Ada Instant Development Environment

Version 1.03
Édition du 17 août 2005

Stéphane Rivière

Cette distribution est dédiée à Dino Risi, à l'inénarrable Vittorio Gasmann et à Jean-Louis Trintignant pour un film météore, sublime, grave, infiniment léger et d'une absolue nostalgie : *Le Fanfaron*.

Les citations en tête de chapitre sont des extraits de *Murphy's Law and other reasons why things go wrong de A. Bloch*. Elles proviennent du site www.adalog.fr créé par Jean-Pierre Rosen.

Toute ma reconnaissance à l'équipe d'AdaCore et à leurs développeurs ; Émmanuel Briot, Joël Brobecker et Arnaud Charlet, sans qui cette distribution n'aurait pu voir le jour. Remerciements à Jeffrey Creem, amical *gourou* GCC et GNAT, à Earnie Boyd, le créateur de l'environnement MSys, plus généralement à tous les auteurs des outils GNU et plus particulièrement à tous les abonnés des listes de diffusion d'AdaCore et d'Ada-France. Merci à Fabrice Popineau, mainteneur de fpTEX, ainsi qu'à toute l'équipe du TEXLive.

Je tiens à remercier également : Enrique Bellido, François Bergeret, Ludovic Brenta, Martin Carlisle, Daniel Feneuille, Gautier de Montmollin, Pascal Obry, Thomas Quinot, Jean-Pierre Rosen, Samuel Tardieu et Luigi Zaffalon.

Ainsi que : Adalog, l'académie de l'US Air Force et l'École d'ingénieurs de Genève.

Ce manuel est destiné à AIDE, Ada Instant Development Environment, version 1.03

Contact :

Stéphane Rivière
Élevage de Rochebrune
La Morinière
17550 Dolus d'Oléron
France

email : stephane@rochebrune.org
http://stephane.rochebrune.org

mailing-list AIDE : aide@rochebrune.org

Table des matières

1 Introduction

On peut écrire très proprement dans n'importe quel langage, y compris C. On peut écrire salement dans n'importe quel langage, y compris Ada. Mais Ada est le seul langage où il soit plus fatigant d'écrire salement que proprement. Jean-Pierre Rosen

1.1 Préalables

L'auteur n'est pas un expert Ada, ni un utilisateur éclairé des outils libres ACT et GNU : n'hésitez pas à améliorer AIDE par vos suggestions.

Toute *aide à AIDE* est *adamantimement* souhaitée, en particulier dans les directions évoquées au paragraphe "Évolutions", situé en dernière page du manuel.

Voici tout d'abord quelques précisions sur le sens donné à certains termes utilisés dans ce document. Ces définitions sont proposées à titre indicatif ; elles ne prétendent pas être correctes d'un point de vue technique ou juridique :

- *Libre* désigne tout logiciel disponible en sources et utilisable sans restriction dans un contexte commercial, selon une licence de type LGPL, BSD, Mozilla ou apparentée ;
- *Windows* désigne les système d'exploitation Microsoft à noyau NT, tels NT4, W2K ou XP, à l'exclusion des systèmes fondés sur MS-DOS, de type Windows 95, 98 et Me ;
- *Unix* désigne les systèmes d'exploitation Linux, FreeBSD, OpenBSD, NetBSD mais aussi tous les Unix propriétaires ou apparentés, connus pour être supportés par les outils GNU ;
- *Multi-plateforme* est la faculté de créer un logiciel sur et pour différents systèmes d'exploitation (Windows, Unix) et différentes architectures matérielles (x86, power pc, alpha, etc.) ;
- Langage *à la mode* désigne les langages dont les qualités propres ne peuvent expliquer rationnellement leur succès, tels C++ ou Java ;
- Langage *propriétaire* désigne des langages appartenant à une société, tels Visual Basic ou C# pour Microsoft ou Delphi pour Borland.

Dénomination des principaux outils de AIDE :

- *Glide* est l'IDE classique de AIDE. Fondé sur l'éditeur EMACS, Glide comporte un mode Ada évolué et une gestion de projet intégrée, ainsi que le débuggeur graphique *GVD* ou *Gnu Visual Debugger*.
- *GPS* ou *Gnat Programming System*, est un IDE graphique de nouvelle génération et majoritairement écrit en Ada.
- *Console MSys* est le shell Unix disponible avec AIDE, pour lancer les programmes utilisables en mode commande, associé à l'éditeur mode texte *FTE*.

1.2 Syntaxe conventionnelle

Dans une ligne de commande :

- Un paramètre entre crochets [ ] est optionnel ;
- Deux paramètres séparés par | sont mutuellement exclusifs.

1.3 Présentation

Ce document décrit AIDE, Ada Instant Development Environment, un environnement intégré de développement Ada pour Windows utilisant uniquement des outils libres et multi-plateformes et permettant de développer des applications également multi-plateformes.

AIDE est unique par sa licence, son intégration, son installation, son absence d'impact sur le système et le choix des environnements de développement :

- Tous les outils nécessaires sont libres, intégrés et déjà configurés ;
- AIDE est immédiatement utilisable après l'installation ;
- Aucune entrée n'est créée dans la registry ;
- Aucun fichier n'est copié dans un quelconque répertoire système ;
- Aucune variable système n'est créée ou modifiée à l'extérieur de AIDE ;
- AIDE permet de choisir entre trois modes de développements : un mode IDE généraliste fondé sur Emacs (Glide), un mode IDE graphique spécialisé Ada (GPS) ou même une simple ligne de commande (Bash). Pour ce dernier choix, un éditeur très simple est proposé (Fte).

Les principaux composants utilisés sont :

- GNAT 3.15p, GtkAda 2.2.0, GPS 2.1.0, GNU Emacs 21.1.1 avec Glide 3.15p, MSys 1.0.8 ;
- UPX 1.22 et UCL 1.01 ;
- GS 7.05, a2ps 4.13, PSutils 1.17 et File 4.03.

Les versions de ces logiciels ne sont pas nécessairement, et c'est volontaire, les dernières disponibles, mais l'ensemble forme un tout cohérent, fiable et, autant que faire se peut, testé.

1.4 Démarche

Se prémunir des aléas de l'avenir

AIDE est né de la lassitude d'être asservi à des langages *propriétaires* tout en réaffirmant que GNU n'est pas Unix et qu'il devait être possible de montrer aux développeurs Windows qu'une solution commune Windows *et* Unix pouvait exister, sans devoir leur imposer de quitter leur système habituel.

L'important est de travailler avec l'environnement le plus approprié. Et, pour beaucoup d'entre nous, et certainement aussi, malgré nous, Unix n'est pas encore, ou ne sera jamais, l'environnement le plus facilement disponible ou le plus adapté. Les développeurs *hardware*, par exemple, doivent utiliser des outils logiciels propriétaires, incontournables, et seulement disponibles sous Windows.

Enfin, si choisir son environnement de développement peut être le résultat d'une démarche personnelle, c'est quand même le marché qui détermine l'environnement d'exécution. Aussi la portabilité des outils de développement et leur faculté à produire un code réellement portable est une réponse pragmatique aux incertitudes de l'avenir et à la diversité des possibles.

Une voie vers plus de liberté

La solution proposée pérennise les acquis, en réaction à la versatilité, l'inconsistance ou la stratégie des éditeurs commerciaux, qui imposent aux développeurs de nouveaux langages, environnements, paramètrages et autres interfaces de programmation. Cette quête permanente de *l'outil ultime* se traduit surtout dans les faits par la recherche mille fois répétée de la bonne

configuration, de la bonne syntaxe ou du bon paramètre. Et tout ceci n'est que de la cuisine sans intérêt. L'important est ailleurs.

L'idée est donc de proposer au développeur Windows l'environnement le plus proche possible de ce que l'on retrouve dans les systèmes *Libres*. Cette démarche permettra aussi à des novices du monde Unix d'être confrontés en douceur aux remarquables outils de ces environnements.

De ce fait, le développeur Windows pourra, s'il le souhaite, passer ensuite indifférement d'un système à un autre sans être dépaysé et en conservant :

- Tous ses outils habituels ;
- Toute la base logicielle déjà écrite.

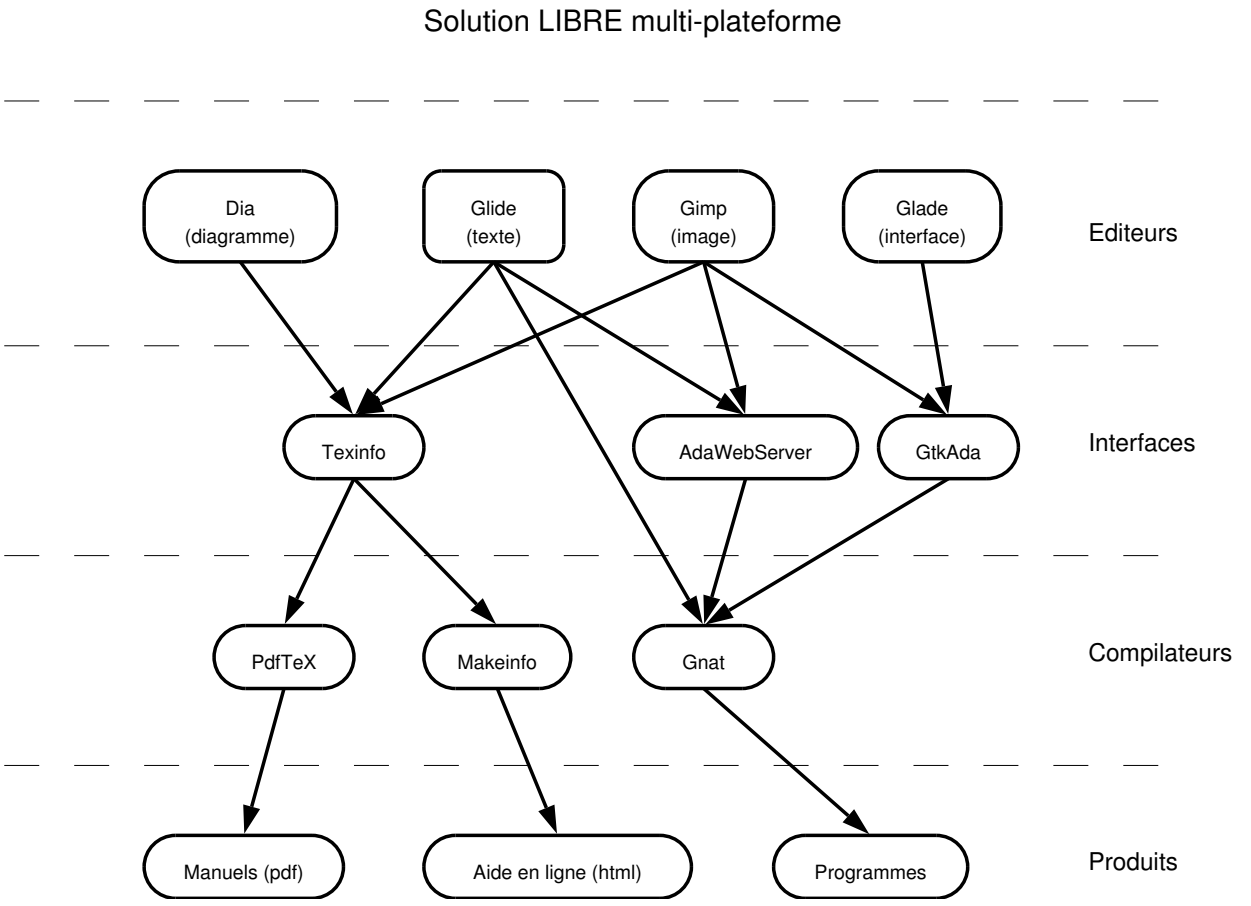
Le dernier atout de cette démarche est, en apparence, moins évident mais il est aussi essentiel. Il s'agit de la *qualité* et de la *puissance* des outils Libres. Quand on y a goûté, il est *vraiment* difficile de s'en passer.

1.5 Solution *Libre* multi-plateforme

Les composants sélectionnés, pour leur aptitude à produire des applications fiables et pérennes, forment une solution multi-plateforme utilisable sur les systèmes Windows et Unix.

Ils ne sont pas sous licence GPL pure, mais sous des licences dérivées, moins contraignantes, qui autorisent la création d'application commerciales vendues sans les sources[1].

Un schéma étant plus parlant qu'un texte, voici un aperçu de ce que permet AIDE, quand Glide est utilisé en association avec les logiciels de documentation proposés dans ce manuel :



[1] C'est pour cette raison que des solutions comme Qt ou MySQL n'auront jamais leur place dans AIDE : il n'est pas agréable de devoir choisir ses outils en fonction de la licence choisie pour la diffusion du logiciel. La véritable liberté est absolue et n'est pas assujettie à une politique, si louable soit-elle.

On distingue quatre couches :

- Les *éditeurs* sont représentés en haut du diagramme :
  - Dia est un éditeur de diagrammes utilisé pour la production des manuels. Ce dessin est un exemple de ses possiblités ;
  - Glide est un éditeur multi-langages qui permet, dans le cas qui nous intéresse, de travailler sur des sources Ada, Texinfo et HTML ;
  - Gimp est un logiciel de traitement d'images utilisé pour la production des manuels, la création de pages Web et d'interfaces graphiques ;
  - Glade est un éditeur d'interface graphique GTK.
- Les *interfaces* sont représentées juste en dessous :
  - Texinfo est utilisé pour la production de la documentation, une description détaillée est proposée un peu plus loin dans le manuel ;
  - AdaWebServer permet de créer une interface utilisateur de type Web, par essence portable et utilisable à distance ;
  - GtkAda autorise la création d'une interface utilisateur graphique, liée à l'application, et donc utilisable principalement en local.
- Les *compilateurs* sont au nombre de trois :
  - PdfTEX et Makeinfo sont utilisés pour la production de la documentation ;
  - GNAT est une suite de compilation Ada complète. Fondé sur GCC, *Gnu Compiler Collection*, GNAT intègre aussi un compilateur C et tous les outils connexes, tels un débuggeur graphique ou un analyseur de performance.

1.5.1 Langage

Présentation de Ada

Ce langage n'est pas encore suffisament connu, au moins pour la grande majorité d'entre nous, au détriment de beaucoup d'utilisateurs potentiels d'ailleurs. Par rapport aux langages *à la mode*, Ada, puisque c'est son nom, est plus portable, plus lisible, autorise des niveaux d'abstraction plus élevés et possède des fonctionnalités inconnues ailleurs. Ada autorise aussi une programmation système plus agréable[2] et se révèle suffisament léger pour être utilisé sur des micro-contrôleurs 8 bits bas de gamme[3].

Ada est le nom du premier programmeur de l'humanité. Et le premier programmeur était une programmeuse : Augusta Ada Byron King, Countess of Lovelace. Née en 1815, fille du grand poète Byron, assistante de Charles Babbage, elle écrivit les programmes destinés à sa célèbre machine.

Ada 95 est une norme militaire américaine[4] et une norme civile internationale[5]. C'est le premier langage orienté objet a avoir été normalisé au niveau international. Tous les compilateurs Ada doivent répondre strictement à cette norme. Il en existe des centaines différents, pour autant de plate-formes différentes, mais tous produisent un code dont le comportement est identique.

---

[2] Grâce, par exemple, à ses clauses de représentation qui suppriment la nécesité d'employer des masques d'octets *XORés* pour la manipulation de bits. Cette fonctionnalité *fondamentale en programmation système* est absente en C ou en assembleur.

[3] Des composants qui ne disposent que de quelques dizaines d'octets de mémoire vive et de quelques kilo-octets de mémoire programme.

[4] MIL-STD-1815B

[5] ISO/IEC 8652:1995

Ada est utilisé partout où la sécurité prime : Airbus (famille civile A3xx et A400 militaire), Alsthom (TGV), Boeing (777, 7E7) , EADS (Eurofighter, Ariane, ATV), STS (Météor ligne 14), NASA (alimentation électrique de la station spatiale internationale). La liste est innombrable. Partout où la sûreté est primordiale, Ada est le langage choisi. Ce lien, entre autres, vous donnera quelques exemples d'utilisation du langage Ada : http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html.

Les effets de la mode

Faisons une pause champêtre quelques instants.

Imaginons une automobile bloquée derrière une *voiture sans permis*, de cette sorte de mini-voiture ayant souvent l'apparence d'un escargot en colère et mené, la plupart du temps, par un pilote hors d'âge. C'est une situation banale dans nos campagnes.

Il est par contre moins crédible d'imaginer le conducteur de la coquille à roulette se gausser de l'automobile qui le suit - de bien près, il est vrai - avec des arguments tels que : *"Ah, ces voitures modernes ! trop solides, trop fiables, trop confortables, trop sûres !"*

C'est pourtant à ce niveau que se situent les critiques habituelles sur Ada, exprimées par des développeurs qui devraient certainement s'asseoir un moment au bord de la route, et songer alors à leurs derniers bugs ordinaires, habituels et quotidiens de leur *langage sans permis*.

Car, au bout du compte, il doit bien y avoir autre chose dans la vie d'un programmeur qu'un long chemin grossièrement pavé de dépassement de tampon, de ces pointeurs en folie qui vont et viennent au gré des saisons et d'anomalies structurellement causées par l'utilisation de langages basés sur des concepts *hors d'âge*.

Y a t'il, autour de nous, des développeurs qui monteraient dans un avion dont les commandes de vol seraient écrites dans un langage *à la mode* ?

Un développeur ne devrait-il pas se montrer aussi exigeant, tant pour sa productivité que pour son bien-être, avec son langage informatique habituel ?

Pourquoi utiliser Ada

Ada a été créé parce que le développement est une activité humaine. Les humains font des erreurs. Le compilateur Ada est l'ami des développeurs. Ada est aussi l'ami des chefs de projet pour les développements à grande échelle. Une application Ada s'écrit, s'étend et se maintient naturellement. Pour ces qualités, la solution Ada est aussi l'amie des gestionnaires.

Ada est le langage des développeurs, des gestionnaires et des utilisateurs heureux.

Parce qu'Ada est un langage agréable par son expressivité et reposant par sa fiabilité, les humains impliqués dans Ada sont aussi à l'image de leur langage. La communauté Ada est une communauté agréable à fréquenter et les rencontres y sont très enrichissantes. Les bibliothèques libres disponibles en Ada sont nombreuses et de grande qualité. Enfin, la communauté Ada francophone est très active.

Pour vous faire rapidement une idée concrète sur Ada, il est conseillé de lire le cours de Daniel Feneuille, dans le répertoire '`c:\aide\doc\courses (fr)\cours Ada Daniel Feneuille`'.

Si vous êtes déjà un développeur confirmé, vous consulterez également avec profit '`c:\aide\doc\faqs, books and tutorials\ada distilled\ada distilled-v2.pdf`'.

Le mot de la fin

Quand Boeing décida, il y a une décennie, que les logiciels du 777[6] seraient exclusivement écrits en Ada, les sociétés partenaires du constructeurs firent remarquer qu'elles utilisaient depuis

[6] Le Boeing 777 est plus gros bimoteur du monde et le premier Boeing civil à commandes de vol électriques, dix ans après l'A320 d'Airbus.

longtemps des langages tels que le C, le C++ ou divers assembleurs et qu'elles en étaient parfaitement satisfaites.

Boeing répondit simplement que seules les sociétés aptes à fournir des logiciels en Ada seraient retenues dans les appels d'offres. Alors les sociétés se convertirent à Ada.

Aujourd'hui, le développement des logiciels du Boeing 777, surnommé *l'Ada plane*, a été effectué et c'est essentiellement grâce au très grand succès commercial de cet avion que Boeing, désormais numéro deux mondial depuis trois ans derrière Airbus, a pu maintenir la rentabilité de son activité civile.

Et que font désormais les sociétés partenaires de Boeing ? Elles continuent à développer leurs nouveaux logiciels en... Ada, et voici pourquoi :

- Elles se sont aperçu que la durée de conversion des développeurs est très courte. En une semaine, le développeur est déjà suffisament à l'aise pour écrire des programmes en Ada et en moins d'un mois, il se sent totalement à l'aise ;
- Ces sociétés ont fait leur comptes : écrits en Ada, les logiciels coûtent moins cher, ils présentent moins d'anomalies, sont prêts plus tôt et sont plus faciles à maintenir.

Pour le futur proche, ces constatations se vérifient encore chez Boeing : le langage principal du 787 "Dreamliner", son nouveau bimoteur, est à nouveau Ada.

1.5.2 Interfaces utilisateur

Beaucoup de développements nécessitent une interface utilisateur. La bibliothèque d'exécution standard propose les briques de base pour une interface utilisateur en mode texte. Pour une interface utilisateur graphique, deux solutions complémentaires sont proposées :

- GtkAda quand une interface locale de haut niveau est nécessaire ;
- AWS (Ada Web Server), quand une interface WEB est préférable.

GtkAda

GtkAda s'appuie sur Gtk (Graphic Tool Kit), la bibliothèque graphique originellement conçue pour GIMP (Gtk Image Manipulation Program, le logiciel de dessin et de retouche d'images bien connu). Gtk a été depuis considérablement développé pour être l'interface graphique de GNOME (une des deux majeures interfaces graphiques de Linux avec KDE).

Gtk n'est pas écrit en C++ mais en "C ANSI orienté objet". C'est à dire du C ANSI (non C++) utilisé dans une conception logicielle orientée objet. Cette approche originale garantit une excellente portabilité et une bonne fiabilité. La notion de conception logicielle orientée objet est indépendante du langage utilisé. On peut programmer objet en assembleur par exemple.

AWS (Ada Web Server)

AWS permet d'intégrer un ou plusieurs serveurs WEB à l'intérieur d'une application Ada. Les protocoles `http` et `https` sont implémentés, ainsi qu'un client `http` et `smtp`.

L'intérêt d'AWS est évident à l'heure du WEB, beaucoup d'applications n'ont pas besoin d'une interface graphique évoluée ou, dans d'autres cas, ont besoin de pouvoir être accédées à partir d'un quelconque navigateur internet.

1.5.3 Bibliothèques spécialisées

Ada se distingue aussi d'autres langages par l'exceptionnelle qualité et la grande diversité du code librement disponible sur internet.

Un petit aperçu est proposé dans une archive annexe dénommée : '`aide-src-1.03.exe`'. Dans ce *digest* d'environ 90 Mo compressés, on trouvera des sources :

- D'ouvrages traitant de Ada ;

- Pour les microcontrôleurs AVR de Atmel et 68HC11/12 de Motorola ;
- De noyaux exécutifs ou de systèmes d'exploitations (Edu-os, Marte, Xada Rtems et Tiny Lovelace) ;
- De plus d'une vingtaine d'applications utilisateurs dans des domaines applicatifs et techniques variés ;
- De plus d'une centaine de packages pour développeurs dans les domaines de l'automatique, de la compression, du chiffrement, des applications distribuées, des bases de données, des structures de données, du déverminage et du test, du traitement du signal, de l'internet, du multimédia, du multi-tâche, sans compter les nombreux packages systèmes nativement portables, portables via posix, spécifiques Windows, Unix, Vms, etc ;
- De plus d'une vingtaine d'outils pour faciliter la liaison avec d'autres langages, de générateurs de documentations, d'éditeurs ou de tests.

