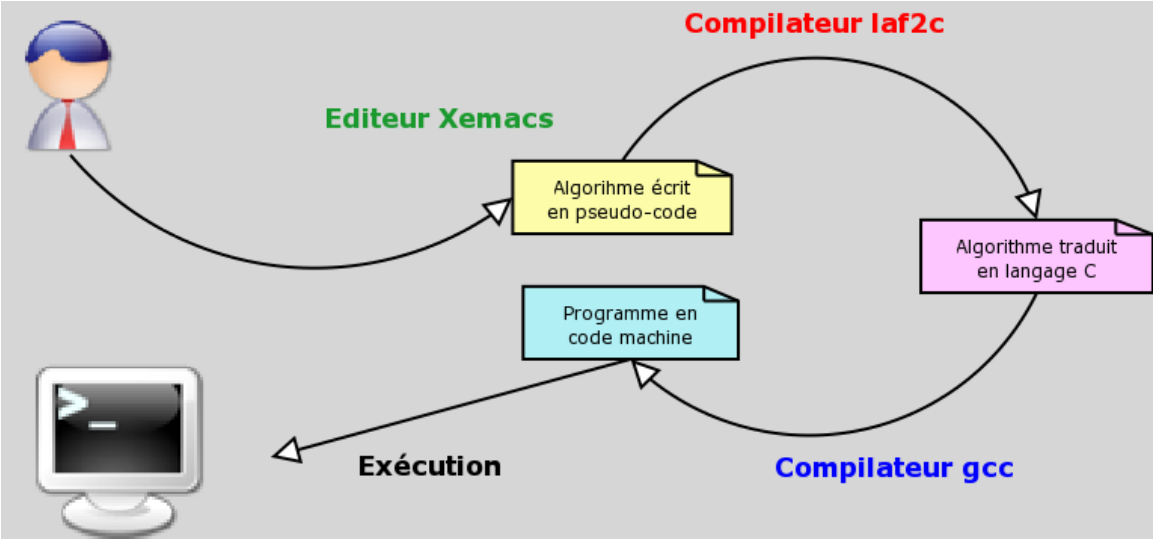
Annexe A

Utilisation des outils du laboratoire

Ce document est destiné à présenter les outils du laboratoire. Il en présuppose une installation réussie.



La création d'un programme comporte les opérations suivantes :

- conception de l'algorithme en pseudo-code (on peut l'encoder avec l'éditeur de texte **xemacs**)
- traduction du pseudo-code en langage C (opération possible avec le programme **laf2c**)
- traduction du langage C en code machine (phase obligatoire avec le compilateur **gcc**)
- exécution du programme sur l'ordinateur.

La phase la plus longue et la plus complexe consiste à encoder le texte du programme. On utilise pour cela un éditeur de texte. N'importe quel éditeur peut convenir, mais **xemacs** présente l'avantage d'intégrer l'édition, les compilations et la correction des erreurs dans le même environnement.

A.1 Installation des outils

A.1.1 Installation des binaires sous Linux

La plupart des distributions Linux comprennent les outils de programmation nécessaires à l'utilisation de notre compilateur. On veillera à les installer avant de procéder à l'installation du compilateur proprement dit. Le programme d'installation fourni (un simple script) s'assure de la présence des composants prérequis :

– le compilateur *GCC*
– l'éditeur *Xemacs* [1]
– l'intégrateur *make.*

Le programme d'installation se compose d'un fichier archivé et d'un script. Il suffit de lancer le script avec des droits d'administrateur (à l'aide `su` ou de `sudo`).

```
jactho@naxos:>sudo install.sh
```

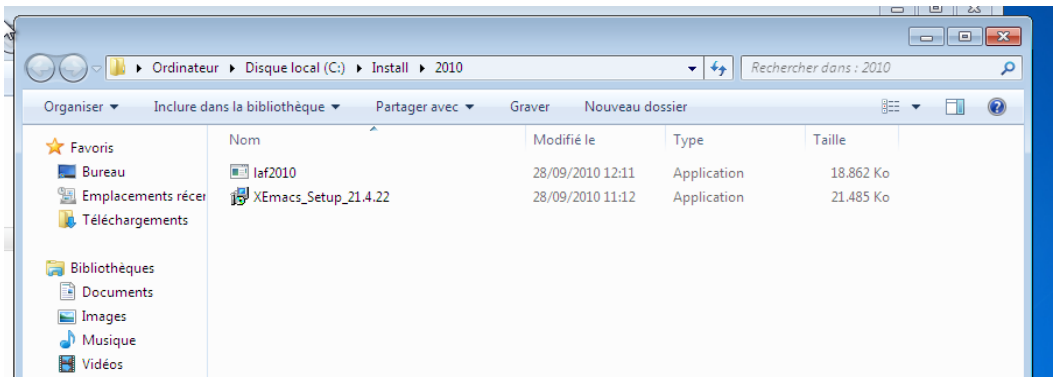
A.1.2 Installation des sources sous Linux

Cette méthode est réservée aux **utilisateurs avertis**. Ils disposent des sources sous forme de programmes C. La séquence traditionnelle `./configure, make` sera réalisée, par sécurité, par un utilisateur normal. Le programme `make install` sera lancé pour l'installation finale par l'administrateur.

A.1.3 Installation sous Windows

Récupération des programmes d'installation

En principe, les deux programmes fournis sur mon site permettent l'installation sur toutes les versions de Windows. Vista et Seven nécessitent une petite manipulation finale que XP et Windows 2000 assurent automatiquement. L'image suivante montre les deux programmes enregistrés sur un disque dur, dans le dossier `Install/2010` du disque C:.

