose comme ce qui suit :

```
#word1  *2   (ag-0) (rkv-0)

?   texte par AlN 18 mars 2005. Corrigé par XXX 28 mai 2005.

    B>1   word1 = "word_kltg (ag)". Anv gourel ...
    B>1[0.0]  Du texte expliquant la signification du nom
              masculin word1 ...
    B>1[0.1]  D'autre texte sur la signification de word1 ...
    B>1[0.1.1] Compléments du texte additionnel ...
    B>1[0.1.2] Encore d'autres compléments ...
    B>1[1.0]  De la grammaire pour word1 ...
    B>1[1.1]  Encore de la grammaire ...
    B>1[2.0]  Des compléments non essentiels, du bavardage ...


    B>2   word1 = "word_kltg (ag)". Rakverb ...
    B>2[0.0]  Du texte expliquant la signification de
              l'adverbe word1 ...
    B>2[0.1]  Du texte supplémentaire sur la signification
              de l'adverbe word1..
    B>2[0.1.1] Compléments du texte supplémentaire...
    B>2[1.0]  De la grammaire pour l'adverbe word1 ...
    B>2[1.1]  Encore de la grammaire ...
    B>2[2.0]  Des compléments non indispensables, du bavardage ...
    B>2[2.0.1]  Encore d'autres compléments non essentiels.
```

La première ligne est la ligne d'entrée du dictionnaire. Elle commence par le signe « # » et nous avons déjà vu sa structure. La première ligne peut être suivie de une ou plusieurs lignes de commentaires commençant par le signe « ? ». Ces lignes ne sont pas affichées à l'écran. Après cela, nous avons une ligne telle que :

```
     B>1   word1 = "word_kltg (ag)". Anv gourel ..
```

où « B>1 » signifie langue bretonne, première signification du mot. À cet endroit, vous pouvez encore écrire le mot avec son orthographe d'origine, donner l'orthographe dans un système moderne tel que le KLTG, le système universitaire ou tout autre système de votre choix puis ajouter le genre grammatical du mot.
Vous introduirez ensuite les paragraphes avec une numérotation telle que :
« B>1[0.x.y] » ou « B>1[2.x.y] » où « x.y » sont des numéros utilisés pour classer les

paragraphes. Le premier chiffre « [0...] » , « [1...] » ou « [2...] » indique le niveau de lecture. Le niveau de lecture 1 est indiqué par « [0...] », le niveau de lecture 2 par « [1...] » et le niveau de lecture 3 est indiqué par « [2...] ». Un niveau de lecture 0 est indiqué par l'absence de « [...] » comme dans « B>1 ». Cette notation est plutôt maladroite, elle est gardée pour des raisons historiques.
L'utilisateur de FURCH peut choisir d'afficher les paragraphes jusqu'à un niveau donné numéroté de -1 (pas d'affichage) à 3 (tout est affiché). Dans le fichier dictionnaire, vous n'êtes pas obligés de mettre les niveaux dans l'ordre croissant. Vous pouvez avoir par exemple :

```
B>1   word1 = "mot_kltg (ag)". Anv gourel ...
B>1[0.0]  Du texte expliquant la signification du nom
          masculin word1 ...
B>1[1.0]  De la grammaire pour word1 ...
B>1[1.1]  Encore de la grammaire ...
B>1[2.0]  Des compléments non essentiels, du bavardage ...
B>1[0.1]  Du texte en plus sur la signification de word1 ...
B>1[0.1.1] Des complements sur le texte en plus ...
B>1[0.1.2] Encore d'autres compléments ...
```