1.5.4 Utilitaires

Navigateur de sources

`Alibrowse` est un navigateur de sources dédié à Ada. Il est très simple d'emploi et ne créée qu'un seul fichier, de taille modeste, par répertoire.

`Alibrowse` permet de lire très rapidement les spécifications, donc l'emploi, des bibliothèques de fonctions (appelées paquetages en Ada) disponibles avec AIDE.

Utilisé dans le cadre d'un développement, `Alibrowse` permet de balayer directement les sources, sans passer par une documentation papier ou `HTML` *peut-être* obsolète.

Voyons, par exemple, comment naviguer dans la librairie de GNAT, située dans '`/lib/gnat`'. À partir de la console MSys, lancer une commande '`aliscan /lib/gnat`'. Un fichier d'indexation '`toc.alb`' est créé dans le répertoire '`/lib/gnat`'.

Pour naviguer dans la librairie de GNAT, il suffit de taper, toujours à partir de la console MSys, '`alibrowse /lib/gnat/toc.alb`'. On peut aussi créer un raccourci sur le bureau pour les librairies les plus souvent examinées.

Dans AIDE, les répertoires '`/lib/gnat`', '`/lib/gtkada`' et '`/lib/win32ada`' sont déjà indexés.

1.5.5 Production de la documentation

AIDE est une solution de développement complète. La documentation fait partie du logiciel. AIDE intègre donc des outils documentaires puissants, cohérents et parfaitement intégrés entre eux.

Le chapitre 6 est entièrement dédié à cet aspect du développement.

2 Ada Instant Development Environment

Les investissements en fiabilité des programmes en C augmenteront jusqu'à dépasser le coût probable des erreurs ou jusqu'à ce que quelqu'un insiste pour tout recoder en Ada. Synthèse des lois de Gilb

2.1 Généralités

2.1.1 Historique

Début 2002, AIDE 0.50 proposait un environnement de développement Ada pour Windows NT avec GtkAda 1.3, immédiatement opérationnel après une simple copie. AIDE 0.50 comportait un environnement fondé soit sur Adagide, soit sur Emacs/Glide. Cette première version posait deux problèmes : un environnement de développement hybride, soit minimaliste et non portable (Adagide) soit puissant mais plus complexe (Glide) et un framework graphique basé sur Gtk 1.3, au *look & feel* Unix, déroutant pour un utilisateur Windows.

Début 2003, la distribution expérimentale AIDE 0.74 permettait, sous windows NT, et avec l'aide de GnatGcc 3.2, GtkAda 2.2 et MSys 1.0.8, d'obtenir un environnement Ada complet, similaire à celui rencontré sur Unix, mais permettant de créer des applications graphiques sous Windows avec GtkAda 2.2. Cet environnement pouvait aussi utiliser Glide.

Début 2004, la distribution AIDE 1.2 a représenté une synthèse des deux démarches précédentes, avec cette fois des outils mieux intégrés et toujours avec GtkAda 2.2, afin d'obtenir un *look & feel* irréprochable sous Windows.

Cette nouvelle solution utilise la version GNAT d'ACT, qui est un compilateur plus sûr et plus complet aujourd'hui que le GnatGcc 3.2 de AIDE 0.74, l'environnement MSys, qui est un environnement Unix sous Windows et GPS, en remplacement de l'IDE non portable Adagide et toujours Glide, pour les développeurs préférant un IDE classique, sans compter de nombreux autres nouveaux outils connexes.

Courant 2005, AIDE 1.3, dernière version de la série 1.x, comporte un programme d'installation, intègre GPS 2.1, un driver Texinfo pour Adadoc et de nombreuses autres améliorations (se référer à la fin du manuel pour le détail des ajoûts).

2.1.2 Organisation de la distribution

La distribution est composée de deux paquets installables multi-volumes.

Le premier paquet, le seul indispensable, est '`aide-1.03`'. Il est composé d'un fichier exécutable '`aide-1.03.exe`' et de plusieurs volumes '`aide-1.03-n.bin`'.

Le second paquet '`aide-src-1.03`' est la collection de sources d'AIDE. Il ne prétends pas être à jour mais plutôt a être une source d'inspiration et d'incitation à des recherches plus approfondies sur internet. Il comprends également plusieurs volumes '`aide-src-1.03-n.bin`'.

Ce dépôt contient une énorme quantité d'applications et de bibliothèques écrites en Ada, toutes en sources. On y trouve même cinq systèmes d'exploitation ou des sources pour micro-contrôleurs 8 bits. C'est un *must-have* et une source d'inspiration permanente.

Le tableau ci-dessous donne le volume indicatif de chaque paquet.

'`aide-1.03`'	115 Mo	AIDE 1.03 complet.
'`aide-src-1.03`'	90 Mo	Sources (applications, packages, OS, etc...).

2.1.3 Installation et désinstallation de AIDE

Installation

Pour installer AIDE, lancez '`aide-1.03.exe`'. Au terme de l'installation dans '`c:\aide`', trois icônes sont placées sur le bureau.

L'installation de AIDE se contente de placer ses fichiers dans '`c:\aide`' sans ajoûts dans d'autres répertoires (hormis les trois icônes et les entrées dans le menu démarrer) ni dans la *registry*.

Un chemin différend de '`c:\aide`' peut être défini en modifiant un paramètre dans '`c:\aide\aide.cmd`'. Veuillez vous référer aux questions courantes, un peu plus loin dans ce chapitre.

Le fichier '`c:\aide\aide.cmd`' permet aussi de lancer directement l'un des trois modes de développement de AIDE. En appelant '`c:\aide\aide.cmd`' sans paramètres, vous obtenez l'usage de la procédure :

```
Usage : aide [glide [file]] or [gps [file]] or [msys]

Examples : aide glide [file] : run Glide IDE with [file]
           aide gps [file]   : run GPS IDE with [file]
           aide msys         : run Msys console

You can also use predefined glide.lnk, gps.lnk and msys.lnk
```

Limitations connues :

- Le nom d'utilisateur de votre session Windows ne doit pas comprendre de caractère accentué.
- Si vous modifiez le chemin de base de AIDE, par défaut '`c:\aide`', le nouveau chemin ne doit pas comporter d'espace ou de caractère accentué.

Désinstallation

Pour désinstaller AIDE, lancez le programme de désinstallation placé dans '`c:\aide`', programme également accessible dans le menu *Démarrer / Programmes / Aide*. La supression est instantanée, mais les fichiers personels créés dans '`c:\aide\home`' sont laissés intacts.

2.1.4 Prise en main

Lancer Aide

Cliquez sur l'une des trois icônes, tout est déjà configuré et prêt à l'emploi : Il n'y a plus qu'à développer !

Icône de Glide

Icône de GPS

Icône de la console Msys

Suivez les conseils des chapitres 3, 4 & 5 pour tester les environnements proposés, afin de choisir celui qui vous convient le mieux.

Multi-utilisateurs

Plusieurs utilisateurs différents peuvent utiliser AIDE. Chacun trouvera son répertoire personnel, automatiquement créé à la première utilisation, derrière le répertoire '`c:\aide\home`'.

Le prompt du shell est de la forme : `[utilisateur] répertoire_courant >`

Impression dans AIDE

L'impression est effectuée grâce à a2ps et ghostscript. Il y a moyen de faire autrement (voir la documentation de GPS, par exemple), mais on est alors dans une logique propriétaire ; non libre et non portable.

La solution proposée ici est plus performante, portable et fonctionne à l'identique sous Windows et sous Unix. Elle fait appel aux outils libres a2ps et ghostscript.

Elle ne nécessite aucune configuration, intègre un pretty-printing avec reconnaissance du type de fichier et lance automatique le sélecteur d'imprimante Windows.

Ces outils permettent aussi au développeur de disposer d'une couche d'impression portable à bon compte, pourvu qu'il n'ait pas besoin d'une mise en page trop évoluée (essentiellement mode texte avec tags suivant une feuille de style choisie).

Cette couche d'impression est suffisante pour éditer des états texte avec les attributs de caractères courants. Elle autorise aussi une sortie au format .PS ou .PDF, ce qui peut se révéler utile.

Validation sommaire de l'installation

Vous pouvez tester rapidement votre installation, aussi bien en mode texte qu'en mode graphique, par la procédure ci-dessous :

- Lancez une console Msys par la commande '`c:\aide\msys.cmd`' ⟨RET⟩ ;
- En vous plaçant dans le répertoire '`/examples/gnat`' par la commande '`cd /examples/gnat`' ⟨RET⟩, lancez la compilation des programmes en mode texte par la commande '`make`' ⟨RET⟩. Les programmes se lancent juste après leur compilation ;
- En vous plaçant dans le répertoire '`/examples/gtkada/testgtk`' par la commande '`cd /examples/gtkada/testgtk`' ⟨RET⟩, lancez la compilation du test complet GtkAda par la commande '`maketest`' ⟨RET⟩. Lancez ensuite le programme par la commande '`testgtk`' ⟨RET⟩ et découvrez la richesse de l'environnement graphique Gtk.

2.2 Structure de la distribution

La structure de la distribution consiste en une arborescence relativement consistante, modulaire et peu profonde, dans la limite des contraintes techniques rencontrées avec les différents composants de la distribution, entre autres, des problèmes de conversion de chemins posix.

Le répertoire /bin et ses descendants ne doivent contenir que des exécutables pouvant utiliser des chemins posix.

Toutes les remarques sont les bienvenues.

```
* = Racine de la distribution, 'c:\aide' par défaut.

*---bin       binaires posix : glide, gps, msys
|
+---binw      binaires natifs : a2ps, ghostscript, ghostview, gnat
|
+---bina      binaires d'archivage : afio, afio_find, afio-cat, etc...
|
+---doc       documentation
|   |
|   +---ada 95                          Documents de référence Ada
|   +---Ada and others languages Ada et C#, C++, Java, pascal
|   +---Ada stuff Bio de Ada Byron, images et logos Ada
|   +---adadoc                          Générateur de documentation Adadoc
|   +---aide
|   |   +---manuals Manuel d'AIDE en français et anglais (partiel)
|   |   \---speechs Intervention aux RMML 2004 à Bordeaux.
|   +---alibrowse "Browser" de sources
|   +---cdrecord
|   +---courses (fr)                    Cours en français
|   |   +---cours Ada Daniel Feneuille
|   |   +---cours IUT
|   |   \---université curie
|   +---faqs, books and tutorials       Faqs, livres et tutauriaux en anglais
|   |   +---ada craft
|   |   +---ada distilled
|   |   +---ada in action
|   |   +---ada library reference
|   |   +---ada lovelace
|   |   +---big online book of ada programming
|   |   +---faqs
|   |   \---richard conn
|   +---faqs, books and tutorials (fr)  Faqs, livres et tutauriaux en français
|   |   +---ada et C++ thread
|   |   \---fr_comp_lang_ada
|   +---fte
|   +---general information             Information générale en anglais
|   +---general information (fr)        Information générale en français
|   +---glide
|   +---gnat
|   +---gps
|   +---gs                              Documentation Ghostscript
|   +---gsview
|   +---gtk
|   +---gtkada
|   +---gvd                             Documentation débugger pour Glide
|   +---ispell                          Correcteur orhtographique pour Glide
|   +---msys
|   +---opengl                          Documentation OpenGL
|   +---real world                      Ada ''dans le monde réel'' en anglais
|   +---real world (fr)                 Ada ''dans le monde réel'' en français
|   \---texinfo                         Documentation Texinfo
|
+---etc       configuration
```

```
|   |
|   +---e            emacs
|   +---emacs        emacs : stuff suppressed from original emacs etc directory
|   +---fte          fte
|   +---icons        emacs
|   +---pango        gtk
|   +---tutor        emacs : gvd tutorial
|   +---gtk-2.0      gtk
|   \---a2ps         a2ps
|       doc-x        emacs
|       gs           gs
|       news         emacs
|       problems     emacs
|       profile      msys
|       tutorial     emacs
|       fstab.smaple msys
|       config.site  msys
|       rgb.txt      emacs
|
+---examples  exemples et tutoriaux
|   |
|   +---ada_gate
|   +---adadoc
|   +---aflex-ayacc
|   +---aide
|   +---gnat
|   +---gps
|   +---gs
|   +---gtkAda
|   +---opengl
|   \---win32Ada
|
+---home      racine des répertoires utilisateurs
|
+---info      emacs : emplacement des fichiers infos
|
+---lib       bibliothèques gnat, gtkada et win32ada
|   |
|   +---aide
|   +---gnat
|   +---gs
|   +---gtkada
|   +---include
|   +---opengl
|   +---gtk-2.0
|   +---gtkada
|   +---xmlada
|   \---win32ada
|
+---lisp      emacs : emplacement des fichiers lisp
|
+---share     fichiers partagés
|   |
|   +---a2ps
|   +---aide
|   +---file
|   +---gps
|   +---ispell
|   +---make
|   +---texinfo
|   \---themes
|
\---tmp
```

Notes :

- Les exécutables GPS sont : '`cbrowser.exe`', '`dbdump.exe`', '`dbimp.exe`', '`gnudiff.exe`', '`gprmake.exe`', '`gps.exe`' et '`winps.exe`' ;
- Les exécutables Glide sont : '`fns-21.1.1.el`', '`addpm.exe`', '`cmdproxy.exe`', '`ctags.exe`', '`ddeclient.exe`' '`ebrowse.exe`', '`emacs.exe`', '`getfile.exe`', '`glideint.exe`', '`hex1.exe`', '`movemail.exe`' et '`runemacs.exe`' ;
- Les sources lisp de Glide, adaptées à Aide, sont : '`ada-aunit.el`', '`ada-mode.el`', '`ada-prj.el`', '`ada-stmt.el`', '`ada-support.el`' '`ada-vms.el`', '`ada-xref.el`', '`cua-mode.el`', '`glide.el`', '`glide-debug.el`', '`glide-externel.el`', '`subdirs.el`' et '`glide_version.txt`'. Le fichier '`dlgopen.el`' a été rajouté pour rétablir une fonctionnalité de compatibilité supprimée dans la nouvelle version de Glide.
- Les exécutables de la chaîne d'impressions sont '`gs.exe`', '`a2ps.exe`' et '`file.exe`'.
- Les exécutables de la chaîne de sauvegarde AFIO sont '`afio.exe`', '`afio_find.exe`', '`afio_split.exe`', '`afio_cat.exe`', '`afio_gzip.exe`', '`cygwin1.dll`', '`cygiconv-2.dll`', '`cygintl-3.dll`'.
- Les exécutables de la chaîne de gravure de CD-R et CD-RW sont '`cdrecord.exe`', '`mkisofs.exe`', '`devdump.exe`', '`isodump.exe`', '`isovfy.exe`', '`madplay.exe`', '`readcd.exe`', '`oggdec.exe`', '`cdda2wav`', '`md5sum.exe`' et '`cygwin1.dll`'.
- Les utilisateurs sont créés au lancement de AIDE par la Console MSys. Leur répertoire de travail est situé derrière le répertoire '`home`'. Le nom utilisé pour le répertoire de travail correspond au nom de l'utilisateur.

2.3 Questions courantes

2.3.1 Généralités

Changer le répertoire par défaut de AIDE

Question : Le répertoire par défaut '`c:\aide`' ne me convient pas.

Réponse : Le répertoire racine '`c:\aide`' peut être modifié dans '`c:\aide\aide.cmd`', grâce au paramètre `ADA_ROOT`.

Si vous changez autre chose que l'unité disque, il vous faudra aussi modifier, dans le fichier '`c:\aide\etc\a2ps`', le paramètre `LibraryPath` qui contient par défaut le chemin `/aide/share/a2ps`.

Le nouveau chemin ne doit pas comporter d'espace ou de caractère accentué.

Notez alors que le programme de désinstallation ne fonctionnera plus, mais la suppression d'AIDE se limite à la suppression du répertoire où est placé AIDE , des icônes du bureau et des entrées dans le menu de démarrage.

Compilation avec certains répertoires

Question : J'ai des problèmes de compilation dans certains répertoires (mais pas tous) ?

Réponse : Vérifiez que le chemin de votre projet ne comporte pas d'espaces dans les noms (si c'est le cas, remplacez les espaces par des soulignés `"_"`)

Problème avec make

Question : J'ai créé mon premier fichier makefile, mais ça ne marche pas.

Réponse : Dans les règles de dépendances, il faut utiliser ⟨TAB⟩ (la touche de tabulation).

Exemple : les trois lignes saisies telles que ci-dessous...

```
hello.exe:⟨RET⟩
⟨TAB⟩gnatmake hello.adb⟨RET⟩
⟨TAB⟩hello.exe⟨RET⟩
```

...apparaîtront ainsi...

```
hello.exe:
      gnatmake hello.adb
      hello.exe
```

...et tout rentrera dans l'ordre.

2.3.2 GNAT

Compréhension des messages d'erreur

Question : Je ne comprends pas le message d'erreur du compilateur...

Réponse : Utilisez l'option -gnatf qui rendra Gnat beaucoup plus bavard.

Réduction de la taille des exécutables mode texte

Question : Quand je compile le programme suivant :

```
-- hello.adb

withText_IO;

procedure Hello is

begin

Text_IO.Put("Hello");

end Hello;
```

J'obtiens un exécutable de taille importante, peut-on la réduire ?

Réponse : Tout dépend des options de compilation et d'édition des liens utilisées, ainsi que de l'usage ou non d'un compresseur d'exécutable.

Si l'on compile sans options particulières par `gnatmake hello`, on obtient un exécutable d'environ 210 Ko.

En utilisant la commande suivante, `gnatmake hello -f -a -g -gnato -gnatwa -fstack-check`, on obtient un exécutable beaucoup plus gros : 520 Ko

Options utilisées :

- -a : intègre les .ali, même en lecture seule ;
- -f : force la compilation des packages non prédéfinis ;
- -g : intègre dans l'exécutable les informations nécessaires au débuggage ;
- -gnato : contrôle des dépassements ;
- -gnatwa : valide tous les avertissements ;
- -fstack-check : contrôle de la pile.

Par contre, en utilisant d'autres options `gnatmake hello -f -a -O2 -m386 -ffast-math -gnatp -largs -s`, on obtient un exécutable minuscule : 65 Ko. En utilisant le packer UPX avec cette dernière version, on tombe à 30 Ko.

Options utilisées :

- -a : intègre les .ali, même en lecture seule ;
- -f : force la compilation des packages non prédéfinis ;
- -O2 : optimise au niveau 2 ;
- -m386 :
- –fast-math :
- -gnatp : Supprimer tous les contrôles à l'exécution ;
- -largs -s : demande à l'éditeur de liens de supprimer tous les symboles.

Quelle est la meilleure approche ? Tout dépend du but recherché. Avec les exemples ci-dessus et la documentation livrée avec AIDE, en particulier les manuels du compilateur GNAT, vous pourrez choisir les options les plus appropriées au but fixé.

Comment débugguer

Question : Comment débugguer mon application ?

Réponse : Compilez avec la ligne suivante :

gnatmake testgtk -g -s

-g pour intégrer les informations de debug -s pour demander à gnatmake de recompiler les packages déjà créés mais sans l'option -g

Gagner du temps en débuggage

Question : Mon développement est assez important et le basculement de la version débug à la version finale et vice versa m'oblige à recompiler toute mon application.

Réponse : Au niveau de votre répertoire source :

- Créez deux sous-répertoires (nommés dans cet exemple '`ACU_debug`' et '`ACU_final`', ACU = Ada Compiled Units) ;
- Créez un batch pour la version debug : '`md.cmd`', contenant la ligne : `gnatmake -s -g -aOACU_debug <monsource>` ;
- Créez un batch pour la version finale : '`mf.cmd`', contenant la ligne : `gnatmake -s -O2 -aOACU_final <monsource> -largs -s` ;
- Compilez une première fois en "final", avec '`mf`', puis déplacez tous les '`.ali`' et les '`.o`' dans '`.\ACU_final`' ;
- Compilez une première fois en "debug", avec '`md`', puis déplacez tous les '`.ali`' et les '`.o`' dans '`.\ACU_debug`'.

Vous basculez de mode simplement en changeant de batch, seuls les packages modifiés entre temps seront recompilés.

Debugger dans le runtime GNAT

Question : Je voudrais suivre une session de débuggage jusque dans la bibliothèque d'exécution GNAT.

Réponse : Compilez avec la ligne suivante : `gnatmake testgtk -a -f`.

Options utilisées :

- -a pour intégrer les .ali, même en lecture seule ;
- -f pour forcer la compilation des packages non prédéfinis.

Développement multi-langage

Question : Peut-on mélanger dans un même développement des sources Ada avec celles d'autres langages ?

Réponse : Le support C est natif et ne pose aucun problème, voir les exemples. Le support de la plupart des autres langages pose peu de problèmes. Une exception notable pour l'instant : le support C++ semble difficile à réaliser sous win32, l'auteur n'y est personnellement pas arrivé, de la façon évoquée dans les exemples. Le mieux est de mettre tout le code C++ dans une DLL et ensuite d'appeler cette DLL[1].

Accès aux DLL

Question : Peut-on accéder à des DLL écrites dans un autre langage ou au contraire créer des DLL en Ada accessible à d'autres langages ?

Réponse : Vous trouverez toutes les réponses sur http://www.adapower.com (entre autres).

2.3.3 GtkAda

Console texte dans une application GtkAda

Question : Quand je compile une application GtkAda, j'ai toujours une console qui s'ouvre à l'exécution du programme.

Réponse : Compilez avec la ligne suivante :

`gnatmake testgtk -largs -mwindows`.

Bien sûr, il ne faudra pas que votre programme sorte aussi du texte sur la console.

Réduction de la taille des exécutables GtkAda

Question : Quand je compile une application GtkAda, j'obtiens des exécutables énormes.

Réponse : Pour un programme faisant 2 Mo, compilez avec la ligne suivante :

`gnatmake <options> -O2 <programme> -largs -s`

Options utilisées :

- -O2 : optimise au niveau 2 ;
- -largs -s : demande à l'éditeur de liens de supprimer tous les symboles.

L'application ne fait plus que 1 Mo.

Puis utilisez le packer upx fourni.

L'application ne fait plus que 250 Ko.

2.4 Anomalies

Message d'erreur lors de l'impression avec a2ps

Anomalie : L'impression délivre un message d'erreur : *file.exe: Cannot map 'c:/aide/share/file/magic.mgc' (Permission denied).*

Solution : Dans l'attente d'en comprendre l'origine et de traiter le problème (sic), l'ignorer, puisqu'il semble sans conséquence.

Implémentation OpenGL avec GtkAda

Anomalie : L'implémentation d'OpenGL avec GtkAda n'est pas fonctionnelle sous Windows.

Solution : Mettre à jour le code OpenGl de GtkAda pour Windows.

