

E.S.D.F.F.M.

P.L. Nageoire

Fri Oct 20 23:35:31 2006

Emacs 21.4a
(Emacspeak 24)
Speech Dispatcher 0.6.1
Festival 1.96-beta
Franfest 1.96-beta-rc01
Mbrola 3.01h

J'ai choisi de regrouper sur cette page les informations concernant tous les projets relatifs aux solutions vocales sous *Gnu/Linux* dont je suis utilisateur et occasionnellement auxquels je contribue modestement. Certaines adresses autrefois référencées par les moteurs de recherche et relatives à d'autres pages vous conduiront ici. Ces pages contenaient en effet des informations devenues obsolètes. Je ferai bien entendu en sorte que les informations disponibles ici soient à jour ; mais elles sont encore parcellaires souvent incomplètes. N'hésitez pas à *m'écrire* pour tout complément d'information.

0 . – Introduction

0.1 . – Où trouver ce document

La version la plus récente de ce document se trouve à *E.S.D.F.F.M. 0.96*. Vous pouvez également trouver ce document au format .info, au format postscript compressé, au format PDF ainsi qu'une archive contenant les fichiers au format HTML.

0.2 . – Une solution vocale sous *Linux* : qu'est-ce à dire ?

Il ne s'agit en aucun cas de quelque bricolage permettant tant bien que mal d'utiliser de temps en temps, épisodiquement et en appoint le système Linux. Il s'agit de disposer d'un outil de niveau professionnel. Il s'agirait, même si, ne rêvons pas, cela n'est pas possible en totalité, d'accéder à toutes les applications disponibles sous Linux :

- Édition de texte,
- applications web et mail,
- Administration système,
- voire des choses plus élaborées comme le traitement du son.

0.3 . –Bavardage et « philosophie du café du commerce »

Dans le principe donc, un système vocale doit pouvoir collecter les informations disponibles dans le système (Linux) et les communiquer à l'utilisateur sous forme de messages vocaux. Il semblera dès lors évident à tout un chacun qu'il faut répondre à un certain nombre de questions telles que « quelles informations collecter ? » « Parmi celles-ci lesquelles communiquer à l'utilisateur ? » « Enfin ce dernier choix étant opéré, comment les lui communiquer de la manière la plus intelligible pour lui ? » Au vu de ces questions, une constatation tout aussi immédiate s'impose c'est que chaque concepteur de système vocal aura fait des choix en fonction de ce qu'il estime le plus efficace pour répondre aux dites questions. Un corrolaire encore plus immédiat est que l'utilisateur peut adhérer ou ne pas adhérer aux choix du concepteur ainsi qu'il en va d'ailleurs de tout système informatique voire de tout outil. La conclusion qui ne consiste pas pour moi à me dédouaner de tout parti pris mais au contraire à les assumer s'impose d'elle-même : il n'y a pas de « bon système vocal » mais une « bonne adéquation entre un système vocal et un utilisateur ». Empressons-nous cependant de nous contredire pour affirmer sans plus de complexe qu'il y a quand même des critères objectifs pour affirmer qu'un outil est efficace ou non. Personne ne songerait à affirmer qu'une cuillère percée remplira son rôle si ce n'est qu'au moins on ne risque pas de la trop emplir ...

Il n'en reste pas moins que la ou les solutions que je présenterai ici sont celles que j'ai personnellement utilisées et jugées adaptées à mes besoins.

0.4 . –Comment ça marche ?

Il ne s'agit en aucun cas ici d'entrer dans les détails techniques mais de donner une idée qui pourra aussi servir de guide pour s'y retrouver dans les différentes pièces du puzzle. En effet, la solution que je propose ne se présente pas (pas encore) sous la forme d'un seul bloc qu'il suffirait d'installer pour que d'un coup la machine se mette à parler. À cela au moins trois raisons : D'abord un tel système nécessite de nombreuses opérations dont certaines sont quand-même d'une certaine complexité ce qui explique qu'elles aient été implémentées dans des outils séparés ; le même développeur n'ayant pas forcément le temps et ou les capacités de s'intéresser à la totalité du mécanisme. Ensuite certaines étapes du processus peuvent présenter un intérêt en soi et être utilisées dans d'autres applications ce qui expliquerait encore qu'elles aient été développées séparément. Enfin n'oublions pas les raisons historiques.

Un système vocal tel que je le présente est donc une sorte de mille-feuilles dont chaque couche réalise une ou plusieurs des opérations qui vont de la collecte des données dans le système à leur transmission à l'utilisateur. À noter qu'aucune raison supérieure n'impose de séparer et/ou regrouper les opérations d'une manière ou d'une autre. Ceci a d'ailleurs évolué au cours du temps et en fonction des capacités des machines. Par exemple les boîtiers ou cartes de synthèse vocale qui intégraient plusieurs fonctions aujourd'hui réparties entre d'autres couches ont disparu ou presque quand certaines des opérations qu'ils effectuaient ont pu être prises en charges par des processeurs de plus en plus puissants. À noter qu'il s'agit d'un point de vue grossier ou plutôt d'un point de vue purement utilitaire car sait-on bien comment étaient réparties les diverses tâches dans ces outils « hard » et surtout a-t-on vraiment envie de le savoir ...

Un ordre en valant bien un autre commençons la revue par la couche directement en contact avec l'utilisateur :

0.4.1 . – La synthèse vocale

Désormais le son final est produit par le système de la machine : carte son etc. Mais ces systèmes, n'intègrent pas, à ma connaissance du moins, la possibilité de convertir des informations codées, même sous forme de texte, en son intelligible. Il ne peuvent convertir que des informations dans des formats bien particuliers (wav, mp » etc) en son et même moyennant une couche logiciel. Ces informations accessibles à la carte son doivent donc être produites en amont à partir d'autres fournies par les couches supérieures. Cette dernière conversion est évidemment fortement dépendante de la langue toutes les langues n'utilisant pas les mêmes sons. Dans le système que je présente ici, elle est effectuée pour l'anglais par une partie de *Festival 1.96-beta* (voir I.5 ;) pour le français, l'allemand ou l'italien par *Mbrola 3.01h* (voir I.4.)

0.4.2 . – Du texte aux phonèmes

Les outils décrit ci-dessus, pour parler rapidement, n'acceptent en entrée que des phonèmes alors que les couches supérieures fournissent essentiellement du texte. La conversion est effectuée ici encore par *Festival 1.96-beta* (voir I.5,) pour l'anglais et son module *Franfest 1.96-beta-rc01* (voir I.3.) On peut aussi utiliser un outil du type *Cicero*.

0.4.3 . – Une « boîte noire »

Le rôle de *Speech Dispatcher 0.6.1* (voir I.8,) est comme son nom l'indique de répartir convenablement les flux sonores pour éviter les collisions de messages etc.

0.4.4 . – D'où vient l'information ?

À ce point certains choisirons de travailler sous un environnement graphique intégré type *Gnome* ou *kde* sonorisé par un outil de type *Orca* qui sont des choses auxquelles je ne connais rigoureusement rien et à propos desquelles je n'ai rigoureusement rien envie d'apprendre puisque je les considère à peu près comme des contre sens et des anomalies qui n'ont d'autre que but que de rassurer les ex utilisateurs de Windows et de son célèbre Jaws auquel, Dieu m'en garde, je n'ai jamais touché et ne toucherai jamais ni de près ni de loin. « On ne dîne pas avec le Diable, même avec une longue cuillère ! »

La solution qui a été retenue par la plupart de ceux qui se refusaient à la concession graphique, a été d'utiliser *Emacs 21.4a* (voir I.9) comme intégrateur de tâches. Il va sans dire que ce qui semblerait le plus naturel au premier abord, ce serait de rendre le *shell* (type *bash* ou autre) capable de transmettre ses informations aux couches décrites ci-avant. Je ne crois pas trop m'avancer en affirmant que cette idée a sans doute effleuré T.V.Raman le concepteur initial d'*Emacspeak 24*. Mais il me semble qu'il suffit de considérer l'aspect technique des choses, sur lequel je ne m'étendrai pas, pour réaliser que, la tâche à peu près irréalisable en shell devenait sinon aisée du moins praticable sous *Emacs 21.4a* eu égard à la nature de l'interpréteur de langage. Ceci ne devait occasionner aucune restriction pour l'utilisateur dans la mesure où *Emacs* est essentiellement tout aussi capable d'intégrer diverses tâches que le *shell*.

À *Emacspeak* qui fut le précurseur en la matière et reste un outil d'une grande puissance, on préfère *Speechd-el 2.1* (voir I.10,) qui reprend la même philosophie mais dans une implémentation nettement plus moderne et permet surtout, ce qui n'est pas négligeable pour des francophones, une utilisation multilingue ce qui s'avère extrêmement compliqué avec *Emacspeak* qui n'est pas du tout conçu pour ça.

Speechd-el est une application écrite dans le langage interne d'*Emacs* (*emacs-lisp*) qui permet à *Emacs* de communiquer avec *Speech Dispatcher* (une application client/serveur quoi.)

0.5 . – Comment réaliser une installation minimale

J'ai regroupé le système vocal que je vous propose d'installer sous l'acronyme *E.S.D.F.F.M.* pour *Emacs Speech Dispatcher Festival Franfest Mbrola* .

Les instructions d'installation se trouvent donc en I.11. La procédure d'installation est essentiellement du type LFS Book et je serais ravi d'avoir des retour d'expérience relatifs à des installations sous les distributions usuelles dont je ne pratique aucune trouvant extrêmement déplaisante la manière dont les concepteurs desdites distributions se plaisent à réempaqueter les packages si bien que lorsqu'on connaît le contenu des packages d'origine on n'y retrouve jamais ses petits.

I . – Outils principaux

I.1 . – E.F.M. 1.95-beta-rc01

I.1.1 . – Introduction à E.F.M.

E.F.M. est un environnement sonore intégré basé sur *Emacspeak* (voir I.12) *Festival* (voir I.5) et *Mbrola* (voir I.4.)

Il a été développé sur le principe d'un client *Festival* directement implémenté dans *Emacspeak*. Ceci impose de recompiler *Emacspeak* pour l'installer mais toutes ces tâches ont été automatisées ce qui en fait l'outil pour l'instant le plus facile d'installation et d'utilisation parmi ceux présentés sur cette page. Il souffre cependant de nombreux manques, et son fonctionnement n'est pas optimal dans toutes les situations. Il constitue néanmoins une solution "clef en main" pour ceux qui ne se sentiraient pas le courage de se lancer dans l'installation et la configuration manuelles d'un composite de packages.

Ce projet ne verra pas de développement ultérieur dans l'état actuel des choses puisqu'une autre solution (voir I.11) lui a été préférée pour atteindre au même but.

I.1.2 . – Information sur le package

Télécharger : `efm-1.95-beta-rc01.tar.bz2`

I.1.3 . – Dépendances pour E.F.M.

Aucun fichier supplémentaire n'est à rapatrier manuellement puisque toutes les sources nécessaires seront téléchargées par l'installeur automatique contenu dans l'archive ci-dessus.

I.1.4 . – Installation de E.F.M.

Décompresser l'archive par

```
# tar -xjf efm-1.95-beta-rc01.tar.bz2
```

entrer dans le répertoire qui vient d'être créé avec

```
# cd efm-1.95-beta-rc01
```

Ensuite vous avez le choix entre

I.1.4.1. – Une procédure complètement automatique

```
# ./efm_test_fr
```

ou

```
# ./efm_test_fr_noaudio
```

Ou

I.1.4.2. – Une procédure semi automatique

```
# ./configure [options]
```

```
# make
```

suffisent à compiler et installer le package.

I.1.4.3. – Explication des commandes Les commandes décrites en I.1.4.1 installent *E.F.M.* dans le sous-répertoire *efm* du répertoire *home* de l'utilisateur qui fait l'installation. La première commande lance un script qui cherche à savoir si *Festival* est utilisable sur le système et dans ce cas produit un compte rendu sonore (en anglais) de l'installation en cours. Cette méthode étant relativement rudimentaire elle peut considérablement augmenter la durée de l'installation. On peut autrement utiliser la deuxième commande *./efm_test_fr_noaudio* qui ne produira pas de compte rendu sonore mais tout comme la première, un fichier de log *efm.log* dans lequel on pourra identifier les éventuelles erreurs d'installation.

Si on utilise la méthode I.1.4.2, on est amené à utiliser la commande *./configure* qui, si on la lance avec l'option *-help*, produit

I.1.5 . – Contenu de *E.F.M.*

I.2 . – *Fr Sd TTS 1.0*

I.2.1 . –Introduction à *Fr Sd TTS*

Fr Sd TTS est un petit script conçu comme *text2wave* (voir I.5.5.1.) qui reçoit en entrée un flux sous forme de texte et émet en sortie un flux sous forme de données audio ; le texte ayant été synthétisé par *Festival* (voir I.5.)

Ce qui le distingue de *text2wave* est le contrôle d'un plus grand nombre de paramètres et notamment la langue dans laquelle est synthétisé le texte.

En ce sens, il ne fait rien de plus, au contraire, que le client *spd-say* de *Speech Dispatcher* (voir I.8.5.1.) mais il dispense précisément de devoir faire tourner les serveurs *Speech Dispatcher* et *Festival*.

En contrepartie, évidemment, un process *Festival* doit être lancé à chaque opération ce qui augmente considérablement le temps de réponse et n'en fait pas du tout un outils adapté à une synthèse en temps réelle.

Il convient davantage à des applications pour lesquelles le texte une fois synthétisé est stocké sous la forme d'un fichier audio utilisé par la suite. Je ne m'intéresse guère personnellement à ce genre d'applications mais cet outil m'a été demandé et ne représentait pas un travail de programmation considérable.

I.2.2 . –Information sur le package

Télécharger :

- frsdttts-1.0.tar.bz2
- franfest-2006-06.patch
- freebsoft-2006-06.patch

I.2.3 . –Dépendances pour *Fr Sd TTS*

Fr Sd TTS dépend évidemment de *Festival* (voir I.5,) qui est le moteur, mais aussi de *Freebsoft Utils* (voir I.7,) et de manière optionnelle, s'il ont veut du français, de *Franfest* (voir I.3.)

I.2.3.1 ATTENTION! Pour permettre à *Fr Sd TTS* de fonctionner, *Franfest* et *Freebsoft Utils* doivent être patchés. Ce sont hélas des artefacts dus à des questions de mise à jour. Pour *Franfest* surtout, il faut patcher avant de compiler (voir I.3.4 :) Après avoir décompressé l'archive et être entré dans le répertoire source et avant de lancer le script *./configure* , fait

```
# patch -Np1 -i ../franfest-2006-06.patch
```

Pour ce qui concerne *Freebsoft Utils*, après avoir décompressé l'archive et être entré dans le répertoire *festival-freebsoft-utils-0.7*, faire

```
# patch -Np1 -i ../freebsoft-2006-06.patch
```

I.2.4 . –Installation de *Fr Sd TTS*

I.2.5 . –Contenu de *Fr Sd TTS*

I.3 . –*Franfest 1.96-beta-rc01*

I.3.1 . –Introduction à *Franfest*

Franfest est essentiellement un patch de *Festival* (voir I.5) qui permet de disposer d'une synthèse de parole française. *Franfest* peut s'utiliser dans l'environnement de *Festival* mais est utilisé aussi en conjonction avec d'autres outils dans *E.F.M.* (voir I.1) *Fr Sd TTS* (voir I.2) et *E.S.D.F.F.M.* (voir I.11.)

Attention : des informations concernant l'installation de *Franfest* sont disponibles à <http://www.culte.org/proj> mais elles ne sont pas tout à fait à jour et surtout ne correspondent pas aux package que nous vous proposons d'installer ici, mais à des version antérieures.

I.3.2 . –Information sur le package

Télécharger : `franfest-1.96-beta-rc01.tar.bz2`

I.3.3 . –Dépendances pour *Franfest*

Il n'est a priori pas nécessaire de télécharger d'autres archives, même si *Franfest* utilise entre autre *Festival* (y compris lors de la compilation), et *Mbrola* I.4. Ces outils seront téléchargés automatiquement (ou ne le seront pas s'ils sont présents déjà sur le système) lors de l'installation (voir I.3.4.)

I.3.4 . –Installation de *Franfest*

Décompresser l'archive par

```
# tar -xjf franfest-1.96-beta-rc01.tar.bz2
```

entrer dans le répertoire qui vient d'être créé avec

```
# cd franfest-1.96-beta-rc01
```

configurer le package avec

```
# ./configure
```

compiler et installer avec

```
# make
```

ATTENTION ! L'utilisateur qui effectue la commande *make* *DOIT* être autorisé à *ÉCRIRE* dans le répertoire défini par l'option *-prefix* du script *./configure*

ATTENTION ! Il n'y a pas de target *install*. Ceci suffit à configurer, compiler et installer *Franfest* cependant, il est conseillé de prendre connaissance du commentaire de la commande

```
# ./configure --help
```

en I.3.4.1.