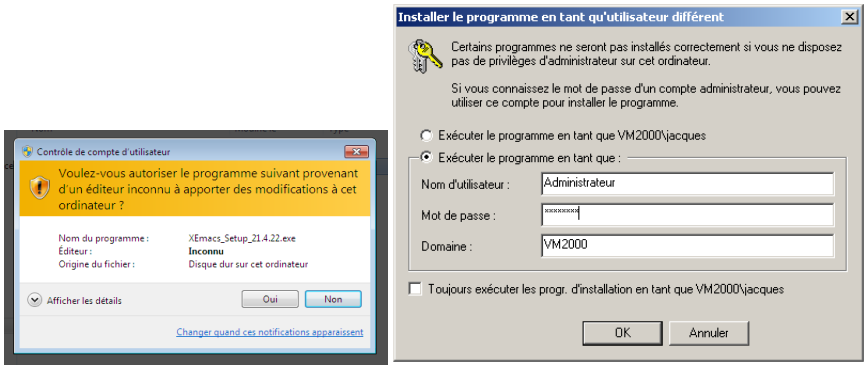


Installation de Xemacs

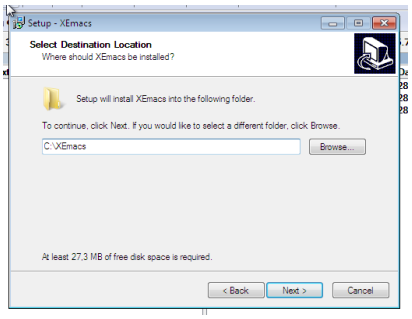
Le programme Xemacs est fourni à l'aide d'un installateur qui ne demande quasiment aucune intervention de la part de l'utilisateur. La seule intervention consiste en fait à choisir le répertoire de base de Xemacs. Pour éviter des arborescences trop complexes, je suggère de l'installer directement dans la racine au lieu du répertoire « Program Files », option préprogrammée dans le progamme d'installation. Ce point est souvent négligé par les étudiants et cause fréquemment des dysfonctionnement.

1. Lancement du programme : en cliquant sur l'icône du programme d'installation.
2. Si vous n'êtes pas administrateur de la machine, on peut vous suggérer de le devenir ou de confirmer votre intention de modifier l'ordinateur.

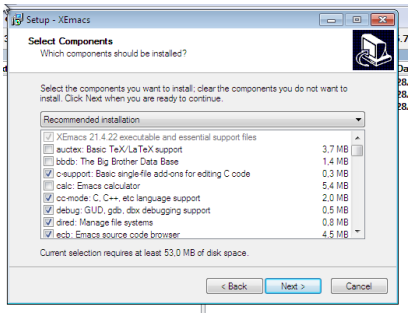
1. Cette version est généralement disponible avec les grandes distributions. Les macros telles qu'elles sont livrées sont conçues pour fonctionner avec Xemacs. Certaines d'entre elles ne fonctionnent pas avec GnuEmacs. Il est parfaitement possible d'installer les deux versions de l'éditeur.



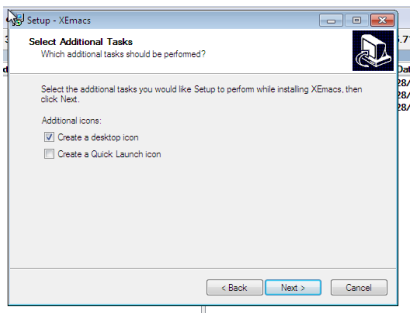
3. Le programme procède à la décompression des fichiers
4. On vous propose ensuite d'accepter la licence d'utilisation (en l'occurence la General Public License)
5. Il faut ensuite choisir le dossier d'installation. **Ne pas accepter le dossier par défaut !**



6. Remplacez le choix par défaut (`c:\Program files\Xemacs`) par `c:\XEmacs`.
7. Les composants proposés dans un écran ultérieur conviennent parfaitement à nos besoins.



8. Je conseille de créer une icône sur le bureau (« desktop icon »). L'icône « Quick launch » risque d'encombrer la barre du bas de l'écran. Il n'est pas souhaitable de l'installer.



9. La suite du programme d'installation se déroule sans autres incidents.

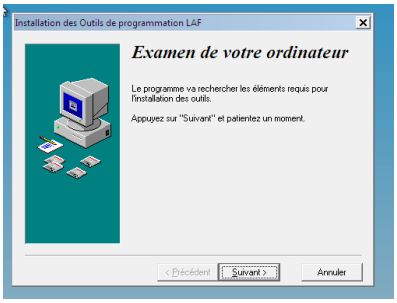
Installation des autres outils

La compilation en C nécessite les outils suivants :
- le compilateur C `gcc`
- le programme `make` qui permet d'automatiser le processus depuis la sauvegarde du fichier en pseudo-code jusqu'à l'ouverture de la console.
- divers outils Unix
- le compilateur `laf2c` lui-même

Tous ces composants doivent interagir et il faut donc qu'ils sachent où ils se trouvent respectivement. Un programme d'installation composé par mes soins répond à toutes ces contraintes [2]. Il va essayer de trouver Xemacs et proposera d'installer les autres outils dans des dossiers situés au même endroit.

1. Le programme à lancer se nomme `laf2010.exe`
2. Un écran de bienvenue s'affiche.

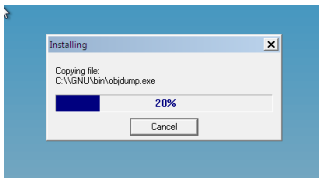
3. Le programme examine l'ordinateur...



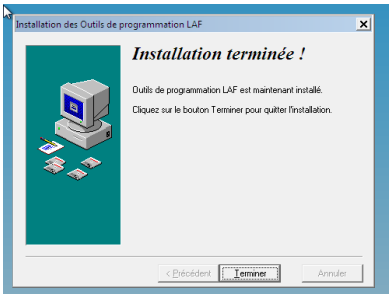
4. Il propose d'installer les programmes parallèlement à Xemacs (ici le disque C : choisi lors de l'installation de l'éditeur).