[1] Cette situation est appellée à évoluer avec l'intégration de GNAT dans GCC

2.5 Fichiers remarquables

2.5.1 'c:\aide\aide.cmd'

Le paramètrage de la partie Windows de AIDE est dans ce fichier, essentiellement sous forme de variables d'environnement. C'est par ce principe que AIDE se passe de toute configuration du système et ne nécessite aucune installation. Si tous les programmes agissaient de même, *quand ils le peuvent*, la vie serait plus simple.

```
echo off
rem no special masking character behind the previous command
rem to allow automatic include of this file in texinfo manual
rem ------------------------------------------------------------------------
rem                                                                        --
rem                 AIDE - ADA INSTANT DEVELOPMENT ENVIRONMENT             --
rem                                                                        --
rem                                    *                                   --
rem                                                                        --
rem                                AIDE 1.xx                               --
rem                                                                        --
rem                  Copyright (C) 2001-2004 Stephane Riviere              --
rem                                                                        --
rem                      http://stephane.rochebrune.org                    --
rem                                                                        --
rem                                    *                                   --
rem                                                                        --
rem AIDE is free software; you can  redistribute it  and/or modify  it under --
rem terms of the  GNU General Public License as published  by the Free Soft- --
rem ware  Foundation;  either version 2,  or (at your option) any later ver- --
rem sion. AIDE is distributed in the hope  that it will be useful, but WITH- --
rem OUT ANY WARRANTY;  without even the  implied warranty of MERCHANTABILITY --
rem or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License --
rem for  more details.  You should have  received  a copy of the GNU General --
rem Public License  distributed with AIDE;  see file COPYING.  If not, write --
rem to  the Free Software Foundation,  59 Temple Place - Suite 330,  Boston, --
rem MA 02111-1307, USA.                                                    --
rem                                                                        --

rem ------------------------------------------------------------------------
rem ENVIRONMENT VARIABLES DEFINES
rem ------------------------------------------------------------------------

rem ------------------------------------------------------------------------
rem Aide root path [USER DEFINED]

set ADA_ROOT=c:\aide

rem ------------------------------------------------------------------------
rem Aide path control

set ADA_TARGET=%ADA_ROOT%\aide.cmd

if exist %ADA_TARGET% goto CMD_CNT1

echo.
echo File "%ADA_TARGET%" not found !
echo Path "%ADA_ROOT%" could be invalid.
echo.

pause
goto CMD_END

:CMD_CNT1
```

```
rem ------------------------------------------------------------------------
rem Aide parameter control

if not "%1" == "" goto CMD_CNT2

echo.
echo Usage : aide [glide [file]] or [gps [file]] or [msys]
echo.
echo Examples : aide glide [file] : run Glide IDE with [file]
echo            aide gps [file]   : run GPS IDE with [file]
echo            aide msys         : run Msys console
echo.
echo You can also use predefined glide.lnk, gps.lnk and msys.lnk
echo.

pause
goto CMD_END

:CMD_CNT2

rem ------------------------------------------------------------------------
rem Main internal paths

set PATH=%ADA_ROOT%\binw;%ADA_ROOT%\bin;%PATH%
set HOME=%ADA_ROOT%\home\%USERNAME%
set TEMP=%ADA_ROOT%\tmp
set TMP=%TEMP%

set GLIDE_BASE=%ADA_ROOT%
set GLIDE_DIR=%ADA_ROOT%\lisp\glide

set GVD_ROOT=%ADA_ROOT%
set ISPELLDICTDIR=%ADA_ROOT%\share\ispell
set WORDLIST=%HOME%\.ispell_word

set GS_LIB=%ADA_ROOT%\lib\gs;%ADA_ROOT%\lib\gs\fonts
set GS_CONFIG=%ADA_ROOT%\etc\gs
set A2PS_CONFIG=%ADA_ROOT%\etc\a2ps
rem set MAGIC=%ADA_ROOT%\share\file\magic.mgc

rem ------------------------------------------------------------------------
rem Lib path

set ADA_INCLUDE_PATH=%ADA_ROOT%\lib
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\gnat
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\gtkada
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\win32ada
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\xmlada
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\opengl

rem ------------------------------------------------------------------------
rem Lib path [USER DEFINED - PLACE HERE YOUR LIBRAIRIES]

set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\aide\v04
set ADA_INCLUDE_PATH=%ADA_INCLUDE_PATH%;%ADA_ROOT%\lib\aide\v04\windows

rem ------------------------------------------------------------------------
rem Src, crt2.o, C path

set ADA_OBJECTS_PATH=%ADA_INCLUDE_PATH%
set GCC_EXEC_PREFIX=%ADA_ROOT%\lib\
set C_INCLUDE_PATH=%ADA_ROOT%\lib\include
```

```
rem ------------------------------------------------------------------------
rem AIDE USER CREATION
rem ------------------------------------------------------------------------

if exist %HOME% goto AUC_END

rem ------------------------------------------------------------------------
rem Create user

md %HOME%
if not exist %HOME% goto AUC_ER1

copy %ADA_ROOT%\share\aide\.emacs %HOME%
if not exist %HOME%\.emacs goto AUC_ER2

rem ------------------------------------------------------------------------
goto AUC_END

rem ------------------------------------------------------------------------
:AUC_ER1
echo Can't create %HOME%, aborting.
pause
goto CMD_END

:AUC_ER2
echo Can't copy %ADA_ROOT%\share\aide\.emacs to %HOME%, aborting.
pause
goto CMD_END

rem ------------------------------------------------------------------------
:AUC_END

rem ------------------------------------------------------------------------
rem APPLICATIONS
rem ------------------------------------------------------------------------

rem ------------------------------------------------------------------------
rem Glide settings and launch, uncomment the following line to use Glide

if not "%1" == "glide" goto no_glide
start /b %GLIDE_BASE%\bin\glideint %GLIDE_DIR% %GLIDE_BASE%\bin\runemacs.exe %2 %3 %4 %5 %6 %7 %8 %9
:no_glide

rem ------------------------------------------------------------------------
rem GPS settings and launch, uncomment the following line to use GPS

if not "%1" == "gps" goto no_gps
start /b gps %2
:no_gps

rem ------------------------------------------------------------------------
rem mSys console settings and launch, uncomment the following line to use mSys console

if not "%1" == "msys" goto no_msys
start /b rxvt -title 'AIDE - Ada Instant Developement Environment - mSys console' -sl 2500 -
fg Navy -bg LightYellow -cr red -sr -fn Courier-12 -tn msys -e /bin/sh --login -i
:no_msys

rem ------------------------------------------------------------------------
:CMD_END
rem eof aide.cmd
```

AIDE root path

La variable `ADA_ROOT` détermine la racine de AIDE. C'est, *a priori*, la seule variable que vous pourriez souhaiter changer dès la copie de AIDE, si le chemin '`c:\aide`' ne vous satisfaisait pas.

Main internal paths

La variable système `PATH` est redéclarée pour prendre en compte les chemins d'exécutables spécifiques à AIDE. Cet ajout n'est effectif que dans l'environnement de AIDE et disparait à la fermeture de AIDE.

La variable `HOME` détermine le répertoire de travail de l'utilisateur courant, elle est fondée sur la variable système `USERNAME`.

Les variables systèmes `TEMP` et `TMP` sont redéclarées pour rediriger les fichiers temporaires vers le répertoire '`tmp`', standard sous Unix.

La variable `GS_LIB` définit les emplacements des scripts, bibliothèques et polices de Ghostscript.

La variable `GS_CONFIG` définit le fichier de configuration de Ghostscript. Ce fichier est très important. Apprendre à le modifier vous ouvrira plein de possibilités, comme par exemple, créer des fichiers '`PDF`', au format Acrobat. Reportez-vous à la documentation.

La variable `A2PS_CONFIG` définit le fichier de configuration de '`a2ps`'.

La variable en commentaire `MAGIC` représente une tentative dénuée de succès pour éviter un message d'anomalie, sans conséquences visibles, lors des impressions avec a2ps.

La variable `ADA_INCLUDE_PATH` contient tous les chemins visités par GNAT pour trouver les paquetages dont il a besoin. Par défaut, la variable `ADA_OBJECTS_PATH` contient les mêmes informations.

La variable `GCC_EXEC_PREFIX` définit l'emplacement du code de démarrage de GCC, en l'espèce le fichier '`crt2.o`'. Cette variable est nécessaire pour éviter d'avoir à placer '`crt2.o`' dans chaque répertoire de travail...

La variable `C_INCLUDE_PATH` définit l'emplacement des fichiers d'en-têtes '`.h`' du langage C. C'est utile lors de projets mixtes ou pour tester les exemples fournis.

2.5.2 ‘c:\aide\etc\profile’

```
# ------------------------------------------------------------------------
#                                                                        --
#                 AIDE - ADA INSTANT DEVELOPMENT ENVIRONMENT             --
#                                                                        --
#                                    *                                   --
#                                                                        --
#                                AIDE 1.xx                               --
#                                                                        --
#                 Copyright (C) 2001-2004 Stephane Riviere               --
#                                                                        --
#                     http://stephane.rochebrune.org                     --
#                                                                        --
#                                    *                                   --
#                                                                        --
# AIDE is free software; you can  redistribute it  and/or modify  it under --
# terms of the  GNU General Public License as published  by the Free Soft- --
# ware  Foundation;  either version 2,  or (at your option) any later ver- --
# sion. AIDE is distributed in the hope  that it will be useful, but WITH- --
# OUT ANY WARRANTY;  without even the  implied warranty of MERCHANTABILITY --
# or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License --
# for  more details.  You should have  received  a copy of the GNU General --
# Public License  distributed with AIDE;  see file COPYING.  If not, write --
# to  the Free Software Foundation,  59 Temple Place - Suite 330,  Boston, --
# MA 02111-1307, USA.                                                    --
#                                                                        --
# ------------------------------------------------------------------------

#  \a     an ASCII bell character (07)
#  \d     the date  in  "Weekday  Month  Date"  format (e.g., "Tue May 26")
#  \e     an ASCII escape character (033)
#  \h     the hostname up to the first '.'
#  \H     the hostname
#  \j     the  number of jobs currently managed by the shell
#  \l     the basename of the shell's terminal  device name
#  \n     newline
#  \r     carriage return
#  \s     the  name  of  the shell, the basename of $0 (the portion following the final slash)
#  \t     the current time in 24-hour HH:MM:SS format
#  \T     the current time in 12-hour HH:MM:SS format
#  \     the current time in 12-hour am/pm format
#  \u     the username of the current user
#  \v     the version of bash (e.g., 2.00)
#  \V     the release of bash,  version  +  patchlevel (e.g., 2.00.0)
#  \w     the current working directory
#  \W     the  basename  of the current working directory
#  \!     the history number of this command
#  \#     the command number of this command
#  \$     if the effective UID is 0, a #, otherwise a $
#  \nnn   the  character  corresponding  to  the octal number nnn
#  \\     a backslash
#  \[     begin a sequence of non-printing characters, which could be used
#               to embed a terminal control sequence into the prompt
#  \]     end a sequence of non-printing characters
#
#  Black       0;30     Dark Gray     1;30
#  Blue        0;34     Light Blue    1;34
#  Green       0;32     Light Green   1;32
#  Cyan        0;36     Light Cyan    1;36
#  Red         0;31     Light Red     1;31
#  Purple      0;35     Light Purple  1;35
#  Brown       0;33     Yellow        1;33
#  Light Gray  0;37     White         1;37
```

```
# [user] current_working_directory >

export PS1='\[\033[0;31m\][\u] \w >\[\033[0m\]'

# others useful environment variables

export PATH=".:/bin:$PATH"
export LOGNAME="$USERNAME"
export HOME="/home/$LOGNAME"
export HISTFILE=$HOME/.bash_history
export CONFIG_SITE=/etc/config.site
export MAKE_MODE=unix
export USERPROFILE=$HOME

export LANGUAGE=en
export LANG=en

# Goto home directory

cd "$HOME"

# eof profile
```

3 Développement avec Glide

Mon système d'exploitation est Emacs et Windows est son driver.

3.1 Présentation

Glide est aujourd'hui la solution de développement Ada la plus aboutie. Fondée sur Emacs, un logiciel surpuissant et complètement configurable, elle n'a, pour l'instant, aucun équivalent.

Emacs est le type même de logiciel qui vous donne mille fois en retour l'investissement de départ que vous avez bien voulu lui consentir. Modifiable et configurable à l'infini, Emacs permet enfin au développeur d'utiliser *une seule et même interface* pour tous ses travaux de saisie : les programmes, la documentation et le courrier. À l'usage, le gain de productivité est flagrant.

3.2 Lancement

Glide peut être lancé par la commande '`c:\aide\aide.cmd glide [nom de fichier]`' ou par l'icône de Glide sur le bureau.

Icône de Glide

3.3 Test de l'installation

Dans Glide, allez dans le *buffer* des nouveaux fichiers par `Buffer/*scratch*`.

Saisir le programme suivant :

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Bonjour_Monde_1 is
begin
   Put_Line ("Bonjour Monde 1 !");
end;
```

Sauvez le programme sous le nom '`c:\aide\home\<votre login>\bonjour_monde_1.adb`' par la commande `File/Save Buffer As...`.

Compilez en appuyant sur ⟨F4⟩.

À l'issue de la compilation, lancez le programme par `Ada/Run`. Une nouvelle fenêtre apparait avec le résultat de l'exécution :

```
Bonjour Monde 1 !

Process run finished
```

3.4 Impression

Une impression de type *pretty-printer* est intégrée dans Glide.

Choisissez une sortie couleur par `File/Postscript Print Buffer` ou noir et blanc par `File/Postscript Print Buffer (B+W)`.

Pour une impression standard, choisissez `File/Print Buffer`.

3.5 Commandes clavier

Les commandes Glide dans AIDE sont fondées sur Wordstar pour les plus importantes, avec quelques arrangements. Les autres commandes restent des classiques d'Emacs.

La bonne nouvelle est que vous pouvez tout changer sans trop d'effort, hormis celui de parcourir la doc d'Emacs, très explicite et d'étudier le fichier '`.emacs`' de votre répertoire personnel dans '`c:\aide\home`'.

Commandes fichier

⟨F2⟩	Sauve le fichier courant
⟨F3⟩	Ouvre un fichier
⟨SHIFT⟩-⟨F3⟩	Ouvre un projet
⟨F4⟩	Construit l'application
⟨SHIFT⟩-⟨F4⟩	Compile le source courant

Commandes de bloc

⟨CTRL⟩-⟨SPACE⟩	Puis sélection du bloc avec ⟨UP⟩ ou ⟨DOWN⟩
⟨F5⟩	Copie un bloc sélectionné
⟨F6⟩	Déplace un bloc sélectionné
⟨F7⟩	Colle un bloc sélectionné

Commandes curseur

⟨HOME⟩	Début de ligne
⟨END⟩	Fin de ligne
⟨CTRL⟩-⟨HOME⟩	Début de fichier
⟨CTRL⟩-⟨END⟩	Fin de fichier
⟨CTRL⟩-⟨LEFT⟩	Mot précédent
⟨CTRL⟩-⟨RIGHT⟩	Mot suivant

Commandes de recherche et de remplacement

⟨CTRL⟩-⟨S⟩	Recherche vers l'avant (poursuivre la recherche en retapant ⟨CTRL⟩-⟨S⟩ à nouveau)
⟨CTRL⟩-⟨R⟩	Recherche et remplacement avec confirmation ('y' pour remplacer, 'n' pour laisser tel quel, ? pour l'aide détaillée)
⟨ESC⟩-⟨P⟩	Dans une recherche en cours, parcours de l'anneau des chaînes de recherche déjà utilisées

Commandes d'édition

⟨CTRL⟩-⟨B⟩	Reformate le paragraphe
⟨CTRL⟩-⟨G⟩	Efface le minibuffer
⟨CTRL⟩-⟨P⟩-lettre	Insère un caractère de contrôle
⟨CTRL⟩-⟨T⟩	Efface le mot à droite
⟨CTRL⟩-⟨E⟩	Efface du curseur à la fin de la ligne
⟨CTRL⟩-⟨Y⟩	Supprime la ligne
⟨CTRL⟩-⟨U⟩	Annule la dernière modification

Commandes CTRL-K-C

⟨CTRL⟩-⟨K⟩-⟨C⟩	Colle un bloc sélectionné
⟨CTRL⟩-⟨K⟩-⟨K⟩	Déplace un bloc sélectionné
⟨CTRL⟩-⟨K⟩-⟨Y⟩	Efface un bloc sélectionné

⟨CTRL⟩-⟨K⟩-⟨R⟩	Insère un fichier au curseur
⟨CTRL⟩-⟨K⟩-⟨W⟩	Exporte un bloc sélectionné dans un fichier

Commandes d'édition pour les caractères français et européens

⟨CTRL⟩-⟨SHIFT⟩-⟨2⟩	É
⟨CTRL⟩-⟨SHIFT⟩-⟨7⟩	È
⟨CTRL⟩-⟨SHIFT⟩-⟨9⟩	Ç
⟨CTRL⟩-⟨SHIFT⟩-⟨0⟩	À
⟨ALT⟩-⟨A⟩	`@ae{}` (æ)
⟨ALT⟩-⟨SHIFT⟩-⟨A⟩	`@AE{}` (Æ)
⟨ALT⟩-⟨O⟩	`@oe{}` (œ)
⟨ALT⟩-⟨SHIFT⟩-⟨O⟩	`@OE{}` (Œ)
⟨ALTGR⟩-⟨E⟩	EUR (symbole monétaire européen)

L'usage de ces caractères est conseillé pour une documentation de qualité. Les caractères æ, Æ, œ et Œ et EUR nécessitent un traitement particulier car ils appartiennent au nouveau jeu de caractère ISO-8859-15 ou Latin-9, remplaçant l'ancien ISO-8859-1 ou Latin-1.

3.6 Questions courantes

Suppression de la *Speedbar*

Question : Je souhaiterais supprimer le lancement par défaut de la *Speedbar* au démarrage de Glide.

Réponse : Lancez Glide avec l'option *-nospeedbar*.

Exemple : '`aide glide -nospeedbar [nom fichier]`'

3.7 Anomalies

Anomalie : Si on ouvre d'abord un fichier source puis un second et que l'on fait F4, la compilation s'effectue dans le répertoire du premier fichier source.adb.

Solution : Ouvrir un Glide par projet.

4 Développement avec GPS

4.1 Présentation

GPS est une solution de développement complètement graphique, majoritairement écrite en Ada, multi-langage et scriptable en python.

4.2 Lancement

GPS peut être lancé par la commande '`c:\aide\aide.cmd gps [nom de fichier]`' ou par l'icône de GPS sur le bureau.

Icône de GPS

4.3 Test de l'installation

Dans GPS, créez un nouveau fichier par `File/New`.

Saisir le programme suivant :

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Bonjour_Monde_2 is
begin
   Put_Line ("Bonjour Monde 2 !");
end;
```

Sauvez le programme sous le nom '`c:\aide\home\<votre login>\bonjour_monde_2.adb`' par la commande `File/Save As...`.

Compilez par la commande `Build/Make/<current file>`.

À l'issue de la compilation, lancez un shell par `Build/Run/Custom...`, saisissez '`cmd`' dans la zone *Enter the command to run* et appuyez sur ⟨RET⟩, un shell s'ouvre dans la fenêtre en bas à droite.

Dans ce shell, vérifiez que vous êtes bien dans le répertoire '`c:\aide\home\<votre login>\`'. Vous pouvez voir vos fichiers par la commande `ls`, suivie de ⟨RET⟩.

Tapez '`bonjour_monde_2`', suivi de ⟨RET⟩ pour voir le résultat.

4.4 Questions courantes

<<<TODO>>>

4.5 Anomalies

Système d'aide HTML

Anomalie : Le système d'aide de GPS n'accepte pas de gros fichiers HTML. Le chargement est très lent et la totalité du fichier n'est pas accessible.

Solution : Fractionner et formater le document HTML à l'image des fichiers d'aide HTML fournis avec GPS.

5 Développement avec Msys

5.1 Présentation

On a toujours besoin d'une petite ligne de commande sur son Windows, surtout quand cette console est une console Unix, capable d'exécuter des scripts bash et toutes les commandes classiques d'un terminal Unix !

5.2 Lancement

Msys peut être lancé par la commande '`c:\aide\aide.cmd msys`' ou par l'icône sur le bureau.

Icône de la console Msys

5.3 Test de l'installation

Lancez l'éditeur de la console par `fte`, suivi de ⟨RET⟩. Appuyez alors sur ⟨F3⟩ et saisissez le nom du fichier, soit '`bonjour_monde_3.adb`', suivi de ⟨RET⟩.