I.3.4.1. –Explication des commandes Ci-dessous, les commentaires sont en italique. On suppose que tout ce qui concerne les options standard de *configure* est connu du lecteur ou que ce dernier se reportera aux nombreuses autres sources disponibles les concernant.

```
`configure' configures FranFest 20050705 to adapt to many kinds of sy
```

```
Usage: ./configure [OPTION]... [VAR=VALUE]...
```

```
To assign environment variables (e.g., CC, CFLAGS...), specify them as VAR=VALUE. See below for descriptions of some of the useful variables.
```

À noter que ces directives de configurations ne sont pas utilisables dans le cadre de l'installation de *Franfest* puisque elles ne sont pas implémentées par les scripts correspondants de *Festival*. Ainsi la spécification de la variable *CC* notamment serait sans effet ici et ne vous permettrait pas d'utiliser un autre compilateur *c* que celui fourni par défaut par le système.

```
Defaults for the options are specified in brackets.
```

Configuration:

-h, --help	display this help and exit
--help=short	display options specific to this package
--help=recursive	display the short help of all the included packages
-V, --version	display version information and exit
-q, --quiet, --silent	do not print 'checking...' messages
--cache-file=FILE	cache test results in FILE [disabled]
-C, --config-cache	alias for '--cache-file=config.cache'
-n, --no-create	do not create output files
--srcdir=DIR	find the sources in DIR [configure dir or `..']

Cette option n'est pas active et a été remplacée par des options décrites plus loin.

Installation directories:

`--prefix=PREFIX` install architecture-independent files in PREFIX
[`/usr/local`]

ATTENTION! Si vous n'indiquez aucun prefix explicitement, vérifier bien avant d'effectuer la commande make, que vous êtes autorisé à écrire dans le `/usr/local` ce qui est assez généralement réservé à l'utilisateur root. Si vous ne souhaitez pas effectuer l'installation en tant que root, indiquez `--prefix=<répertoire>` où `<répertoire>` est un répertoire sur lequel vous avez la permission d'écriture.

`--exec-prefix=EPREFIX` install architecture-dependent files in EPREFIX
[PREFIX]

By default, `'make install'` will install all the files in `'/usr/local/bin'`, `'/usr/local/lib'` etc. You can specify an installation prefix other than `'/usr/local'` using `'--prefix'`, for instance `'--prefix=$HOME'`.

For better control, use the options below.

Fine tuning of the installation directories:

`--bindir=DIR` user executables [EPREFIX/bin]
`--sbindir=DIR` system admin executables [EPREFIX/sbin]
`--libexecdir=DIR` program executables [EPREFIX/libexec]
`--datadir=DIR` read-only architecture-independent data [PREFIX/share]
`--sysconfdir=DIR` read-only single-machine data [PREFIX/etc]
`--sharedstatedir=DIR` modifiable architecture-independent data [PREFIX/com
mon]
`--localstatedir=DIR` modifiable single-machine data [PREFIX/var]
`--libdir=DIR` object code libraries [EPREFIX/lib]
`--includedir=DIR` C header files [PREFIX/include]
`--oldincludedir=DIR` C header files for non-gcc [`/usr/include`]
`--infodir=DIR` info documentation [PREFIX/info]
`--mandir=DIR` man documentation [PREFIX/man]

System types:

`--build=BUILD` configure for building on BUILD [guessed]

Optional Packages:

`--with-PACKAGE[=ARG]` use PACKAGE [ARG=yes]
`--without-PACKAGE` do not use PACKAGE (same as `--with-PACKAGE=no`)
`--with-audiofeedback=y|n`
If `'y'` tries to find festival to provide an audio
feedback during the installation (default is `'y'`)

Produit un suivi sonore de l'installation si Festival peut être trouvé sur le système. La méthode employée étant assez rudimentaire, cela peut considérablement ralentir l'installation.

`--with-sources=PATH` Common source directory. Individual directories might

set as well for each package.. (Default is
'\$ac_cv_pln_prefix/src')

Si Festival I.5 et/ou Mbrola I.4 n'est pas installé sur le système mais que les sources sont cependant disponibles le script d'installation les trouvera de lui-même si elle sont dans prefix/src. Cependant si elles sont ailleurs vous pouvez lui indiquer où elles se trouvent grâce à l'option ci-dessus.

--with-owner=OWNER Value of the --owner option passed to install.
(default depends on installation level)
--with-group=GROUP Value of the --group option passed to install.
(default depends on installation level)

Si un utilisateur lambda fait l'installation et que le premier item renvoyé par la commande groups lambda est grecs, ces valeurs seront utilisées par défaut pour les options ci-dessus.

--with-festival=y|n Allows to inhibit explicitly use of festival. Beware
not to use this option if you don't really know what
you are doing. (Default is y)
--with-festivalprefix Sets a particular prefix for festival installation.
(Default is the prefix set by the --prefix option)
--with-festivaldirectory=PATH
Sets directory by hands (not really a user option)
(Default is '\$ac_cv_pln_default_directory_festival')
--with-festivalsources=PATH
Directory where festival sources are available. If
not set will try to get the sources on internet if
necessary. (Default is '\$prefix/src')

Cette option permet de spécifier où se trouvent les sources Festival en particulier si cela ne peut se faire grâce à l'option --with-sources dans le cas notamment où toutes les sources ne seraient pas au même endroit.

--with-franfest=y|n Allows to inhibit explicitly use of franfest. Beware
not to use this option if you don't really know what
you are doing. (Default is y)
--with-franfestprefix Sets a particular prefix for franfest installation.
(Default is the prefix set by the --prefix option)
--with-franfestdirectory=PATH
Sets directory by hands (not really a user option)
(Default is '\$ac_cv_pln_default_directory_franfest')
--with-mbrola=y|n Allows to inhibit explicitly use of mbrola. Beware
not to use this option if you don't really know what
you are doing. (Default is y)
--with-mbrolaprefix Sets a particular prefix for mbrola installation.
(Default is the prefix set by the --prefix option)
--with-mbroladirectory=PATH

```

        Sets directory by hands (not really a user option)
        (Default is '$ac_cv_pln_default_directory_mbrola')
--with-mbrolasources=PATH
        Directory where mbrola sources are available. If
        set will try to get the sources on internet if
        necessary. (Default is '$prefix/src')

```

Permet de spécifier où se trouvent les sources de Mbrola indépendamment des autres sources.

```

--with-mbrolaprogram=PROGNAME
        Sets program by hands (not really a user option).
        (Default is '$ac_cv_pln_default_program_mbrola')
--with-fr1directory=PATH
        Sets directory by hands (not really a user option)
        (Default is '$ac_cv_pln_default_directory_fr1')
--with-fr2directory=PATH
        Sets directory by hands (not really a user option)
        (Default is '$ac_cv_pln_default_directory_fr2')

```

Report bugs to <biglux@culte.org>.

I.3.5 . – Contenu de *Franfest*

I.3.5.1. – Programmes installés *Franfest* n’installe pas d’autre programmes que l’interface utilisateur de *Festival* I.5 c’est-à-dire *festival*, *festival_client*, *festival_server*, *text2wave*. Néanmoins, ceux-ci sont modifiés puisqu’ils ont été recompilés avec *Franfest*. Ceux-ci se trouvent dans <PREFIX>/share/festival/ et un lien symbolique est créé depuis la commande *festival* sur <PREFIX>/bin/festival.

I.3.5.2. – Bibliothèques installées Aucune bibliothèque particulière n’est installée.

I.3.5.3. – Répertoires installés Les répertoires <PREFIX>/share/festival et <PREFIX>/share/speech_tools sont créés.

I.3.5.4. – Courte description Pour s’assurer du bon fonctionnement de *Franfest* il est raisonnable de s’assurer préalablement du bon fonctionnement de *Festival*. Si l’installation à été convenablement effectuée, la commande *festival* est disponible dans le *PATH* (voir I.3.5.1) Avec la commande

```
# festival
```

Vous vous trouverez devant un prompt du genre :

```
Festival Speech Synthesis System 1.95 :beta July 2004 Copyright (C) University
of Edinburgh, 1996-2004. All rights reserved. For details type `(festival_w
festival>
```

La commande « SayText » comme ci-dessous

```
festival> (SayText "Hello") #<Utterance 0xb6f71588> festival>
```

doit produire un output du genre de celui ci-dessus ainsi que le mot « Hello » qui doit être prononcé.

Pour charger *Franfest* (ce qui n'est peut-être pas strictement nécessaire suivant la version dont vous disposez mais « abondance de bien ne nuit pas ! »)

```
festival> (require 'franfest) t festival>
```

Sélectionnez ensuite la voix française masculine avec

```
festival> (voice_fr1_mbrola) fr1_mbrola festival>
```

Puis

```
festival> (SayText "Bonsoir") #<Utterance 0xb6fd65d8> festival>
```

doit vous faire entendre « bonsoir » avec une voix française.

Enfin

```
festival> (voice_fr2_mbrola) fr2_mbrola festival> (SayText "Bonsoir")  
#<Utterance 0xb6ff3488> festival>
```

doit vous faire entendre le même « bonsoir » avec une charmante voix d'hôtesse de l'air (enfin n'exagérons rien !)

Si ces tests vous semblent concluants, vous pouvez raisonnablement penser que *Franfest* est convenablement installé.

I.4 . – *Mbrola 3.01h*

I.4.1 . – Introduction à *Mbrola*

I.4.2 . – Information sur le package

Télécharger :

- dba/fr1/fr1-990204.zip
- dba/fr2/fr2-980806.zip
- dba/de1/de1-980227.zip
- dba/de2/de2-990106.zip
- bin/pclinux/mbr301h.zip

I.4.3 . – Dépendances pour *Mbrola*

I.4.4 . – Installation de *Mbrola*

I.4.5 . – Contenu de *Mbrola*

I.5 . –*Festival 1.96-beta*

I.5.1 . –Introduction à *Festival*

Festival est un environnement intégré permettant de piloter de nombreuses *synthèses vocales*. Je renvoi à *Festival 1.96-beta* pour de plus amples détails ; mon propos n'étant pas ici de décrire les innombrables possibilités de *Festival* dans une utilisation indépendante. Je ne le considère en effet que sous l'aspect de la synthèse vocale en Français (ou tout au moins multilingue) à travers *Franfest* (voir I.3.)

I.5.2 . –Information sur le package

Télécharger :

- festival-1.96-beta.tar.gz
- festlex_CMU.tar.gz
- festlex_POSLEX.tar.gz
- festvox_kallpc16k.tar.gz
- speech_tools-1.2.96-beta.tar.gz

I.5.3 . –Dépendances pour *Festival*

Festival ne nécessite a priori pas l'installation de packages particuliers. Cependant, il faut prendre garde au fait qu'on risque de rencontrer des difficultés de compilations avec des *gcc* récents et plus précisément, des versions supérieures à 3.4. Il est possible qu'il existe des patches permettant de surmonter ces problèmes et je serais reconnaissant à quiconque pourrait *me les signaler*.

I.5.4 . –Installation de *Festival*

Puisque je n'envisage en fait l'installation de *Festival* que dans le cadre de *Franfest* (voir I.3.) et que celle-ci s'effectue alors automatiquement, je ne donnerai qu'un bref aperçu ici. Les archives indiquées au paragraphe I.5.2 constituent un ensemble minimal permettant de compiler un système *Festival* fonctionnel. Il suffit de les décompresser avec *tar -xzf* , ce qui crée deux sous-répertoires du répertoire courant *speech_tools* et *festival*. Il suffit ensuite, dans chacun d'entre eux, et dans cet ordre, d'effectuer les commandes

```
# ./configure
```

et

```
# make
```

À noter qu'il n'y a pas de *target install* dans le *Makefile* ; les exécutables et les bibliothèques restent donc dans le répertoire où les archives ont été décompressées.

I.5.4.1. – Explication des commandes On se reportera avec profit à la page *Festival 1.96-beta*, où l'on disposera de nombreuses sources d'information quant à l'installation et à l'utilisation de *Festival*.

Cependant, on peut mentionner ici que contrairement à ceux de nombreux autres packages, les scripts *./configure* de *Festival* et *speech_tools* ne permettent pas d'utiliser l'option `CC=` pour sélectionner un autre compilateur que celui par défaut.

I.5.5 . – Contenu de *Festival*

I.5.5.1. – Programmes installés Dans le sous-répertoire *bin* du répertoire *festival* créé lors de l'installation (voir I.5.4,) se trouvent les exécutables, ou liens symboliques vers des exécutables suivants :

- *festival* : le moteur de *Festival* lui-même.
- *festival_client*.
- *festival_server* : un script permettant de lancer *Festival* en mode serveur.
- *festival_server_control*.
- *text2wave* : transforme, par *Festival* un texte en fichier audio pouvant être joué ensuite. Voir *Fr Sd TTS* (I.2) qui est une extension de ce programme permettant de configurer davantage de paramètres et notamment la langue.

I.5.5.2. – Bibliothèques installées Aucune.

I.5.5.3. – Répertoires installés Aucun répertoire n'est installé au sens habituel du terme, mais les répertoires *festival* et *speech_tools* sont créés qui contiennent respectivement et grossièrement l'interface et le moteur de *Festival*.

I.5.5.4. – Courte description Nous conseillons évidemment de se reporter à la documentation en ligne de *Festival*.

Cependant ce document est conséquent et sans doute un peu compliqué pour un débutant et concerne d'ailleurs bien plus que les aspects d'une simple utilisation de *Festival*.

Nous allons simplement décrire ici la manière de faire fonctionner *Festival* en *mode serveur* ce qui est **indispensable** pour une utilisation conjointe avec *Speech Dispatcher* (voir I.8) et par conséquent avec *Speechd-el* (voir I.10.)

A priori la commande

```
# festival_server
```

suffit à lancer *Festival* en mode serveur. Cependant :

- Il se peut que le *script festival_server* ne soit pas accessible dans le *PATH* puisque, si aucun lien symbolique n’a été fait, il se trouve dans le sous-répertoire *bin* de *Festival* c’est-à-dire dans */usr/local/share/festival/bin* dans le cas d’une installation par défaut. On devra donc entrer la commande :

```
# /usr/local/share/festival/bin/festival_server \  
/usr/local/share/festival/bin/festival &
```

- Ensuite, *Festival* doit disposer d’un répertoire où placer certains fichiers temporaires. Le plus naturel est d’utiliser le répertoire */tmp* du système. Vous indiquerez ce choix à *Festival* en complétant la commande ci-dessus par

```
# /usr/local/share/festival/bin/festival_server -l /tmp \  
/usr/local/share/festival/bin/festival &
```

- Je recommanderais alors, si la machine dispose de suffisamment de mémoire, de monter */tmp* en *tmpfs* ; ce qui pourrait augmenter appréciablement la réactivité de *Festival*.
- Enfin le *script festival_server* souffre de certaines imperfections qui laisseront certains messages d’erreur être envoyés essentiellement n’importe où. Pour remédier à ce défaut, je conseillerais d’ajouter encore à la commande de démarrage du serveur :

```
# /usr/local/share/festival/bin/festival_server -l /tmp \  
/usr/local/share/festival/bin/festival \  
2>/var/log/festival.log &
```

Il est évidemment recommandé de démarrer le serveur de *Festival* au démarrage de la machine mais, la variabilité des protocoles relativement aux distributions rend malaisée la description d’une méthode.

I.6 . –*Sound Icons 0.1*

I.6.1 . –Introduction à *Sound Icons*

Un certain nombre d’icônes sonores utilisées par *Freebsoft Utils* (voir I.7) et partant par *Speech Dispatcher* (voir I.8.)

I.6.2 . –Information sur le package

Télécharger : [sound-icons-0.1.tar.gz](#)

I.6.3 . –Dépendances pour *Sound Icons*

I.6.4 . –Installation de *Sound Icons*

Il semble que les fichiers contenus dans l’archive [sound-icons-0.1.tar.gz](#) doivent être accessibles dans le répertoire */usr/share/sounds/sound-icons/*.

Décompressez donc cette archive par

```
# tar -xzf sound-icons-0.1.tar.gz
```

et copiez les fichiers dans le répertoire mentionné ci-dessus.

I.6.5 . –Contenu de *Sound Icons*

I.7 . –*Freebsoft Utils 0.7*

I.7.1 . –Introduction à *Freebsoft Utils*

Freebsoft Utils est une collection d'utilitaires *Festival* (voir I.5) qui étendent les possibilités de *Festival*. Ils fournissent notamment toutes les fonctionnalités nécessaires à l'interaction avec *Speech Dispatcher* (voir I.8.)