L'auteur du dictionnaire peut mettre par exemple la grammaire juste après une définition de base du mot et mettre les compléments de la définition du mot à la fin s'il le désire. Cependant, la numérotation « [.x.y] » doit être dans l'ordre croissant.
Remarque : « x » et « y » commencent à partir de 0 mais le 0 peut être omis.
Par exemple « B>1[0.0] » est équivalent à « B>1[0.] » ou à « B>1[0.0.0] » et bien sûr, « B>1[1.0] » est équivalent à « B>1[1.] » ou à « B>1[1.0.0] »
La seconde signification de « word1 », ici par exemple comme adverbe, aura une numérotation de paragraphe telle que :
« B>2 », « B>2[0.x.y] » , « B>2[1.x.y] » ou « B>2[2.x.y] » où « x.y » sont, comme ci-dessus, des nombres utilisés pour classer les paragraphes. Pour faciliter la séparations des différentes significations ou genres grammaticaux du mot, la première ligne des paragraphes « B>1 », « B>2 », « B>3 » ... apparaîtra dans la fenêtre dictionnaire avec un fond vert clair. Pour tous les autres paragraphes la couleur de fond de la première ligne sera bleue pâle. Nous avons déjà vu comment ajuster ces couleurs dans le fichier « furch.conf ».

Jusqu'à présent, nous avons seulement considéré le cas de paragraphes écrits en breton. Ceux écrits en français seront numérotés : « F>1 » , « F>1[0.x.y] » , « F>1[1.x.y] » , « F>1[2.x.y] » , « F>2 » ... et ceux écrits en anglais : « E>1 » , « E>1[0.x.y] » , « E>1[1.x.y] » , « E>1[2.x.y] » , « E>2 » ...
Vous n'êtes pas obligés de regrouper les paragraphes par langue, vous devez simplement avoir les paragraphes de chaque langue classés de façon naturelle sans vous préoccuper des autres langues. Ci-dessous vous avez un classement acceptable où le breton a été mis en premier et ensuite le français et l'anglais :

```
B>1   word1 = "word_kltg (ag)". Anv gourel ...
B>1[0.0]  Du texte expliquant la signification du
          nom masculin word1 ...

B>2   word1 = "word_kltg (ag)". Rakverb ...
B>2[0.0]  Du texte expliquant la signification de
          l'adverbe word1 ...

F>1   word1 = "word_kltg (ag)". Nom masculin ...
F>1[0.0]  Du texte en français qui est la traduction
          du texte en breton.

F>2   word1 = "word_kltg (ag)". Adverbe ...
F>2[0.0]  Du texte en français (traduction du breton).

E>1   word1 = "word_kltg (ag)". Masculine noun ...
E>1[0.0]  Du texte en anglais qui est la traduction du
          texte en breton.

E>2   word1 = "word_kltg (ag)". Adverbe ...
E>2[0.0]  Du texte en anglais (traduction du breton).
```

Cependant, ce n'est pas la seule possibilité. Vous pouvez aussi regrouper la première signification dans les trois langues et mettre ensuite la deuxième signification avec aussi ses trois langues.

```
B>1   word1 = "word_kltg (ag)". Anv gourel ...
B>1[0.0]  Du texte expliquant la signification du
          nom masculin word1 ...

F>1   word1 = "word_kltg (ag)". Nom masculin ...
F>1[0.0]  Du texte en français qui est la traduction
          du texte en breton.

E>1   word1 = "word_kltg (ag)". Masculine noun ...
E>1[0.0]  Du texte en anglais qui est la traduction du
          texte en breton.



B>2   word1 = "word_kltg (ag)". Rakverb ...
B>2[0.0]  Du texte expliquant la signification de
          l'adverbe word1 ...

F>2   word1 = "word_kltg (ag)". Adverbe ...
F>2[0.0]  Du texte en français (traduction du breton).

E>2   word1 = "word_kltg (ag)". Adverbe ...
E>2[0.0]  Du texte en anglais (traduction du breton).
```

D'autres mélanges sont encore possibles. Bien entendu, vous choisirez la configuration qui est la meilleure pour l'écriture et la correction de votre article. Lorsque vous avez beaucoup d'explication pour chaque signification du mot, la deuxième configuration peut être meilleure.

Particularités spécifiques aux paragraphes de niveau 0.