Puis saisissez le texte du programme :

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Bonjour_Monde_3 is
begin
   Put_Line ("Bonjour Monde 3 !");
end;
```

Sauvez le programme par ⟨F2⟩ et quittez par ⟨Alt-X⟩.

Compilez par la commande `gnatmake bonjour_monde_3`.

À l'issue de la compilation, lancez '`bonjour_monde_3`' pour voir le résultat.

5.4 Questions courantes

Utilisation de find avec des fichiers contenants des caractères spéciaux

Question : Comment utiliser un chemin ou un fichier dont le nom de fichier contient des espaces, points, apostrophes et autres caractères inhabituels ?

Réponse : Encadrer le nom du chemin ou du fichier avec des guillemets. Si le nom contient des guillemets, encadrez-le avec des apostrophes.

Question : Comment utiliser '`find`' avec des fichiers contenants des caractères spéciaux (espaces, points, apostrophes, etc...) ?

Réponse : Pour lister des fichiers dont les noms contiennent des espaces, eux-mêmes situés dans un chemin contenant au moins un espace, utilisez la commande : '`find ‘‘chemin avec espaces/’’ -name ‘‘masque’’ -printf "\"%p\"\n"`'.

Exemple : '`find ‘‘/c/aide/doc/ada 95/’’ -name ‘‘*.pdf’’ -printf "\"%p\"\n"`'

5.5 Anomalies

Anomalie : Après le lancement d'une console MSys, un premier appel à Fte peut être infructueux.

Solution : Relancer la commande une seconde fois. Les appels suivants s'effectueront normalement.

6 Documentation

6.1 Généralités

6.1.1 L'enfer du développeur

La production de la documentation utilisateur et technique d'un logiciel pose souvent problème.

C'est une tâche mal aimée et demandant du temps car elle impose - entres autres - des connaissances typographiques et de mise en page pour la forme, mais aussi pour le fond des contraintes de vocabulaire, de style et des compétences linguistiques propres à la *rédaction technique*.

Des aspects spécifiques de synchronisation viennent aussi affecter cette activité :

- Synchronisation entre documentations identiques mais dans des langues différentes ;
- Synchronisation entre documentations différentes sur la forme mais similaires sur le fond (aide en ligne et manuels utilisateurs) ;
- Synchronisation entre documentation technique et code source.

Le maintien de la cohérence d'une documentation est un problème très complexe. Enfin, pour achever de compliquer la situation, la documentation est, le plus souvent, effectuée avec des outils inadéquats.

Ce tableau d'ensemble une fois dressé, il est aisé de comprendre que la *doc*, ainsi familièrement appelée, est *l'enfer* du développeur.

6.1.2 La solution Texinfo

Compte tenu de la situation évoquée ci-dessus, une solution *Libre* a été développée : Texinfo.

Texinfo est un système qui décharge le rédacteur des problèmes de synchronismes et de mise en page tout en assurant la production de la documentation sous tous les formats utiles, aussi bien pour l'aspect utilisateur que technique.

Voici quelques éléments supplémentaires sur Texinfo :

- Texinfo est très simple à apprendre et à utiliser. Texinfo n'est pas un logiciel WYSIWYG, mais un langage à balises très simple à utiliser[1] et parfaitement adapté au problème à résoudre ;
- Texinfo est capable de produire, à partir d'un source unique, des sorties aux formats '`HTML`', '`PDF`', '`TXT`', '`PS`', '`INFO`' et même '`HLP`', le format des fichiers d'Aide Windows. Avec une seule saisie et un seul fichier, vous obtenez le manuel papier en '`PDF`'[2] ou '`PS`', l'aide en ligne intégrée '`INFO`' ou '`HLP`', et la documentation internet et l'aide en ligne via la sortie '`HTML`' ;
- Texinfo produit des documents de qualité professionnelle : le résultat typographique de la sortie papier est exceptionnel, sans que l'utilisateur ait besoin de s'investir à ce niveau, ce qui n'est pas rien, car la typographie est un art.

Après beaucoup d'années d'expérience, l'auteur croit pouvoir vous assurer que Texinfo est *sans équivalent* pour produire la documentation utilisateur d'un logiciel.

6.2 Installation pour les manuels imprimables

Quoique Texinfo soit pré-installé dans AIDE, Il est nécessaire d'installer TEX pour produire des manuels imprimables.

---

[1] Texinfo est incomparablement plus accessible que TEX et ne demande aucun apprentissage particulier.

[2] Avec création automatique de la hiérarchie de signets pour la navigation et les liens hypertextes à partir du sommaire.

6.2.1 Installation de TEX

Cette étape a été très soigneusement décrite pour éviter tout ambiguïté.

Téléchargez la distribution de fpTEX à l'adresse :
http://www.ctan.org/tex-archive/systems/win32/fptex

Installez-la en suivant *strictement* la procédure ci-dessous :

- Lancer le programme d'installation '`TexSetup`' ;
- Sélectionner *Install for all user* et pressez ⟨Suivant⟩ ;
- Même si fpTEX est placé sur un CD-ROM, sélectionnez *Local directory / ZIP files* et pressez ⟨Suivant⟩ ;
- Choisissez un répertoire d'installation ne comportant pas d'espaces, par exemple '`c:\tex`', et pressez ⟨Suivant⟩, le programme d'installation examine le contenu de fpTEX ;
- Sur la fenêtre *TeXLive Setup Wizard*, dans la liste *Choose a scheme to install*, assurez vous que *Generic basic TeXLive scheme* est bien sélectionné ;
- Cochez *I want to customize selected scheme* et pressez ⟨Suivant⟩ ;
- Ouvrez l'arborescence *texlive* ;
- Effectuez les opérations suivantes :
  - Laisser tex-basic sélectionné ;
  - Laisser tex-documentation sélectionné ;
  - Laisser tex-etex sélectionné ;
  - Ouvrez tex-extrabin, laisser tex-basic sélectionné et sélectionnez texinfo ;
  - Sélectionnez tex-langfrench ;
  - Laisser tex-latex sélectionné;
  - Laisser tex-pdftex sélectionné ;
  - Sélectionner tex-psutils.

Puis pressez ⟨Suivant⟩ afin de poursuivre et laissez ensuite l'installation de fpTEX se terminer.

Vous disposez alors d'un environnement TEX prêt à l'emploi.

6.2.2 Installation de Texinfo dans Tex

Les instructions ci-dessous considèrent que votre installation de TEX est placée dans '`c:\tex`'

Pour installer Texinfo dans TEX, trois étapes suffisent :

- Créez le répertoire '`c:\tex\texmf\tex\texinfo`' ;
- Copiez tous les fichiers situés dans '`c:\aide\share\texinfo`' vers le répertoire '`c:\tex\texmf\tex\texinfo`' ;
- Mettez à jour votre installation de TEX par '`Démarrer / Programmes / TeXLive / Maintenance / Rebuild ls-R filenames databases`' ou par la commande '`c:\tex\bin\win32\mktexlsr.exe`'.

Vous disposez alors d'un environnement Texinfo prêt à l'emploi pour la production de manuels imprimables.

6.2.3 Test de l'installation

Pour tester votre nouvel environnement de production de documentation, lancer Glide et charger le source de la documentation de AIDE : '`c:\aide\doc\aide\aide.texi`', puis compilez-le par la commande ⟨F9⟩.

Une fois la compilation terminée, vous disposez, dans le même répertoire, d'un fichier au format PDF, dénommé '`aide.pdf`'. Cerise sur le gâteau, l'arborescence de signets et les hyperliens sont automatiquement créés !

À partir de ce même fichier source '`aide.texi`', il est possible de créer des sorties HLP, HTML, INFO, PS, TXT, etc...

Habituellement, vous écrivez votre manuel une fois et vous en dérivez *au moins* deux compilations ; la sortie HTML pour l'aide en ligne intégrée dans le logiciel et la sortie PDF pour la documentation papier.

6.3 Formation Texinfo

Le manuel de Texinfo est inclu dans AIDE, les autres documentations pertinentes sont situées dans '`c:\tex\texmf\doc`'.

Ce manuel lui-même utilise Texinfo. Ses sources sont disponibles dans le répertoire '`c:\aide\doc\aide\manuals`' et sont un bon exemple des possiblités de Texinfo.

6.4 Documentation utilisateur

6.4.1 Règles typographiques

La production de manuels en français avec Texinfo est pré-intégrée dans AIDE, par l'intermédiaire de Glide et d'autres utilitaires permettant la création de documentation en français avec une typographie correcte, en particulier capable de traiter les caractères æ, Æ, œ et Œ, mais aussi les majuscules accentuées telles que É, È, À et Ç[3].

6.4.2 Internationalisation

Par l'intermédiaire de la fonction `@ifset` de Texinfo, il est possible de gérer, dans un seul fichier, plusieurs versions linguistiques d'une seule et même documentation. Cette disposition est très intéressante pour maintenir la cohérence du document et faciliter le travail du traducteur.

Texinfo inclut nativement le support du tchèque, de l'allemand, de l'anglais, de l'espagnol, du français, de l'italien, du néerlandais, du norvégien, du polonais, du portugais et du turc. Il est aisé de rajouter une nouvelle langue.

6.5 Documentation technique

Une documentation technique fiable implique d'avoir un référentiel unique : les sources du logiciel documenté. Pour ce faire AIDE propose AdaDoc, logiciel d'extraction de spécifications commentées, avec un module Texinfo spécialement étudié à cette occasion.

Avec son module Texinfo, Adadoc produit, à partir d'une spécification Ada, un document Texinfo automatiquement généré, prêt à inclure dans la documentation technique.

Beaucoups de temps a été consacré à l'écriture de ce module pour proposer un résultat harmonieux, dans la lignée des manuels Texinfo. Toutefois, l'auteur reste à l'écoute pour toute suggestion d'amélioration ou rapport d'anomalie.

Plus généralement, Adadoc est capable de générer des sorties aux formats suivants :

- `XML` ;
- LaTeX ;
- `HTML`, pour l'aide en ligne ou la documentation internet ;
- `Texinfo`, pour les manuels imprimables.

[3] Extraits du *Lexique des règles typographiques en usage à l'imprimerie nationale* : En français, l'accent a pleine valeur orthographique. On veillera à utiliser systématiquement les capitales accentuées, y compris la préposition À. On évitera ainsi de désorienter le lecteur ou même de l'induire en erreur comme ce pourrait être le cas dans les deux exemples suivants : *LÉGITIME et LÉGITIMÉ*, ou bien *MODÈLE et MODELÉ*.

Adadoc dispose d'une documentation complète en anglais et en français et permet l'ajout de nouveaux modules de sortie.

AdaDoc possède son propre *parser* Ada, et ne nécessite pas de compilateur basé *ASIS*. Il n'utilise donc pas les fichiers '`.ALI`'.

6.5.1 Utilisation de Adadoc

Pour générer un document Texinfo '`v04-crt.texi`' à partir de la spécification '`v04-crt.ads`', il suffit de taper '`adadoc -T c:\aide\lib\aide\v04\v04-crt.ads`'. Le document généré pourra être inclu dans la documentation technique au format Texinfo grâce à la commande `@include`.

Pour l'usage général d'Adadoc, se référer à la documentation jointe. Pour l'usage particulier du module Texinfo d'Adadoc, se référer au paragraphe suivant.

6.5.2 Référence du module Texinfo

Usage

Message d'aide Adadoc

```
AdaDoc create a documentation in different format
from an Ada95 specification package.

Use : adadoc (option) (File_Name)

Options :
  -x          : don't delete XML file after generating output.
  -c          : don't try to extract comments from the specification.
  -b          : don't ask to 'press <enter>' for batch use.
  -Hstylsheet : generate HTML file. stylsheet is optional.
  -L          : generate LaTeX file.
  -Tns        : generate Texinfo file. -n : numbered output -s : stand-alone output.

Example : adadoc -x -Hfid spec.ads spec2.ads ...

AdaDoc by Julien Burdy and Vincent Decorges
Texinfo module by Stephane Riviere
Web => http://adadoc.sf.net
AdaDoc is freely available under the GPL license.
Version 2.02t
```

Paramètres

`-c` Supprime l'extraction des commentaires.

`-b` Mode "Batch" sans confirmation par 'press enter' en cas d'ereur, afin d'effectuer des traitements par lots.

`-T` Sélectionne le mode Texinfo

`-n` Option spécifique Texinfo : sélectionne la sortie Texinfo avec numérotation.

`-s` Option spécifique Texinfo : sélectionne la sortie Texinfo autonome. La sortie par défaut est prévue pour être incluse dans un document Texinfo principal par la commance `@include`.

Exemples

- `adadoc -b -T *.ads` : Traiter en mode batch toutes les spécifications du répertoire courant pour une génération Texinfo ;
- `adadoc -Tns test.ads` : Générer une sortie Texinfo autonome et numérotée de la spécification '`test.ads`'.

Caractères spéciaux

Le caractère `@` est utilisé par Adadoc comme tag d'extraction des commentaires d'en-tête de source. Ce même caractère est aussi le tag principal du langage Texinfo. Pour qu'un caractère `@` soit généré en sortie, il est nécessaire de le doubler dans le source, exemple : `nom@@domaine.org`.

Quoique accepté par Ada, l'usage des caractères accentués dans les noms de variables est incompatible avec Adadoc.

Les caractères accentués sont déconseillés dans les commentaires de codes sources, pour des raisons de présentation dans la génération Texinfo. Dans le cas d'une documentation technique en français, il est possible d'y remédier en pré-traitant les fichiers Texinfo générés par Adadoc avec l'utilitaire Free Recode, inclu dans la distribution et utilisé pour la génération des manuels en français.

Commentaires

L'insertion des commentaires, dans le module Texinfo, diffère largement des autres modules Adadoc.

En raison de la largeur limitée des manuels imprimables, l'insertion des commentaires en ligne a été principalement limitée aux en-têtes de fonction et de procédure.

Pour ces deux cas, afin de se conformer aux usages en matière de documentation, les commentaires sont désormais placés après la liste des paramètres (les autres modules affichent les commentaires entre le nom de la fonction et la liste des paramètres).

Cette fonction a aussi été améliorée au niveau de l'extraction. Les commentaires extraits, placés au dessus de la définition, se limitent aux deux premières lignes blanches consécutives rencontrées[4].

Se référer au fichier '`c:\aide\lib\aide\v04\v04-crt.ads`' pour un exemple de commentaires compatibles avec une extraction Adadoc.

Insertion de l'opérateur =>

L'opérateur Ada `=>`, composé des deux caractères `=` et `>`, est traduit par le symbole ⇒ pour une meilleure typographie.

Compilation d'Adadoc

XmlAda, Aflex, Ayacc et Adadoc ont tous été recompilés à partir d'une console MSys pour une meilleure optimisation (-f -a -O2 -m386 -largs -s), puis les exécutables ont été compressés avec UPX.

Pour compiler Adadoc, il suffit de lancer une console MSys, puis de se placer dans le répertoire '`/c/aide/examples/adadoc`' et de taper "makelinux -c -O2 -m386 -largs -s".

Remarque : la compilation de ada95_parser.adb est extrèmement longue.

Anomalie connue

Le module ne met pas encore en page les entités CHOICE, TASK, VARIANT PART, PROTECTED OBJECT, ENTRY, car je suis à la recherche de spécifications de test comportant ces entités.

6.6 Logiciels complémentaires

Afin d'illustrer vos manuels et vos aides en ligne, deux logiciels complémentaires sont suggérés :

- Gimp : Logiciel de traitement d'images généraliste, disponible en version Windows à partir de : http://www.gimp.org/~tml/gimp/win32/downloads.html ;
- Dia : Logiciel d'édition de diagrammes, disponible en version Windows à partir de : http://www.dia-installer.sourceforge.net.

[4] Dans les autres modules, la limite de départ est la première expression Ada rencontrée, ce qui est évidemment gênant si des commentaires généraux ont été insérés au dessus des commentaires de la procédure ou de la fonction.

Ces deux logiciels, comme toujours libres et multi-plateformes, permettent de produire des images et des diagrammes au format '`EPS`', '`PNG`' et '`JPG`'.

Le format '`EPS`' sera converti en '`PDF`' grâce à la commande '`epstopdf nomfic.eps`', qui produira un fichier '`nomfic.pdf`' nécessaire à PdfTEX pour la production des manuels au format Acrobat.

Les formats '`PNG`' et '`JPG`' seront indifféremment utilisés pour la production de l'aide en ligne au format '`HTML`'.

Enfin, il est intéressant de remarquer que Gimp, Dia et Glade sont tous les trois fondés sur GTK et proposent une interface utilisateur conçue dans le même esprit.

6.7 Questions courantes

Correction des erreurs

Question : Où puis-je trouver les messages d'erreurs suite à une compilation Texinfo interrompue par une anomalie ?

Réponse : Dans le fichier *log* de votre document : '`votre_document.log`'. Aller à la fin du fichier et regarder la ligne incriminée. De retour dans votre document, allez à la ligne indiquée et corriger l'erreur de syntaxe.

Mise à jour automatique du sommaire

Question : Le sommaire et les signets de mon manuel ne sont pas à jour.

Réponse : La concordance entre le texte du manuel d'une part et, d'autre part, le sommaire et l'arborescence des signets nécessite une compilation supplémentaire !

La première compilation produit un manuel dont le texte est à jour. La seconde compilation[5], *qui doit être effectuée sans que le texte ait changé*, créera un sommaire et des signets à jour par rapport au texte.

Caractères accentués dans les titres

Question : Je n'arrive plus à compiler mon document mais il ne semble pas qu'il y ait d'erreur de syntaxe.

Réponse : Vous employez peut être un caractère accentué non valide, tels le *ç*, dans un titre ou un sous titre. Supprimez le problème en deux étapes :

- Nettoyez votre répertoire de travail en supprimant les fichiers '`*.fn`', '`*.ky`', '`*.log`', '`*.pg`', '`*.toc`', '`*.tp`', '`*.vr`' ;
- Modifier le titre ou le sous titre contenant le caractère non valide.
- Recompilez, le problème doit avoir disparu.

[5] La seconde compilation pourra être forcée par l'ajout puis la suppression d'un espace à la fin d'une ligne.

7 Sauvegarde des données

7.1 Généralités

7.1.1 L'enfer du développeur (bis)

La sauvegarde des développements pose, comme la documentation, de nombreux problèmes.

Après avoir essayé de nombreux procédés de sauvegarde, il faut reconnaître que la plupart d'entre eux ne sont pas fiables, surtout lorsque le volume des données est important.

Il est fascinant de constater qu'un procédé de sauvegarde fonctionnant parfaitement pour un volume de quelques dizaines de Mo présente des anomalies (souvent à la restauration) dès que les données dépassent le Go.

Avant de présenter la solution retenue, voici quelques règles, dictées par l'expérience :

- *Règle 1 : Les supports de sauvegarde ne sont pas fiables.* Il faut privilégier un procédé de sauvegarde qui permettra une récupération de toutes les données qui ne sont pas sur la partie du support endommagée. Cette règle exclut d'emblée tous les procédés compressant en un seul bloc des arborescences entières (zip, tar.gz, tar.bz2, etc... sont dans ce cas) car un seul bit altéré dans l'archive peut entraîner la perte de la totalité de l'archive ;
- *Règle 2 : Les supports de sauvegarde non standards sont à proscrire.* Il faut privilégier un support de sauvegarde standard car, par définition, une sauvegarde doit pouvoir être lue à partir d'une machine standard. Si vous avez utilisé pour vos sauvegardes un lecteur DAT très spécifique et que vos bureaux ont brûlé, mêmes si vos sauvegardes étaient prudemment dupliquées à votre domicile, vous aurez le plus grand mal à restaurer vos données ;
- *Règle 3 : Le logiciel de sauvegarde doit être libre et standard.* Pour des raisons de pérennité des sauvegardes, il faut privilégier un logiciel de sauvegarde standard, afin de pallier à la perte du logiciel et ce dernier doit être libre, pour s'assurer de sa disponibilité permanente et universelle ;
- *Règle 4 : Le logiciel de sauvegarde ne doit pas être lié à un type de support.* Suivant le contexte de la sauvegarde, le support pourra être différent et il n'est pas sain de se limiter à tel ou tel support de sauvegarde, comme pour certains logiciels dédiés aux graveurs de CD ou aux lecteurs de DAT. Un bon logiciel de sauvegarde est capable de travailler avec tous les types de supports reconnus par le système d'exploitation. En particulier, les archives créées pourront être multi-volumes, afin de s'accomoder des contraintes du support (taille de fichier de 2 Go maximum pour FAT32 ou volume limité à 650 ou 700 Mo pour un CD-R). Les noms longs, comprenant des espaces ou des caractères spéciaux doivent également êtres acceptés ;
- *Règle 5 : Le logiciel de sauvegarde ne doit pas être lié à un système d'exploitation.* Le logiciel doit exister sur toutes les plates-formes courantes, Windows, Linux, FreeBSD, les Unix courants, etc...

7.2 La solution de sauvegarde AFIO

AFIO[1] est un logiciel multi-plateforme, porté sous Windows par les soins de l'auteur, et qui permet de créer des sauvegardes standard, multi-plateformes, multi-volumes en toute fiabilité. En effet, AFIO compresse séparément chaque fichier et une archive endommagée est toujours récupérable (hormis le fichier altéré dans l'archive).

Un fichier de log est créé à l'archivage et à la restauration. Les données restaurées le sont toujours à partir du répertoire courant et il n'existe donc pas de risque d'un écrasement accidentel des

[1] Home page AFIO : http://freshmeat.net/projects/afio.

données à jour par des données archivées. Enfin, AFIO ne se bloque pas lors de l'archivage pour des fichiers inaccessibles et se contente d'inscrire une ligne d'avertissement dans le fichier de log.

La documentation détaillée d'AFIO, qui est un programme aux multiples possibilités, est disponible dans '`c:\aide\doc\afio.pdf`'.

Une limitation de la console MSys entraine un comportement dangereux en restauration : les fichiers sont restaurés par défaut à leur place d'origine.

Pour contourner cette limitation, mettez le chemin '`c:\aide\bina`' dans le PATH de Windows et utilisez AFIO à partir d'une console Windows.

Si cette limitation ne vous concerne pas, copiez simplement les fichiers présents dans '`c:\aide\bina`' vers '`c:\aide\binw`' et utilisez AFIO à partir d'une console MSys.

7.2.1 Sauvegarde

Pour effectuer une sauvegarde, il faut utiliser la commande '`backup`' :

```
USAGE : backup [chemin a archiver] [nom de l'archive] [[Volume d'un
segment (1024m)]]

Exemple : backup /c/donnees/prv prv

Produit : prv.afio.00           1er  bloc de 1 Go
          prv.afio.01           2eme bloc de 1 Go
          prv.afio.nn           Neme bloc de 1 Go
          prv.backup.afio.log  Fichier log de la sauvegarde

L'archive est toujours creee dans le repertoire courant de la console.
```

Le volume d'un segment d'une archive multi-volume est par défaut réglé à 1 Go. Cette taille peut être modifiée par le troisième paramètre, avec comme unité le Mo ou le Ko. Pour une disquette, choisir *1440k* (1.44 Mo). Pour un CD, choisir *700m* (700 Mo).

7.2.2 Archiver AIDE avec AFIO

Le provider *Free* met à disposition de ses utilisateurs un espace Web de 1 Go, mais les fichiers ne peuvent dépasser 80 Mo et la durée d'une connexion FTP ne peut dépasser 1 heure.

Donc, pour mettre à disposition AIDE, compte tenu du débit d'upload disponible, il faut l'archiver par tranches d'environ 30 Mo. Compte tenu de la fiabilité aléatoire de la connexion du serveur FTP mis à la disposition des abonnés pour l'upload des fichiers dans le site Web, il est même préférable de réduire encore ce volume...

Par exemple, pour faire une sauvegarde multi-volume de AIDE, dans '`/c/archive`', avec des volumes de 20 Mo :

```
>mkdir /c/archive
>cd /c/archive
>backup /c/aide aide 20m
```

Pour faire une sauvegarde multivolume de '`Documents and settings`' dans '`/c/sauvegarde`', avec des volumes de 1440 Ko :

```
>backup ''/c/documents and settings'' das 1440k
```

7.2.3 Restauration

Pour effectuer une restauration, il faut utiliser la commande '`restore`' :

```
USAGE : restore [nom de l'archive]
        restore nom_archive

La restauration s'effectue a partir du répertoire courant.

Si l'on sauve '/d/données/prv' dans l'archive 'prv', et que l'on
restaure 'prv' dans '/c/restaure', la restauration se fera dans
'/c/restaure/d/données/prv'.
```

Pour faire une restauration multi-volume de AIDE, à partir de '`/c/sauvegarde`' :

```
>restore aide
```

Les fichiers seront restaurés dans '`c:\sauvegarde\c\aide\...`'.

7.2.4 Fichier backup et backup.cmd

Backup script :

```
#!/bin/sh
#
# Standard cross-platform multi-volume backup with high reliability capabilities.
#
# Volume size : parameter -b of split, 1024m = 1Go, 1O24k = 1 Mo
#
# Needs       : afio, afio_find, afio_split, afio_cat, afio_gzip (or bzip2, see afio man)
#               : cygwin1.dll, cygiconv-2.dll cygintl-3.dll
#
# AFIO Windows Port : Stephane Riviere with Cygwin.