I.7.2 . –Information sur le package

Télécharger : `festival-freebsoft-utils-0.7.tar.gz`

Attention, les versions antérieures à 0.7 présentent des incompatibilités avec le français.

I.7.3 . –Dépendances pour *Freebsoft Utils*

Dans la mesure où *Freebsoft Utils* est une extension de *Festival* écrite dans le langage interne de *Festival* (*Scheme*.) dans le but d'étendre les possibilités de *Festival*, il n'y a aucun sens à l'installer sans *Festival* (voir I.5.) En revanche, aucune modification du code source et conséquemment aucune recompilation de *Festival* ne sont nécessaires.

Les outils *Recode* (voir II.5) et *Sox* (voir II.6) sont nécessaires ainsi que le package *Sound Icons* (voir I.6.)

I.7.4 . –Installation de *Freebsoft Utils*

Après avoir décompressé l'archive `festival-freebsoft-utils-0.7.tar.gz` avec la command

```
# tar -xzf festival-freebsoft-utils-0.7.tar.gz
```

entrer dans le répertoire `festival-freebsoft-utils-0.7` par la commande

```
# cd festival-freebsoft-utils-0.7
```

Le *Makefile* de ce paquet est encore très incomplet et ne peut être que partiellement utilisé. Cependant si vous voulez disposer de la documentation au format *info*, il est recommandé de faire

```
# make info
```

Ensuite copiez tous les fichiers *.scm* dans un répertoire du *load-path* de *Festival*.

Vous pouvez, par exemple, créer un sous-répertoire du répertoire */lib* de *Festival* (*/usr/local/share/festival/lib*) si vous avez effectué l'installation par défaut. Placez-vous dans ce répertoire et créez un sous-répertoire « *freebsoft* » apr :

```
# mkdir freebsoft
```

Ajoutez ensuite la ligne

```
(set ! load-path (cons (string-append libdir "/freebsoft") load-path)
```

à la fin du fichier *siteinit.scm* situé dans le sous-répertoire *lib* de *Festival* c'est-à-dire */usr/local/share/festival/lib* dans le cas de l'installation par défaut.

I.7.5 . –Contenu de *Freebsoft Utils*

I.7.5.1. – Programmes installés Aucune commande spécifique accessible depuis le *shell* n'est installée.

I.7.5.2. – Bibliothèques installées Les bibliothèques écrites en *Scheme* le langage interne de *Festival* sont installées comme décrit en I.7.4.

I.7.5.3. – Répertoires installés Voir I.7.4.

I.7.5.4. – Courte description *Freebsoft Utils* est avant tout conçu pour être un support à *Speech Dispatcher* (voir I.8,) et est aussi nécessaire pour faire fonctionner *Fr Sd TTS* (voir I.2.)

Néanmoins, les utilitaires implémentés par *Freebsoft Utils* peuvent être utilisés dans le mode interactif de *Festival*. Voir I.3.4.1 pour entrer dans ce mode. On peut alors faire

```
festival> (require 'speech-dispatcher) t festival>
```

pour charger ces utilitaires (et notamment ceux relatifs à *Speech Dispatcher*) dans mode interactif de *Festival*. Si la sortie n'est pas « t » comme ci-dessus, c'est que les bibliothèques de *Freebsoft Utils* ne sont pas accessibles sans doute pour cause de mauvaise configuration. Il faut alors se reporter au paragraphe I.7.4. Il est d'abord recommandé de s'assurer que l'anglais est bien disponible par

```
festival> (speechd-set-language 'en) ISO-8859-1 festival>
```

puis qu'il est bien fonctionnel par

```
festival> (utt.play (speechd-speak* "Hello")) #<Utterance 0xb7594f98>
festival>
```

qui doit faire entendre « Hello ».

On doit ensuite s'assurer que le français est également disponible par

```
festival> (speechd-set-language 'fr) ISO-8859-1 festival> (utt.play (speechd-speak*
"Bonjour")) #<Utterance 0xb75ec7c8> festival>
```

doit faire entendre « Bonjour » avec une voix française.

Vous pouvez tester d'autres réglages comme la vitesse d'élocution par exemple :

```
festival> (speechd-set-rate 50) 1 festival> (utt.play (speechd-speak*
"Bonjour")) #<Utterance 0xb7594438> festival>
```

produit le même « Bonjour » mais prononcé plus rapidement et :

```
festival> (speechd-set-rate -50) 1.41421 festival> (utt.play (speechd-speak*
"Bonjour")) #<Utterance 0xb75e2318> festival>
```

produit le même « Bonjour » mais cette fois plus lentement.

Le maximum de vitesse est obtenu 100 et le minimum pour -100 tandis que la vitesse moyenne correspond à 0.

Pour une revue complète de tous les réglages possibles, reportez-vous au fichier *festival-freebsoft-utils.info* du sous-répertoire *festival-freebsoft-utils-0.7* (voir I.7.4.)

À noter qu'il n'est pas nécessaire d'approfondir l'apprentissages des commandes internes de *Freebsoft Utils* si l'on projette une utilisation de *Speechd-el* (voir I.10) sous *Emacs* (voir I.9.)

Si les tests ci-dessus se sont avérés concluants, vous pouvez considérer que *Freebsoft Utils* est correctement installé et configuré.

I.8 . – *Speech Dispatcher 0.6.1*

I.8.1 . – Introduction à *Speech Dispatcher*

Speech Dispatcher est une application *client/serveur* qui se comporte comme *serveur* vis-à-vis des applications et comme *clients* vis-à-vis des systèmes de synthèse vocale. Il permet à plusieurs applications d'utiliser plusieurs systèmes de synthèse vocale de manière multilingue. Il optimise autant que faire se peut la gestion des flux sonores.

I.8.2 . – Information sur le package

Télécharger : `speech-dispatcher-0.6.1.tar.gz`

I.8.3 . – Dépendances pour *Speech Dispatcher*

- *GLib* II.3
- *DotConf* II.1
- *Pkg-Config* II.4

I.8.4 . –Installation de *Speech Dispatcher*

Décompresser l'archive par

```
# tar -xzf speech-dispatcher-0.6.1.tar.gz
```

entrer dans le répertoire qui vient d'être créé avec

```
# cd speech-dispatcher-0.6.1
```

configurer le package avec

```
# ./configure
```

compiler avec

```
# make
```

ATTENTION ! Pour installer, vous *DEVEZ* être autorisé à *ÉCRIRE* dans le répertoire défini par l'option `–prefix` du script `./configure`

Tapez :

```
# make install
```

Vous devez ensuite configurer *Speech Dispatcher* en fonction de vos besoins. Vous pouvez vous reporter à la documentation fournie dans le package mais je vous propose un modèle de configuration `sdconf-0.6.1.tar.gz` que vous pouvez décompresser dans le répertoire correspondant à l'option `–localstatedir` de `./configure` (`/usr/local/etc/` par défaut.)

I.8.5 . –Contenu de *Speech Dispatcher*

I.8.5.1. – Programmes installés *Speech Dispatcher* installe essentiellement les programmes *speech-dispatcher* et *spd-say* dans le sous-répertoire *bin* du répertoire de l'installation déterminé par l'option `–prefix` du script `./configure` (`/usr/local/bin` par défaut.)

I.8.5.4. – Courte description Avant de lancer *Speech Dispatcher* qui installe une application *client/serveur* il est **indispensable** de démarrer le serveur *Festival* (voir I.5.5.4) dans le cadre d'une utilisation conjointe avec *Festival*.

Ensuite la commande :

```
# speech-dispatcher
```

lance l'application.

Le programme *spd-say* est un *client* rudimentaire pour *Speech Dispatcher* et qui permet au moins de savoir si *Speech Dispatcher* a démarré correctement et s'est correctement connecté à *Festival*. La commande

```
# spd-say "Hello"
```

doit vous faire entendre « Hello ». Et la commande

```
# spd-say -l fr "Bonjour"
```

doit faire entendre « Bonjour » avec la voix française. D'autres réglages plus fins sont disponibles par *spd-say* mais nous n'envisageons ici que son utilisation que comme test et si les deux commandes précédentes ont réussi, on peut considérer à juste titre que « tout va bien ! »

I.9 . – Emacs 21.4a

I.9.1 . – Introduction à Emacs

Emacs est un logiciel suffisamment connu pour qu'on ne donne pas de détails. Il est installé sur la plupart des systèmes et on utilise une installation tout à fait standard aussi on ne donne aucun détail quant à son installation pour l'instant.

I.9.2 . – Information sur le package

Télécharger : `emacs-21.4a.tar.gz`

I.9.3 . – Dépendances pour Emacs

I.9.4 . – Installation de Emacs

I.9.5 . – Contenu de Emacs

I.10 . –*Speechd-el* 2.1

I.10.1 . –Introduction à *Speechd-el*

Speechd-el permet à *Emacs* (voir I.9) de se comporter comme *client* vis-à-vis de *Speech Dispatcher* (voir I.8) et donc de devenir un environnement sonore multilingue intégré. *Speechd-el* est écrit en *emacs-lisp* le langage de programmation interne d'*Emacs*.

I.10.2 . –Information sur le package

Télécharger : `speechd-el-2.1.tar.gz`

I.10.3 . –Dépendances pour *Speechd-el*

Il va sans dire qu'il n'y a absolument aucun intérêt à installer ce package sans *Emacs* I.9. Vous devez aussi installer le package *Eieio* (voir II.2.)

I.10.4 . –Installation de *Speechd-el*

L'installation de ce package doit se faire manuellement. Il faut qu'*Emacs* puisse trouver les fichiers `.el` constituant le code. Ce genre de fichiers sont généralement placés dans un sous-répertoire de `/usr/local/share/emacs/site-lisp/` ou `/usr/share/emacs/site-lisp/`.

Vous pouvez par exemple copier l'archive `speechd-el-2.1.tar.gz` dans l'un des deux répertoires ci-dessus puis la décompresser avec

```
# tar -xzf speechd-el-2.1.tar.gz
```

ce qui crée un sous-répertoire `speechd-el-2.1`.

Pour indiquer ensuite à *Emacs* où avoir accès aux fichiers de *Speechd-el* placez dans votre fichier `.emacs`

```
(add-to-list 'load-path "/usr/local/share/emacs/site-lisp/speechd-el-2.1")
```

si vous avez choisi l'option « locale » ou

```
(add-to-list 'load-path  
"/usr/share/emacs/site-lisp/speechd-el-2.1")
```

sinon.

I.10.5 . –Contenu de *Speechd-el*

I.10.5.1. –Programmes installés Aucun programme autonome n'est installé par ce paquet.

I.10.5.2. –Librairies installées Les librairies écrites en *emacs-lisp* de ce paquet se trouvent dans */usr/local/share/emacs/site-lisp/speechd-el-2.1* ou */usr/share/emacs/site-lisp/speechd-el-2.1* suivant l'installation que vous avez choisie (voir I.10.4.)

I.10.5.3. –Répertoires installés Voir I.10.5.2

I.10.5.4. –Courte description Pour que *Speechd-el* soit actif au moment du lancement d'*Emacs* vous pouvez placer dans le fichier *.emacs* après les commandes déjà décrites en I.10.4 la séquence de commandes suivante :

```
(load-library "speechd-speak")
(speechd-speak)
(load-library "speechd-ssip")
```

Lancez ensuite *Emacs* par la commande

```
# emacs
```

et il **PARLE** ! (euh du moins j'espère !)

I.11 . –*E.S.D.F.F.M. 0.96*

I.11.1 . –Introduction à *E.S.D.F.F.M.*

Cette page contient, en attendant mieux, c'est-à-dire un utilitaire d'installation intégré pour les divers packages constituant le système *E.S.D.F.F.M.*, des indications aussi détaillées que possible pour l'installation desdits packages.

Ces packages constituent un système multilingue intégré basé sur *Emacs* (voir I.9) *Multilingue* signifie, dans ce contexte, nettement plus que la possibilité d'obtenir un feedback sonore dans telle ou telle langue, puisque le système convenablement configuré, permet que la langue adéquate soit utilisée en fonction du texte qui doit être prononcé.

Observons, que nous nous restreignons d'emblée au cas d'une installation pour francophones c'est-à-dire pour un système fonctionnant en anglais et français.

I.11.2 . –Information sur le package

Télécharger :

–

I.11.3 . –Dépendances pour *E.S.D.F.F.M.*

- *Franfest* I.3
- *Sound Icons* I.6
- *Freebsoft Utils* I.7
- *Speech Dispatcher* I.8
- *Speechd-el* I.10
- *Emacs* I.9

I.11.4 . –Installation de *E.S.D.F.F.M.*

L'endroit où chacun souhaite installer ces paquets regarde chacun mais je formulerais cependant quelques observations : Personnellement je les installe dans */usr* suivant en cela les recommandations du LFS Book mais je conçois tout à fait qu'on veuille séparer la maintenance de ces paquets particuliers de la maintenance du reste du système. Le meilleur compromis me semble alors */usr/local* plutôt que le répertoire de l'utilisateur qui pourrait être plus vulnérable à de fausses manoeuvre de la part d'un utilisateur peu expérimenté. De plus le choix de */usr/local* étant toujours le choix par défaut, cela évite de passer aucune option supplémentaire à *./configure* .

L'ordre dans lequel les packages sont installés est, le plus souvent, sans importance sauf mention explicite du contraire mais je vous propose de suivre le schéma suivant en vous référant à chaque fois au paragraphe adéquat :

- *Franfest* I.3 ;
- *Sound Icons* I.6
- *Freebsoft Utils* I.7 ;
- *Speech Dispatcher* I.8 ;
- *Emacs* I.9 ;
- *Speechd-el* I.10.

I.11.5 . –Contenu de *E.S.D.F.F.M.*

I.11.5.4. –Courte description Pour disposer de l'environnement sonore intégré sous *Emacs* vous devez démarrer **successivement** :

- *Festival* en mode serveur (voir I.5.5.4 ;)
- *Speech Dispatcher* (voir I.8.5.4 ;)
- puis *Emacs* configuré avec la librairie *Speechd-el* (voir I.10.5.4.)

I.12 . –*Emacspeak 24*

I.12.1 . –Introduction à *Emacspeak*

I.12.2 . –Information sur le package

Télécharger : emacspeak-24.tar.bz2

I.12.3 . –Dépendances pour *Emacspeak*

I.12.4 . –Installation de *Emacspeak*

I.12.5 . –Contenu de *Emacspeak*

II . –Packages annexes

II.1 . –*DotConf 1.0.13*

II.1.1 . –Introduction à *DotConf*

II.1.2 . –Information sur le package

Télécharger : dotconf-1.0.13.tar.gz

II.1.3 . –Dépendances pour *DotConf*

II.1.4 . –Installation de *DotConf*

Il se peut que le package *DotConf* soit déjà installé. Après vous en être assuré, vous pouvez simplement ignorer ce paragraphe.

II.1.5 . –Contenu de *DotConf*

II.2 . –*Eieio 0.17*

II.2.1 . –Introduction à *Eieio*

Eieio un paquet écrit en *emacs-lisp* permettant de disposer de possibilités de programmation objet en *emacs-lisp*.

II.2.2 . –Information sur le package

Télécharger : [eieio-0.17.tar.gz](#)

II.2.3 . –Dépendances pour *Eieio*

Ce paquet ne peut fonctionner qu'avec *Emacs* (voir I.9.)

II.2.4 . – Installation de *Eieio*

L'installation de ce package doit se faire manuellement. Il faut qu'*Emacs* puisse trouver les fichiers .el constituant le code. Ce genre de fichiers sont généralement placés dans un sous-répertoire de */usr/local/share/emacs/site-lisp/* ou */usr/share/emacs/site-lisp/*.

Vous pouvez par exemple copier l'archive *eieio-0.17.tar.gz* dans l'un des deux répertoires ci-dessus puis la décompresser avec

```
# tar -xzf eieio-0.17.tar.gz
```

ce qui crée un sous-répertoire *eieio-0.17*.

Pour indiquer ensuite à *Emacs* où avoir accès aux fichiers de *Eieio* placez dans votre fichier *.emacs*

```
(add-to-list 'load-path "/usr/local/share/emacs/site-lisp/eieio-0.17")
```

si vous avez choisi l'option « locale » ou

```
(add-to-list 'load-path  
"/usr/share/emacs/site-lisp/eieio-0.17")
```

sinon.

II.2.5 . – Contenu de *Eieio*

II.3 . – *GLib 2.9.1*

II.3.1 . – Introduction à *GLib*

II.3.2 . – Information sur le package

Télécharger : *glib-2.9.1.tar.bz2*

II.3.3 . –Dépendances pour *GLib*

II.3.4 . –Installation de *GLib*

Il se peut que le package *GLib* soit déjà installé. Après vous en être assuré, vous pouvez simplement ignorer ce paragraphe.