2. J'ai utilisé pour le réaliser un logiciel gratuit de Microsoft, assez ancien mais très efficace : *SMS Installer version 1.0*. D'autres versions de ce produit sont sorties depuis longtemps, mais pourquoi remplacer un programme qui fonctionne et que j'ai appris à utiliser ? Mes écrans d'installation ont un petit côté démodé qui surprendra certains. L'important n'est-il pas que le programme final fonctionne ? En outre, ce programme tourne parfaitement sous Linux à l'aide de Wine.

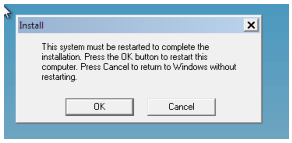
5. L'installation commence



6. L'installation s'est terminée.

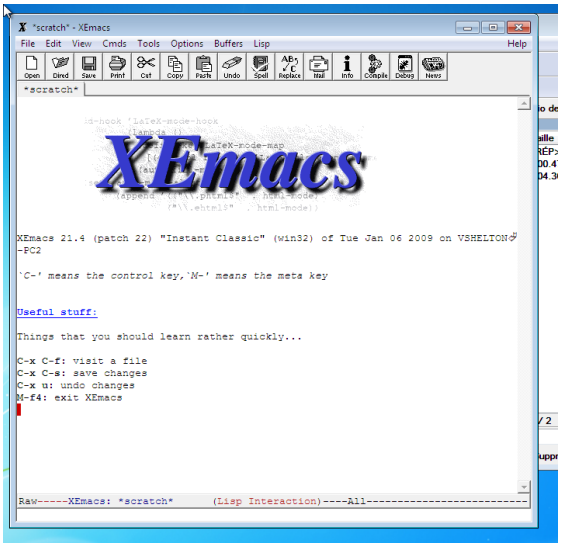


7. L'ordinateur propose de relancer Windows. Ce n'est nécessaire que sous 2000 et X, pour lesquels l'installation est terminée. Sous Vista et Seven, ce serait une perte de temps inutile. Nous allons voir comment terminer l'installation sous ces deux dernières versions de Windows.

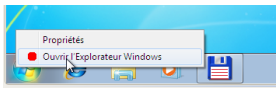


Fin de l'installation sous Vista et Seven

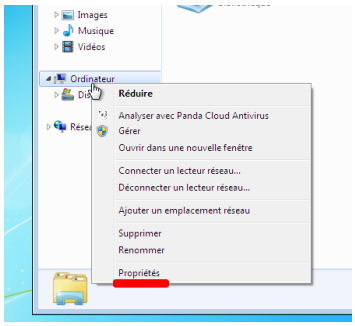
Windows utilise comme Unix des variables d'environnement, mais s'efforce de les masquer à son public décébré. Il faut modifier deux variables pour faire fonctionner le système. Si le travail n'a pas été convenablement réalisé, l'écran d'accueil de Xemacs reste désespérément gris et aucun raccourci ne fonctionne.



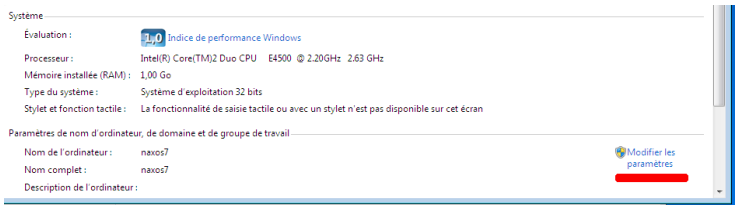
– par un clic droit sur le bouton « Démarrer », ouvrir l'explorateur Windows (la combinaison Windows-E donne le même résultat).



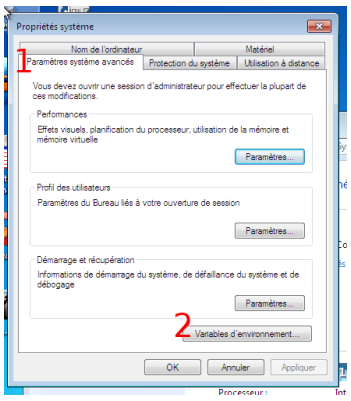
– un autre clic droit sur l'icône représentant l'ordinateur permettra d'accéder à ses propriétés (dernière ligne du menu).



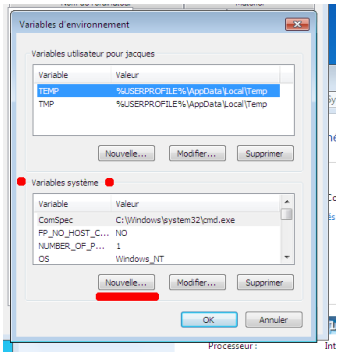
– parmi les propriétés, on trouvera dans la zone « Système », un lien nommé « modifier les paramètres »



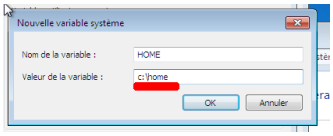
– dans la fenêtre « Propriétés système » qui s'ouvre, on choisira l'onglet « Paramètres système avancés » avant de finalement cliquer sur le bouton « Variables d'environnement ».



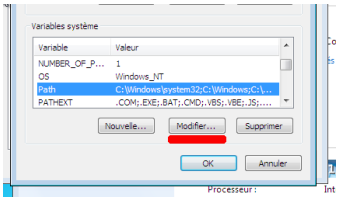
– parmi les variables d'environnement, nous allons modifier les variables systèmes, communes à tous les utilisateurs.



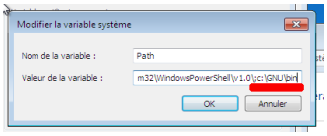
– il faut d'abord créer un première variable `HOME` (par le bouton « Nouvelle »). Nous lui donnons la valeur `c:\home` avant de cliquer sur OK. Ce dossier contiendra quelques paramètres et nous y placerons nos exercices.