AFIO_EXE=1

if [ "$1" = "" ]; then
   AFIO_EXE=0
fi
if [ "$2" = "" ]; then
   AFIO_EXE=0
fi
if [ "$3" = "" ]; then
   AFIO_VOL=1024m
else
   AFIO_VOL=$3
fi

if [ "$AFIO_EXE" -eq 1 ]; then

   echo
   echo "Backup volume size : $AFIO_VOL"

   if [ -f "$2.backup.afio.log" ]; then
      echo "Deleting previous $2.backup.afio.log"
      rm $2.backup.afio.log
   fi
   echo afio_find "$1" | afio -o -b10k -v -Z -G6 -20 -L $2.backup.afio.log - | afio_split -b$AFIO_VOL -d
   afio_find "$1" | afio -o -b10k -v -Z -G6 -20 -L $2.backup.afio.log - | afio_split -b$AFIO_VOL -d - $2.

else

   echo
   echo "USAGE : backup [backup path] [backup name] [[volume size : 1024 Mo default]]"
   echo
   echo "Example : backup /c/datas/prv prv"
```

```
   echo
   echo "Create  : prv.afio.00           1st  1 Go volume"
   echo "          prv.afio.01           2nd  1 Go volume"
   echo "          prv.afio.nn           Nth  1 Go volume"
   echo "          prv.backup.afio.log   Backup log file"
   echo ""
   echo "The backup files are already created in the current console directory."
   echo

fi

# eof
```

Backup.cmd batch :

```
@rem  Standard cross-platform multi-volume backup with high reliability capabilities.
@rem
@rem  Volume size : parameter -b of split, 1024m = 1Go, 1024k = 1 Mo
@rem
@rem  Needs : afio, afio_find, afio_split, afio_cat, afio_gzip (or bzip2, see afio man)
@rem        : cygwin1.dll, cygiconv-2.dll cygintl-3.dll
@rem
@rem  AFIO Windows Port : Stephane Riviere with Cygwin.

@set AFIO_VOL=1024m

@if '%1'==' goto USAGE
@if '%2'==' goto USAGE
@if not '%3'==' set AFIO_VOL=%3

@echo.
@echo Backup volume size : %AFIO_VOL%

@if exist %2.backup.afio.log del %2.backup.afio.log

afio_find %1 | afio -o -b10k -v -Z -G6 -20 -L %2.backup.afio.log - | afio_split -b%AFIO_VOL% -d - %2.afio

@goto END

:USAGE
@echo.
@echo USAGE : backup [backup path] [backup name] [[volume size (1024 par defaut)]]
@echo.
@echo Example : backup /c/datas/prv prv
@echo.
@echo Create  : prv.afio.00           1st  1 Go volume
@echo           prv.afio.01           2nd  1 Go volume
@echo           prv.afio.nn           Nth  1 Go volume
@echo           prv.backup.afio.log   Backup log file
@echo.
@echo The backup files are already created in the current console directory.
@echo.

:END
```

7.2.5 Fichier restore et restore.cmd

Restore script :

```
#!/bin/sh
#
# Standard cross-platform multi-volume backup with high reliability capabilities.
#
```

```
# Volume size : parameter -b of split, 1024m = 1Go, 1024k = 1 Mo
#
# Needs       : afio, afio_find, afio_split, afio_cat, afio_gzip (or bzip2, see afio man)
#              : cygwin1.dll, cygiconv-2.dll cygintl-3.dll
#
# AFIO Windows Port : Stephane Riviere with Cygwin.

if [ "$1" = "" ]; then

   echo
   echo USAGE : restore [backup name]
   echo         restore aide
   echo
   echo Restore is already done from the current console directory.
   echo
   echo Saving '/d/datas/prv' in 'prv', and restoring 'prv'
   echo in '/c/restore', effectively restores datas in
   echo '/c/restaure/d/datas/prv'.
   echo

else

   if [ -f "$2.restore.afio.log" ]
   then
      echo "Deleting previous $2.restore.afio.log"
      rm $2.restore.afio.log
   fi

   afio_cat $1.afio.* | afio -i -b10k -v -Z -n  -L $1.restore.afio.log -

fi

# eof
```

Restore.cmd batch :

```
@rem  Standard cross-platform multi-volume backup with high reliability capabilities.
@rem
@rem  Needs         : afio, afio_find, afio_split, afio_cat, afio_gzip
@rem                : cygwin1.dll, cygiconv-2.dll cygintl-3.dll
@rem
@rem  AFIO Windows Port : Stephane Riviere with Cygwin.

@if "%1"=="" goto USAGE

@if exist %2.restore.afio.log del %2.restore.afio.log

afio_cat %1.afio.* | afio -i -b10k -v -Z -n  -L %1.restore.afio.log -
@goto END

:USAGE
@echo.
@echo USAGE : restore [backup name]
@echo         restore prv
@echo.
@echo Restore is already done from the current console directory.
@echo.
@echo Saving '/d/datas/prv' in 'prv', and restoring 'prv'
@echo in '/c/restore', effectively restores datas in
@echo '/c/restaure/d/datas/prv'.
@echo.

:END
```

7.3 Création de CD-R ou de CD-RW

Il peut être intéressant d'archiver des applications sur CD-R, pour des raisons de coûts et de souplesse[2].

Il peut être aussi opportun de graver des CD pour diffuser une application, même si internet est aujourd'hui le medium idéal.

La gravure de CD-R ou de CD-RW n'est pas disponible en standard sous Windows 2000. La procédure native sous Windows XP manque de souplesse. Les logiciels propriétaires livrés avec les graveurs posent souvent des problêmes, en particulier pour la gravure de CD à partir de fichier '`.iso`'.

Différents outils libres permettent de réaliser des gravures rapidement et en toute fiabilité. Le propos de ce chapitre est de vous familiariser avec eux.

7.3.1 La solution CDrecord

CDrecord[3] permet de graver des CD-R et des CD-RW, d'effacer rapidement (table d'allocation) ou totalement (toute la surface) les CD-RW et, d'une manière plus générale, d'effectuer toutes les opérations courantes sur ces media.

Liste des principaux utilitaires :

- '`cdrecord`' : Gravure des CD ;
- '`mkisofs`' : Génération de systèmes de fichiers hybrides ISO9660, JOLIET, HFS (extension '`.iso`' ;
- '`devdump`' : Affichage interactif du contenu d'un périphérique ou d'une image de système de fichier ;
- '`isodump`' : Affichage interactif d'une image iso9660 contenue dans un fichier '`.iso`' ;
- '`isovfy`' : Vérifie un fichier '`.iso`' ;
- '`readcd`' : Lit ou écrit un CD ;
- '`md5sum`' : Contrôle de somme pour valider des '`.iso`' téléchargés ;
- '`md5sum.md5`' : Placé dans le répertoire '`c:\aide\binw\`', afin de valider l'exécutable '`md5sum.exe`'.

Liste des utilitaires annexes :

- '`cdda2wav`' : Extraction des pistes audio d'un CD vers des '`.wav`' ;
- '`madplay`' : Lecteur '`mp3`' (et convertisseur vers d'autres formats) ;
- '`coggdec`' : Lecteur '`ogg`'.

La documentation de CDrecord au format pdf est située dans '`c:\aide\doc\cdrecord`'. D'autres informations peuvent être trouvées sur le site de cdrecord.

7.3.2 Identifier le numéro de votre graveur

Le numéro de *dev* de votre graveur peut être trouvé par la commande '`cdrecord -scanbus`'. Dans ce cas, le graveur est un graveur de DVD de marque MATSHITA et son numéro de *dev* est *1,0,0*.

```
> cdrecord -scanbus

Cdrecord-Clone 2.01 (i686-pc-cygwin) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.8'.
scsibus0:
        0,0,0     0) 'IC25N080' 'ATMR04-0        ' 'M040' Disk
```

[2] Notez à ce sujet qu'il est déconseillé, fortement déconseillé, d'utiliser des CD-RW pour cet usage.

[3] Home page de cdrecord : http://cdrecord.berlios.de/old/private/cdrecord.html.

```
            0,1,0     1) *
            0,2,0     2) *
            0,3,0     3) *
            0,4,0     4) *
            0,5,0     5) *
            0,6,0     6) *
            0,7,0     7) HOST ADAPTOR
      scsibus1:
            1,0,0   100) 'MATSHITA' 'DVD-RAM UJ-820S ' '1.00' Removable CD-ROM
            1,1,0   101) *
            1,2,0   102) *
            1,3,0   103) *
            1,4,0   104) *
            1,5,0   105) *
            1,6,0   106) *
            1,7,0   107) HOST ADAPTOR
```

7.3.3 Effacer un CD-RW

Un CD-RW peut être effacé par la commande :
'`cdrecord dev=x,y,z -v [-eject] blank=fast|all`'

L'option '`fast`' efface juste rapidement la table d'allocation du CD-RW.

L'option '`all`' efface complètement le CD-RW.

7.3.4 Créer un CD de AIDE avec CDrecord

'`mkisofs -o /c/aide.iso -J -R /c/aide`'

7.3.5 Vérifier l'image créée

'`isovfy /c/aide`'

```
> isovfy /c/aide.iso

Root at extent 20, 4096 bytes
[0 0]
[100 206]
[200 350]
No errors found
```

7.3.6 Graver le CD

'`cdrecord dev=x,y,z -v speed=16 -tao -data aide.iso`'

7.4 Informations complémentaires

7.4.1 Création d'un CD bootable Knoppix

Si vous ne disposez pas encore de GNU/Linux mais seulement de Windows, vous pouvez facilement graver un CD bootable Knoppix, qui est la distribution idéale pour découvrir Linux.

Télécharger la dernière distributions stable de Knoppix et son fichier de somme de contrôle[4]:
'`KNOPPIX_Vx.y.z-yyyy-mm-dd-EN.iso`'
'`KNOPPIX_Vx.y.z-yyyy-mm-dd-EN.iso.md5`'

À partir d'une console MSys, contrôler le fichier reçu par la commande :
'`md5sum KNOPPIX_Vx.y.z-yyyy-mm-dd-EN.iso`'

Vérifier le résultat grâce au fichier :
'`KNOPPIX_Vx.y.z-yyyy-mm-dd-EN.iso.md5`'

Gravez votre CD-R ou votre CD-RW bootable par la commande :
'`cdrecord dev=x,y,z -v speed=16 -tao -data KNOPPIX_Vx.y.z-yyyy-mm-dd-EN.iso`'

[4] Une adresse parmi d'autres : ftp://ftp.free.fr/pub/Distributions_Linux/knoppix.

7.4.2 Ecouter des mp3

Pour écouter un fichier :

```
>madplay --verbose "/z/cds/01/01. Can't We Be Friends.mp3"

MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-2004 Robert Leslie et al.
          Title: Can't We Be Friends?
         Artist: Ella Fitzgerald and Louis Armstrong
          Album:
          Track: 1
           Year:
          Genre: Jazz
        Encoder:
        Comment: Track 1
 00:00:15 Layer III, 192 kbps, 44100 Hz, joint stereo (MS), no CRC
```

Pour écouter un répertoire :

```
>madplay /z/cds/01/*.mp3
```

7.4.3 Intégration de CDrecord

Afin que Windows n'interfère pas avec CDrecord lors de l'insertion des CD dans le graveur, puis ne risque pas d'altérer le disque en cours de gravure, il est nécessaire de supprimer la détection automatique en modifiant la registry :

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Cdrom]
Autorun=0
```

Si le graveur n'est pas visible, les changements ci-dessous, à effectuer dans la registry, devraient corriger le problème :

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Aspi32]
"ErrorControl"=dword:00000001
"Start"=dword:00000001
"Type"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Aspi32\Parameters]
"ExcludeMiniports"=""
```

Une commande '`cdrecord -scanbus`' qui déclenche des anomalies est le symptôme d'un pilote ASPI défectueux.

8 Initiation au développement

Si les architectes construisaient les maisons comme les programmeurs écrivent les programmes, le premier picvert venu détruirait la civilisation. Seconde loi de Weinberg

8.1 Présentation

Ce chapitre n'a aucune prétention pédagogique. Toutefois, si vous assimilez la méthode proposée, vous en saurez déjà beaucoup plus que la plupart des *informaticiens de gestion* des années 70, grands utilisateurs de *goto* et adeptes de la programmation *spaghetti*.

Une méthode pour construire des fondations solides

Pour devenir un développeur, il faut posséder une méthode, posséder au sens de maîtriser, pour arriver, à terme, à une utilisation de la méthode comme un réflexe si bien ancré qu'on ne s'en aperçoit plus.

C'est la maîtrise d'une méthode qui distingue l'authentique développeur de l'amateur sans formation. C'est encore elle qui permet d'écrire de très gros programmes sans effort ni problème de fiabilité.

La méthode de programmation proposée est simple et efficace. Cette méthode de programmation *modulaire et structurée* permet de créer facilement et de maintenir harmonieusement des programmes de taille quelconque, du plus simple au très complexe. Elle fut très en vogue dans les années 80, avant l'apparition des méthodes orientées objet.

Cette méthode sera accompagnée d'une initiation à l'algèbre de Boole, algèbre très simple et indispensable à l'écriture de programmes bien construits.

Ce qu'il faut éviter pour débuter

La *mode* est désormais à l'objet, mis à toutes les sauces et sans discernement. En effet, les langages et les méthodes utilisant des objets, qui rajoutent toujours de la complexité dans le code créé par le compilateur, peuvent être néfastes dans certaines circonstances : développement embarqué à petit budget ou à très haute fiabilité, grandes contraintes d'occupation mémoire, etc... Cette approche objet peut être aussi complètement inutile en regard du problème à résoudre.

Il en va des méthodes comme des outils, il faut toujours utiliser la méthode appropriée au problème à résoudre. Il est des cas où l'approche objet est de loin préférable à tout autre, par exemple pour conçevoir des interfaces graphiques.

De même, la programmation évènementielle, grande affaire des interfaces graphiques, a fait apparaître une génération de développeurs qui éprouvent des difficultés à écrire une application autonome, gèrant seule ses entrées-sorties et écrite dans les règles de l'art.

Quand vous aurez assimilé la méthode proposée, nécessaire et suffisante pour faire de vous de bons développeurs tout terrain, il sera toujours temps de monter d'un cran et de regarder du côté de chez HOOD[1].

De même, quand vous serez devenu un familier de la programmation linéaire[2], c'est à dire des programmes se lisant du début à la fin, avec une continuité dans les sources, il sera aussi temps pour vous de découvrir la programmation évènementielle, par exemple avec GtkAda !

En conclusion, l'important est d'apprendre à faire les fondations avant d'apprendre à poser une toiture. Et pour bien commencer, il faut commencer modestement, et manipuler des concepts simples.

---

[1] Se reporter à la bibliographie en fin de manuel.

[2] Linéaire, par opposition à évènementielle, dans ce contexte.

8.2 Débuter en Ada

8.2.1 Livre compagnon

Si vous êtes déjà un développeur, la documentation associée vous suffira. Par contre si vous débutez en programmation, un livre compagnon vous est proposé (à l'inverse de l'usage, où c'est plutôt le CD qui est le compagnon du livre ;-).

C'est un livre recommandé pour les débutants adultes , dans la collection *Reprendre ses études*. Il se nomme *Initiation et auto-formation à la programmation informatique* de *Canesi et Suc*, aux éditions *Ellipses*.

C'est un livre de prix modique (20 EUR) et aux qualités pratiques exceptionnelles (les auteurs sont professeurs au CNAM) pour convaincre un péri-développeur, genre administrateur système qu'il est capable de devenir un véritable développeur.

Ce n'est pas un livre permettant d'acquérir une méthode, ni un cours Ada, c'est uniquement, et c'est sa grande qualité, un livre destiné à faire le premier pas, celui qui est le plus difficile.

Quoique tous les exemples soient en Ada, Initiation et auto-formation à la programmation informatique donne aussi quelques éléments de comparaison en évoquant les autres langages courants, avec quelques listings, comparaison qui tourne naturellement à l'avantage de Ada, car le lecteur néophyte préfèra instinctivement un code naturellement lisible et expressif à un code cryptique et inexpressif.

Initiation et auto-formation à la programmation informatique est disponible en 48h sur http://www.eyrolles.com (où il a été commandé), mais n'importe quel autre site en ligne fera aussi bien l'affaire.

8.2.2 Exercices pour faire le premier pas

Toujours pour faire le premier pas, avant de poursuivre avec l'apprentissage d'une méthode d'analyse, des exemples en Ada vous sont proposés, afin de débuter et de se familiariser avec la syntaxe.

Ils proviennet du CNAM[3], plus précisément de l'équipe de l'enseignant-chercheur François Barthélemy email : barthe@cnam.fr. De plus amples informations sont disponibles sur http://lmi17.cnam.fr/~barthe/FB/index.html. La totalité du cours Ada, bien conçu pour les débutants, est ici : http://lmi17.cnam.fr/~barthe/APA_EAD.

Suivant notre propre expérience, l'indentation et la présentation des programmes a été sensiblement améliorée et certains programmes ont été ajoutés, tels '`bonjter.adb`'.

Comment écrire et tester les exercices ?

Reportez vous aux sections "test de l'installation" des chapitres 3, 4 & 5, suivant l'environnement que vous avez adopté, afin de lire un exemple complet de saisie d'un programme.

Bonne découverte de AIDE !

[3] Selon http://lmi17.cnam.fr/~barthe/APA_EAD/common/apropos.html, tous les documents du site web de l'UV Algorithmique-Programmation A1 sont sous copyright du CNAM et de l'auteur. Leur reproduction sans modification est autorisée pour un usage non commercial. Tout autre utilisation est soumise à l'accord de l'auteur.

Programme 1

Écrire un programme qui affiche bonjour.

```
-- bonj.adb

with ADA.TEXT_IO;  -- necessaire pour faire put (affichage) et get (saisie) ..
                   -- des chaines de caractères
procedure bonj is

begin

  ADA.TEXT_IO.put("Bonjour");
  ADA.TEXT_IO.new_line; -- retour à la ligne

end bonj;
```

Une variante avec l'utilisation de `use`.

```
-- bonjbis.adb

with ADA.TEXT_IO;  -- necessaire pour faire put (affichage) et get (saisie) ...
                   -- des chaines de caractères
use ADA.TEXT_IO;   -- evite de mettre ADA.TEXT_IO. avant put, get, ...

procedure bonjbis is

begin

  put("Bonjour");
  new_line; -- retour à la ligne

end bonjbis;
```

Une autre variante avec l'utilisation de `renames`.

```
-- bonjter.adb

with ADA.TEXT_IO;  -- necessaire pour faire put (affichage) et get (saisie) ...
                   -- des chaines de caractères

procedure bonjter is

package tio renames ADA.TEXT_IO;  -- permet d'ameliorer la lisibilite
                                  -- en gardant le nom du package lie
                                  -- aux procedures ou au fonctions
begin

  tio.put("Bonjour");
  tio.new_line; -- retour à la ligne

end bonjter;
```

Programme 2

Saisie et affichage d'entiers. Utilisation du `if then else endif` : Écrire un programme qui demande deux nombres entiers à l'utilisateur et affiche le plus grand des deux.

```
with ADA.TEXT_IO; use ADA.TEXT_IO;
with ADA.INTEGER_TEXT_IO; use ADA.INTEGER_TEXT_IO;

procedure compare is

a, b : integer; -- declaration des variables

begin

  put_line("Bonjour, je peux vous calculer le plus grand de deux nombres ");
  put("Entrez le premier nombre : ");
  get(a);

  put("Entrez le second nombre : ");
  get(b);

  put("Le plus grand est ");

  if (a >= b) then
     put(a);

  else
     put(b);

  end if;

  new_line;

end compare;
```

Programme 3

Saisie et affichage de réels : Écrire un programme qui demande deux nombres réels à l'utilisateur et affiche le plus grand des deux.

```
-- compare_reel.adb

with ADA.TEXT_IO; use ADA.TEXT_IO;
with ADA.FLOAT_TEXT_IO; use ADA.FLOAT_TEXT_IO; -- pour saisie et affichage de reels

procedure compare_reel is

a, b : float; -- declaration des variables

begin

  put_line("Bonjour, je peux vous calculer le plus grand de deux nombres ");
  put("Entrez le premier nombre reel : ");
  get(a);

  put("Entrez le second nombre reel : ");
  get(b);

  put("Le plus grand est ");

  if (a >= b) then
     put(a, fore => 3, aft => 1, exp => 0);

  else
     put(b, fore => 3, aft => 1, exp => 0);

  end if;

  new_line;

end compare_reel;
```

Programme 4

Saisie et affichage sur un type énuméré : Écrire un programme qui demande un jour de la semaine à l'utilisateur sous la forme Lundi, Mardi, Mercredi, ...ou Dimanche. Le programme doit ensuite afficher le jour rentré par l'utilisateur et son lendemain.

```
-- jours.adb

with ADA.TEXT_IO; use ADA.TEXT_IO;

procedure jours is

type tjour is (Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche);

-- pour faire des put et des get sur le type tjour, voici ce qu'il faut faire

package tjour_io is new ADA.TEXT_IO.ENUMERATION_IO(tjour); use tjour_io;

jour : tjour;

begin

  put("entrez un jour : ");   -- put qui vient de ADA.TEXT_IO
  get(jour);                  -- get qui vient de tjour_io

  -- on affiche ici le jour rentre par l'utilisateur
  put(jour);                  -- put qui vient de tjour_io
  new_line;

  -- et ici le lendemain
  put("Le lendemain est : ");

  if jour = Dimanche then
     put(Lundi);

  else
     put(tjour'succ(jour));

  end if;

  new_line;

end jours;
```

Programme 5

Exercice sur les boucles : écrire un programme qui affiche :

```
1 2 3
4 5 6
7 8 9
```

```
-- aff_carre.adb

with ADA.TEXT_IO;
use ADA.TEXT_IO;

procedure aff_carre is

begin

  for i in 1..9 loop

      put(integer'image(i));

      if (i rem 3) = 0 then
         new_line;
      end if;

  end loop;

end aff_carre;
```

Programme 6

Exercice sur les boucles et les conversions de types : Écrire un programme qui affiche la correspondance entre livres et kilogrammes pour les nombres allant de 1 à 10 :

```
1 livre(s) = 0.45 Kg
2 livre(s) = 0.91 Kg
etc...
```

Indication : Float(n) convertit l'entier n en le nombre réel correspondant.

```
-- conversion.adb

with ADA.TEXT_IO, ADA.FLOAT_TEXT_IO;
use ADA.TEXT_IO, ADA.FLOAT_TEXT_IO;

procedure conversion is

taux : constant float := 0.45;

begin

  for i in 1..10 loop

      put (integer'image(i) & " livre(s)  = ");
      put (Float(i)* taux, fore => 2, aft => 2, exp =>0);
      put (" Kg");

      new_line;

  end loop;

end conversion;
```

Programme 7

Exercice sur les boucles : écrire un programme qui affiche la table de multiplication de 8

```
-- mult_8.adb

with ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;
use ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;

procedure mult_8 is

mult : constant integer := 8;

begin

  put ("Voici la table de multiplication de ");
  put(mult);
  new_line;

  for i in 1..9 loop
      put (mult); put (" x "); put(i); put ( " = " );
      put (mult * i);

      new_line;

  end loop;

end mult_8;
```

Programme 8

Exercice sur les boucles : Écrire un programme qui affiche les tables de multiplication des nombres allant de 2 à 9.