Les paragraphes de niveau 0 sont ceux introduits par « B>1 », « B>2 », « B>3 », ... , « F>1 », ... , « E>1 ». La première ligne de ces paragraphes est affichée avec un fond vert clair. Généralement, je ne mets dans ces paragraphes que des informations grammaticales, les traductions en d'autres langues ou la transcription du mot dans le système orthographique KLTG. En l'état actuel de son développement, le logiciel FURCH n'utilise pas ces informations. Cependant ces informations sont déjà étiquetées pour un futur usage. Prenons un exemple extrait de « dico2_quere », le mot « komz » peut être un verbe, le radical d'un verbe ou un mot :

```
#komz  *3  (v-0) (pgv-0) (aw-0)
    B>1  komz = "komz (v)". Verb, anv-gwan-verb : "komzet (agv)".
    B>1[0.0]   ..................
    B>1[1.0]   ..................
    B>1[1.1]   ..................
    B>1[1.2]   ..................

  F>1  komz = "parler (v)". Verbe, participe pass\'e : < komzet >.
    F>1[0.0]   ..................
    F>1[1.0]   ..................
    F>1[1.1]   ..................
    F>1[1.2]   ..................

   E>1  komz = "to speak (v)". Verb, past participle : < komzet >.
    E>1[0.0]   ..................
    E>1[1.0]   ..................
    E>1[1.1]   ..................
    E>1[1.2]   ..................

    B>2  komz = "komz (pgv)". Penngef ar verb < komz >.
    F>2  komz = Radical du verbe < komz > (parler).
    E>2  komz = Stem of the verb < komz > (to speak).

    B>3  komz = "komz (aw)". Anv gwregel, liester : "komzo\`u
(awl)".
    B>3[0.0]   ..................
    B>3[0.1]   ..................
    B>3[0.2]   ..................
    B>3[0.3]   ..................
```

```
F>3  komz = "parole (nf)". Nom f\'eminin (en breton),
          pluriel : < komzo\`u >.
F>3[0.0]   ..................
F>3[0.1]   ..................
F>3[0.2]   ..................
F>3[0.3]   ..................

E>3  komz = "speech, spoken words". Feminine noun (in Breton),
          plural : < komzo\`u >.
E>3[0.0]   ..................
E>3[0.1]   ..................
E>3[0.2]   ..................
E>3[0.3]   ..................
```

Comme le mot « komz » a trois valeurs grammaticales, nous avons pour chaque langue trois paragraphes de niveau 0. Ces paragraphes ont été reproduits ci-dessus. Vous pouvez remarquer que des informations ont été placées entre guillemets, par exemple :

```
B>1  komz = "komz (v)". Verb, anv-gwan-verb : "komzet (agv)".

E>1  komz = "to speak (v)". Verb, past participle : < komzet >.
F>1  komz = "parler (v)". Verbe, participe pass\'e : < komzet >.
```

Ces mots, écrits en rouge ci-dessus, avec aussi les identifications grammaticales abrégées, seront utilisés comme entrées supplémentaires de la base de données dans une future version du logiciel FURCH. Les mots en français et en anglais peuvent aussi être utilisés comme entrées pour de pseudo dictionnaires français-breton et anglais-breton. Cela ne remplacera pas de véritables dictionnaires mais sera utile tant que ces dictionnaires n'auront pas été écrits.

Les petits trucs de la norme ordinaire.

Comme pour la norme simplifiée, des artifices spéciaux ont été introduits pour permettre un certain contrôle de l'affichage sur l'écran. Comme chaque utilisateur final peut choisir la largeur de chaque colonne pour l'affichage des dictionnaires, les auteurs ne peuvent pas savoir quand une nouvelle ligne commencera. Il est cependant parfois nécessaire d'imposer le commencement d'une nouvelle ligne à un endroit précis du texte pour en faciliter la lecture. Cela peut être fait en introduisant un nouveau paragraphe mais nous avons alors une première ligne colorée ce qui n'est peut-être pas souhaité. Nous avons trois autres possibilités pour introduire une césure dans le texte et imposer le commencement d'une nouvelle ligne à cet endroit. Dans le

texte de notre dictionnaire nous pouvons commencer une ligne :

1) par un point : « . »
2) par un chiffre *0 → 9.*
3) par un signe « < »

Quand je dis qu'une ligne commence par un tel caractère, je veux dire que le premier caractère qui n'est pas un espace est ce caractère, ici un point, un chiffre ou le signe « < ».

Remarque 1 : Le point au commencement d'une ligne dans le texte d'un dictionnaire est un caractère de commande qui n'est jamais affiché à l'écran.