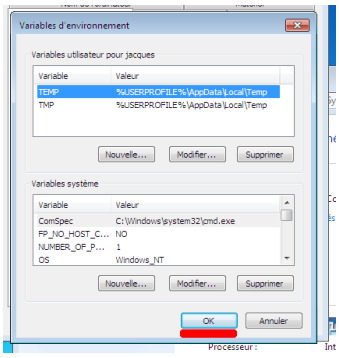
– parmi les variables existantes, nous sélectionnons ensuite « Path », qui indique le nom des dossiers qui contiennent les programmes exécutables de Windows. Nous allons modifier cette variable en y ajoutant le dossier `c:\GNU\bin`. Il faut pour cela enfoncer le bouton « Modifier »



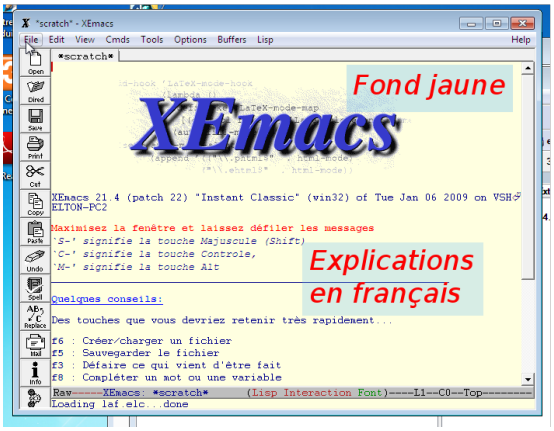
– la suite est délicate, car toute erreur pourrait endommager le système
  – **ne pas remplacer le contenu actuel de la variable**, mais se placer à la fin de la chaîne (sur ma machine, elle se termine par v1.0, mais cela peut être différent sur votre système).
  – ajouter `;C:\GNU\bin` à la fin du texte en n'oubliant pas le point-virgule initial, qui sert de séparateur.
  – on peut alors valider par OK.



– il reste encore à valider les modifications en cliquant sur OK dans la fenêtre des variables d'environnement. Sans cette manoeuvre, les changements ne seront pas pris en compte.



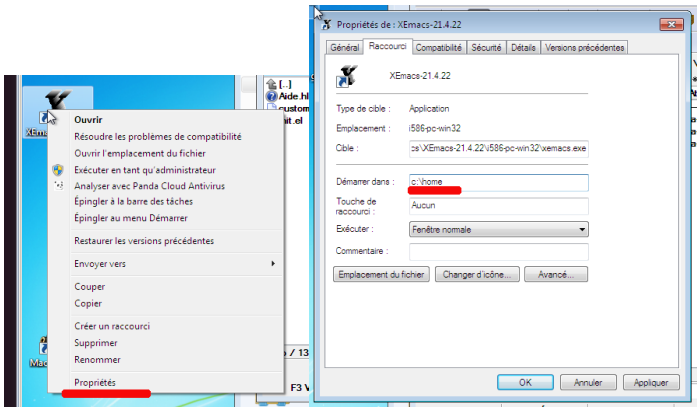
Si on lance maintenant Xemacs, il doit ressembler à ceci :



Changer le répertoire de travail

Lorsqu'on clique sur l'icône de l'éditeur située sur le bureau, le programme a la fâcheuse tendance à nous le proposer comme dossier par défaut. Il peut être utile de renseigner le dossier `c:\home` comme dossier de base. Il suffit de modifier les propriétés de l'icône.

- un petit clic droit sur l'icône
- choisir propriété
- remplir la zone « Démarrer dans ». Vous pouvez choisir un autre dossier de travail, mais vous devez taper le chemin au clavier.



A.2 Utiliser l'éditeur xemacs

Emacs , prononcé [imaks] est le premier programme « libre » réalisé par Richard STALLMAN et, à ce titre, il occupe un place importante dans l'histoire des logiciels libres. Il s'agit d'un éditeur, c'est-à-dire un programme servant à écrire des programmes, bien que ce ne soit pas sa seule vocation. Il présente les caractéristiques suivantes :

- il est *multi-plateformes*, ce qui garantit qu'on pourra l'utiliser sur n'importe quel ordinateur. Il existe une version d'Emacs pour chaque système d'exploitation. Nous utiliserons des versions pour Linux et pour Windows 32 bits.
- il est *modulaire*, ce qui signifie qu'il se compose de plusieurs centaines de fonctions (qui historiquement étaient des *macros*, qui lui ont d'ailleurs donné son nom). Chaque fonction porte un nom par lequel on peut l'activer. L'utilisateur peut parfaitement créer de nouvelles fonctions, dans un langage dérivé de Lisp, et ainsi adapter le comportement de son éditeur.