```
-- mult_tout.adb

with ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;
use ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;

procedure mult_tout is

begin

  for mult in 2..9 loop

      put ("Voici la table de multiplication de ");
      put(mult);
      new_line;

      for i in 1..9 loop

          put (mult);
          put (" x ");
          put(i);
          put ( " = " );
          put (mult * i);

          new_line;

      end loop;

      put ("-------------------"); new_line;

  end loop;

end mult_tout;
```

Programme 9

Exercice sur le type caractère : écrire un programme qui affiche les caractères représentés par les nombres de 32 à 126.

Indication : `character'val(n)` donne le caractère dont le code ASCII est le nombre n.

```
-- car_32_126.adb

with ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;
use ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;

procedure car_32_126 is

begin

  for n in 32..126 loop
      put ("Le caractere dont le code ASCII est ");
      put (n, width => 3);
      put_line (" est " & character'val (n));

  end loop;

end car_32_126;
```

Programme 10

Exercice sur l'alternative `case` : écrire un programme qui demande un caractère à l'utilisateur et affiche le type de ce caractère (lettre, chiffre, caractère de ponctuation ou autre).

```
-- car_tri.adb

with ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;
use ADA.TEXT_IO, ADA.INTEGER_TEXT_IO;

procedure car_tri is

c : character;

begin

 loop

    put ("Entrez un caractere (* pour finir) : ");
    get (c);

    exit when c='*';

    put ("Ce caractere est " );

    case c is
         when 'a'..'z'|'A'..'Z'        =>
              put_line ("une lettre");

         when '1'..'9'                 =>
              put_line ("un chiffre");

         when '.'|';'|','|'?'|'!'|':'  =>
              put_line ("un carac. de ponctuation");

         when others                   =>
              put_line ("un carac. non classe");

    end case;

    new_line;

  end loop;

end car_tri;
```

Programme 11

Exercice sur la saisie, l'affichage et l'utilisation de chaînes de caractères : écrire un programme qui demande la saisie d'une chaîne de 40 caractères au maximum, puis l'affiche et affiche la chaîne *miroir*.

```
-- chaine.adb

with ADA.TEXT_IO; use ADA.TEXT_IO;

procedure chaine is

taille : constant integer := 40;
ch : string(1..taille);

long : integer; --longueur de chaine

x : character;

begin

  put("Donnez la chaine : ");
  get_line (ch, long);   -- après l'appel, long contient le nombre de
                         -- caractères effectivement saisis

  put("Voici votre chaine : ");
  put_line (ch(1..long));  -- uniquement les caractères utiles

  -- on inverse la chaine

  for i in 1 .. (1+long)/ 2 loop

      -- on echange les contenus des cases i et long+1-i
      x := ch(i);
      ch(i) := ch(long+1-i);
      ch(long+1-i):= x;

  end loop;

  put("Voici la chaine miroir : ");
  put_line (ch(1..long));

end chaine;
```

Programme 12

Exercice sur les tableaux : écrire un programme qui déclare un tableau de 5 entiers contenant les valeurs 10, 20, 30, 40 et 50 puis calcule et affiche la somme des élèments de ce tableau.

```
-- somme_tab.adb

with ADA.TEXT_IO; use ADA.TEXT_IO;
with ADA.INTEGER_TEXT_IO; use ADA.INTEGER_TEXT_IO;

procedure somme_tab is

-- declaration des variables
tab : array (1..5) of integer := (10,20,30,40,50);
S : integer := 0;

begin
  for i in tab'range loop
      S := S + tab(i);
  end loop;

  put (" La somme des elements du tableau est : ");
  put (S);
  new_line;

end somme_tab;
```

8.3 Débuter en conception

8.3.1 Inventaire des outils

Pour concevoir un programme, il faut :

- Posséder une méthode d'analyse ;
- Connaître quelques rudiments d'algèbre de boole ;
- Maîtriser un langage.
- Avoir une bonne culture générale et du *savoir faire*, c'est à dire posséder les trucs & astuces du métier ;

De ces quatre éléments, le premier est le plus difficile à acquérir, mais j'espère que les lignes qui vont suivre vont vous aider dans ce domaine.

J'aurais pu rajouter *du papier, un crayon et une gomme*, tant il est vrai que ces trois objets sont toujours à la base d'un bon programme et qu'il ne faut surtout pas se précipiter à coder ce qui vous passe par la tête.

Vous remarquerez que la connaissance d'un langage vient après la théorie. C'est normal. Comme l'analyse précède l'écriture, la maîtrise de la conception précède la maîtrise d'un langage.

Toutefois, la joie de programmer doit rester le moteur de votre motivation, c'est pour cette raison que nous avons commencé ce chapitre par quelques programmes très simples : pour vous donner envie de poursuivre !

8.3.2 Un premier exemple

Tout le monde programme, le garagiste, le facteur et le cuisinier. Vous ne le saviez pas ? Alors commençons donc par nous faire cuire un œuf !

Maîtriser une méthode d'analyse permet d'analyser un problème, même très complexe, et de le décomposer, *par raffinements successifs*, en une somme de problèmes un à un si évidents à résoudre que l'on arrête alors l'analyse en déclarant qu'elle est terminée !

Donc, nous allons nous faire cuire un œuf, un œuf dur pour être précis. Mais pourriez-vous détailler sans hésitation un tel processus, en apparence très simple ? Voyons cela ensemble.

La décomposition du problème

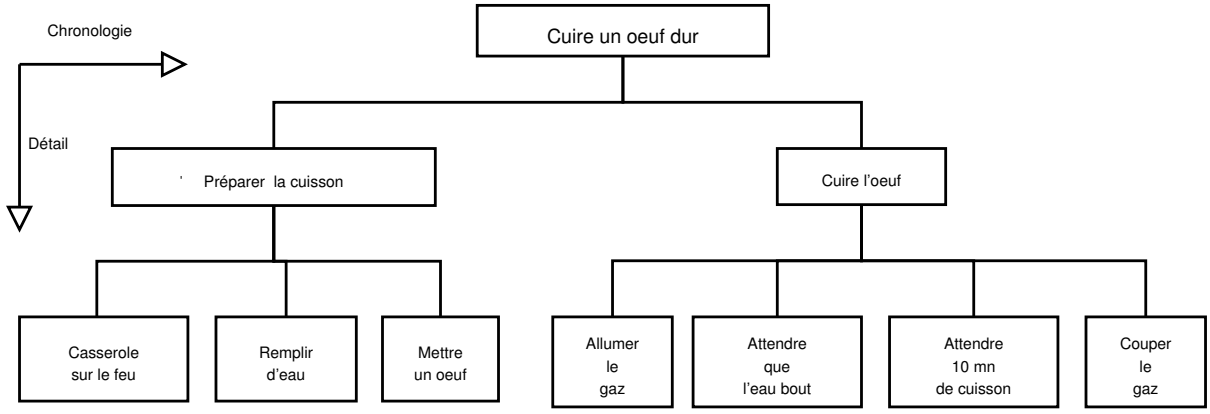
On pourrait, par exemple, commencer par décomposer, *par raffiner*, l'action de cuire un œuf en deux grandes étapes : la *préparation* et la *cuisson*.

Puis on pourrait reprendre ces deux grandes étapes en les raffinant à leur tour :

- La *préparation*, c'est poser la casserole sur le feu, la remplir d'eau et mettre un œuf dedans ;
- La *cuisson*, c'est allumer le gaz, attendre que l'eau se mette à bouillir, attendre 10 mn pour la cuisson[1], puis couper le gaz.

[1] C'est beaucoup, mais ça n'est pas gênant, sauf si vous les aimez mollets.

Cette démarche est connue sous l'appellation de *décomposition par raffinements successifs*. Cette décomposition achevée, il est indispensable de la représenter visuellement, grâce au DSP, le *Diagramme Structuré de Programme*.



Ce DSP est très pratique car il fonctionne en deux dimensions :

- Dans le plan vertical, on descend du plus complexe au plus simple ;
- Dans le plan horizontal, le sens de la lecture représente naturellement, *chronologiquement*, les tâches à effectuer.

Et il est à double usage :

- Dans un premier temps, il vous permet d'acquérir une *vue d'ensemble* sur le problème à résoudre et de vous assurer que votre analyse est cohérente et complète ;
- Dans un deuxième temps, puisque chaque boîte représente une action si simple à résoudre qu'elle ne requiert pas d'analyse supplémentaire, le DSP vous permet de passer directement à la seconde phase : *le pseudo-code* !

En créant ce DSP, nous avons *modularisé* notre problème. C'est à dire que nous avons décomposé notre problème en une suite de *modules* élémentaires. En écrivant le pseudo-code, nous allons décrire le fonctionnement de chaque *module* en utilisant des *structures*. Ces *structures* forment les briques de base de la programmation *structurée*, sans *goto*, ni code *spaghetti*.

Le pseudo-code

Le pseudo-code est la traduction informatique, sous formes de structures, du DSP déjà écrit. Le DSP et le pseudo-code sont liés. Ils doivent être en cohérence l'un envers l'autre et c'est souvent en contrôlant cette cohérence, au moment d'écrire le pseudo-code, que l'on s'aperçoit que le niveau de détail du DSP est incorrect. S'il est trop élevé, le pseudo-code contient des modules inutiles, qui ne contiennent pas de traitement méritant d'être modularisé. Par contre, si le niveau de détail n'est pas assez élevé, le pseudo-conde contient des modules beaucoup trop chargés.

Avant d'entrer dans le détail de l'écriture générale d'un pseudo code, voyons un petit exemple, avec notre œuf dur, pour se mettre en bouche justement.

```
debut *** cuire un oeuf ***

      faire *** préparer la cuisson ***
      faire *** cuire l'oeuf ***

fin   *** cuire un oeuf ***
```

Dans ce premier pseudo-code, qui représente le module principal du programme *cuire un œuf*, l'analogie entre DSP et pseudo-code apparait clairement. Le terme *faire* devant *préparer la cuisson* représente l'appel au module *préparer la cuisson* et chaque module commence par *debut 'nom du module'* et finit par *fin 'nom du module'*.

Passons à l'écriture du module *préparer la cuisson*.

```
debut *** préparer la cuisson ***

      faire *** poser la casserole sur le feu ***

      tant que "casserole n'est pas remplie"
               remplir d'eau
      fin tant que

      faire *** mettre un oeuf ***

fin   *** préparer la cuisson ***
```

C'est alors qu'un problème apparaît. Il apparaît que le module *poser la casserole sur le feu* représente une action très simple. Elle est si simple qu'elle ne mérite pas, en fait, d'être isolée dans un module. Laisser l'analyse telle que, sans rien changer, aboutirait à rendre le programme beaucoup plus complexe qu'il ne le mérite.

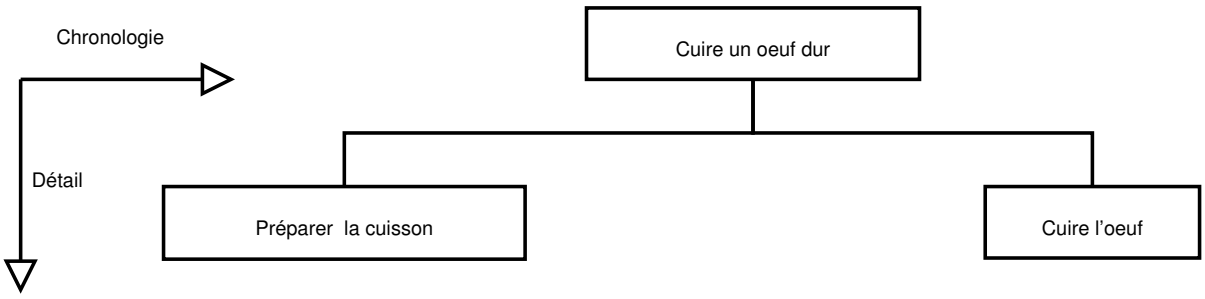
Nous allons donc simplifier le pseudo-code.

```
debut *** preparer la cuisson ***

      poser la casserole sur le feu

      tant que "casserole n'est pas remplie"
               remplir d'eau
      fin tant que

      mettre un oeuf

fin   *** preparer la cuisson ***
```

Il est donc apparu que le niveau de détail du DSP était trop élevé. Les actions du dernier rang : *casserole sur le feu*, *remplir d'eau*, *etc...* ne méritaient pas, à elles seules, un module séparé.

Il faut les regrouper dans les modules de rang supérieur : *préparer la cuisson* et *cuire l'œuf*.

Le DSP devient donc :



L'analyse de la cuisson de l'œuf se termine avec le pseudo-code du dernier module :

```
debut *** cuire l'oeuf ***

      allumer le gaz

      tant que "l'eau ne bout pas"
               attendre
      fin tant que

      tant que "pas 10 minutes écoulées
               attendre
      fin tant que

      couper le gaz

fin   *** cuire l'oeuf ***
```

Après cet exemple d'introduction, nous allons examiner en détail cette méthode d'analyse.

8.4 Programmation Générale, Modulaire et Structurée

Cette méthode s'appelle PGMS. On retrouve cette démarche dans des dizaines d'autres méthodes, se différentiant essentiellement par leurs signes, leurs vocabulaires, l'esthétique des dessins de diagrammes, etc...

L'acronyme PGMS trouve son origine dans l'enseignement dispensé par le regretté Institut Control Data[2], installé dans les années 80 à *Chinatown*, dans le XIIIème arrondissement de Paris.

8.4.1 Le DSP

Écrire un DSP, *Diagramme Structuré de Programme*, c'est identifier, décomposer et hiérarchiser des fonctions dans un ensemble cohérent, afin de permettre l'écrire du pseudo-code du programme.

Détail de la démarche

La démarche de création du DSP est un procédé itératif, *qui boucle sur lui-même*, pour identifier toutes les tâches à effectuer, jusqu'à épuisement des possibilités de raffinement, c'est à dire jusqu'à ce qu'on ne puisse plus détailler le problème à résoudre.

Cette démarche est appellée démarche *top-down*, afin de bien montrer que l'on part du problème global, *en haut du diagramme*, pour aller vers le plus petit détail, *vers le bas du diagramme*.

Á chaque fois que l'on rajoute un niveau de détail, on créé une nouvelle ligne.

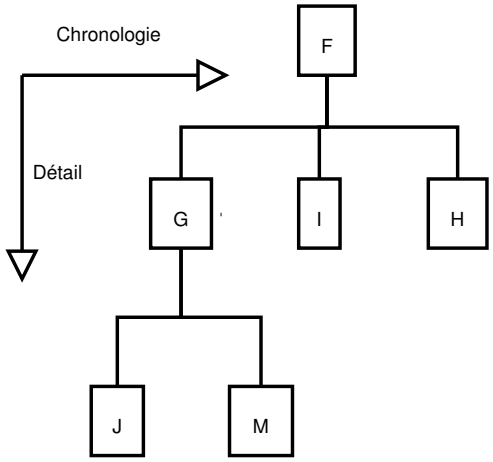
Pour chaque niveau de détail, les tâches identifiées sont écrites dans le sens de lecture, par ordre d'exécution. Elles sont placées dans des *boîtes*. Pour la clarté du DSP, toutes les boîtes d'un rang inférieur sont reliées par des traits à la boîte de rang supérieur.

Les DSP s'écrivent et se lisent toujours :

- De haut en bas, pour le niveau de détail ;
- De gauche à droite, pour les étapes chronologiques.

[2] Aussi connu sous l'acronyme CDI, pour *Control Data Institute*. L'auteur n'a pas gardé le manuel traitant de PGMS et le regrette bien aujourd'hui.

Exemple :



En aucun cas un DSP ne fait apparaître les tests et autre actions de bas niveau qui sont du ressort de la programmation.

Un DSP est à la fois la vue d'ensemble et l'ossature de l'analyse.

L'écriture du DSP est la partie la plus difficile de l'analyse.

8.4.2 Le pseudo-code

L'écriture du pseudo-code est effectuée à partir du DSP. À chaque *boîte* du DSP correspondra un *module* dans le pseudo-code.

L'écriture d'un pseudo code s'effecture à partir de briques élémentaires, que nous allons examiner dès maintenant.

Le module maître

Un programme commence et se termine au module maître.

Voici le pseudo-code, aussi appelé *PC*, du DSP précédent, décrivant le module principal du programme `F` :

```
debut *** F ***

      faire *** G ***

      tant que P (est vrai)
           faire *** I ***
      fin tant que

      faire *** H ***

fin   *** nom du programme ***
```

Le début d'un module est représenté par `*** debut 'nom du module' ***` et la fin d'un module est représenté par `*** fin 'nom du module' ***`.

Le nom du module principal reprend le nom du programme.

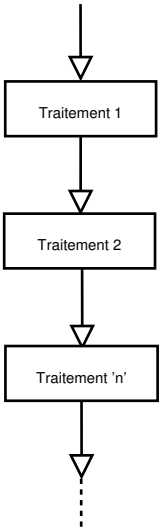
Les autres modules

Les autre modules sont écrits de la même manière. Voici le pseudo-code du module `G` du DSP précédent, décrivant le programme `F` :

```
debut *** G ***

      faire *** J ***

      si Q (est vrai)
         faire *** M ***

      finsi

fin   *** G ***
```

La séquence

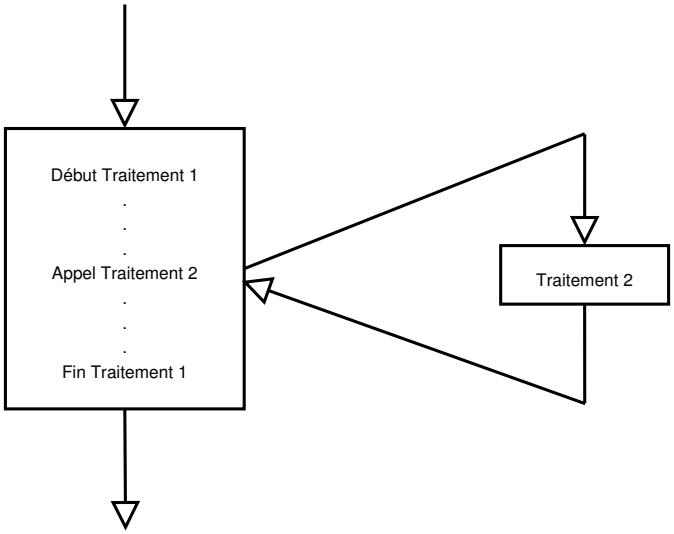
La séquence est la forme la plus simple d'un pseudo-code. Elle représente juste l'enchaînement de plusieurs traitements, qui s'exécutent les uns après les autres.



L'appel à un module

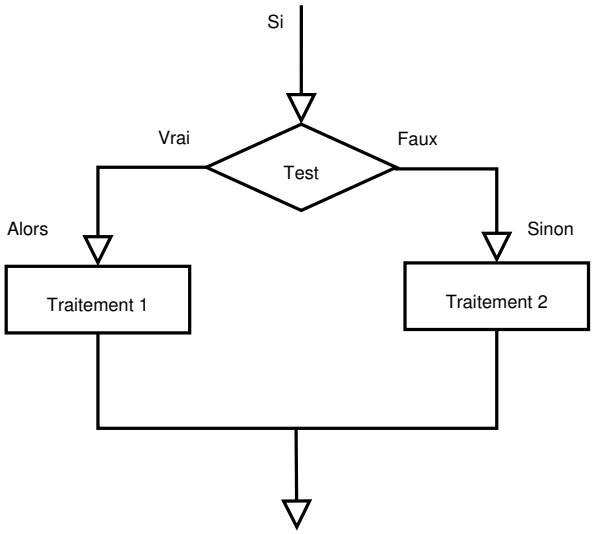
L'appel à un module est représenté par `faire *** 'nom du module' ***`. Les traitements du module appelant s'arrêtent à la ligne de l'appel et le module appelé s'exécute.

Á la fin du module appelé, ce dernier rend la main au module appelant et l'exécution de ce dernier reprend à la ligne suivant l'appel qui vient d'être exécuté.



Le si...sinon...finsi

L'alternative est le test le plus simple d'un pseudo-code. En fonction de la véracité de la condition de test, le flux du programme est aiguillé vers un traitement ou un autre.



L'alternative est représentée ainsi dans un pseudo-code :

```
si condition de test (est vraie)
   Traitement 1

sinon
   Traitement 2

finsi
```

Le si...sinonsi...sinon...finsi

Cette structure est une extension de l'alternative :

```
si      condition de test 1
        Traitement 1

sinonsi condition de test 2
        Traitement 2

sinonsi condition de test 3
        Traitement 3
            .
            .
            .

sinonsi condition de test 'n'
        Traitement 'n'

sinon
        Traitement par défaut

finsi
```

Le traitement par défaut est exécuté quand aucune condition de test n'a été vérifiée.

Cette structure est équivalente à une imbrication d'alternatives. Mais ces imbrications sont beaucoup moins lisibles, comme le montre l'exemple ci-dessous :

```
si condition de test 1
        Traitement 1

   sinon
        si condition de test 2
           Traitement 2

        sinon
             si condition de test 3
                Traitement 3

             sinon
                Traitement par défaut

             finsi
        finsi
   finsi
finsi
```

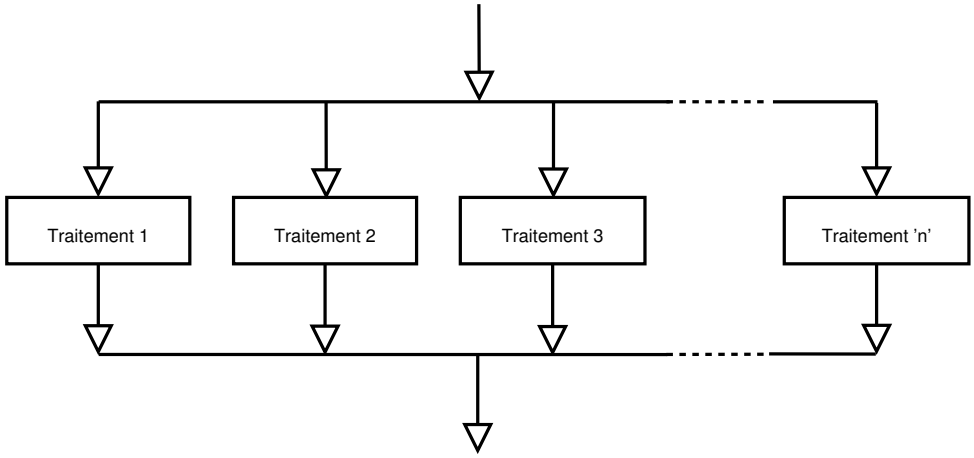
La sélection...quand...sinon...fin sélection

La sélection est une forme différente de l'alternative car le test n'est plus booléen (vrai ou faux) mais fonction du contenu de la valeur testée.

Un pseudo-code étant plus parlant :

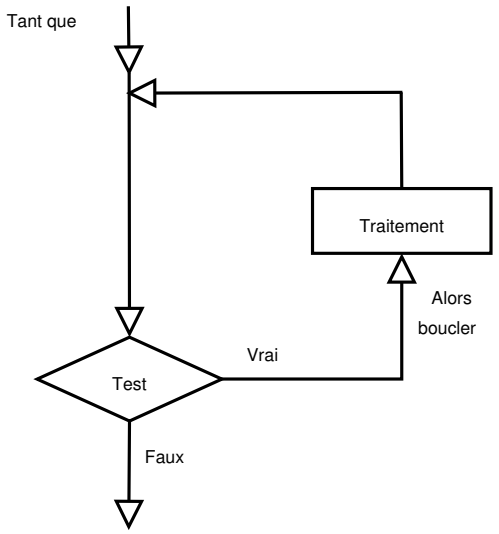
```
selection valeur à tester

          quand valeur 1
                Traitement 1

          quand valeur 2
                Traitement 2

          quand valeur 3
                Traitement 3
                   .
                   .
                   .