Remarque 2 : J'utilise le signe « < « pour introduire les citations. Il est souvent pratique que ce signe induise le commencement d'une nouvelle ligne pour mettre en évidence le début de la citation. Cependant ce n'est peut-être pas toujours ce que nous désirons. Nous pouvons empêcher le saut de nouvelle ligne en commençant la ligne par une virgule « , ». Cette virgule étant un caractère de commande ne sera pas affichée à l'écran. Donc, avoir « , < » au commencement d'une ligne empêchera le saut de ligne et la citation sera écrite dans le texte sans séparation. La virgule peut aussi être utilisée devant les chiffres avec le même effet.

Peut-être d'autres normes pour FURCH ?

Comme nous l'avons vue plus haut, la norme ordinaire est la norme native de FURCH et la norme simplifiée a seulement été introduite pour faciliter l'utilisation des listes de mots déjà existantes. On a supposé que ces listes étaient simples, sans indications grammaticales. Certaines personnes pourraient désirer faire des dictionnaires avec une norme plus simple que la norme ordinaire mais aussi avec plus de possibilités que la norme simplifiée. Elles peuvent le faire et leurs normes peuvent être introduites dans FURCH. Cependant, gardez à l'esprit que l'introduction d'une nouvelle norme dans le logiciel est beaucoup de travail et cela ne peut être envisagé que pour des dictionnaires assez gros.

Maintenant une dernière remarque à propos des normes ordinaires et simplifiées de FURCH. Certaines entreprises commerciales considèrent leurs normes comme des propriétés intellectuelles privées et elles essaient d'obtenir des licences logicielles pour des choses évidentes. Contrairement à cette tendance, les normes de FURCH sont déclarées « ouvertes » et vous pouvez les utiliser dans vos propres produits. Il n'y a cependant aucune garantie d'aucune sorte avec ces normes, excepté une faible promesse morale que FURCH fera de son mieux pour rester compatible avec ses anciennes normes, ou que les dictionnaires seront transformés pour devenir compatibles avec les nouvelles normes.

Installation du logiciel FURCH.

Installation dans les systèmes Windows.

FURCH est un projet « GNU/Linux/Gtk/gcc ». Cependant « gcc » et « Gtk » ont été portés dans Windows et donc FURCH marchera aussi pour Windows 98 SE, 2000 and XP. Pour le moment FURCH utilise « gcc » et « Gtk » comme ils ont été implantés dans « Dev-Cpp » .

.

Les programmes professionnels ont des logiciels d'installation automatiques et ceux des amateurs doués aussi. Mes connaissances sur cette question particulière sont trop limitées et donc vous aurez à installer FURCH à la main. Cependant pour Windows c'est facile et cela peut vous éviter des conflits avec vos logiciels déjà installés.

Comme pour toute installation d'un programme Windows, vous avez besoin de librairies dynamiques. Pour FURCH vous télé-chargerez les « dll » suivantes pour MS-XP et Win2000 :

iconv.dll
intl.dll
libatk-1.0-0.dll
libcairo-2.dll
libgdk-win32-2.0-0.dll
libgdk_pixbuf-2.0-0.dll
libglib-2.0-0.dll
libgmodule-2.0-0.dll
libgobject-2.0-0.dll
libgtk-win32-2.0-0.dll
libpango-1.0-0.dll
libpangocairo-1.0-0.dll
libpangowin32-1.0-0.dll
libpng13.dll
zlib1.dll

Maintenant, la question est : Où mettre ces librairies ? La réponse est : Quelque part dans votre PATH (chemin). La variable PATH est une liste de dossiers où le système cherche les librairies dont il a besoin. Le système cherche d'abord dans le premier

dossier de sa liste, puis dans le second, etc ... La recherche s'arrête à la première trouvaille. Vous pouvez visualiser votre liste PATH en ouvrant une fenêtre DOS et en tapant « path » (pas les « » s'il vous plait !). Le système écrira la liste des dossiers PATH séparés par des points-virgules « ; ». Vous aurez un dossier « system » ou « system32 » et peut-être un dossier « .\ » au commencement de la liste. Vous pouvez modifier votre PATH en éditant le fichier « autoexec.bat » dans Windows 98 ou par les actions suivantes dans win2000 ou winXP :