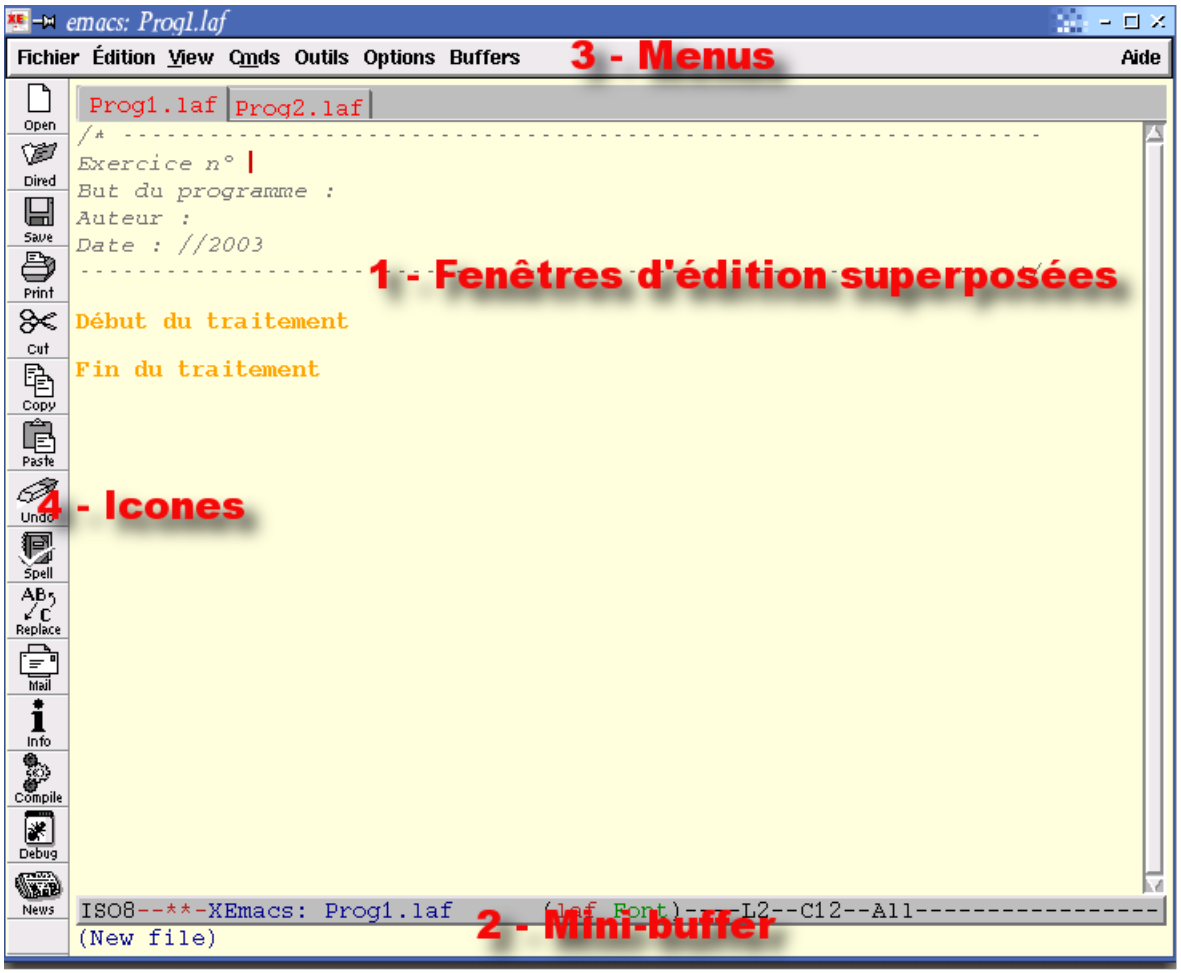


FIGURE A.1 – Le programme Xemacs

- il est centré sur une utilisation de *raccourcis-clavier* qui nécessitent un peu de mémoire humaine, mais procurent assez rapidement une grande rapidité à l'utilisateur. Les raccourcis sont modifiables dans une grande mesure. La version que nous utilisons a été modifiée pour faire un usage presque exclusif des touches de fonction, qui n'existent pas sur toutes les plateformes.
- certaines versions ont été adaptées pour tirer profit d'un environnement graphique, c'est le cas de Xemacs que nous utiliserons.
- il est sensible au contexte, notamment par le biais des extensions des noms de fichiers. Celui lui permet de modifier son comportement et la présentation du texte selon le langage employé par l'utilisateur. Ce sont les *modes*. Nous utiliserons essentiellement deux modes : le mode LAF, défini par mes soins pour rédiger du pseudo-code, et le mode C, l'un des plus anciens et des plus élaborés.

Nous utiliserons au laboratoire une version graphique du programme, nommée **xemacs**.

Le programme Emacs comporte une fenêtre principale dans laquelle nous pouvons distinguer quatre zones principales :

1. la **fenêtre d'édition**, dans laquelle nous tapons le texte du programme. En fait, nous disposons de plusieurs fenêtres superposées ou disposées côte à côte.
2. le **mini-buffer**, une zone d'une seule ligne dans laquelle nous donnons des indications au programme (nom du fichier à charger, de la fonction à exécuter).
3. des **menus**, semblables à ceux qu'on trouve dans tous les programmes (notons qu'ils ne

sont que partiellement traduits).

4. des **icônes** reprenant quelques fonctions élémentaires (sauvegardes, impressions...)

Une erreur fréquente consiste à utiliser une fonctionnalité quand le curseur est placé au mauvais endroit : par exemple lancer une sauvegarde pendant qu'on est en train de taper le nom d'un fichier à charger.

A.3 Création, compilation et exécution d'un programme simple

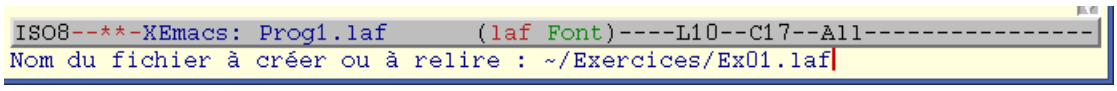
Lancer xemacs

Sous Linux, on utilisera au choix le menu KDE (*Utilitaires-Éditeurs-Xemacs*) ou le lanceur d'application appelé à l'aide de Alt f2. Il n'est pas conseillé de lancer Xemacs depuis une console, parce que la fermeture de celle-ci tue le programme sans possibilité de sauvegarder les textes.

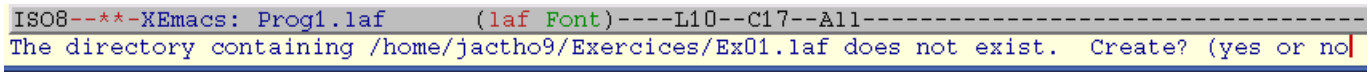
Sous Windows, on trouvera une icône de Xemacs dans le menu démarrer et, si on l'a autorisé lors de l'installation, sur le bureau.

Création d'un fichier

Lors de la **création** d'un nouveau fichier, nous allons devoir fournir le nom de ce fichier. La touche f6 nous permet de taper, dans le mini-buffer, le nom du fichier. Pour placer le fichier dans un nouveau dossier, il suffit de faire précéder le nom du fichier par le nom du dossier. Attention, sous Linux, les minuscules et les majuscules sont distinctes. Le fichier doit comporter une extension correspondant au langage employé. Rappelons que Emacs se sert de cette extension pour choisir son *mode*.



Si le dossier n'existe pas, on nous demande de confirmer sa création (cela pourrait être une faute de frappe).