II.3.5 . –Contenu de *GLib*

II.4 . –*Pkg-Config 0.20*

II.4.1 . –Introduction à *Pkg-Config*

II.4.2 . –Information sur le package

Télécharger : [pkg-config-0.20.tar.gz](#)

II.4.3 . –Dépendances pour *Pkg-Config*

II.4.4 . –Installation de *Pkg-Config*

Il se peut que le package *Pkg-Config* soit déjà installé. Après vous en être assuré, vous pouvez simplement ignorer ce paragraphe.

II.4.5 . –Contenu de *Pkg-Config*

II.5 . –*Recode* 3.6

II.5.1 . –Introduction à *Recode*

II.5.2 . –Information sur le package

Télécharger : `recode-3.6.tar.gz`

II.5.3 . –Dépendances pour *Recode*

II.5.4 . –Installation de *Recode*

Il se peut que le package *Recode* soit déjà installé. Après vous en être assuré, vous pouvez simplement ignorer ce paragraphe.

II.5.5 . –Contenu de *Recode*

II.6 . –*Sox 12.17.7*

II.6.1 . –Introduction à *Sox*

II.6.2 . –Information sur le package

Télécharger : `sox-12.17.7.tar.gz`

II.6.3 . –Dépendances pour *Sox*

II.6.4 . –Installation de *Sox*

Il se peut que le package *Sox* soit déjà installé. Après vous en être assuré, vous pouvez simplement ignorer ce paragraphe.

II.6.5 . –Contenu de *Sox*

III . –Tutoriels

III.0 . – Introduction

Je ne prétends pas donner plus, ici et maintenant, qu’une esquisse de tutoriel pour l’utilisation d’*Emacs 21.4a* dans un environnement sonore. Deux options sont possibles :

- *Speechd-el 2.1* III.2 et
- *Emacspeak 24* III.3.

Signalons d’emblée que j’ai choisi actuellement la première plutôt que la deuxième solution parce qu’*Emacspeak* en dépit de ses innombrables qualités, présente le défaut d’être très mal adapté à une utilisation multilingue. *Speechd-el* en revanche, a été conçu dans l’optique de l’utilisation conjointe de plusieurs langues.

En conséquence, le document relatif à *Emacspeak* n’a pas été mis à jour sinon un nettoyage purement technique consistant à réactualiser les références.

Les deux outils mentionnés plus haut, ont tout de même ceci en commun qu’ils sont des « extensions » d’*Emacs 21.4a* et que, par conséquent, leur utilisation, requiert quelques notions quant à l’utilisation d’*Emacs* lui-même. Nous donnons quelques conseils, que nous espérons les plus utiles possibles dans la sections III.1.

III.1 . – Utilisation élémentaire de *Emacs 21.4a*

III.1.0 . – Avant-propos

Rappelons qu’*Emacspeak* est un programme *Emacs* ; autrement dit qu’en lançant *Emacspeak* vous lancez en fait *Emacs* qui exécute le programme *Emacspeak*. Par conséquent vous utilisez une interface sonore de *Emacs*. Certaines commandes sont spécifiques d’*Emacspeak* comme nous en avons déjà mentionné certaines dans la section III.3.6.2 et à propos desquelles nous donnerons plus de détails dans la section III.3.

D’autres sont simplement des commandes de *Emacs* qui sont adaptées par *Emacspeak* afin de produire, en plus de leur destination originelle un retour sonore comme nous l’avons aperçu dans la section III.3.6.1.

Dorénavant, nous ne distinguerons que s’il y a vraiment lieu, entre ces deux types de commandes et nous nous bornerons à parler des commandes de *Emacs* en prenant soin à chaque fois de spécifier quel type de retour sonore est produit par *Emacspeak*.

Le logiciel *Emacs* est certes, au départ, un *éditeur de texte* mais son langage de programmation *emacs-lisp* a permis aux développeurs de le doter d’un grand nombre de possibilités supplémentaires disponibles dans une installation standard. Ainsi, *Emacs* est amené à manipuler un certain nombre d’*objets* différents dans un certain nombre de *contextes* différents.

- Ce que nous appelons *objet* est assez proche de ce que les programmeurs appellent *objet* dans le cadre des langages *objet*. Ceux à qui cette notion ne serait pas du tout familière, peuvent se faire une très bonne idée de ce que je veux dire en se référant à la notion usuelle, tout comme on a une très bonne idée de la théorie élémentaire des ensembles sans jamais avoir lu Bourbaki ou de la prose sans avoir eu le moindre maître de rhétorique ! Il s’agit d’une entité sur laquelle peuvent être effectuées un certain nombre d’opérations. Dans ce sens, un *caractère* (cf. III.1.1.e) est un *objet* qui peut être inséré, effacé, déplacé etc ... C’est un *sous-objet* d’un *mot* lequel peut également être inséré, effacé, déplacé ... suivant la situation dans laquelle on se trouve.

- La notion de *contexte* s'apparente à la notion de *mode* sous *Emacs* (cf. III.1.8,) mais le terme de *contexte* est peut-être plus intuitif. Une même commande, dont le but est de modifier un *objet* donné peut, ou non s'appliquer suivant les circonstances. Par exemple, si vous avez choisi de lire cette documentation dans son format *info* comme indiqué à la section III.3.2.2.2 vous constaterez que vous ne pouvez procéder à l'insertion d'un *caractère* comme indiqué dans la section III.1.1.

J'ai également dans tout ce paragraphe, parlé de *commandes*. *Emacs* étant, au moins en fonctionnement normal, un logiciel interactif, une fois qu'il est lancé, même s'il exécute un certain nombre de tâches sans intervention de l'*utilisateur*, il attend cependant de celui-ci un certain nombre d'instructions. Nous verrons en détail dans la section III.1.2.

III.1.1 . – Caractères

Répetons-le une dernière fois, *Emacs* étant, à l'origine au moins, un *éditeur de texte*, l'*objet* de base est bien évidemment le *caractère*. En pressant n'importe laquelle des touches du clavier correspondant à un *caractère* alphabétique « a b c A B C » etc, celui-ci est inséré dans le texte courant à la position où se trouve le *curseur*.

ATTENTION !

Si vous avez choisi de lire la version *info* ou *html* de cette documentation avec *Emacspeak* comme indiqué dans les sections III.3.2.2.2 et III.3.2.2.3 vous vous trouvez déjà dans une situation particulière où les manipulations précédentes ne sont pas possibles ! Veuillez m'excuser par avance, pour ce que vous pourrez assimiler à un manque de pédagogie. Cependant, vous touchez d'emblée au coeur de la spécificité de *Emacs* ! Vous remarquez que, si vous êtes en train de lire la version *info* du tutoriel et si vous pressez *u*, vous n'allez pas insérer un *caractère* “u” dans le texte que vous êtes en train de lire, mais remonter d'un niveau dans son arborescence. Sans détailler le fonctionnement des *modes* sous *Emacs* comme je le ferai un peu plus dans la section III.1.8, disons que le processus qui vous a conduit à charger la version *info* du tutoriel a réaffecté la touche « u » à une autre fonction que l'*insertion* du *caractère* « u ».

Si donc vous voulez bénéficier des possibilités d'*insertion* de *caractères* il vous faut changer de *buffer actif* (cf. III.1.4.)

Les plus hardis d'entre vous peuvent tenter la manipulation suivante :

- Pressez d'abord la séquence *Ctrl-x b*. Un message prononcé par *Emacspeak* vous demandera d'indiquer un nom de *buffer*.
- Utilisez alors les touches alphabétiques du clavier pour saisir un nom. Tiens c'est possible maintenant ? Oui car le *buffer actif* est désormais *minibuffer* (cf. III.1.5o) où l'*insertion* de *caractères* est permise et même recommandée !
- Pressez ensuite la *touche* <ENTRÉE> pour terminer votre saisie.
- Vous pouvez alors taper quelques *caractères*.
- Si vous utilisez la version *info* du tutoriel que vous avez lancée avec le script *info-start* comme expliqué en III.3.2.2.2 la séquence *Ctrl-h i* vous ramènera à la lecture du dis tutoriel, là où vous l'avez laissée. Si vous utilisez une version *html* très vraisemblablement, en pressant à nouveau *Ctrl-x b* puis en pressant la *touche* <ENTRÉE> sans saisir aucun nom, c'est-à-dire en sélectionnant le nom du *buffer* par défaut, qui doit être celui où vous vous trouviez précédemment vous reviendrez également à votre lecture, là où vous l'aviez laissée. Je ne peux, dans cette deuxième situation, pas garantir à cent pourcents, le succès de l'opération ; celle-ci pouvant dépendre du

navigateur internet que vous avez adjoint à *Emacspeak* pour lire ce document (voir la section III.3.8.)

Si donc vous êtes dans une situation où l'*insertion* de caractères est possible, ce qui sera le cas si vous avez simplement lancé *Emacspeak*, vous obtiendrez un écho sonore soit *caractère* à caractère, soit *mot* à mot (suivant les réglages d'*Emacspeak* (cf. III.3.6.) Ce dernier écho est obtenu lorsque vous pressez la *touche* <ESPACE> après avoir tapé un certain nombre de caractères qui sont alors regroupés et prononcés ensembles. Le type de résultat sonore obtenu dépend évidemment très fortement de la *synthèse vocale* que vous utilisez. Vous ne pouvez par exemple pas espérer qu'*Emacspeak* obtienne de votre *synthèse vocale* une prononciation à la française si vous utilisez une *synthèse vocale* anglophone ou réciproquement.

III.1.2 . – Les commandes

Nous avons déjà amplement mentionné les *commandes* de *Emacs* sans expliquer tout à fait systématiquement comment entrer une *commande* et à quoi elles servent. Ne comptez évidemment pas sur ce texte pour répondre exhaustivement à la deuxième partie de la question ! d'autant plus qu'on peut à volonté, fabriquer soi-même des *commandes Emacs* III.1.7. À l'exception de la *commande* d'*insertion* d'un *caractère* dont nous avons déjà parlé dans la section III.1.1, et qu'on pourrait il est vrai considérer séparément des autres, il y a essentiellement trois manières de faire exécuter une *commande* par *Emacs* :

III.1.2.1 À partir d'un menu Certaines *commandes Emacs* peuvent être lancées à partir de *menus*. En pressant la touche *F10*, par exemple, vous ouvrez le *menu principal* dont tous les items seront prononcés par *Emacspeak* jusqu'à ce que vous pressiez à nouveau une touche. Celle-ci peut être, soit un *caractère* correspondant à l'un des items du *menu* qui sera alors sélectionné (on voit que lorsque l'objet manipulé par *Emacs* est un *menu* l'*insertion* de *caractère* n'a évidemment pas de sens), soit l'une des flèches de déplacement vertical *touche* <FLÈCHE HAUT> ou *touche* <FLÈCHE BAS> lesquelles provoquent le déplacement du *curseur* d'un item vers le haut ou le bas. Après chacun de ces déplacement, l'item du *menu* sur lequel se trouve le *curseur* est prononcé par *Emacspeak*. En pressant ensuite la *touche* <ENTRÉE> vous sélectionnez l'item et effectuez la *commande* correspondante.

En ouvrant le *menu principal* avec *F10*, vous constaterez que le premier item est *file*. En pressant *touche* <ENTRÉE> ou l'initiale *f*, vous ouvrirez un nouveau *menu* dont le premier item est *Open File*. En pressant *touche* <ENTRÉE> ou *o*, vous exécutez une *commande* pour laquelle *Emacs* a besoin de connaître un nom de *fichier* qu'il vous demande d'entrer grâce au *minibuffer*. La suite des opérations relève de la section III.1.6.

Vous pouvez également sélectionner l'item *HELP* dans le *menu principal* qui ouvre un *menu* dont le premier item est un tutoriel en anglais pour *Emacs* (cf. III.3.6.5.3.) Si l'anglais n'est pas un obstacle pour vous, vous lirez ce document avec profit. À tout moment, au cours de cette lecture, la déjà connue *commande* *Ctrl-x b* ouvrira un dialogue grâce au *minibuffer* où vraisemblablement le nom du *buffer* que vous avez quitté pour lire le tutoriel vous sera proposé, c'est-à-dire celui où se trouve la présente documentation à laquelle vous pourrez alors revenir.

III.1.2.2 Raccourci clavier Un certain nombre de *commandes Emacs* peuvent être déclenchées grâce à des séquences de touches. Par exemple *Ctrl-x Ctrl-c* provoquera la fin d'une session *Emacs* et vous rendra la ligne de commande du *shell*. De même *Ctrl-x Ctrl-s* vous permettra de sauvegarder,

(enregistrer sur le disque) le *fichier* correspondant au *buffer actif* (voir la section III.1.6.) ou *Ctrl-x Ctrl-f* vous permettra d'ouvrir un nouveau *fichier*. Un message sera alors prononcé et *Emacs* ouvrira (rendra actif) le *minibuffer* (cf. III.1.5p) pour vous permettre d'entrer le nom du *fichier* que vous voulez visiter. La *commande quit* déclenchée par *Ctrl-g* est particulièrement utile car elle permet d'annuler une *commande* en cas de mauvaise manipulation et de revenir au *buffer actif* avant le début de la séquence de *commandes*. Nous avons déjà vu également la *commande Ctrl-x Ctrl-b* qui permet de change de *buffer actif*.

Certains pourraient être rebutés par ces *commandes* à base de *Ctrl* obligeant parfois à adopter des positions inconfortables sur le clavier. Il faut savoir, que les combinaisons de touches auxquelles sont affectées les *commandes* peuvent être modifiés à volonté par l'*utilisateur* (voir la section III.1.7.) et qu'il pourrait par exemple affecter certaines *commandes* utilisées particulièrement souvent aux touches de fonctions du clavier s'il le souhaite.

III.1.2.3 Lancer une commande par son nom Les *commandes Emacs* ont toute un nom. Elles peuvent être appelées par ce dernier grâce à la procédure suivante :

La commande particulière *Esc x*, la touche d'échappement suivie du caractère « x », ces deux touches étant pressées successivement, provoque l'ouverture du *minibuffer* en attente du nom d'une commande. Celui-ci peut être saisi partiellement puis la touche *TAB* sollicitera *Emacs* pour qu'il complète automatiquement ; ce qu'il fera jusqu'au point où il n'y aura pas ambiguïté pour lui. Dès que la touche *touche <ENTRÉE>* est pressée, la correspondance est établie par *Emacs* entre la chaîne de caractères que vous venez d'entrer et une *commande* qui est exécutée en cas de succès.

Ainsi *Esc x* puis « find-file » puis *touche <ENTRÉE>* provoquera un effet exactement semblable à celui de *Ctrl-x Ctrl-f*. Vous pourriez encore entrer *Esc x* puis « find-fi » puis *TAB*. Alors *Emacs* ajoutera les caractères « le » et en pressant finalement *touche <ENTRÉE>* le *minibuffer* s'ouvrira pour que vous puissiez entrer le nom du *fichier* que vous voulez visiter.

III.1.3 . – Fenêtres

III.1.4 . – Buffers

Si les *mots*, *phrases* et *paragraphes* apparaissent naturellement comme les *objets* dont les caractères sont les *sous-objet*, nous avons entrevu au paragraphe précédent le rôle suffisamment important que jouent les *buffers* dans *Emacs* pour leur consacrer dès à présent un paragraphe avant même d'autres entités. Une compréhension élémentaire du maniement des *buffers* est nécessaire pour utiliser *Emacs*. Un *buffer* est un ensemble de données (en général du texte,) qui peut être manipulé par *Emacs*. Tous les *buffers* ne peuvent cependant pas être manipulés exactement de la même manière. À chaque *buffer* est, en effet associé un *mode* (cf. III.1.8,) c'est-à-dire, pour faire bref avant de plus amples développements, un ensemble de *commandes* autorisées à l'*utilisateur* (ainsi d'ailleurs qu'un certain nombre de comportements spontanés.) L'exemple de la section III.1.1 montre que si l'on a lancé *Emacspeak* de manière à pouvoir lire la version *info* de ce tutoriel, on se trouve devant un *buffer* dans lequel l'*insertion* de caractère n'est pas autorisée.

Les espaces de données constitués par les *buffers* sont, en général, dédiés à une tâche particulière. Dès que *Emacs* en effet, doit exécuter une tâche particulière, il y affecte un *buffer* ; la plus courante de ces tâches étant de visiter un *fichier* (cf. III.1.6.) Un *buffer Emacs* possède toujours un nom (auquel on peut faire référence notamment pour le sélectionner) et un *mode* qui spécifie les opérations possibles (autorisées à l'*utilisateur*) vis-à-vis de ce *buffer*.

Si *Emacs* peut manipuler (ou garder une référence) à un nombre quasi-infini de *buffers* simultanément, un seulement d'entre eux est actif. Ceci signifie qu'une *commande* entrée par l'*utilisateur*

concernera ce *buffer* et non un autre ! Un certain nombre d'opérations sous *Emacs* peuvent bien entendu changer le *buffer actif*.