          quand valeur 'n'
                Traitement 'n'

          quand autres
                Traitement par défaut

fin selection
```

Cette structure peut se repésenter ainsi :



Le tant que...fin tant que

Cette structure de boucle est à employer quand on souhaite que le flux du programme évite le traitement dans la boucle si la condition est fausse au premier passage dans la boucle.



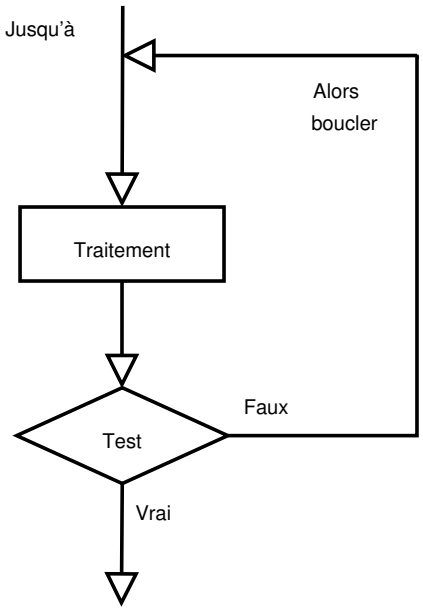
Le pseudo code d'une telle structure est le suivant :

```
tant que condition de test
   Traitement

fin tant que
```

Le boucler... jusqu'à

Cette structure de boucle diffère de la précédente car le traitement dans la boucle est effectué une fois *avant* que la condition de boucle soit testée.

De la sorte, on passera toujours au moins une fois dans ce type de boucle.



Voici la notation du *boucler...jusqu'à* en pseudo-code :

```
boucler
   Traitement

jusqu'a condition de test
```

Il apparait nettement que le test est effectué *après* un premier passage dans la boucle.

8.5 Algèbre de Boole

Voici un résumé pratique concernant l'algèbre de Boole, qui devrait être connue par tous les développeurs.

8.5.1 Identités, propriétés et lois de De Morgan

Deux conventions sont utilisées :

- $\equiv$ pour l'équivalence. A $\equiv$ B signifie que A et B sont deux conditions équivalentes et qu'elles sont interchangeables ;
- NON A pour la négation de A. Si A est vrai, NON A est faux.

Identités

```
A OU 0 ≡ A          NON (NON A) ≡ A
A OU 1 ≡ A          A OU A ≡ A
A OU (NON A) ≡ 1    A ET A ≡ A
A ET (NON A) ≡ 0
```

Propriétés

```
A ET B ≡ B ET A
A OU B ≡ B OU A
A OU (B OU C) ≡ (A OU B) OU C
A ET (B ET C) ≡ (A ET B) ET C
A OU (B ET C) ≡ (A OU B) ET (A OU C)
A ET (B OU C) ≡ (A ET B) OU (A ET C)
```

Lois de De Morgan

```
NON (A OU B) ≡ (NON A) ET (NON B)
NON (A ET B) ≡ (NON A) OU (NON B)
```

8.5.2 Conseils pratiques

Dans la description de votre langage courant, vous trouverez certainement la description des priorités dans l'évaluation des expressions logiques.

Voici un exemple de priorités d'évaluation :

1. Expression situées dans les parenthèses les plus internes ;
2. Négation ;
3. ET et OU (Dans le langage Ada, ces deux opérateurs sont sur un pied d'égalité, ce qui n'est pas la règle générale dans d'autres langages où *ET* a généralement une priorité supérieure à *OU*.) ;
4. À priorité égale, évaluation des expressions de la gauche vers la droite.

On pourrait être tenté de tenir compte de ces priorités pour rédiger une condition de test la plus courte possible mais c'est à éviter absolument pour des raisons de clarté.

Voici trois règles de base à suivre en toute circonstance :

1. Il faut pas hésiter à recourir aux parenthèses pour accroître la lisibilité et la fiabilité.
2. Pour travailler ou inverser une condition complexe, il faut, avant toute chose rétablir les parenthèses implicites.
3. Une simplification de condition complexe se fait en appliquant les lois de De Morgan.

8.6 Rédaction des programmes

Fichiers sources

Un programme est constitué essentiellement de fichiers sources. Ces derniers doivent tous être rédigiés avec les mêmes conventions d'écriture.

Chaque source doit contenir un en-tête d'identification et une marque de fin de fichier. Ils doivent occuper une bonne partie de la largeur du source, afin d'être facilement visualisables.

Identifiants

Les identifiants doivent être parlants. Il est d'usage en Ada de séparer les éléments d'une variable composée par un souligné. Exemple : `serial_line_rx`. D'autres préfèrent capitaliser : `Serial_Line_Rx`. Quoique l'amélioration de la lisibilité ne soit pas évidente, la plupart des sources Ada

sont présentés ainsi. La vitesse de frappe peut s'en trouver ralentie, mais les bons éditeurs, tels Glide ou GPS, peuvent reformater à la volée selon cette convention.

Indentation

Un programme lisible est un programme aéré (indentation verticale et horizontale). Les exemples ci-dessous vont du moins lisible au plus lisible.

Absence d'indentation : dans ce premier exemple, l'absence complète d'indentation rend la lecture particulièrement difficile.

```
début "identifier l'opérateur"
si opérateur = "+"
résultat = premier nombre + second nombre
sinonsi opérateur = "-"
résultat = premier nombre - second nombre
sinonsi opérateur = "*"
résultat = premier nombre * second nombre
sinonsi opérateur = "/"
résultat = premier nombre / second nombre
finsi
fin "identifier l'opérateur"
```

Indentation verticale seulement : dans ce second exemple, l'ajout de lignes vides rend la lecture plus aisée.

```
début "identifier l'opérateur"

si opérateur = "+"
résultat = premier nombre + second nombre

sinonsi opérateur = "-"
résultat = premier nombre - second nombre

sinonsi opérateur = "*"
résultat = premier nombre * second nombre

sinonsi opérateur = "/"
résultat = premier nombre / second nombre

finsi

fin "identifier l'opérateur"
```

Indentation horizontale seulement : dans ce exemple, la lecture n'est pas vraiment améliorée, mais le décalage horizontal des lignes permet de mieux voir la structure du programme.

```
début "identifier l'opérateur"
      si      opérateur = "+"
              résultat = premier nombre + second nombre
      sinonsi opérateur = "-"
              résultat = premier nombre - second nombre
      sinonsi opérateur = "*"
              résultat = premier nombre * second nombre
      sinonsi opérateur = "/"
              résultat = premier nombre / second nombre
      finsi
fin   "identifier l'opérateur"
```

Indentation horizontale et verticale : comparez le premier et le dernier exemple de cette section consacrée à l'indentation, l'amélioration de la lecture est évidente. Imaginez vos yeux après une journée de travail sur des sources non ou mal indentés !

```
début "identifier l'opérateur"

      si      opérateur = "+"
              résultat = premier nombre + second nombre

      sinonsi opérateur = "-"
              résultat = premier nombre - second nombre

      sinonsi opérateur = "*"
              résultat = premier nombre * second nombre

      sinonsi opérateur = "/"
              résultat = premier nombre / second nombre

      finsi

fin   "identifier l'opérateur"
```

8.7 Développement portable

Un des grands avantages de Ada est de permettre l'écriture, sans grand effort, de programme réellement portables.

L'exemple donné est très simple, mais le principe reste valable pour des cas bien plus complexes. Ici, il s'agit d'un programme en mode texte qui doit pouvoir fonctionner aussi bien sous Windows que sous Linux.

Le problème

Le problème est le suivant : dès que l'on souhaite écrire à un endroit précis de l'écran, la gestion d'écran en mode texte, commune aux systèmes Unix et Ms-Dos, via les commandes *ANSI*, s'avère malheureusement complètement différente sous Windows.

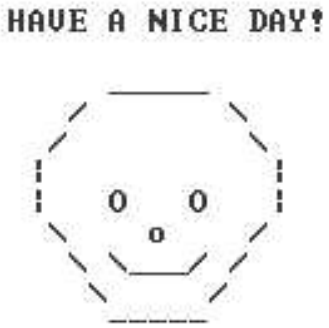
On ne peut plus écrire un code source unique pour tous les systèmes. Heureusement, Ada va nous sortir de cet écueil.

La solution

La solution consiste à créer un paquetage *screen*, dont la spécification est commune '`screen.ads`' mais dont les corps '`screen.adb`', adaptés pour chaque système d'exploitation, sont placés dans deux répertoires distincts : '`linux`' et '`windows`'.

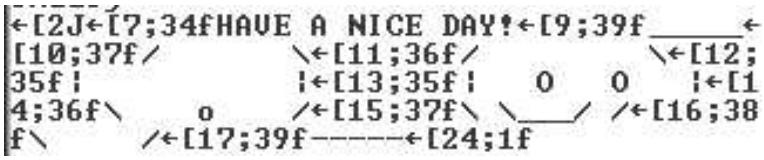
```
* = racine du répertoire contenant le programme,
    'c:\aide\examples\aide\crossplatform\prg01'

*---linux
|   |
|   \---screen.adb
|
+---windows
|   |
|   \---screen.adb
|
+---screen.ads
\---smiley.adb
```

Dès lors, pour compiler le programme '`smiley.adb`' pour Windows, il suffira de taper, dans une console MSys : `gnatmake smiley -aIwindows`, pour obtenir :



Pour vérifier, on peut aussi compiler la version Linux *sous windows* par la commande : `gnatmake smiley -aIlinux`. Ne pas oublier d'effacer au préalable les fichiers '`.o`', '`.ali`' et '`.Exe`'.

On obtient alors un affichage complètement incompréhensible :



9 Exemples de programmes

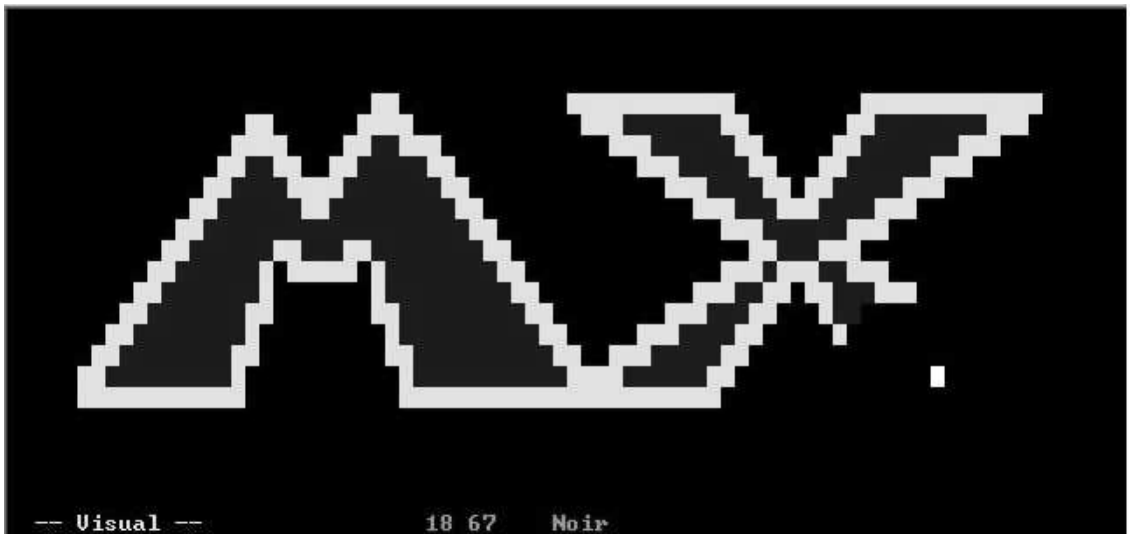
Doubler le nombre de programmeurs sur un projet en retard ne fait que doubler le retard. Loi de Brook

9.1 Visual

Visual a été développé par Martin, 13 ans en 2004, alors qu'il avait découvert la programmation et Ada cinq mois auparavant.

Visual est un éditeur d'écran en mode texte. Les images créées peuvent être sauvées dans des fichiers d'image écran avec l'extension '`.vsl`'. Ces fichiers sont directement utilisables comme des ressources externes par des applications tierces.

Les sources de Visual sont disponibles dans '`c:\aide\examples\aide\projects\visual`'.



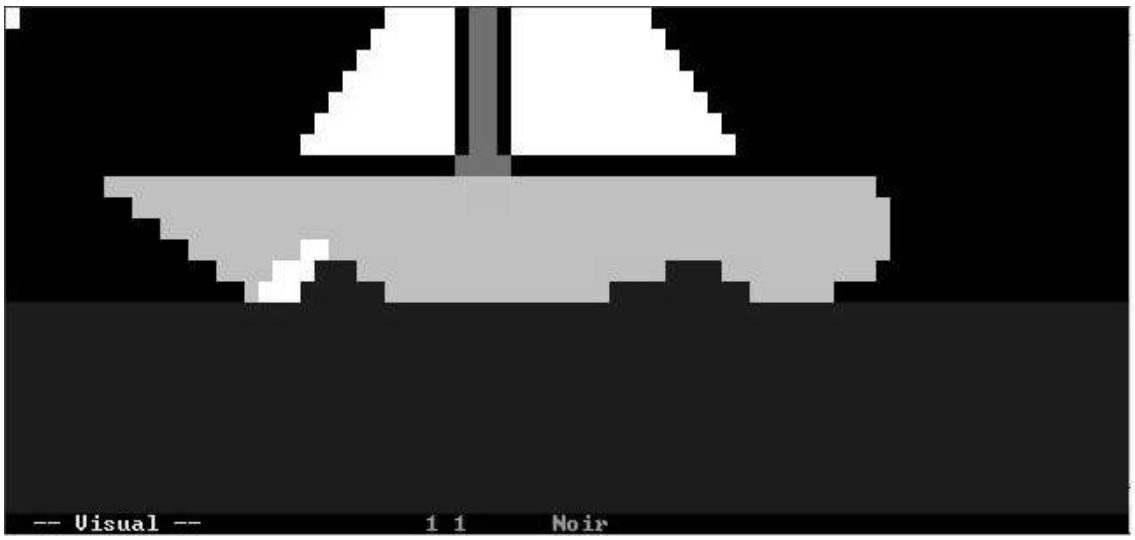
Commandes générales :

- ⟨Esc⟩ ou ⟨Alt⟩-⟨F4⟩ : Quitter
- ⟨Ctrl⟩-S : Sauver dans un fichier
- ⟨Ctrl⟩-O : Ouvrir un fichier
- ⟨Ctrl⟩-N : Nouveau fichier

Sélection des couleurs du "pinceau" :

- ⟨F1⟩ : Noir
- ⟨F2⟩ : Bleu
- ⟨F3⟩ : Vert
- ⟨F4⟩ : Cyan
- ⟨F5⟩ : Rouge
- ⟨F6⟩ : Magenta
- ⟨F7⟩ : Marron
- ⟨F8⟩ : Gris
- ⟨F9⟩ : Jaune
- ⟨F10⟩: Blanc
- ⟨Ctrl⟩-⟨F2⟩ : Bleu clair

- ⟨Ctrl⟩-⟨F3⟩ : Vert clair
- ⟨Ctrl⟩-⟨F4⟩ : Cyan clair
- ⟨Ctrl⟩-⟨F5⟩ : Rouge clair
- ⟨Ctrl⟩-⟨F6⟩ : Magenta clair

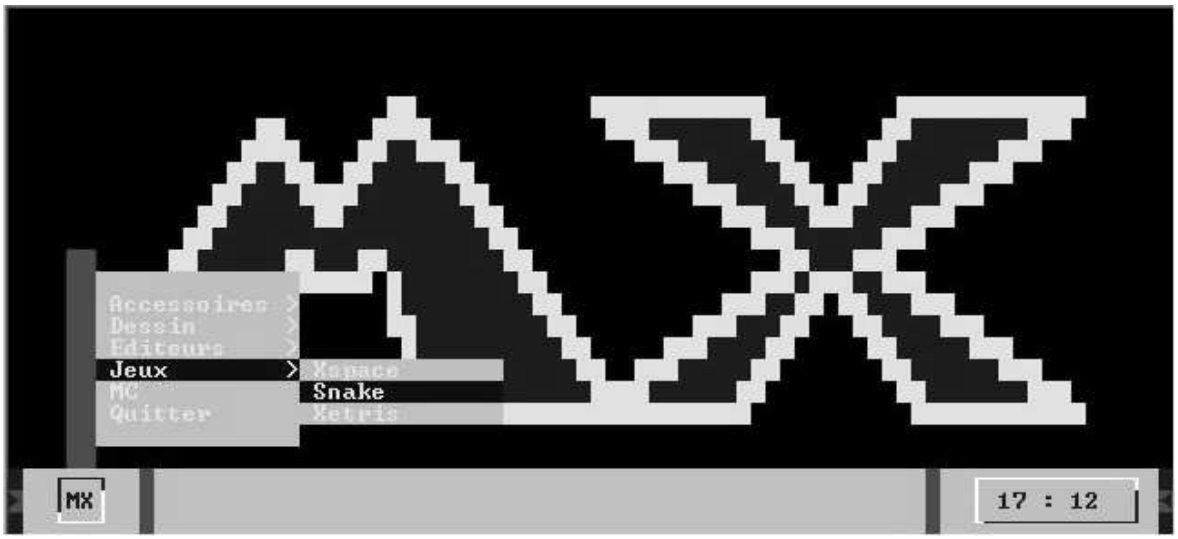


9.2 Mx

Mx a été développé par Xavier, 13 ans en 2004, alors qu'il avait découvert la programmation et Ada cinq mois auparavant.

Mx est un lanceur d'application. Il utilise les fichiers de ressources '`.vsl`' créé par Visual. Le bouton "Démarrer", baptisé ici "Mx" est en relief. Les menus sont imbriqués.

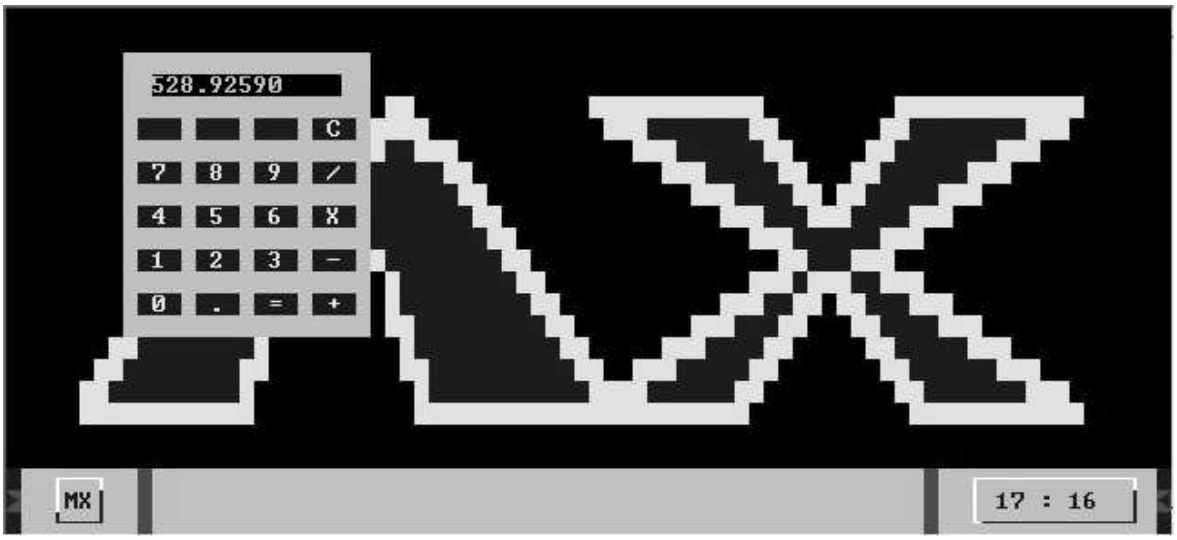
Les sources de Mx sont disponibles dans '`c:\aide\examples\aide\projects\mx`'.



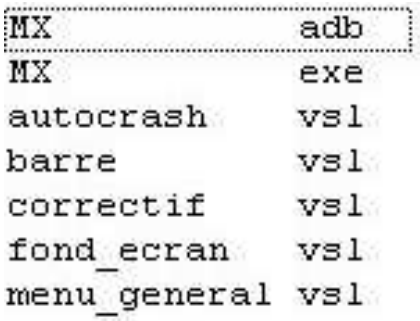
Commandes générales :

- ⟨Esc⟩ : Quitte
- ⟨Entrée⟩ : Validation
- ⟨Flèches⟩ : Déplacement

Exemple avec une “applette” calculatrice :



Détail d’un menu :



9.3 AdaOpenGL

AdaOpenGL a été créé par Étienne Baudin (email : ebaudin@mail.com).

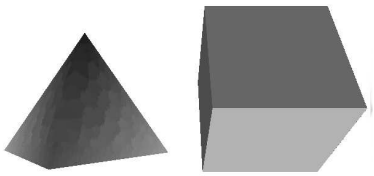
Étienne souhaite que le site de son inspirateur, Nate Robin, soit cité : http://www.xmission.com/~nate/tutors.html, ainsi qu’un article sur le *cell shading* disponible ici : http://www.gamedev.net/reference/articles/article1438.asp.

Les sources d'AdaOpenGL sont disponibles dans '`c:\aide\examples\opengl`'.

À partir d'une console MSys, placez-vous dans le répertoire '`c:\aide\examples\opengl`', taper '`make cglp`' pour compiler l'exemple '`Newg`', qui accepte un paramètre : le nom de l'objet sans extension pourvu qu'il soit dans le répertoire "objets".

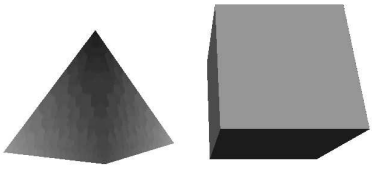
Dans le sous répertoire '`adaopengl`', compiler l'exemple par la commande '`make cglp`'.

Dans le sous répertoire '`glut32`', compiler les exemples par la commande '`make`'.



Quelques indications d'emploi :

- ⟨ctrl⟩-⟨C⟩ pour quitter à partir de la console ;
- double : clic droit arrête l'animation ;
- csg : clic droit pour la combinaison des figures ;
- ada_sphere : clic droit pour quitter ;
- Si l'on souhaite rendre un exemple indépendant de AIDE, il suffit de mettre glut32.dll dans un chemin accessible par le PATH du système.



10 Exemples de documentations

10.1 Généralités

La documentation de AIDE est, bien sûr, basée Texinfo. Les avantages de ce format ont été abondamment détaillés dans le chapitre *Documentation* de ce manuel.

La documentation des packages V04 & Visual a été effectuée *complètement automatiquement* à partir des spécifications '`v04.ads`', '`v04-crt.ads`' et '`visual.ads`'.

Leur insertion dans le manuel de AIDE a juste nécessité trois lignes :