Si le fichier existe déjà, il sera automatiquement chargé.

Nous devons veiller à placer les fichiers au bon endroit. La version Windows de Xemacs place systématiquement les fichiers sur le bureau de l'utilisateur, une idée pas très heureuse. On peut utiliser le dossier de base, noté par la convention Unix `~/`, pour placer les fichiers dans le dossier de l'utilisateur (le dossier `\home` sous Windows). Sous Linux, le dossier `/home/nomutilisateur` est le seul endroit où un utilisateur normal puisse écrire.

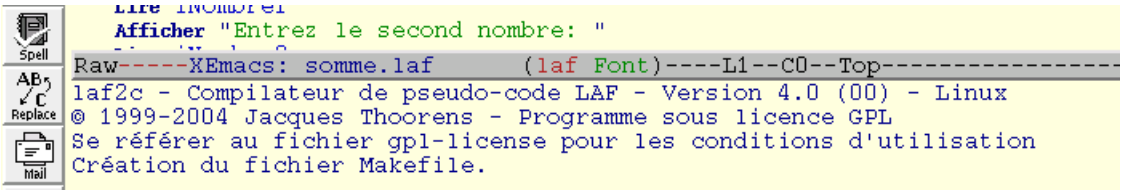
Sauvegarde

Emacs opère des sauvegardes automatiques pour nous protéger d'une coupure de courant. Nous devons cependant sauvegarder le fichier quand nous avons terminé de l'écrire. Cette opération doit également précéder la compilation d'un programme. La touche f5 sauvegarde le fichier sous le nom sous lequel il a été créé. Pour sauvegarder le fichier sous une autre nom,

on peut utiliser la combinaison Shift f5. Attention de ne pas employer un nom existant car on risque de perdre le contenu du fichier existant. Nous verrons que la touche de compilation effectue en fait une sauvegarde, ce qui rend inutile l'usage de *f5*.

Compilation

Lorsqu'on travaille dans un nouveau dossier, il convient de créer un petit fichier nommé `Makefile` qui permettra de réaliser la compilation de tous les programmes contenus dans ce dossier. L'appui sur la combinaison Shift f9 réalise cette tâche.



Dans la suite, on pourra utiliser directement la touche f9, qui réalise le travail suivant :

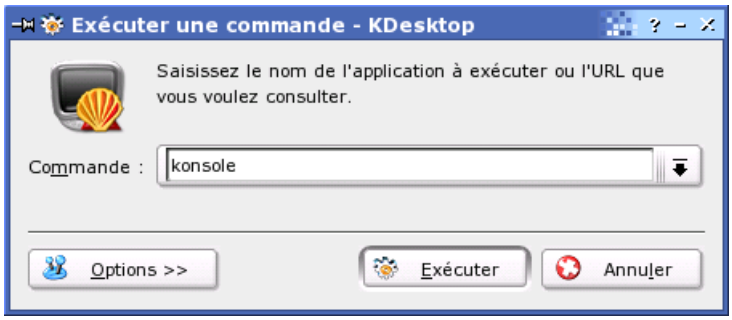
- sauvegarder le fichier en cours
- placer le nom du fichier en cours dans une variable spéciale
- appeler le programme make qui va utiliser le fichier `Makefile` et le contenu de la variable spéciale pour créer le programme exécutable.

Exécuter le programme

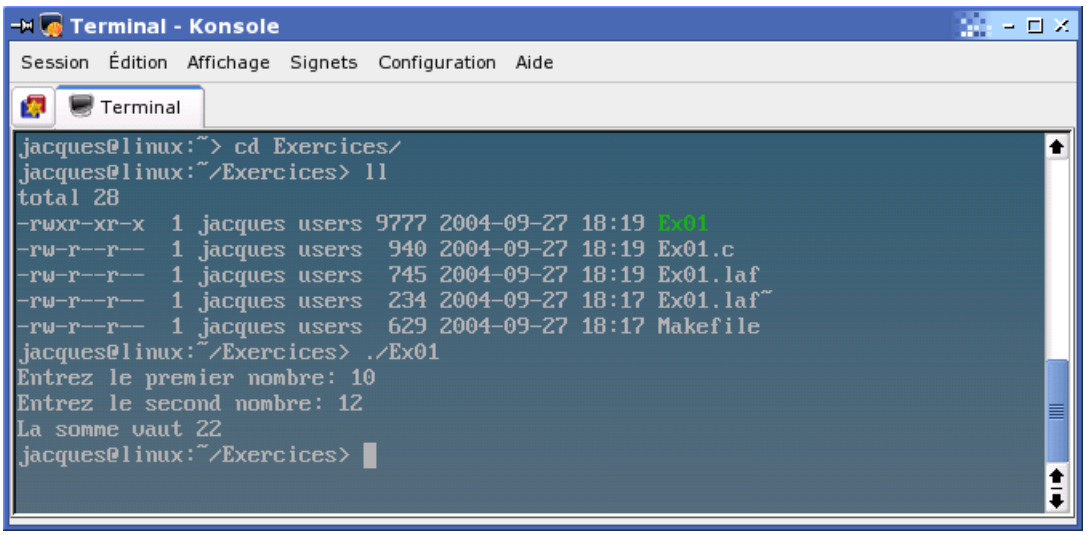
Si l'environnement le permet, la touche de compilation ouvre, en cas de réussite, une console dans laquelle le programme est automatiquement lancé. Petit problème, la console ouverte automatiquement est liée à l'exécution du programme. Lorsque ce dernier s'achève, la console se ferme, empêchant parfois de lire la dernière ligne affichée par le programme. On peut remédier à cet inconvénient en terminant systématiquement les programmes par la commande `Attente`.