Le procédé que nous indiquions dans la section III.1.1 lorsque l'utilisateur avait lancé *Emacspeak* afin de lire le tutoriel en *mode info*, consistait à sélectionner une autre *buffer* que le *buffer* courant (où l'insertion de caractère n'était pas possible) afin de tester comment insérer des caractères dans un texte. La commande *Ctrl-x b* consiste en effet à demander à *Emacs* de sélectionner un autre *buffer actif* en utilisant le *minibuffer* (cf. III.1.5p) pour procéder au dialogue afin de déterminer quel *buffer* l'utilisateur veut choisir comme *buffer actif*.

Il ne faut pas confondre la notion de *buffer Emacs* et de *fichier* même si, dans bien des cas, ces deux notions ont tendance à coïncider. *Emacs* peut créer, détruire, modifier etc ... des *buffers* ne correspondant à aucun *fichier* c'est-à-dire à aucun groupe de données référencé par le système d'exploitation.

Cependant, comme vous le verrez au paragraphe III.1.6, lorsque *Emacs* veut manipuler un *fichier* il commence par lui associer un *buffer*.

III.1.5 . – Le *minibuffer*

Le *minibuffer* est un *buffer Emacs* particulier qu'en général l'utilisateur ne peut sélectionner directement comme *buffer actif* mais qui est automatiquement sélectionné par *Emacs* lorsque celui-ci veut entamer un dialogue avec l'utilisateur. En général, si la commande que vous venez d'entrer appelle naturellement une question de la part de *Emacs* à laquelle il s'attend à ce que vous répondiez (en entrant un nom de *fichier* par exemple), il sélectionne le *minibuffer* comme *buffer actif* y place le curseur en attendant que vous saisissiez un chaîne de caractère que vous terminerez par la touche *<ENTRÉE>* et qu'il utilisera comme paramètre. Si plusieurs paramètres sont nécessaires à une opération donnée, *Emacs* ouvrira autant de fois qu'il le faut le *minibuffer* afin de vous solliciter à lui fournir les données dont il a besoin. L'édition dans le *minibuffer* se fait de manière très très intuitive mais de capacités supplémentaires sont adjointes qui vous permettent de ne taper par exemple que le début d'une chaîne de caractère suivi de *TAB* si *Emacs* peut compléter de manière non ambiguë en fonction du contexte.

III.1.5.1 Oui/non Une des principales utilisations du *minibuffer* par *Emacs* est la réponse aux alternatives que peut poser le logiciel c'est-à-dire des questions appelant une réponse du type *oui* ou *non* ! C'est en effet un point de fonctionnement très important de *Emacs*. Les questions appelant une réponse de type *Oui/Non* sont de deux niveaux différents suivant l'importance que peut avoir la réponse.

Soit la question est terminée par (*y or n ?*) et dans ce cas la simple pression sur la touche *y* va faire continuer le process, soit la question est terminée par (*yes or no ?*), et dans ce cas il faut réellement entrer *yes* ou *no* en toute lettres puis touche *<ENTRÉE>*. Ces dernières questions concernent en général des manipulations dont l'issue peut amener à la perte de donnée ou des choses de cette importance.

Par exemple si des *buffers* ont été modifiés et non sauve gardés, et qu'on essaye de sortir de *Emacs* par la commande *Ctrl-x Ctrl-c*, *Emacs* demandera d'abord si on veut sauver les *buffers* qui ne l'ont pas été en demandant de répondre par (*y or n ?*) mais si l'on répond *n* pour l'un d'entre eux, il demandera confirmation de l'abandon des modifications, cette fois par (*yes or no ?*) et on n'en sortira pas tant qu'on n'aura pas répondu par l'une ou l'autre de ces réponses. Ainsi en va-t-il également de la fermeture d'un *buffer* (avec le menu ou avec *Ctrl-x k*) qui n'aurait pas été sauvé.

III.1.6 . – Fichiers

Nous l'avons mentionné dans la section consacrée aux *buffers* (cf. III.1.4.) un *buffer* c'est-à-dire un ensemble de données manipulable par *Emacs* n'est pas nécessairement lié à un *fichier* c'est-à-dire un ensemble de données manipulable par le *système d'exploitation* et le plus souvent correspondant à un espace physique sur le disque dur de la machine. Deux opérations peuvent cependant créer ce lien :

- Si l'on essaye d'enregistrer un *buffer* (commande *Ctrl-x Ctrl-s*) encore lié à un *fichier*, *Emacs* demandera un nom de fichier à l'utilisateur où enregistrer les données contenues dans le *buffer*. Après quoi, ce *buffer* sera lié au *fichier* qui vient d'être créé et toute opération de sauvegarde par *Ctrl-x Ctrl-s* s'effectuera dans ce *fichier*. Il se peut qu'alors le nom du *buffer* soit modifié pour correspondre plus intuitivement au nom du *fichier*.
- Si l'on visite un *fichier* sous *Emacs*, par la commande *Ctrl-x Ctrl-f*, le contenu de ce *fichier* est chargé dans un *buffer* dont le nom correspond au nom du *fichier*. On peut alors éditer le contenu de ce *fichier* si toutefois l'utilisateur dispose des permissions requises. Les modifications apportées au contenu du *fichier* ne seront enregistrées sur le support de mémoire de masse qu'après avoir effectué la commande de sauvegarde *Ctrl-x Ctrl-s*.

III.1.6.1 Sauve-garde des fichiers Remarquons cependant, que si l'utilisateur oublie de sauvegarder le contenu d'un *buffer* lié à un *fichier*, au moment de la sortie commandée par *Ctrl-x Ctrl-c*, *Emacs* demandera confirmation de l'abandon des modifications.

Notons encore qu'évidemment, si un incident (ceux-ci sont pour ainsi dire inexistant) provoquait l'arrêt inopiné de *Emacs*, celui-ci conserve malgré tout une trace des modifications apportées à un *fichier*. Évidemment, un incident matériel sur le disque pourrait anéantir les efforts de *Emacs* pour conserver les données.

III.1.7 . – Comment personnaliser *Emacs*

III.1.8 . – Les modes de *Emacs*

III.1.8.1. – Qu'est-ce qu'un mode ? Nous avons apporté, dans les descriptions précédentes, quelques restrictions à la permanence des *commandes* sous *Emacs* en spécifiant, ici ou là, que certaines opérations sous *Emacs* étaient liées au *contexte*. Ce dernier est implémenté par la notion de *mode*. Chaque *buffer* possède un *mode* qui conditionne les opérations permises à l'intérieur de ce *buffer* et la manière dont elles s'exécutent. Nous ne détaillerons pourtant pas cette notion qui est à la fois une des plus complexes, mais une des plus performantes sous *Emacs*.

Donnons simplement quelques exemples. Nous avons déjà mentionné que l'utilisateur peut redéfinir l'affectation de certaines *combinaisons de touches* à certaines *commandes*. Ceci peut-être fait de manière globale ou pour un mode particulier. Ainsi la *combinaison de touche* peut être actif quelque soit les circonstances ou bien uniquement si le *buffer actif* est d'un type particulier. Ceci sera contrôlé par le *mode*. Dans le *mode info* par exemple, les *buffers* utilisés pour afficher les pages d'information sont ouverts uniquement en lecture puisqu'il ne s'agit pas de modifier ces pages. Certains *caractères* qui déclenchent ordinairement la commande d'*insertion* seront alors réaffectés à d'autres tâches comme *u* qui permet de remonter d'un niveau dans l'arborescence de l'information.

III.1.8.2. – Quels sont les modes ? À cette question on répondra de manière partielle et on renverra à la documentation en ligne de *Emacs* pour un panorama complet de modes existant sous *Emacs*. Ce panorama ne sera d'ailleurs jamais complet puisque il est permis à n'importe quel utilisateur de

créer ses propres *modes*. Il est hors de question de dire quoi que ce soit de la manière de procéder ici, puisque ces opérations appartiennent à une utilisation experte de *Emacs* et que lorsqu'on s'attaque à un tel travail on doit être capable de lire la documentation dans le détail.

Cependant la probabilité de devoir réaliser une telle opération au moins dans le cadre d'une utilisation élémentaire de *Emacs* est extrêmement faible. Un grand nombre de *modes* prédéfinis et dédiés aux tâches les plus courantes existent déjà. Le plus simple d'entre eux est dit *fondamental* et correspond au *mode* dans lequel sera ouvert un *buffer* dans lequel on voudrait réaliser des tâches d'édition standard. Certains raffinements existent qui permettent d'éditer avec des facilités très appréciables des *fichiers* de type particulier, comme des sources C ou JAVA, des sources \LaTeX , etc ...

Certains *modes* sont encore plus particuliers comme le *mode shell* auquel, étant donnée son importance, surtout pour les utilisateurs d'*Emacspeak*, nous réserverons une section particulière (cf. III.1.8.6.)

Mentionnons encore, pour mémoire, le *mode info* qui permet de lire de façon pratique les *fichiers* conçus pour le programme *info*.

Le *minibuffer* (cf. III.1.5.) a son propre *mode* qui permet certaines opérations spécifiques comme la compléssion par exemple.

III.1.8.3. – Sélection du *mode* Le plus souvent, le *mode* d'un *buffer* est sélectionné automatiquement par *Emacs* suivant des critères préétablis. Par exemple si vous souhaitez lire des *fichiers info*, *Emacs* affichera leur contenu dans un *buffer* ouvert sous un mode particulier.

Si vous visitez une *fichier* avec *Emacs*, dans un grand nombre de cas, l'extension détermine le *mode* dans lequel sera ouvert le *fichier*. Par exemple un fichier comportant l'extension *c* ou *C* ou *cc* sera ouvert dans un mode spécifique à l'édition de code source C. Ainsi en va-t-il également de *fichiers* comportant l'extension *tex* ou *java*.

III.1.8.4. – Le *mode fondamental*

III.1.8.5. – Le *mode info*

III.1.8.6. – Le *mode shell* Ce paragraphe est destiné à ceux qui, dès le début, aurait pu se sentir captifs de *Emacs* et qui auraient craint de ne pouvoir accéder qu'à une surcouche du *système d'exploitation* sans pouvoir accéder au système lui-même. Cependant *Emacs* est ouvert sur le système et *Emacspeak* permet de profiter de cette ouverture dans le cadre d'un environnement sonore. Il se comporte, dans cette situation, à peu près comme une *revue d'écran*. En effet, il existe sous *Emacs* un *mode* spécial appelé *mode shell* dans lequel le *buffer actif* se comporte exactement comme un *shell* système. Comme tout *buffer* il peut être parcouru et sonorisé par *Emacspeak*. Grâce à ce mode particulier, lorsque la *touche* <ENTRÉE> est pressée, la partie de la ligne située avant le *curseur* est envoyé au système comme une *shellcommandtxt* est le texte retourné par la *commande* est inséré dans le *buffer*.

Pour accéder au *mode shell*, il faut entrer la *commande* *Emacs shell* c'est-à-dire (cf. III.1.2.3.) faire *Esc x* puis *shell* puis *touche* <ENTRÉE>. On se retrouve alors dans un *buffer* qui peut ressembler à quelque chose comme ça :

```
[lorenzon@jabberwocky emacspeak]$
```

Si l'on entre *ls*, puis *touche* <ENTRÉE>, le contenu du *buffer* doit ressembler à

```
[lorenzon@jabberwocky emacspeak]$ ls  
auto                emacspeak-info      introduction.tex
```

```

connexes.tex      emacspeak-info.tar  Makefile
doc.tex          emacspeak.log       modes.tex
efm.mac          emacspeak.tex       modes.tex~
efm.sty          emacspeak.txt       readme-festival.index.tex
efm.sty~         emacs.tex           survol.tex
emacspeak.aux    espeak-features.tex syntheses.tex
emacspeak.dvi    install.tex
emacspeak.idx    interactive.tex
[lorenzon@jabberwocky emacspeak]$

```

La *commande ll* aurait donné :

```

[lorenzon@jabberwocky emacspeak]$ ll
total 472
drwxr-x---   4 lorenzon users      4096 Jul 10 08:34 .
drwxr-x---   6 lorenzon users      4096 Jun 23 08:21 ..
drwxr-x---   2 lorenzon users      4096 Jul 10 08:34 auto
-rw-r-----   1 lorenzon users        201 Apr 22 11:32 connexes.tex
-rw-r-----   1 lorenzon users        943 Apr 22 16:40 doc.tex
lrwxrwxrwx   1 lorenzon users         40 May  7 06:33 efm.mac -> /home/lo
-rw-r-----   1 lorenzon users      8749 Jul 10 08:34 efm.sty
-rw-r-----   1 lorenzon users      8025 Jun 24 13:16 efm.sty~
-rw-r--r--   1 lorenzon users     11191 Jul 10 08:20 emacspeak.aux
-rw-r--r--   1 lorenzon users    90588 Jul 10 08:20 emacspeak.dvi
-rw-r--r--   1 lorenzon users   13654 Jul 10 08:20 emacspeak.idx
drwxr-x---   2 lorenzon users      4096 Apr 22 14:21 emacspeak-info
-rw-r--r--   1 lorenzon users   102400 Jul  2 06:22 emacspeak-info.tar
-rw-r--r--   1 lorenzon users    16901 Jul 10 08:20 emacspeak.log
-rw-r-----   1 lorenzon users     1226 Jun 23 05:32 emacspeak.tex
-rw-r--r--   1 lorenzon users    67111 Jul  2 06:22 emacspeak.txt
-rw-r-----   1 lorenzon users    21570 Jun 24 13:26 emacs.tex
-rw-r-----   1 lorenzon users      155 Apr 21 09:58 espeak-features.tex
-rw-r-----   1 lorenzon users    10515 Apr 21 09:47 install.tex
-rw-r-----   1 lorenzon users     4896 Apr 22 15:57 interactive.tex
-rw-r-----   1 lorenzon users     9482 Apr 22 13:15 introduction.tex
-rwxr-----   1 lorenzon users     1929 Jun 24 13:30 Makefile
-rw-r-----   1 lorenzon users     6779 Jul 10 08:27 modes.tex
-rw-r-----   1 lorenzon users     5198 Apr 22 15:46 modes.tex~
-rw-r-----   1 lorenzon users      717 May  7 21:55 readme-festival.inc
-rw-r-----   1 lorenzon users    12846 Jun 23 05:31 survol.tex
-rw-r-----   1 lorenzon users      410 May  7 21:59 syntheses.tex
[lorenzon@jabberwocky emacspeak]$

```

Je tiens à signaler que cette utilisation du *mode shell* sous *Emacs* est tout à fait **ARTIFICIELLE** ! et encore plus sous *Emacspeak*. En effet, l'information renvoyée par la *commande ls*, est sous un format qu'on pourrait qualifier de brut. C'est encore plus vrai sous *Emacspeak* où le texte contenu dans le *buffer* est prononcé à la file. Les *commandes* de manipulation de *fichier* comme *mv*, *cp*, *rm*, *cd*, etc .. peuvent évidemment être lancées dans un *buffer* en *mode shell* mais la philosophie de *Emacs* et surtout d'*Emacspeak* est plutôt d'utiliser, pour ces opérations, le *mode dired* (cf. III.1.8.7.)

III.1.8.7. –mode dired Le mode *dired* comme *directory edit* implémente sous *Emacs* un gestionnaire de fichiers assez sophistiqué. Un *buffer* en mode *dired* est ouvert par la commande *Ctrl-x d* (cf. III.1.2.2) où *Esc x* puis *dired* puis *touche <ENTRÉE>* (cf. III.1.2.3,) qui ouvre un dialogue dans lequel l'utilisateur est amené à entrer le nom du répertoire qu'il veut lister. Ces opérations sont bien évidemment sonorisées par *Emacspeak* et s'effectuent avec toutes les facilités d'édition habituelles (cf. III.1.5.)