Changez de session pour devenir administrateur ;

Faire un clic droit sur l'icône bureau puis un clic gauche sur propriétés ;

Puis choisissez l'onglet « avancé » ;

Puis cliquez le bouton « variables d'environnement »

Dans la fenêtre « Variables d'environnement » vous mettrez en valeur la variable « Path » dans la section « Variables système » (la section du bas). Cliquez « Modifier » . Vous pouvez maintenant ajouter des dossiers dans votre variable PATH mais vous ne serez autorisé à le faire que si vous êtes un membre du groupe des administrateurs.

Si l'idée de modifier votre variable PATH vous rend nerveux, vous pouvez ajouter les librairies dans l'un des dossiers : system, system32, WINNT\system32, etc... Si l'une de ces librairies est déjà là, vous ne la changerez pas pour éviter tout futur conflit. Avec un peu de chance les librairies seront les mêmes ou compatibles.

Une autre possibilité est d'introduire les caractères « .\; » au début du PATH. Ainsi, le système cherchera d'abord dans le dossier ou est votre exécutable avant toute autre place. Vous mettrez alors toutes les librairies dans le dossier où vous avez votre exécutable « furch.exe ». Cependant, pour Linux, les experts signalent un petit problème de sécurité avec cette solution.

Une fois que vous avez installé les librairies, vous créerez un nouveau dossier, par exemple « Q2-11 », dans ce dossier vous mettrez l'exécutable « furch.exe » et les autres fichiers textes dont le logiciel a besoin. Vous avez besoin de :

dibenn1 : une liste de terminaisons de mots reconnues

furch.conf : le fichier de configuration de FURCH

gtkb.rc : le fichier de configuration de gtk.

Vous avez besoin aussi des dictionnaires référencés dans le fichier « furch.conf » . Au moment de la première sortie de la version 2, vous avez besoin des dictionnaires :

dico1_quere : le premier volume du dictionnaire « Quéré » .

dico2_quere : le second volume du dictionnaire « Quéré » .

roll1_kltg : un lexique pour des mots écrits en KLTG.

roll2_kltg : le second volume des mots en KLTG.

yezh1_kltg : un lexique pour les verbes irréguliers, etc...,
en KLTG.

splf1_ex : un exemple de dictionnaire en norme simplifiée.

Ensuite, vous avez besoin des fichiers textes qui seront affichés dans la fenêtre livre. Chaque fichier est considéré comme étant un livre. Ces fichiers doivent être codés en texte latin1. Ils peuvent être formatés par des balises comme .NP, .NPO... ou ne pas être formatés. Ainsi, vous pouvez afficher des textes générés par des logiciels comme OpenOffice, MS-Words, etc...

Un exemple de texte formaté est : sarmoniou1.txt que vous pouvez télécharger sur les sites de FURCH ou GUTENBERG.

Le code source a aussi été inclus dans le paquet proposé au téléchargement. Vous n'avez pas besoin de ces fichiers, sauf si vous désirez faire vous-même une compilation avec le logiciel « Dev-Cpp » . Vous devez noter que ce code utilise la version GTK.2.8.12 qui n'est pas la plus récente pour Windows.

Installation pour Linux.

Linux est un merveilleux logiciel libre, cependant il ne peut pas être considéré comme pleinement opérationnel pour un utilisateur final ordinaire. Linux a trois défauts principaux :

1. La compatibilité de Linux avec le matériel est incomplète. Les constructeurs ne se préoccupent pas de développer des pilotes pour Linux car le nombre de systèmes installés reste faible. Ce nombre reste faible dans le grand public pour les deux raisons suivantes :
2. Il est souvent nécessaire de recompiler le noyau pour introduire un nouveau pilote. Cette action est dangereuse car vous pouvez définitivement ruiner votre système. Pour la configuration de votre nouveau noyau vous aurez à choisir entre des milliers d'options ésotériques. On peut contourner toutes ces difficultés (sauvegardez votre vieux noyau, commencez avec un noyau Knoppix, etc). Cependant un non professionnel aura probablement de grosses difficultés avec la mise en œuvre du noyau.
3. L'introduction de tout nouveau logiciel dans votre système est pratiquement impossible. La seule exception est lorsque quelqu'un d'autre a déjà mis en paquet le logiciel pour votre distribution et qui plus est, pour le numero de version correspondant exactement à celle installée sur votre ordinateur. Il y a quelques efforts pour améliorer cette situation (par exemple la procédure « klik » pour Knoppix ou le logiciel « autopackage ») mais l'utilisation de tels outils ne s'est pas encore généralisée.