Si l'ouverture automatique de la console n'est pas possible, il faut se résoudre à exécuter le programme manuellement. Il existe plusieurs manières d'exécuter le programme réalisé :

- ouvrir une console, se placer dans le dossier de travail et invoquer le programme par son nom (il n'a pas d'extension sous Linux, on peut ne pas écrire l'extension `exe` sous Windows). Sous Windows, la console s'obtient en lançant le programme `command.com` ou `cmd.exe`, ou en cliquant sur une icône proposant le mode MS-DOS. Sous Linux, la console s'appelle par l'icône représentant une coquille (*shell* en anglais) ou par l'invocation du programme *konsole* à l'aide de Alt f2 dans l'interface KDE et *terminal* dans l'interface Gnome.



– ouvrir un shell à l'intérieur d'Emacs par la commande Alt x `shell`[3]. Ce shell ne reconnaît pas les extensions ANSI et n'autorise donc aucune mise en page. Les effacements d'écran ou modifications d'aspect sont tout simplement ignorés.



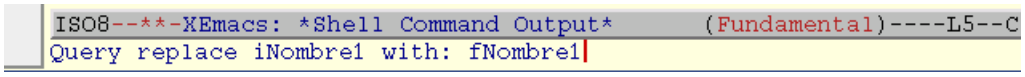
A.4 Fonctions avancées

Recherche

Emacs dispose d'une recherche incrémentale, ce qui signifie qu'on peut préciser au fur et à mesure le texte recherché. La combinaison ctrl s place le curseur dans le mini-buffer où on peut taper le texte à rechercher. La recherche commence dès la première lettre.

Remplacement

La touche f4 permet de taper dans le mini-buffer la chaîne à rechercher et la chaîne à lui substituer.



Pour chaque occurrence du texte, on peut soit :

– accepter le remplacement espace
– refuser le remplacement et passer à l'occurrence suivante del
– abandonner Echap
– accepter tous les prochains remplacements !

3. Sous les versions 95, 98 et Millenium, le lancement d'un shell dans Emacs entraîne un comportement erratique du PC, pouvant aller jusqu'au plantage. C'est dû au fait que Emacs tourne en 32 bits alors que le shell tourne en 16 bits.

Copie et effacement de texte

C'est probablement le comportement du programme qui surprendra le plus les habitués de Windows. Emacs ne dispose pas du presse-papier habituel, mais d'un *kill-ring*. Tout ce qui est placé dans cet « anneau de la mort » est mémorisé et peut être récupéré plus tard [4]. Comme sous Windows, on utilise un bloc pour marquer le texte à effacer ou à copier [5].

effacer le contenu de la sélection et la copier dans l'anneau (équivalent d'un *couper*)	ctr-w
copier simplement le contenu de la sélection dans l'anneau (équivalent d'un *copier*)	alt-w
effacer la fin de la ligne et en placer le contenu dans l'anneau,	ctr-k
effacer une ligne entière et en placer le contenu dans l'anneau	home ctr-k ctr-k
récupérer le dernier contenu placé dans l'anneau	ctr-y
employé répétitivement après *ctrl-y*, récupérer les contenus précédents placés dans l'anneau. Cette touche permet de se rendre compte de la réalité de l'anneau : en effet, si on a placé cinq contenus dans l'anneau, après avoir vu apparaître les cinq derniers contenus, on revient au premier et on part pour un nouveau tour. La touche f3 permet d'inverser le parcours.	alt-y

Annulation des dernières opérations

La touche f3 annule la dernière modification effectuée. Employée plusieurs fois successivement, elle permet d'annuler de nombreuses opérations (en fonction de la mémoire disponible).

Abandon d'une opération

Une opération commencée peut être abandonnée au moyen de la touche ctr-g. Dans certaines circonstances, l'éditeur semble se bloquer et afficher le message suivant :

```
Minibuffer already active: abort it with `^]', enable new one with `n':
```

Pour en sortir (et éviter de devoir taper *Ctrl-]*, difficile d'accès sur un clavier français), on tapera deux fois sur la touche pause.

4. En fonction des limites de la mémoire. En pratique, sur un petit programme, on peut considérer le *kill-ring* comme infini.
5. Les méthodes de marquage sont identiques :
   - à l'aide de la souris
   - à l'aide des touches de déplacement en maintenant la touche majuscule enfoncée
   - en marquant le début du bloc par ctrl-espace puis en utilisant les touches de déplacement.

Il en existe d'autres qu'il n'est pas utile de détailler ici.

Complétion automatique

C'est sans doute l'une des caractéristiques les plus utiles du programme. Si on tape les premières lettres d'un mot (nom de commande ou nom de variable), l'appui sur la touche f8 termine le mot en se basant sur les mots déjà tapés. Plusieurs appuis sur cette touche permettent de faire défiler les autres possibilités. Pour ce qui concerne les noms de fichiers et les macros, la touche magique de Unix est utilisable : tab.

Accès aux macro-commandes

Emacs dispose de plusieurs centaines de macros dont seulement un faible pourcentage dispose d'un raccourci-clavier. Pour accéder à ces macros, on utilise la combinaison alt-x. Ici encore la complétion automatique fait merveille.

Réorganisation de la disposition du programme

Lorsqu'on passe à la ligne suivante, le programme remet automatiquement la ligne dans l'alignement. Un simple appui sur la touche tab reformate la ligne courante (à ne pas faire à l'intérieur d'une paire de guillemets). Pour reformater tout le texte, il suffit d'employer la touche f7.

Portions de programmes

La touche ctrl-c offre dans chaque mode des fonctionnalités particulières. En mode LAF, j'ai prévu d'associer à ce préfixe une série de touches permettant d'écrire des embryons de structures. Ces embryons ne nous seront utiles qu'au fur et à mesure de l'avancement de notre cours :

- ctrl-c p : embryon de programme,
- ctrl-c s : embryon de structure alternative (`si..alors...sinon...fin de si`),
- ctrl-c i : embryon de boucle à test initial (`init...tant que.. répéter... moteur... fin de boucle`),
- ctrl-c f : embryon de boucle à test final (`init... répéter... moteur... boucler tant que...`),
- ctrl-c c : embryon de boucle avec compteur (`init... compter avec de à... fin de compter`)
- ctrl-c shift-P : embryon de code de procédure,
- ctrl-c shift-F : embryon de code de fonction.
- ctrl-c l : bloc de variables locales.