Cela donne par exemple :

```
Dired (directory): ~/web/gnu/emacspeak/
```

Si on presse simplement *touche <ENTRÉE>* à ce moment là on ouvrira un *buffer* qui aura cette allure :

```
/home/lorenzon/web/gnu/emacspeak:
used 488 available 17025988
drwxr-x---   4 lorenzon users      4096 Jul 10 08:46 .
drwxr-x---   6 lorenzon users      4096 Jun 23 08:21 ..
drwxr-x---   2 lorenzon users      4096 Jul 10 08:44 auto
-rw-r-----  1 lorenzon users        201 Apr 22 11:32 connexes.tex
-rw-r-----  1 lorenzon users        943 Apr 22 16:40 doc.tex
lrwxrwxrwx   1 lorenzon users         40 May  7 06:33 efm.mac -> /home/
-rw-r-----  1 lorenzon users      8749 Jul 10 08:34 efm.sty
-rw-r-----  1 lorenzon users     8025 Jun 24 13:16 efm.sty~
-rw-r--r--   1 lorenzon users    11191 Jul 10 08:20 emacspeak.aux
-rw-r--r--   1 lorenzon users   90588 Jul 10 08:20 emacspeak.dvi
-rw-r--r--   1 lorenzon users  13654 Jul 10 08:20 emacspeak.idx
drwxr-x---   2 lorenzon users      4096 Apr 22 14:21 emacspeak-info
-rw-r--r--   1 lorenzon users  102400 Jul  2 06:22 emacspeak-info.ta
-rw-r--r--   1 lorenzon users   16901 Jul 10 08:20 emacspeak.log
-rw-r-----  1 lorenzon users     1226 Jun 23 05:32 emacspeak.tex
-rw-r--r--   1 lorenzon users   67111 Jul  2 06:22 emacspeak.txt
-rw-r-----  1 lorenzon users   21570 Jun 24 13:26 emacs.tex
-rw-r-----  1 lorenzon users      155 Apr 21 09:58 espeak-features.t
-rw-r-----  1 lorenzon users   10515 Apr 21 09:47 install.tex
-rw-r-----  1 lorenzon users    4896 Apr 22 15:57 interactive.tex
-rw-r-----  1 lorenzon users    9482 Apr 22 13:15 introduction.tex
-rwxr-----  1 lorenzon users    1929 Jun 24 13:30 Makefile
lrwxrwxrwx   1 lorenzon users        37 Jul 10 08:44 .#modes.tex -> lc
-rw-r-----  1 lorenzon users    9834 Jul 10 08:46 #modes.tex#
-rw-r-----  1 lorenzon users    9509 Jul 10 08:44 modes.tex
-rw-r-----  1 lorenzon users    5198 Apr 22 15:46 modes.tex~
-rw-r-----  1 lorenzon users     717 May  7 21:55 readme-festival.i
-rw-r-----  1 lorenzon users   12846 Jun 23 05:31 survol.tex
-rw-r-----  1 lorenzon users     410 May  7 21:59 syntheses.tex
```

Comparez-le avec le *buffer* obtenu dans la section III.1.8.6 après avoir exécuté la commande *ll*. Il est évidemment très semblable mais dès l'entrée le rendu sonore d'*Emacspeak* est radicalement différent puisque le *buffer* ne sera pas prononcé à la file, ce qui peut être un avantage certain si vous listez un répertoire de plusieurs centaines de fichiers. Ensuite lorsque vous commencerez à déplacer

le *curseur* avec les touches de déplacement vous vous apercevrez de la différence. Seul le nom du *fichier* situé sur une ligne est prononcé et de manière différente selon qu'il s'agit d'un *fichier*, d'un *répertoire* ou d'un lien symbolique (si toutefois votre *synthèse vocale* permet ce genre de choses.)

Les autres renseignements concernant le *fichier* situé sur la lignes où se trouve le *curseur* sont accessibles par un certain nombre de *commandes*. Vous pouvez toujours lire cette ligne d'un bout à l'autre, comme dans n'importe quel *buffer* ou presque par la *commande Emacspeak Ctrl-e Ctrl-l* (cf. III.3.6.2) mais c'est encore la manière la moins fine de procéder.

Le *buffer* en *mode dired* ne permettant pas l'insertion de *caractères* (cf. III.1.1.) un certain nombre de touches ont été réaffectées à des tâches particulières. En voici un petit résumé (reportez-vous à la section correspondante du manuel de *Emacs* accessible par *Ctrl-h i* (cf. III.3.6.5.2.) pour une information complète.)

Certaines de ces *commandes* sont spécifiques à *Emacspeak* c'est-à-dire pour faire bref produisent un effet sonore sans déplacement du *curseur* (cf. III.3.6) :

- *z* prononce la taille du *fichier* courant.
- *a* prononce la date du dernier accès au *fichier*.

D'autres *commandes* sont des *commandes* standard du *mode dired* de *Emacs* sonorisées par *Emacspeak* :

- *touche <ENTRÉE>* exécute ici une tâche particulière : si le *fichier* situé sur la *ligne* courante est un *fichier*, il est visité par *Emacspeak* (cf. III.1.6.) S'il s'agit d'un *répertoire* c'est-à-dire d'un sous-répertoire de celui qu'on est en train de lister, il est listé à son tour dans un nouveau *buffer*.
- *C* implémente la *commande cp* et permet donc de copier le *fichier* courant vers un autre endroit. Ceci se passe à travers un dialogue avec *Emacs* (cf. III.1.5) avec toutes les facilités d'édition habituelles.
- *D* implémente la *commande rm* et détruira donc le *fichier* sans oublier de vous demander confirmation bien évidemment !
- *R* implémente la *commande mv* qui permet de déplacer un *fichier* avec les mêmes facilités que pour *C*.

D'autres *commandes* sont disponibles qui permettent par exemple de traiter des *fichiers* par groupes mais il me paraît abusif de les détailler toutes dans le cadre de cette présentation sommaire.

III.2 . – À propos de *Speechd-el 2.1*

III.2.0 . – Introduction

Cette section n'est, pour l'instant, qu'une FAQ.

III.2.1 . – La langue

Pour configurer la langue sous *Speechd-el*, on peut utiliser *Ctrl-E d l* ou la *commande speechd-set-language*. Il faut ensuite dans l'un et l'autre cas entrer le code de la langue : *fr* pour le français, *en* pour l'anglais ...

III.3 . – Quelques notes à propos d'*Emacspeak 24*

III.3.1 . – Qu'est-ce qu'*Emacspeak* et pourquoi l'utiliser ?

À ces deux questions ce document tente de répondre de la manière la plus simple et la moins technique possible. Je me suis placé pour le rédiger, du moins ai-je fait le maximum efforts dans cette

direction, dans le point de vue d'un lecteur qui serait d'abord et essentiellement un utilisateur de ce logiciel et qui ne voudrait a priori pas avoir à intervenir lui-même, ou alors au minimum, pour adapter ce système à ses besoins. Quelques sections seront cependant plus particulièrement réservées à ceux qui voudraient contribuer ici ou là au développement de ce logiciel. Néanmoins, ce n'est pas du tout le propos principal de ce document. Ces paragraphes seront donc bien balisés et en fait renverrons le plus souvent à d'autres documents de fond. Ils pourront être soigneusement évités par les lecteurs uniquement soucieux de disposer d'un outil puissant répondant à leur demande.

III.3.1.1. – Pourquoi ce document et quel est-il ? J'ai rédigé ce document à la demande des animateurs de la liste de diffusion *carrefourblinux* à destination d'utilisateurs non nécessairement expérimentés. La liste de diffusion *carrefourblinux* est un espace de discussion et d'échanges pour des utilisateurs mal ou non voyants de *Linux*. Le logiciel *Emacspeak* permet, en effet, à des personnes non ou mal voyantes d'utiliser le *système d'exploitation Linux* grâce à un environnement sonore intégré. Il est hors de question ici d'entrer dans les détails concernant *Linux* lui-même et nous conseillons au lecteur qui ne serait pas suffisamment familier avec ce système de se reporter aux adresses le concernant mentionnées ci-dessus. S'il veut s'assurer que ses connaissances sont suffisantes qu'il prenne connaissance du paragraphe III.3.1.2 de ce document.

Nous nous intéresserons principalement à *Emacspeak* dans ce texte même si, ici ou là, nous serons peut-être obligés de faire quelques incursions dans le fonctionnement du *système d'exploitation* lui-même.

J'ai voulu concevoir le présent document plutôt comme un tutoriel pour *Emacspeak* et non comme un manuel de référence. Ceux-ci existent (en anglais certes) et je suggère aux utilisateurs qui chercheraient une documentation approfondie de se reporter simplement à la page officiel d'*Emacspeak*. Le présent texte est d'abord conçu pour les commençants et non pour des utilisateurs confirmés qui voudraient affiner encore leur utilisation du logiciel. L'idéal pour le débutant qui voudrait s'initier à *Emacspeak* serait de lire ce document justement grâce à *Emacspeak*. Se reporter pour cela à la section III.3.2.

III.3.1.2. – Les prérequis pour poursuivre la lecture de ce document Je supposerai par la suite que mon lecteur est assez familier avec le *système d'exploitation Linux* pour au moins savoir s'identifier (se loguer), et lancer une commande dans le *shell* (interpréteur de commandes.) Le logiciel *Emacspeak* est en effet un programme qui s'exécute sous le *système d'exploitation Linux*.¹

Nous supposerons également que lecteur est suffisamment familier avec la manière dont *Linux* organise les données en *répertoires* et *fichiers*.

Le lecteur sera plus à l'aise s'il sait déjà utiliser des utilitaires tels que *gnu tar* ou *gzip* mais cela n'a aucun caractère indispensable.

Il est temps de décrire plus précisément, quoi qu'encore assez sommairement ce qu'est ce logiciel.

III.3.1.3. – Qu'est-ce qu'EMACSPEAK ? Nous l'avons déjà mentionné, *Emacspeak* est un environnement sonore intégré destiné principalement aux personnes non et mal voyantes. Il permet d'effectuer les tâches les plus courantes de la bureautique domestique mais possède des possibilités qui peuvent également le rendre très appréciable dans un cadre professionnel. On peut grâce à *Emacspeak* éditer des textes de manière extrêmement sophistiquée, manipuler les fichiers du système (fonction "file manager"), recevoir et envoyer des courriers électroniques, naviguer sur internet mais aussi éditer des codes sources pour le développement de logiciel de manière très agréable et efficace.

¹ Il peut peut-être utilisé avec d'autres systèmes, mais ce n'est pas du tout le propos de ce document que de décrire ces possibilités somme toute assez artificielles. Il a été conçu pour et s'adapte particulièrement bien à *Linux*.

À l'origine *Emacspeak* est d'abord un *éditeur de texte* puisqu'il est basé sur *Emacs 21.4a* qui est lui-même un *éditeur de texte*. Cependant les possibilités très étendues de *Emacs* et notamment son langage de programmation interne ont permis de développer de nombreuses autres applications. Le logiciel *Emacspeak* lui-même est un programme *Emacs* qui sonorise complètement ce dernier.

Il faut cependant bien garder à l'esprit qu'*Emacspeak* n'est pas un logiciel de synthèse vocale. Il peut piloter un certain nombre de synthèses vocales matériel et logiciel mais n'effectue lui-même aucune des tâches spécifiques à ces engins. Se reporter à la section III.3.5.

Le système vocale intégré qui vous permettra d'utiliser *Linux* est, au moins à première vue, composé de trois couches superposées : *Emacs* > *Emacspeak* > une *synthèse vocale*. Quel est donc le rôle de cette couche intermédiaire placée entre l'*éditeur de texte Emacs* et la *synthèse vocale* ? C'est d'organiser de la manière la plus efficace possible l'information recueillie dans *Emacs* pour la transmettre à la *synthèse vocale* afin que l'utilisateur puisse en tirer le plus de profit.

Si nous prenons un exemple extrême, il est inutile dès qu'une page d'écran a été localement modifiée de la relire dans son intégralité. Il appartient donc à *Emacspeak* de repérer les modifications significative de l'environnement de *Emacs* et de transmettre à la *synthèse vocale* un message le plus intelligible et le plus signifiant possible.

Pour cela, *Emacspeak* qui a été conçu par et pour des personnes handicapées visuel a développé un certain nombre d'astuces comme des changements de voix (si la *synthèse vocale* le permet bien évidemment,) pour rendre compte de changements de police de caractères par exemple.

Emacspeak n'est pas non plus une *revue d'écran* au sens où l'entend d'habitude et l'utilisateur pourrait se sentir un peu frustré d'être ainsi séparé du *système d'exploitation* par une couche logiciel supplémentaire. Il pourrait avoir l'impression que certaines manipulations propres au système lui sont interdites par cette situation. On verra par la suite qu'en fait il n'en est rien et qu'*Emacspeak* peut se comporter presque comme une *revue d'écran* (cf. III.1.8.6) même si nous n'insisterons pas beaucoup sur cette possibilité puisque *Emacs* dispense le plus souvent, par le grand nombre de ses possibilités internes, de recourir à ce genre de manipulation.

Disons d'emblée que je n'écris pas non plus ce document pour les experts du système qui ont développé à grands coups de "shell-scripts" sophistiqués des solutions qu'ils jugent efficace et considéreraient que le jeu ne vaut pas la chandelle d'apprendre à manipuler *Emacspeak*. Néanmoins il faut bien qu'ils gardent en tête les avantages d'un système intégré qui permet d'effectuer les tâches décrites plus haut, édition, mail, navigation internet etc ... en connexion les unes avec les autres. Par exemple on peut prendre un morceau du mail ou de la page WEB qu'on est entrain de lire pour l'insérer dans le *fichier* qu'on est en train d'éditer sans perdre à aucun de ses endroits le pointeur ...

III.3.1.4. – Remarque concernant le rendu sonore d'*Emacspeak* Disons d'emblée qu'il ne faut pas attendre de miracles d'*Emacspeak*. Dans la mesure où il n'effectue pas les tâches de *synthèse vocale* lui-même la qualité du rendu sonore est fortement subordonnée aux possibilités de la *synthèse vocale*. *Emacspeak* n'a aucune aptitude à influencer sur la prosodie ou l'intonation du texte en fonction de sa nature. Il se contente de demander de telles performances à la *synthèse vocale* en fonction de ses exigences ou des vôtres (cf. III.3.6.3.)

Par exemple il pourra solliciter une prosodie différente de la part de la *synthèse vocale* pour mettre en évidence une partie de texte en italique ou en gras. Il pourra même, si la *synthèse vocale* le permet lui demander de changer de voix pour rendre compte de certaines différences de mise en forme des données. Si la *synthèse vocale* est limitée dans ses possibilités *Emacspeak* ne pourra rien faire pour améliorer la situation. Tout au plus peut-il corriger quelques petits manques d'une *synthèse vocale*. Par exemple, si votre *synthèse vocale* est incapable de reconnaître des caractères particuliers comme "J" *Emacspeak* peut s'arranger, dès qu'il rencontrera ce caractère pour que la *synthèse vocale* prononce

“crochet fermant”.

Vous voyez bien que ceci ne procède que de la substitution de caractères dans des chaînes et pas du tout de modifications fines de la prosodie ou de l’intonation. Il va donc de soit que pour profiter au maximum des possibilités d’*Emacspeak* il vaut mieux disposer d’une *synthèse vocale* performante (voir la section III.3.5.)

III.3.2 . – Lecture interactive de ce document

III.3.2.1. – Où trouver ce document et sous quelle forme ? La dernière version de ce document se trouve à l’adresse `http://www.pollock-nageoire.net/emacspeak/`.

Il existe sous les formes suivantes :

III.3.2.1.1. – Format texte compressé

III.3.2.1.2 Un seul fichier *HTML*

III.3.2.1.3 Une archive contenant plusieurs fichiers *HTML* Si vous téléchargez sous cette forme il vous faudra décompresser l’archive *emacspeak-html.tar.gz* en effectuant la commande

```
# tar -xzvf emacspeak-html.tar.gz
```

qui créera un *répertoire emacspeak-html* dans lequel vous trouverez un *fichier index.html* contenant la racine de l’arborescence de ce texte et vous donnant accès à toutes ses composantes.

III.3.2.1.4 Une archive contenant un fichier info et quelques utilitaires Il vous faudra dans ce cas aussi décompresser l’archive *emacspeak-info.tar.gz* par la commande

```
# tar -xzvf emacspeak-info.tar.gz
```

qui créera un *répertoire emacspeak-info*.

III.3.2.2. – Comment lire ce document de manière interactive ? Je le répète, l’idéal, serait de pouvoir lire ce tutoriel avec *Emacspeak* lui-même, afin de pouvoir, au fur et à mesure de la lecture, effectuer les opérations décrites ici et se familiariser ainsi avec le maniement d’*Emacspeak*. Moyennant que vous disposiez d’un *Emacspeak* installé et fonctionnel (sinon reportez-vous à la section III.3.4.) plusieurs options sont possibles avec chacune ses avantages et ses inconvénients :

III.3.2.2.1 Avec la version texte Vous avez téléchargé la version texte de ce document (cf. III.3.2.1.1.) Vous pouvez alors, simplement vous placer dans le répertoire où se trouve le fichier *emacspeak.txt* que vous aurez obtenu en dézipant *emacspeak.txt.gz*. Pour cela il vous suffit d’effectuer la commande

```
# gunzip emacspeak.txt.gz
```

Ensuite démarrez *Emacspeak* sur ce fichier en tapant

```
# emacspeak emacspeak.txt
```

à l’invite du *shell* puis la *touche <ENTRÉE>*. Voir la section III.3.4.3 pour plus de détails. Cette solution offre le mérite de la plus grande simplicité, mais, contrairement aux deux suivantes, ne vous permettra pas une lecture *hypertexte* de cette documentation.