Ce qui a été dit ci-dessus était vrai il y a quelques années et la situation c'est beaucoup améliorée avec des distributions comme Ubuntu mais vous aurez encore à

installer des paquets pré-compilés spécifiques à votre distribution.

Pour le logiciel FURCH, nous allons essayer de contourner ces difficultés et donner des indications de façon à ce qu'un non spécialiste ait des chances raisonnables d'installer avec succès le logiciel.

Le logiciel FURCH est très simple, il utilise seulement les fonctions C standard et la librairie GTK2 . Il utilise aussi seulement les fonctions standard de GTK. Comme gcc et GTK sont dans toutes les distributions, vous aurez seulement à installer ces deux librairies (les versions complètes aussi appelées versions de développement). Les deux librairies seront compatibles car elles proviennent de la même distribution. Vous aurez aussi besoin du paquet « make » qui est dans toutes les distributions. Une fois que vous aurez installé ces trois paquets et souvent ils sont déjà là d'avance, vous ferez la compilation :

make -f makegtkb2

Avec cette ligne l'ordre « make » utilise le Makefile « makegtkb2 » que vous pouvez télécharger en même temps que les deux fichiers headers et que les fichiers sources du logiciel :

gtkb2_0.h
gtkb2_ftexte.h
gtkb2.c
gtkb2_1.c
gtkb2_2.c
gtkb2_3.c
gtkb2_4.c
gtkb2_5.c
gtkb2_6.c
gtkb2_7.c
gtkb2_8.c
gtkb2_9.c

L'ordre make compilera, fera l'édition de liens et générera l'exécutable « furch.x » . Il lancera aussi l'exécution du logiciel.

Comme pour Windows, vous avez besoin, dans le même dossier que « furch.x », des trois fichiers suivants :

dibenn1 : une liste de terminaisons de mots reconnues

furch.conf : le fichier de configuration de FURCH

gtkb.rc : le fichier de configuration de gtk.

Vous avez besoin aussi des dictionnaires référencés dans le fichier « furch.conf » et bien entendu des fichiers correspondants aux pages que vous désirez lire.

Remarque : Les fichiers textes Linux et Dos-Windows sont différents. Chaque ligne d'un fichier Dos-Windows se termine par CR/LF (Ascii 13 + Ascii 10) et chaque ligne dans un fichier Linux se termine par un simple LF. Cependant, FURCH-Windows accepte des fichiers textes Linux et FURCH-Linux accepte des fichiers textes Dos. Ainsi, pour économiser la place et simplifier la distribution du logiciel, les textes, données, fichiers de configuration et dictionnaires sont seulement distribués sous leur forme Dos-Windows. Ils seront acceptés par les deux versions de FURCH

Si vous avez le moindre doute, vous pouvez convertir (sous Linux) les fichiers Dos-Windows en fichiers Linux avec les utilitaires « dos2unix » ou « fromdos » et l'inverse peut aussi se faire à l'aide des utilitaires « unix2dos » and « todos ». Sous Windows vous pouvez convertir un fichier texte Linux en utilisant NotePad, en éditant et en re-sauvant le fichier texte. Le caractère CR sera automatiquement ajouté à la fin de chaque ligne par NotePad. Le nombre de lignes que NotePad peut manipuler est cependant sévèrement limité et vous pouvez choisir d'utiliser pour cette conversion d'autres éditeurs de textes équivalents et libres. Word2003 peut aussi faire le travail.