A.5 Touches de raccourcis

Je ne parle pas ici des touches de navigation habituelles (flèches et touches situées à droite du clavier)

Raccourci	Utilité	Touche « normale »
f1	Aide de l'éditeur (en anglais). Voir F10.	Ctrl-h
f2	Fermer le tampon en cours (avec éventuellement une proposition de sauvegarde).	Ctrl-x k enter
f3	Annuler la/les dernière(s) opération(s)	Ctrl-/
f4	Recherche et remplacement sélectif	Alt-%
f5	Sauvegarde du tampon en cours (m)	Ctrl-x Ctrl-s
Maj-f5	Sauvegarde du tampon sous un autre nom	-
f6	Ouverture ou création d'un fichier. Lecture dans un nouveau tampon.	Ctrl-x Ctrl-f
f7	Reformatage de tout le texte	-
f8	Compléter le mot en cours (plusieurs fois si la suite ne convient pas)	Alt-/
f9	Compilation du programme (m)	-
Maj-f9	Création du fichier Makefile dans un nouveau dossier (m).	-
f10	Affichage d'un résumé des commandes principales	-
f11	Retour au tampon précédent	Ctrl-b Enter
Maj-f11	Passer dans l'autre tampon de la fenêtre	Ctrl-x o
Ctrl-f11	Affichage de la liste des tampons	Ctrl-x Ctrl-b
f12	Un seul tampon d'édition visible	Ctrl-x 1
Ctrl-s	Recherche incrémentale	Ctrl-S
Ctrl-g	Abandon de la tâche en cours	Ctrl-G
Pause Pause	Abandon et déblocage	Ctrl-[
Tab	Formatage de la ligne en cours	Tab

Les fonctions marquées à l'aide de (m) ont été modifiée (notamment francisation des messages) dans la version que je distribue.

A.6 Erreurs possibles

Tous les exemples supposent que l'on compile un programme nommé `MonTest.laf`.

Mauvaise installation de `laf2c`

```
make: laf2c : Commande introuvable
```

Le programme `laf2c`, qui permet la transformation du fichier de pseudo-code en langage C n'est pas installé sur votre machine.

Absence du fichier `Makefile`

```
make: *** Pas de cibles spécifiées et aucun makefile n'a été trouvé. A
```

Le dossier de travail ne contient pas le fichier `Makefile` indispensable à la compilation du programme. Il faut le réinstaller. C'est ce qu'il faut faire en principe lorsqu'on utilise pour la

première fois un nouveau dossier. Il suffit de taper la combinaison Shift f9. En ligne de commande, on peut taper `laf2c -m`. On doit voir s'afficher le message suivant :

```
laf2c - Compilateur de pseudo-code LAF - Version 4.0 (00) - Linux
© 1999-2004 Jacques Thoorens - Programme sous licence GPL
Se référer au fichier gpl-license pour les conditions d'utilisation
Création du fichier Makefile
```

Absence du fichier `laf2c.h`

```
MonTest.c:5:19: laf2c.h: Aucun fichier ou répertoire de ce type
```

Le fichier `laf2c.h` est normalement contenu dans le dossier `/usr/local/include` dans lequel seul l'administrateur peut écrire. Pour corriger cette erreur, deux solutions sont possibles.

– créer à nouveau le fichier, en étant administrateur. Depuis une console, taper :

```
moi@athlon18:/> su
Password:
root@athlon18:/>laf2c -i
Compilateur de pseudo-code LAF - Version 3.9 (03) - Linux
© 1999-2003 Jacques Thoorens - Programme sous licence GPL
Se référer au fichier gpl-license pour les conditions d'utilisation
Création du fichier laf2c.h
root@athlon18:/>exit
```

– créer le fichier dans une zone privée d'un utilisateur normal. On suppose que l'utilisateur jactho2 est placé dans le dossier exercices.

```
jactho2@athlon18:exercices> laf2c --user -i
laf2c - Compilateur de pseudo-code LAF - Version 3.9 (03) - Linux
© 1999-2003 Jacques Thoorens - Programme sous licence GPL
Se référer au fichier gpl-license pour les conditions d'utilisation
Création du fichier laf2c.h
jactho2@athlon18:exercice> laf2c --user -m
laf2c - Compilateur de pseudo-code LAF - Version 3.9 (03) - Linux
© 1999-2003 Jacques Thoorens - Programme sous licence GPL
Se référer au fichier gpl-license pour les conditions d'utilisation
Création du fichier Makefile.
```

Cette procédure devra être répétée dans chaque nouveau dossier. On notera que l'option `--user` permet de travailler avec un fichier `laf2c.h` disposé dans le dossier de l'utilisateur. Cette technique est beaucoup plus lourde que la première.

Fichier .c existant

```
Un fichier d'origine inconnue MonTest.c existe déjà.
```

Ce message peut se produire dans deux circonstances :

1. on tente de compiler un programme `MonTest.laf` mais il existe déjà un fichier `MonTest.c`. Le compilateur est conçu de manière à ne pas effacer ce fichier, qui peut avoir de la valeur pour son propriétaire. Il faut renommer le fichier `.laf`, ou renommer ou détruire le fichier `.c`.
2. le fichier `MonTest.c` résulte d'une précédente compilation, mais sa première ligne a été modifiée. C'est elle que le compilateur teste pour vérifier que le fichier a été créé par lui-même.

Attention, si vous modifiez un fichier en C généré par `laf2c`, toutes les modifications seront perdues lors de la prochaine exécution du compilateur. Sauf si vous modifiez aussi la première ligne.

Compilation terminée

```
Compilation finished at Tue Aug 26 11:17:34
```

La date sera évidemment différente ! La compilation s'est bien terminée, aussi bien du pseudo-code vers le langage C que du langage C vers le langage machine. En principe, le programme va pouvoir s'exécuter.