III.3.2.2.2 Sous forme *info* Vous avez téléchargé la version *info* de cette documentation sous la forme du fichier *emacspeak-info.tar.gz* (cf. III.3.2.1.4.) Procédez tout d’abord au dépaquetage du fichier mentionné ci-dessus par la commande

```
# tar -xzvf emacspeak-info.tar.gz
```

qui va créer un répertoire *emacspeak-info*. Sélectionnez ce répertoire comme répertoire courant par

```
# cd emacspeak-info
```

puis lancez le script *info-start* par la commande

```
# ./info-start
```

Vous avez alors lancé *Emacspeak* dans un *mode* (cf. III.1.8,) particulier qui va vous permettre de naviguer de manière satisfaisante dans le tutoriel. Cette navigation *hypertexte* est certes moins performantes qu’une navigation dans un fichier *html* comme décrit ci-après, mais néanmoins très raisonnable et ne suppose pas d’adjoindre le moindre outil supplémentaire à *Emacspeak* puisque le seul présumé est que le programme *info* existe sur le site où vous travaillez ce qui est généralement le cas dans une installation standard.

III.3.2.2.3 Sous forme *html*

III.3.2.3. –Contacter l’auteur Vous pouvez toujours *me contacter* si vous avez des questions à propos de ce document ou si vous n’y trouvez pas la réponse à une question que vous vous posez à propos d’*Emacspeak*.

III.3.3 . –Documentation à propos d’*Emacspeak*

Il existe un certain nombre de documents relatifs à *Emacspeak*.

III.3.3.1. –Un tutoriel en anglais Vous pouvez consulter un tutoriel en anglais qui se trouve également, si votre installation d’*Emacspeak* a été convenablement réalisée dans le répertoire *emacspeak/user-guide/* qui est normalement placé dans */usr/share/emacs/site-lisp/* ou */usr/local/share/emacs/site-lisp/* selon votre installation. En pointant sur le fichier *index.html* du répertoire *emacspeak/user-guide/* vous accédez à ce tutoriel.

III.3.3.2. –La documentation au format *info*

III.3.4 . –Installation

Nous l’avons dit plus haut, *Emacspeak* est un environnement intégré mais il ne réalise pas lui-même toutes les couches de l’intégration et a besoin pour fonctionner d’un minimum d’autres logiciels. Nous ne mentionnerons ici que ceux qui lui sont absolument indispensables et permettent un fonctionnement standard comme décrit dans les sections III.3.6 et III.1.

Comme l’on peut élargir ses possibilités à l’infini, il est hors de question que nous en présentions d’autres que celles dont nous avons déjà fait l’expérience. Voir pour cela la section outils connexes qui permet d’augmenter les capacités d’*Emacspeak* III.3.8.

Voici donc ce dont vous devez disposer sur votre ordinateur pour installer et utiliser *Emacspeak* :

III.3.4.1. – Les différents composants du système

III.3.4.1.1 Le système Vous devez disposer du *système d'exploitation Linux* ou équivalent en état de fonctionnement.²

III.3.4.1.2 Emacs L'*éditeur de texte Emacs* doit être également installé. Il ne fait pas partie de la distribution standard d'*Emacspeak* et doit être installé préalablement. En revanche il fait partie de toute presque les distributions de *Linux* et est généralement proposé lors de la plupart des installations interactives.

III.3.4.1.3 synthèse vocale Un système de *synthèse vocale* matériel ou logiciel avec lequel *Emacspeak* puisse communiquer. Un grand nombre de synthèse vocale matériel est supporté et quelques synthèse vocal logiciel (hélas trop peu nombreuses.) (Voir la section III.3.5 pour plus de détails.)

III.3.4.1.4 Speech server Les synthèses vocales (qu'elles soient matériel ou logiciel) ne disposant pas d'un système de commandes standardisées, *Emacspeak* ne peut les piloter sans l'aide d'une interface appropriée et vous devez donc vous procurer (ou écrire vous-même mais là bon courage) l'interface « speech server » dans le jargon *Emacspeak* correspondant à votre *synthèse vocale*. Voir la page de Jim van Zandt pour le package *deb* ou la version *tar.gz* des packages contenant les speech servers ainsi que la liste des synthèses vocales supportées. Pour d'éventuels utilisateurs de *Festival*, se reporter à la section III.3.5.2 tout en sachant que cette solution est encore tout à fait expérimentale et d'une installation encore délicate.

III.3.4.2. – Installation Une fois que vous aurez rassemblés les éléments indispensables au fonctionnement d'*Emacspeak* comme décrit dans le paragraphe III.3.4.1, vous êtes prêt à installer *Emacspeak* lui-même. Un package correspondant à *Emacspeak* est disponible dans un grand nombre de distributions récentes de *Linux* (Debian RedHat ou SUSE.) Vous pouvez évidemment également télécharger les sources en *tar.gz* sur la page d'*Emacspeak* compiler et installer vous-même³. Il est probable que, si vous disposez d'une distribution convenablement installée, il sera plus aisé d'utiliser le processus d'installation propre à votre distribution (rpm pour RedHat ou SUSE par exemple.)

Vous devez évidemment disposer d'une installation de *Emacs*. Il se pourrait bien qu'une version antique de *Emacs* pose des problèmes de compatibilité avec des versions récentes d'*Emacspeak*. Si vous êtes l'administrateur du système je vous conseillerais d'upgrader votre version de *Emacs* vers une version 20.x (voire 21.x, mais je ne suis pas sûr que le support de la nouvelle norme iso-8859-15 soit vraiment parfait dans ces versions !) Si vous n'êtes pas l'administrateur du système, je serais vraiment tenté de vous dire de passer cette section et de demander à votre administrateur système de procéder à l'installation.

Il vous faudra ensuite installer le *speech server* correspondant à la synthèse vocale que vous avez choisie et configurer *Emacspeak* pour qu'il utilise ce speech server. Ici, mieux vaut consulter la documentation relative à l'installation. Pour donner un ordre d'idée de la complexité de l'opération à réaliser, je mentionnerai simplement qu'avec un boîtier apollo de Dolphin System, et un *shell bash*, il suffit d'ajouter les lignes suivantes au *.bashrc* de l'utilisateur d'*Emacspeak* :

```
export DTK_PROGRAM=apollo
```

²L'utilisation d'*Emacspeak* sous un autre *système d'exploitation* est peut-être possible, mais pour n'en jamais avoir fait l'expérience, nous n'en parlerons pas dans ce document.

³En cas d'utilisation avec *Festival*, cette méthode est indispensable reportez-vous dans ce cas à la section III.3.5.2

Encore une fois pour *Festival*, la procédure sera un peu différente et voir la section III.3.5.2

III.3.4.3. – Démarrage d’*Emacspeak* Une fois que vous avez correctement installé les diverses composantes du système *Emacspeak* mentionnées dans la section III.3.4.1, et que vous disposez d’un prompt *shell*, vous pouvez démarrer *Emacspeak* par la commande :

```
# emacspeak
```

avec éventuellement des options sur la ligne de commande. Si tout fonctionne normalement, vous devez entendre un message de bienvenue sur le périphérique que vous utilisez pour la synthèse vocale.

III.3.4.3.1 It crashes! Si ce n’est pas le cas, c’est que l’une ou plusieurs des composantes du système est (sont) défectueuse(s). Pour déterminer laquelle, l’aide d’un voyant vous sera vraisemblablement nécessaire, à moins que vous n’utilisiez conjointement un autre moyen de contrôle des sorties de votre ordinateur comme un terminal Braille par exemple. En admettant que vous ayez accès à l’output consécutif à la commande *Emacspeak*, un grand nombre de cas de figure peuvent se présenter dont je ne peux rendre compte exhaustivement ici. Voir la section III.3.4.4.

III.3.4.3.2 All right! Vous avez entendu le message d’entrée, tout va bien et vous êtes prêt à travailler avec *Emacspeak*. Je serai moins superficiel dans cette section tant pour ne pas frustrer ceux qui, lorsqu’ils se trouvent devant une application aiment savoir un minimum comment elle fonctionne que pour rendre les explications ultérieures plus claires et compréhensibles. Devant quoi se trouve-t-on lorsqu’on a lancé la commande

```
# emacspeak
```

à l’invite du *shell*. En fait, on est entré dans l’éditeur de texte *Emacs*. Comme je l’ai déjà mentionné, *Emacs* est en fait, (au moins en première approximation) un éditeur de texte programmable. Il possède son propre langage de programmation (*emacs-lisp*) au sujet duquel il est hors de propos dans ce document introductif de donner le moindre détail. Toujours est-il qu’*Emacspeak* est un programme en *emacs-lisp* donc exécutable par *Emacs*. Lorsque vous avez lancé *Emacspeak* vous avez en fait exécuté un script *shell* qui indique à *Emacs* de se lancer en exécutant les programmes d’*Emacspeak*. La conséquence la plus immédiate est que vous disposez de tout le corpus de commandes standard de *Emacs* plus les commandes d’*Emacspeak*. Voir la section III.3.6 pour une présentation rapide de ces dernières. Ça fait beaucoup à apprendre ! Je vous l’accorde, mais si je me suis donné la peine de rédiger ce texte c’est que probablement ça vaut la peine ou que je suis affecté de dangereuses manies et souffre d’une attirance perverse pour la complexité inutile ! Il est temps pour vous de faire un choix ! et d’interrompre immédiatement la lecture de ce texte ou de me faire un peu confiance !

Je dirais pour ma défense, que l’apprentissage des commandes de *Emacs* et d’*Emacspeak* peut se faire de manière très très progressive. Ces outils peuvent être utilisés à plusieurs niveaux depuis un usage très élémentaire jusqu’à des astuces extrêmement élaborées. Disons qu’en dix minutes vous saurez comment écrire un texte, le sauvegarder et le recharger la prochaine fois que vous en aurez besoin, tout en ayant un feed-back sonore sur toutes ces opérations. En une demi heure vous aurez achevé la lecture des paragraphes essentiels de ce document et vous connaîtrez les commandes élémentaires d’*Emacspeak* et de *Emacs* et vous saurez où trouver l’aide en ligne et comment la consulter. Vous aurez appris à repérer où se trouve l’information correspondant à la tâche précise que vous voulez effectuer. N’hésitez cependant toujours pas à *me contacter* si vous vous trouvez démunis devant une situation.

III.3.4.4. – Problèmes au démarrage

- La commande n'est pas reconnue par le *shell* : Très vraisemblablement, l'installation n'est pas correcte, *Emacs* n'est pas présent (ou le script de lancement n'est pas dans un répertoire du path,) ou bien le script de lancement d'*Emacspeak* n'est pas dans un répertoire du path ... Vérifier ces problèmes système.
- *Emacspeak* se lance mais aucun message n'est prononcé. Très vraisemblablement un beep sera émis signalant une erreur interne de *Emacs*. Cette dernière peut être identifiée en examinant le *buffer* « *Messages* » qui doit normalement apparaître dans la liste qui s'affiche si l'on ouvre le menu principal de *Emacs* par F10 et l'on clique sur l'item *buffer*. Le contenu du message d'erreur peut être relativement clair, et il s'agira, du moins de l'expérience que j'en ai, vraisemblablement d'un mauvais lien entre *Emacspeak* et le speech server ou d'un problème d'accès au périphérique de synthèse vocale. Vérifiez entre autre les permissions pour ce dernier.

III.3.5 . – Les synthèses vocales supportées par *Emacspeak*

III.3.5.1. – *ViaVoice*

III.3.5.2. – *Festival 1.96-beta* Cette partie n'existe pour l'instant qu'en anglais.

III.3.6 . – Survol des possibilités d'*EMACSPEAK*

Si vous avez installé correctement *Emacspeak* et que vous l'avez bien connecté à une *synthèse vocale*, (cf. III.3.4.) vous pouvez lancer la commande

```
# emacspeak
```

À l'entrée dans *Emacspeak*, un message de bienvenue doit normalement se faire entendre. Il peut dépendre de la *synthèse vocale* que vous utilisez et par conséquent de la couche intermédiaire (*speech server* (cf. III.3.4.1.4.))

Si vous n'entendez aucun message ou si un beep est émis, il y a tout lieu de croire que quelque chose fonctionne de travers. Reportez vous donc à la section III.3.4.4.

Sinon, vous êtes dans *Emacspeak*. Nous l'avons déjà mentionné ici ou là, *Emacspeak* qui est construit sur *Emacs*, est d'abord un *éditeur de texte*. Vous pouvez donc dès l'entrée, saisir du texte à partir du clavier. *Emacspeak* sonorise vos commandes voici comment.

III.3.6.1. – Sonorisation des commandes de *Emacs* Dans son fonctionnement par défaut, *Emacspeak* produit un *écho caractère* et un *écho mot* c'est-à-dire qu'après la frappe de chaque caractère au clavier, celui-ci est prononcé par la *synthèse vocale*. De même lorsqu'on presse la *touche* <ESPACE> le groupe de lettres précédant l'espace blanc qu'on vient d'insérer et limité par le blanc précédent est prononcé comme un mot.

On peut également utiliser les flèches pour se déplacer dans le texte qu'on est en train d'écrire. Chacun de ces mouvements est sonorisé par *Emacspeak*. Par exemple si l'on presse la *touche* <FLÈCHE HAUT>, le *curseur* se déplace sur la ligne précédente et celle-ci est prononcée. Si la *touche* <FLÈCHE DROITE> est pressée le *curseur* se déplace sur le caractère suivant et le ce dernier est prononcé.

Citons encore pour mémoire les commandes

- *Esc b* Déplace le *curseur* sur le *mot* précédent et *Emacspeak* le prononce.
- *Esc f* Déplace le *curseur* sur le *mot* suivant et *Emacspeak* le prononce.

- *Esc a* Déplace le *curseur* au début de la *phrase* sur laquelle il se trouve. Celle-ci est alors prononcée.
- *Esc e* Déplace le *curseur* à la fin de la *phrase* sur laquelle il se trouve et prononce la *phrase* suivante.

Voir le *paragraphe Commands for Human Languages* et le sous-*paragraphe Words* (resp. *Sentences*) dans la documentation de *Emacs* accessible grâce à la *commande Ctrl-h i* (cf. III.3.6.5.2.)

Le logiciel *Emacs* possède un ensemble très étendu de commandes et celles-ci sont sonorisées comme nous venons de le voir sommairement pour les plus élémentaires d’entre elles. Nous n’irons pas plus loin dans ce *paragraphe* et nous donnerons plus détails dans les sections consacrées aux diverses possibilités de *Emacs*. Nous indiquerons à chaque fois comment les commandes de *Emacs* ont été sonorisées par *Emacspeak*.

Un certain nombre de commandes sont cependant spécifiques à *Emacspeak*.

III.3.6.2. – Les commandes les plus élémentaires d’*Emacspeak* La manière de fonctionner d’*Emacspeak* décrite dans le *paragraphe* III.3.6.1 est, pourrait-on dire, spontanée. L’utilisateur effectue des tâches d’édition qui sont sonorisées par *Emacspeak*. Cependant ce retour sonore sur ses actions peut s’avérer insuffisant pour l’utilisateur et il peut vouloir obtenir des informations sur le texte qu’il a devant lui sans pour autant modifier celui-ci et même sans y déplacer le *curseur*. Pour cela *Emacspeak* dispose d’une panoplie extrêmement étendue de commandes. Nous ne détaillerons ici que les plus importantes d’entre elles. Ces commandes peuvent toutes être effectuées grâce à une succession (combinaison) de touches du clavier. Il existe de multiples moyens d’exécuter des commandes sous *Emacs* mais nous ne les détaillerons pas dans cette section. Se reporter à la section III.1.2. Toutes les commandes spécifiques à *Emacspeak* sont déclenchées par une succession de combinaisons de touches commençant par *Ctrl-e* la touche contrôle et la touche “e” pressées simultanément.