Le logiciel FURCH et les Mac OSs

La présente version de FURCH pour Linux est fondée sur GTK2.x. Cette version de GTK a été portée sur Mac OS X. Donc, il devrait être possible d'utiliser la version Linux de FURCH et de la compiler sur un Mac. Je n'ai cependant pas une telle machine à ma disposition, aussi quelqu'un d'autre utilisant Mac OS X devra faire le travail. Je serais heureux d'ajouter l'exécutable dans la page web de FURCH ou d'ajouter un lien vers une autre page web comportant cet exécutable.

Conclusion.

FURCH est un projet libre, son code est publié et distribué selon les termes de la licence GPL. Vous pouvez améliorer ce code et en utiliser des parties dans vos propres applications. Vos modifications doivent cependant être publiées et distribuées selon les termes de cette même licence. Pour la partie littéraire du projet, les textes et les dictionnaires, chaque auteur peut faire ce qu'il veut. Certains textes sont dans le domaine publique, d'autres peuvent être sous copyright. Le dictionnaire « dico_quere » est sous copyright mais il est libre pour tout usage non commercial. En fait, c'est pour que je reste libre d'en extraire des parties et de les développer.

C'est pour éviter aussi des corrections anarchiques et des modifications par d'autres auteurs. Les gens peuvent envoyer des corrections, si elles sont acceptées le nom des correcteurs sera ajouté à ce dictionnaire de façon à reconnaître leur travail. Si ces corrections n'étaient pas acceptées (ce qui est peu probable), les auteurs auraient encore la possibilité de publier leurs propres dictionnaires avec les mêmes mots et les compléments et les corrections qui ne furent pas acceptés. Je serais certainement heureux d'ajouter ces nouveaux dictionnaires à la base de données de FURCH. De façon plus générale j'encourage fortement tout le monde à écrire des dictionnaires, dans le système orthographique de son choix, avec peut-être seulement quelques mots dans chaque dictionnaire. Tout est utile. Laissez-moi prendre un exemple. Voici quelques mots de la famille de la faux :

1. « gwignet » : une petite faucille pour couper l'herbe, pour chercher les lançons, pour les druides pour cueillir le gui ...
2. « falzig » : une faucille.
3. « fosilhon » : une grosse faucille pour couper le bois, pour élaguer les arbres.
4. « falz » : une faux.

Tous ces mots sont utilisés dans la région de Tréguier. Sont-ils compris ailleurs ? Y a-t-il d'autres mots en usage dans d'autres régions ? Ces quatre mots peuvent constituer un dictionnaire et vous pouvez avoir autant de tels dictionnaires que vous avez de cantons en Bretagne. Que pouvons nous faire avec cela ? Et bien à l'aide de logiciels appropriés très utilisés maintenant dans les études génétiques, tracer des arbres philogéniques. Avec ce type de méthodes, nous mettrons en évidence les transmissions, les évolutions et les transformations des mots et parce que les mots voyagent avec les gens, nous pourrons tracer les routes commerciales en Bretagne. Imaginez aussi que la langue de la Cornouailles britannique soit encore vivante, en comparant les noms locaux des poissons et d'autres animaux marins des deux côtés de la manche, nous pourrions retracer l'origine des populations en Bretagne, faire la différence entre les origines cornouaillaises et galloises, etc. Le temps cours très vite contre le breton traditionnel, nous devons noter rapidement ces humbles mots. Ils sont la mémoire de notre Nation. Les moyens électroniques modernes de communication nous permettent de les sauver et de les diffuser presque sans frais. Utilisons ces possibilités nouvelles. Beaucoup de personnes ont déjà enregistré des conversations, des contes, des chansons, etc. Nous pouvons maintenant disséminer ce qui a été recueilli. Le logiciel FURCH est un outil dans cette grande entreprise. Prenez part à ce projet, améliorez la présente documentation, écrivez vos propres dictionnaires, envoyez des corrections aux autres auteurs, notez des mots, enregistrez des histoires sur bande magnétique ou sur CD, prenez des photos, scannez de vieux livres. Tout est précieux et peut maintenant être publié, distribué gratuitement. Nous ne devons pas laisser passer cette chance !

Alphonse Nandert, traduction du texte anglais achevée le 23 décembre 2007

Texte rédigé avec OpenOffice 2.2 qui mélange allègrement les guillemets anglais et français ...