Une fois ces deux touches relâchées, en pressant sur une autre touche, on peut obtenir les effets suivantes :

- En pressant “l” après avoir relâché les touches contrôle et “e” c’est-à-dire en pressant la séquence de touches *Ctrl-e l* on obtient d’*Emacspeak* qu’il prononce la *ligne* sur laquelle se trouve le *curseur*.
- *Ctrl-e w* prononce le *mot* sous lequel se trouve le *curseur*. Si cette commande est effectuée deux fois de suite au même endroit (sans déplacer le *curseur*.) la seconde fois le *mot* est épilé.
- *Ctrl-e W* épelle systématiquement le *mot* sous lequel se trouve le *curseur*.
- *Ctrl-e <FLÈCHE HAUT>* prononce la *ligne* précédente sans y déplacer le *curseur* pour autant. La séquence *Ctrl-e* suivie des autres touches de déplacement produit des effets du même genre.
- *Ctrl-e {* Permet de prononcer le *paragraphe* dans lequel se trouve le *curseur* c’est-à-dire la portion de texte (de *buffer* (cf. III.1.4)) comprise entre deux groupes de *retour chariots* successifs. Faire *Ctrl-h k* puis *Ctrl-e {* pour obtenir plus de détails concernant cette *commande* et notamment pour lire la partie de *paragraphe* se trouvant avant (resp. après) le *curseur*.
- *Ctrl-e [* Permet, suivant un mode tout à fait analogue à celui de la *commande* précédente, de lire la page dans laquelle se trouve le *curseur*.
- *Ctrl-e c* permet d’identifier le *caractère* placé sous le *curseur* en le prononçant grâce à l’alphabet international : (par exemple a donne alpha.) Attention pour les utilisateurs français cette *commande* est un peu limitée puisqu’elle est inopérante sur les *caractères* accentués.

III.3.6.3. – Réglages du synthétiseur Il est bien évident que l’utilisateur peut souhaiter modifier certains paramètres de la *synthèse vocale* (la vitesse d’élocution, le timbre de la voix etc ...) sans être obligé d’intervenir par le biais d’un autre outil qu’*Emacspeak* qu’il est en train d’utiliser. Le

programme offre bien entendu de nombreuses possibilités pour cela. Bien évidemment, comme nous le disions déjà dans le paragraphe III.3.1.4 “la plus jolie fille ne peut donner que ce qu’elle a” ! et *Emacspeak* ne pourra pas accroître les possibilités de la *synthèse vocale* que vous utilisez. Les fonctions prévues par *Emacspeak* pour les réglages de la *synthèse vocale* ne seront opérantes que si la *synthèse vocale* implémente des possibilités adéquates. Nous nous bornerons à donner ici les fonctions les plus courantes qu’il devrait avoir un support avec n’importe quelle *synthèse vocale*.

Toutes les séquences permettant ces réglages commencent par *Ctrl-e d* (touches contrôle et “e” pressées simultanément puis relâchées, puis la touche “d” elle-même relâchée avant qu’une nouvelle touche déclenchant une commande spécifique ne soit pressée.)

- *Ctrl-e d r* “*dtk-set-rate*” permet de configurer la vitesse d’élocution de la *synthèse vocale*. Vous devez alors entrer une valeur au clavier suivie de la touche <ENTRÉE>. Cette valeur dépend en fait de la *synthèse vocale* que vous utilisez néanmoins, pour la plupart des *speech servers*, elle s’exprime en un rapport nombre de mots par minute et des valeurs entre 300 et 500 sont raisonnables. Le plus simple est que vous fassiez quelques essais pour savoir quel est la vitesse vous convient.

Pour des *synthèse vocale* comme *ViaVoice* (cf. III.3.5.1o) ou *Festival* (cf. III.3.5.2l) les réglages sont un peu particuliers, et je vous conseille de vous reporter aux paragraphes spécifiquement consacrés à ces outils.

III.3.6.4. – Comment connaître la commande déclenchée par une touche ? Une commande particulièrement intéressante, qui n’est pas spécifique à *Emacspeak* mais est une commande de *Emacs* sonorisée par *Emacspeak* est *Ctrl-h k*. En pressant cette séquence de touches, suivie d’une séquence quelconque, *Emacs* affiche (dans une autre fenêtre) l’aide concernant la séquence suivant *Ctrl-h k*. Ce court texte est alors prononcé par *Emacspeak*. Si vous ne parvenez pas à saisir immédiatement tous les renseignements contenus dans ce texte, vous pouvez placer le *curseur* dans la fenêtre contenant cette aide en effectuant la séquence *Ctrl-x o*. Dès lors, en déplaçant le *curseur* grâce aux flèches vous pourrez lire ce texte plus lentement, *ligne* à *ligne* et même *mot* à *mot*. En pressant de nouveau *Ctrl-x o* vous replacerez le *curseur* dans la fenêtre où vous vous trouviez auparavant (voir la section III.1.3 pour plus de détails concernant les fenêtres ⁴.)

III.3.6.5. – Comment obtenir de l’aide de manière plus générale ?

III.3.6.5.1 Une aide sommaire et spécifique à *Emacspeak* La combinaison de touches *Ctrl-h Ctrl-e* ouvre un *buffer* (cf. III.1.4c) contenant une aide sommaire pour *Emacspeak*. Le *curseur* est automatiquement placé dans ce *buffer* et vous pouvez le lire à loisir *ligne* à *ligne*.

Des renseignements sommaires vous seront donnés pour un certain nombre de séquences de touches spécifiques à *Emacspeak*. Vous pourrez compléter les informations concernant chacune d’entre elles en utilisant la séquence *Ctrl-h k* comme décrit au paragraphe III.3.6.4. Vous pouvez aussi placer le *curseur* le nom de la commande affectée à une séquence de touches particulière puis presser la séquence *Ctrl-h f*. *Emacspeak* vous demandera alors si vous voulez effectivement obtenir l’aide correspondant à cette commande. En pressant la touche <ENTRÉE>, vous confirmerez que vous voulez bien de l’aide concernant cette commande et un court paragraphe sera alors affiché et prononcé.

III.3.6.5.2 L’info sous *Emacs* La combinaison de touches *Ctrl-h i* ouvre un *buffer* (cf. III.1.4) en *mode info* (cf. III.1.8.5) contenant la racine de la documentation disponible sous *Emacs*. Cette

⁴Il se peut qu’il y ait plus de deux fenêtres ouvertes, auquel cas la commande *Ctrl-x o* les parcourt les unes après les autres de manière circulaire.

documentation est organisée de manière arborescente avec quelques possibilités hypertexte. Après avoir entré la séquence *Ctrl-h i*, vous pouvez vous déplacer à travers une sorte de table des matières de cette documentation. Il y a, normalement, si l'installation des programmes est correcte, un item correspondant à *Emacs* et un autre correspondant à *Emacspeak*. En plaçant le *curseur* dessus, puis en pressant la *touche* <ENTRÉE>, vous accédez au niveau de l'arborescence de la documentation spécifiquement consacré à *Emacs* ou *Emacspeak*. Vous constaterez qu'un grand nombre d'autres items sont disponibles, vous donnant accès à des informations sur nombre d'autres utilitaires *Linux*.

ATTENTION Dans le *buffer* que vous venez d'ouvrir, les touches du clavier n'ont plus tout à fait les mêmes fonctions que dans un *buffer* standard (en *mode fondamental* (cf. III.1.8.4.)

- Comme nous l'avons déjà dit, la *touche* <ENTRÉE> vous permet de progresser d'un niveau dans l'arborescence.
- La *touche* *u* vous permet de remonter au noeud supérieur.
- *n* vous permet d'accéder à l'item suivant (next).
- *p* item précédent.

III.3.6.5.3 Le tutoriel de *Emacs* En pressant la combinaison de touches *Ctrl-h t*, vous entrez dans un ttutoriel de *Emacs* (en anglais) que vous pouvez lire comme n'importe quel *buffer* en vous y déplaçant avec les flèches et en utilisant les commandes spécifiques à *Emacspeak* (cf. III.3.6.2.)

III.3.7 . –Utilisation avancée d'*Emacspeak*

III.3.8 . –Outils connexes

III.3.8.1. –*navigateur internet*

IV . –Notes pour les développeurs

This section is written in English since developpers understand this language most of the time. Many of them are not French people.

IV.1 . –About *E.F.M.*

Instructions to install an use *E.F.M.* are contained in paragraph I.1 (in French). The goals of this project and the technical solutions I implemented are described here (in English).

The only question to answer now is *why decided not to continue this project ?*

The quickest answer is that I discovered *Speech Dispatcher 0.6.1* during the development of this project.

To sum up, I planed in the *E.F.M.* project to implement a *Festival 1.96-beta* client for *Emacspeak 24* directly in *Emacs 21.4a*. Indeed I wanted to use *Emacspeak* with *Festival* since I first think that *Festival* has good speech qualities and secondly since *Festival* can be multilingual which is a appreciable feature for a French guy.

However, *Speech Dispatcher* is a very powerful interface between the applications and the speech systems and in a certain sens developing an *Emacs* client for the *Festival* server was reimplementing *Speech Dispatcher*. Indeed *Speech Dispatcher* comes with a well developed *Emacs* client *Speechd-el 2.1*. *Speech Dispatcher* is able to manage the messages in a very clever manner defining priorities and a full featured communication protocol *SSIP*. Implementing such a protocol is a tricky and not very funny work and moreover it was done !

However there are two objections : First introducing a layer between *Emacs* and *Festival* could make the process slower and decrease its responsiveness. Sure but computers are faster everyday as well as cheaper ! According to my own experience using *Speech Dispatcher* between the application and the server doesn't not make the computer talk slower !

The second objection is that *Speechd-el* is not *Emacspeak* and that many *Emacspeak* users don't find their favorite *Emacspeak* features in *Speechd-el*. I have been an *Emacspeak* user for a long long time and I can understand this point of view. I do agree with the idea that it could be a goal to implement the well known and powerful *Emacspeak* features for *Speechd-el*.

However I think I sufficiently explained before why for me the solution is not to rewrite a full featured *Emacs* client for *Festival*.

The direction in which I think that it is reasonable to work is to use the high level *Emacspeak* modules and to connect them with *Speechd-el*. However it is a tricky work as well since the philosophy of both projects is definitely not the same and someone could say definitely incompatible ! I don't think so since *Speechd-el* provides a very flexible library which can be used to reimplement the low level *Emacspeak* methods.

The module which is mainly concerned is *emacspeak-speak* which is meant to protect the high level modules from the speech-server dependency. The job to be done in a certain sense is to rewrite completely this module with the *Speechd-el* routines. I already detected some traps in such a work : for instance certain tasks which were dedicated to *Emacspeak* (like punctuation filtering for instance) are now accomplished by the *Speech Dispatcher* server and must simply be called by the client. So before writing any line of code we should list all these features.

Other features were hard coded in *Emacspeak* and can be obtained simply by customizing⁵ certain variables with *Speechd-el*.

Finally I must say that I am a quite good experimented *emacs-lisp* programmer and that I investigated deeply both *Emacspeak* and *Speechd-el* codes.

⁵I mean using the powerful customize *Emacs* mechanism.

Index

- <PREFIX>/bin/festival, 9
- <PREFIX>/share/festival, 9
- <PREFIX>/share/festival/bin, 9
- <PREFIX>/share/speech_tools, 9
- (y or n ?), 31, 32
- (yes or no ?), 31, 32
- ./configure, 2–5, 11, 16, 17, 20
- ./efm_test_fr, 2
- ./efm_test_fr_noaudio, 2, 3
- .emacs, 19, 23
- /lib, 14
- /tmp, 12
- /usr, 20
- /usr/local, 6, 20
- /usr/local/bin, 17
- /usr/local/etc/, 17
- /usr/local/share/emacs/site-lisp/, 19, 23
- /usr/local/share/festival/bin, 12
- /usr/local/share/festival/lib, 14
- /usr/share/emacs/site-lisp/, 19, 23
- /usr/share/sounds/sound-icons/, 13
- écho caractère, 44
- écho mot, 44
- éditeur de texte, 27, 28, 38, 41–43

- a, 36

- bin, 12, 17
- buffer, 28–36, 43, 45, 46
- buffer actif, 28, 30–33

- C, 36
- caractère, 28–32, 36, 45
- CC, 6
- cd, 2, 5, 14, 16, 35
- client, 17, 18
- client/serveur, 16, 17
- clients, 16
- combinaison de touche, 32
- combinaisons de touches, 32
- commande, 28–36, 44, 45
- contexte, 27, 28, 31, 32
- cp, 35, 36
- Ctrel-e w, 44
- Ctrl, 30
- Ctrl-e, 44
- Ctrl-e [, 45
- Ctrl-e {, 45
- Ctrl-e <FLÈCHE HAUT>, 44
- Ctrl-e c, 45
- Ctrl-e Ctrl-l, 36
- Ctrl-e d, 45
- Ctrl-E d l, 36
- Ctrl-e d r, 45
- Ctrl-e l, 44
- Ctrl-e W, 44
- Ctrl-g, 30
- Ctrl-h Ctrl-e, 46
- Ctrl-h f, 46
- Ctrl-h i, 29, 36, 44, 46
- Ctrl-h k, 45, 46
- Ctrl-h t, 46
- Ctrl-x b, 28, 29, 31
- Ctrl-x Ctrl-b, 30
- Ctrl-x Ctrl-c, 30–32
- Ctrl-x Ctrl-f, 30, 32
- Ctrl-x Ctrl-f., 30
- Ctrl-x Ctrl-s, 30, 32
- Ctrl-x d, 35
- Ctrl-x k, 32
- Ctrl-x o, 45
- curseur, 28, 29, 31, 33, 36, 44–46

- D, 36
- DotConf, 16, 22

- E.F.M., 1–3, 5, 46, 47
- E.S.D.F.F.M., 5, 20, 21
- Eieio, 18, 22, 23
- Emacs, 16–21, 23, 27–33, 35, 36, 38, 41–47
- emacs-lisp, 18, 19, 22, 27, 42, 47
- Emacspeak, 1, 2, 21, 27–31, 33, 35–47
- en, 36
- Esc a, 44
- Esc b, 44
- Esc e, 44
- Esc f, 44
- Esc x, 30, 33, 35

- f, 29
- F10, 29
- Festival, 1–9, 11–15, 17, 21, 41, 42, 45, 47

festival, 9, 11, 12
 festival-freebsoft-utils.info, 16
 festival_client, 9, 12
 festival_server, 9, 12
 festival_server_control, 12
 fichier, 29–33, 35–40
 fr, 36
 Fr Sd TTS, 3–5, 12, 15
 Franfest, 4–6, 9–11, 20
 Freebsoft Utils, 4, 13–16, 20

gcc, 11
 gestionnaire de fichiers, 35
 GLib, 16, 24
 gnu tar, 37
 Gnu/Linux, 1
 groups lambda, 7
 gzip, 37

html, 28, 29, 40
 hypertexte, 40

info, 14, 28–30, 33, 40
 insertion, 28–32

lib, 14
 ligne, 36, 44–46
 Linux, 37, 38, 41, 46
 ll, 34, 36
 load-path, 14
 ls, 34, 35

make, 2, 5, 6, 11, 14, 16
 make install, 17
 Makefile, 11, 14
 Mbrola, 1, 5, 7, 8, 10
 menu, 29
 menu principal, 29
 minibuffer, 28–31, 33
 mkdir, 14
 mode, 28, 30–33, 40
 mode dired, 35, 36
 mode fondamental, 33, 46
 mode info, 31–33, 46
 mode serveur, 12, 21
 mode shell, 33, 35
 mot, 28–30, 44, 45
 mv, 35, 36

n, 32, 46

navigateur internet, 29, 46

o, 29
 objet, 27, 28, 30

p, 46
 paragraphe, 30, 44, 45
 patch -Np1 -i ../franfest-2006-06.patch, 4
 patch -Np1 -i ../freebsoft-2006-06.patch, 4
 PATH, 9, 12
 phrase, 30, 44
 Pkg-Config, 16, 24, 25

quit, 30

R, 36
 répertoire, 35–37, 39, 40
 Recode, 14, 25, 26
 retour chariot, 45
 revue d'écran, 33, 38
 rm, 35, 36
 root, 6

Scheme, 14, 15
 script, 12, 17
 serveur, 16
 shell, 15, 30, 33, 37, 40, 42, 43
 Sound Icons, 13, 14, 20
 sous-objet, 28, 30
 Sox, 14, 26
 spd-say, 3, 17
 Speech Dispatcher, 3, 12–18, 20, 21, 47
 speech server, 42, 43, 45
 speech-dispatcher, 17
 speech_tools, 11, 12
 Speechd-el, 12, 16, 18–21, 27, 36, 47
 speechd-set-language, 36
 SSIP, 47
 synthèse vocale, 29, 36, 38, 39, 41, 43–45
 synthèses vocales, 11
 système d'exploitation, 31–33, 37, 38, 41

TAB, 30, 31
 tar -xjf, 2, 5
 tar -xzf, 11, 13, 14, 16, 19, 23
 target install, 11
 text2wave, 3, 9, 12
 tmpfs, 12
 touche <ENTRÉE>, 28–31, 33–36, 40, 45, 46
 touche <ESPACE>, 29, 44

touche <FLÈCHE BAS>, 29
touche <FLÈCHE DROITE>, 44
touche <FLÈCHE HAUT>, 29, 44

u, 28, 33, 46
utilisateur, 28, 30–32, 35

ViaVoice, 43, 45
visite, 30–32

y, 31

z, 36