```
@include texi/v04.texi
@include texi/v04-crt.texi
@include texi/visual.texi
```

10.2 Documentation de AIDE

Description du fichier source

<<<TODO>>>

Insertion d'images avec GIMP

Pour insérer des images avec GIMP, sauver au format '`.eps`'.

Insertion de diagrammes avec DIA

Pour insérer des diagrammes avec DIA, choisissez une exportation au format '`.eps`'.

Génération PDF

À partir d'une console MSys, traduire les fichiers d'images au format '`.eps`' vers le format '`.pdf`' par l'utilitaire '`epstopdf`'.

<<<TODO>>>

Génération HTML

Les fichiers d'images au format '`.eps`' seront convertis au format '`.png`' ou '`.jpg`'.

<<<TODO>>>

10.3 Package v04

Author	Stephane &Xavier Riviere, Martin Cattoen
Date	03/09/04
Description	V04 library, parent package
File name	v04.ads

10.3.1 Definitions

10.4 Package v04.crt

Author	Stephane &Xavier Riviere, Martin Cattoen
Date	03/09/04
Description	V04 library, CRT functions
File name	v04-crt.ads

10.4.1 Basic objects

Name	*Type*	*Default*	
Max_Line	Integer	24	constant
Max_Column	Integer	79	constant
Oem	Integer	850	constant
Latin_1	Integer	1252	constant
Key codes (Direct)			
K_Null	Integer	0	constant
K_BackSpace	Integer	8	constant
K_Tab	Integer	9	constant
K_Return	Integer	13	constant
K_Escape	Integer	27	constant
K_Left	Integer	331	constant
K_Right	Integer	333	constant
K_Up	Integer	328	constant
K_Down	Integer	336	constant
K_Home	Integer	327	constant
K_End	Integer	335	constant
K_PageUp	Integer	329	constant
K_PageDown	Integer	337	constant
K_Insert	Integer	338	constant
K_Delete	Integer	339	constant
K_F1	Integer	315	constant
K_F2	Integer	316	constant
K_F3	Integer	317	constant
K_F4	Integer	318	constant
K_F5	Integer	319	constant
K_F6	Integer	320	constant
K_F7	Integer	321	constant
K_F8	Integer	322	constant
K_F9	Integer	323	constant
K_F10	Integer	324	constant
K_F11	Integer	389	constant
K_F12	Integer	390	constant
Key codes (Shift)			
K_Shift_F1	Integer	340	constant
K_Shift_F2	Integer	341	constant
K_Shift_F3	Integer	342	constant
K_Shift_F4	Integer	343	constant
K_Shift_F5	Integer	344	constant
K_Shift_F6	Integer	345	constant
K_Shift_F7	Integer	346	constant
K_Shift_F8	Integer	347	constant
K_Shift_F9	Integer	348	constant
K_Shift_F10	Integer	349	constant
K_Shift_F11	Integer	391	constant
K_Shift_F12	Integer	392	constant

Key codes (Ctrl)

K_Ctrl_Left	Integer	371	constant
K_Ctrl_Right	Integer	372	constant
K_Ctrl_Up	Integer	397	constant
K_Ctrl_Down	Integer	401	constant
K_Ctrl_Home	Integer	375	constant
K_Ctrl_End	Integer	373	constant
K_Ctrl_PageUp	Integer	390	constant
K_Ctrl_PageDown	Integer	374	constant
K_Ctrl_Insert	Integer	402	constant
K_Ctrl_Delete	Integer	403	constant
K_Ctrl_A	Integer	1	constant
K_Ctrl_B	Integer	2	constant
K_Ctrl_C	Integer	3	constant
K_Ctrl_D	Integer	4	constant
K_Ctrl_E	Integer	5	constant
K_Ctrl_F	Integer	6	constant
K_Ctrl_G	Integer	7	constant
K_Ctrl_H	Integer	8	constant
K_Ctrl_I	Integer	9	constant
K_Ctrl_J	Integer	10	constant
K_Ctrl_K	Integer	11	constant
K_Ctrl_L	Integer	12	constant
K_Ctrl_M	Integer	13	constant
K_Ctrl_N	Integer	14	constant
K_Ctrl_O	Integer	15	constant
K_Ctrl_P	Integer	16	constant
K_Ctrl_Q	Integer	17	constant
K_Ctrl_R	Integer	18	constant
K_Ctrl_S	Integer	19	constant
K_Ctrl_T	Integer	20	constant
K_Ctrl_U	Integer	21	constant
K_Ctrl_V	Integer	22	constant
K_Ctrl_W	Integer	23	constant
K_Ctrl_X	Integer	24	constant
K_Ctrl_Y	Integer	25	constant
K_Ctrl_Z	Integer	26	constant
K_Ctrl_F1	Integer	350	constant
K_Ctrl_F2	Integer	351	constant
K_Ctrl_F3	Integer	352	constant
K_Ctrl_F4	Integer	353	constant
K_Ctrl_F5	Integer	354	constant
K_Ctrl_F6	Integer	355	constant
K_Ctrl_F7	Integer	356	constant
K_Ctrl_F8	Integer	357	constant
K_Ctrl_F9	Integer	358	constant
K_Ctrl_F10	Integer	359	constant
K_Ctrl_F11	Integer	393	constant
K_Ctrl_F12	Integer	394	constant

Key codes (alt)

K_Alt_F1	Integer	360	constant
K_Alt_F2	Integer	361	constant
K_Alt_F3	Integer	362	constant
K_Alt_F4	Integer	363	constant
K_Alt_F5	Integer	364	constant
K_Alt_F6	Integer	365	constant
K_Alt_F7	Integer	366	constant
K_Alt_F8	Integer	367	constant
K_Alt_F9	Integer	368	constant
K_Alt_F10	Integer	369	constant
K_Alt_F11	Integer	395	constant
K_Alt_F12	Integer	396	constant
K_Alt_Left	Integer	411	constant
K_Alt_Right	Integer	413	constant
K_Alt_Up	Integer	408	constant
K_Alt_Down	Integer	416	constant
K_Alt_Home	Integer	407	constant
K_Alt_End	Integer	415	constant
K_Alt_PageUp	Integer	409	constant
K_Alt_PageDown	Integer	417	constant
K_Alt_Insert	Integer	418	constant
K_Alt_Delete	Integer	419	constant
Key codes (Ctrl Alt)			
K_Ctrl_Alt_Backspace	Integer	270	constant

10.4.2 Definitions

Types

Name	*Declaration*
Color_Type	(Black, Blue, Green, Cyan, Red, Magenta, Brown, Gray, Light_Blue, Light_Green, Light_Cyan, Light_Red, Light_Magenta, Yellow, White)
Line	Integer range 1..Max_Line
Column	Integer range 1..Max_Column

10.4.3 Subprograms

10.4.3.1 Beep

procedure Beep

Parameter name	*Type*	*Mode*	*Default*
Frequency	Integer	**in**	2000
Period	Integer	**in**	100

Description : Send a beep.
Frequency : Tone frequency (in Hertz)
Period : Tone duration (in millisecond)

10.4.3.2 Clear

procedure Clear

Parameter name	*Type*	*Mode*	*Default*
Back	color_type	**in**	black

Description : Clear the screen.
Back : Background color.

10.4.3.3 New_Line

procedure New_Line

Parameter name	*Type*	*Mode*	*Default*
Lines	Line	**in**	1

Description : Send a new line.
Lines : Number of lines.

10.4.3.4 Cursor_Move

procedure Cursor_Move

Parameter name	*Type*	*Mode*	*Default*
Row	Line	**in**	
Col	Column	**in**	

Description : Move the cursor at the specified coordinates.
Row : Row coordinate.
Col : Column coordinate.

10.4.3.5 Color_Set

procedure Color_Set

Parameter name	*Type*	*Mode*	*Default*
Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black

Description : Set the current color.
Fore : Foreground color.
Back : Background color.

10.4.3.6 put

procedure put

Parameter name	*Type*	*Mode*	*Default*
text	string	**in**	
fore	color_type	**in**	gray
back	color_type	**in**	Black
Period	Duration	**in**	0.0

Description : Put a text on the console with given colors attributes.
Fore : Foreground color.
Back : Background color.
Period : Wait duration between character output

10.4.3.7 Put

procedure Put

Parameter name	*Type*	*Mode*	*Default*
Number	integer	**in**	
Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black

Description : Put a number on the console with given colors attributes.
Fore : Foreground color.
Back : Background color.

10.4.3.8 Put_Line

procedure Put_Line

Parameter name	*Type*	*Mode*	*Default*
Text	String	**in**	
Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black

Description : Put a text on the console with given colors attributes followed by a new line.
Fore : Foreground color.
Back : Background color.

10.4.3.9 Put_Line

procedure Put_Line

Parameter name	*Type*	*Mode*	*Default*
Number	integer	**in**	

Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black

Description : Put a number on the console with given colors attributes followed by a new line.
Fore : Foreground color.
Back : Background color.

10.4.3.10 Put_Lc

procedure Put_Lc

Parameter name	*Type*	*Mode*	*Default*
Row	Line	**in**	1
Col	Column	**in**	1
Number	Integer	**in**	
Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black

Description : Put a number on the console at given coordinates with specified colors attributes followed by a new line.
Row : Row coordinate.
Col : Column coordinate.
Fore : Foreground color.
Back : Background color.

10.4.3.11 Put_Lc

procedure Put_Lc

Parameter name	*Type*	*Mode*	*Default*
Row	Line	**in**	1
Col	Column	**in**	1
Text	String	**in**	
Fore	Color_Type	**in**	Gray
Back	Color_Type	**in**	Black
Period	Duration	**in**	0.0

Description : Put a text on the console at given coordinates with specified colors attributes followed by a new line.
Row : Row coordinate.
Col : Column coordinate.
Fore : Foreground color.
Back : Background color.
Period : Wait duration between character output

10.4.3.12 Codepage_Set

procedure Codepage_Set

Parameter name	*Type*	*Mode*	*Default*
Codepage	integer	**in**	Latin_1

Description : Set the codepage character, works with Windows NT OS family only.
Codepage : Codepage identifier, currently `Oem` or `Latin_1`.

10.4.3.13 Codepage_Get

function Codepage_Get *return* integer

Description : Get the codepage character, works with Windows NT OS family only.
Return : Codepage identifier, currently `Oem` or `Latin_1`.

10.4.3.14 Key_Read

function Key_Read *return* Integer

Description : Read the keyboard.
Return : The last key pressed.

10.4.3.15 Key_Available

function Key_Available *return* Boolean

Description : Check key availability.
Return : `True` if a key is available, `False` otherwise.

10.5 Package Visual

Author	Martin Cattoen
Date	16/08/04
Description	Draw in text mode. Visual files can be used as resources by others programs
File name	visual.ads

10.5.1 Clauses

Context clauses

- V04.Crt

Use clauses

- V04

10.5.2 Basic objects

Name	*Type*	*Default*	
Constants			
Screen depth			
MAX_LINE	Natural	Crt.Max_Line	constant
MAX_COLUMN	Natural	Crt.Max_Column	constant
Cursor			
FORE_COLOR_CURSOR	Crt.Color_Type	Crt.White	constant
BACK_COLOR_CURSOR	Crt.Color_Type	Crt.Black	constant
CHARACTER_CURSOR	Character	Character'Val (177)	constant
Definite colors			
MAGIC_COLOR	Crt.Color_Type	Crt.Black	constant
Keys (functions)			
KEY_CHOICE_MACRO	Integer	Crt.K_Shift_F1	constant
KEY_CHOICE_COLOR_FORE	Integer	Crt.K_Ctrl_F	constant
KEY_INSERT_CODE_CHARACTER	Integer	Crt.K_Ctrl_I	constant
KEY_INITIALIZE_FILE	Integer	Crt.K_Ctrl_N	constant
KEY_OPEN_FILE	Integer	Crt.K_Ctrl_O	constant
KEY_SAVE_FILE	Integer	Crt.K_Ctrl_S	constant
KEY_APPLY_COLOR_BACK	Integer	Crt.K_Delete	constant
KEY_APPLY_COLOR_FORE	Integer	Crt.K_Delete	constant
KEY_SELECT_COPY	Integer	Crt.K_Alt_F5	constant
KEY_COPY	Integer	Crt.K_F5	constant
KEY_PASTE	Integer	Crt.K_Alt_F7	constant
KEY_MODE_INSERT_REPLACE	Integer	Crt.K_Insert	constant
KEY_EXIT_VISUAL	Integer	Crt.K_Escape	constant
Keys (colors)			
KEY_BLACK	Integer	Crt.K_F1	constant
KEY_BLUE	Integer	Crt.K_F2	constant
KEY_GREEN	Integer	Crt.K_F3	constant
KEY_CYAN	Integer	Crt.K_F4	constant
KEY_RED	Integer	Crt.K_F5	constant
KEY_MAGENTA	Integer	Crt.K_F6	constant
KEY_BROWN	Integer	Crt.K_F7	constant
KEY_GRAY	Integer	Crt.K_F8	constant
KEY_LIGHT_BLUE	Integer	Crt.K_Ctrl_F2	constant
KEY_LIGHT_GREEN	Integer	Crt.K_Ctrl_F3	constant
KEY_LIGHT_CYAN	Integer	Crt.K_Ctrl_F4	constant
KEY_LIGHT_RED	Integer	Crt.K_Ctrl_F5	constant
KEY_LIGHT_MAGENTA	Integer	Crt.K_Ctrl_F6	constant
KEY_YELLOW	Integer	Crt.K_F9	constant
KEY_WHITE	Integer	Crt.K_F10	constant

NULL_FILE	Visual_File	(Tab_Fore ⇒ (others ⇒ (others ⇒ Crt.Gray)), Tab_Back ⇒ (others ⇒ (others ⇒ Crt.Black)), Tab_Char ⇒ (others ⇒ (others ⇒ ' ')))	constant

10.5.3 Definitions

Types

Name	*Declaration*
T_Tab_Color	array (Line,Column) of Crt.Color_Type
T_Tab_Char	array (Line,Column) of Character
Line	Natural range 1..MAX_LINE
Column	Natural range 1..MAX_COLUMN

Record Visual_File

Component	*Type*	*Default*
Tab_Char	T_Tab_Char	
Tab_Back	T_Tab_Color	
Tab_Fore	T_Tab_Color	

Exceptions

- Open_Visual_File_Error

Pragmas

Elaborate_Body

10.5.4 Subprograms

10.5.4.1 Open

function Open *return* Visual_File

Parameter name	*Type*	*Mode*	*Default*
Name	String	**in**	

Description : Load a visual file
Name : The visual file name to load
Return : The loaded visual file
Exceptions : Raise Open_Visual_File_Error if Name is not available or file is not a Visual file

10.5.4.2 Open

function Open *return* Visual_File

Parameter name	*Type*	*Mode*	*Default*
Name	String	**in**	
Success	Boolean	**access**	

Description : Load a visual file

Name : The visual file name to load
Success : Tre if open succeed, false otherwise
Return : The loaded visual file
Exceptions : Raise Open_Visual_File_Error if Name is not available or file is not a Visual file

10.5.4.3 Create

procedure Create

Parameter name	*Type*	*Mode*	*Default*
Name	String	**in**	
Item	Visual_File	**in**	

Description : Create a visual file
Name : The visual file name to create
Item : The visual file to create

10.5.4.4 Text_File_To_Visual

function Text_File_To_Visual *return* Visual_File

Parameter name	*Type*	*Mode*	*Default*
From	String	**in**	

Description : Load a visual file from an external text file
From : The text file name to load
Return : The visual file

10.5.4.5 Visual_To_Text_File

procedure Visual_To_Text_File

Parameter name	*Type*	*Mode*	*Default*
From	Visual_File	**in**	
To	String	**in**	

Description : Write an external text file from a visual file
From : The visual file to write
To : The external test file

10.5.4.6 Clean_Screen

procedure Clean_Screen

Parameter name	*Type*	*Mode*	*Default*
Item	Visual_File	**in**	
Min_C_Scr	Column	**in**	1

Min_L_Scr	Line	**in**	1
Max_C_Scr	Column	**in**	MAX_COLUMN
Max_L_Scr	Line	**in**	MAX_LINE
Min_C_Fil	Column	**in**	1
Min_L_Fil	Line	**in**	1
Put_Magic_Color	Boolean	**in**	True

Description : Display item to screen
Item :
Min_C_Scr :
Min_L_Scr :
Max_C_Scr :
Max_L_Scr :
Min_C_Fil :
Min_L_Fil :
Put_Magic_Color :

10.5.4.7 Processing

procedure Processing

Colors
Black : ⟨F1⟩
Blue : ⟨F2⟩
Green : ⟨F3⟩
Cyan : ⟨F4⟩
Red : ⟨F5⟩
Magenta : ⟨F6⟩
Brown : ⟨F7⟩
Gray : ⟨F8⟩
Yellow : ⟨F9⟩
White : ⟨F10⟩
Light_Blue : ⟨C-F2⟩
Light_Green : ⟨C-F3⟩
Light_Cyan : ⟨C-F4⟩
Light_Red : ⟨C-F5⟩
Light_Magenta : ⟨C-F6⟩
Main commands
⟨Esc⟩ : Quit; ⟨F5⟩ : Copy; ⟨F7⟩ : Paste;
⟨C-s⟩ : Save; ⟨C-o⟩ : Load; ⟨C-n⟩ : New file;
⟨C-f⟩ : Shape color; ⟨Del⟩ : Fill background color;
⟨C-i⟩ : Character code; ⟨C-m⟩ : macro character code
Editing commands
⟨Up⟩, ⟨Down⟩, ⟨Right⟩, ⟨Left⟩ : Write;
⟨C-Up⟩, ⟨C-Down⟩, ⟨C-Right⟩, ⟨C-Left⟩ : Move cursor;
⟨M-Up⟩, ⟨M-Down⟩, ⟨M-Right⟩, ⟨M-Left⟩ : Write macro character;

10.5.4.8 Code_Char

function Code_Char *return* Character

Description : Get character code
Code_Char : Character code
Return : ASCII.NUL if character code is not available

Annexe A Collection de sources Ada de AIDE

Généralités

Tous les logiciels, bibliothèques et paquetages décrits ci-dessous sont disponibles en source et libres d'emploi dans un contexte commercial.

Sauf mention contraire, le langage employé est Ada et dans ce cas, le logiciel est disponible dans '`aide-src-1.03.exe`'.

Les version proposées ne sont les dernières disponibles. Aucune effort n'est fait en ce sens, par manque de contribution. Si un package vous intéresse, cherchez la dernière version sur internet.

Base de données SQL

Il existe des bases de données écrites en Ada, mais le temps a manqué pour les évaluer et examiner les licence de celles qui étaient disponibles en sources, telles ADAISM. Par contre les deux bases ci-dessous, complémentaires, s'accordent parfaitement avec Ada.

SQLite

SQLite est une mini base de données SQL embarquable. L'empreinte est minimale et la fiabilité remarquable. Portable et libre, SQLite bénéficie d'un excellent support.

SQLite peut être utilisée en Ada grâce à un binding natif disponible dans Gnade. Un driver ODBC existe aussi. Il autorise également un accès via l'interface ODBC de Gnade. SQLite est écrit en C.

Firebird

Firebird est la version libre d'Interbase, base de données mature existant réputée pour sa fiabilité, sa portabilité et sa compacité. Firebird, comme Postgres, mais *a contrario* de MySQL, peut être utilisé librement pour une application commerciale. Firebird peut aussi être utilisé en base locale, via un jeu de quelques fichiers totalisant environ 1,5 Mo.

Il n'existe pas d'interface native de haut niveau en Ada pour Firebird. Firebird peut toutefois être utilisée directement avec Ada grâce à un driver ODBC et l'interface ODBC de Gnade. Firebird est écrit en C.

PostgreSQL

Postgres est certainement la base de données de choix pour un développement d'envergure en Ada. PostgrSQL est désormais nativement porté sous Windows et dispose d'une interface native de haut niveau avec APQ. Que demander de plus ?

Gnade

Gnade est l'interface ODBC de choix sous Ada. Toutefois, différents bindings natifs sont aussi proposés. C'est l'interface idéale pour SQLite ou Firebird, par exemple.

'`c:\aide\src\packages\data base management\relational\gnade`'

APQ

APQ, binding natif, autorise une interface portable de haut niveau avec Postgres et MySQL. Une documentation très soignée accompagne ce paquetage.

'`c:\aide\src\packages\data base management\relational\apq`'

Interface système

système de fichier

‘c:\aide\src\packages\system & general\system\os_service’

écran mode texte

‘c:\aide\src\packages\system & general\system*’

ports série

‘c:\aide\src\packages\system & general\serial’

Formats de fichiers

btree

‘c:\aide\src\packages\data base management\sequential\btree’

Le format dBase est accessible en lecture et écriture par une bibliothèque. Un btee

‘c:\aide\src\packages\data base management\sequential\dbf’

printing

‘c:\aide\src\packages\printing’

Unicode

Différentes implémentations existent.

‘c:\aide\src\packages\unicode’

xml

Le format xml est accessible en lecture et écriture par une bibliothèque existante.

‘c:\aide\src\packages\miscellaneous\xmlAda’

zlib & gzip

Les formats zlib et gzip sont accessibles en lecture et écriture par une bibliothèque existante.

‘c:\aide\src\packages\compress’

Interface d’accès aux services internet

ftp

Une implémentation existe, non directement utilisable.

‘c:\aide\src\packages\internet’

http

Une implémentation cliente et serveur existe.

‘c:\aide\src\packages\internet’

https

Une implémentation cliente et serveur existe.

‘c:\aide\src\packages\internet’

pop3

‘c:\aide\src\packages\internet’

smtp

Différentes implémentations client existent.

‘c:\aide\src\packages\internet’

sockets tcp/ip

Différentes implémentations existent.

‘c:\aide\src\packages\internet’

tcp/ip

Différentes implémentations existent, non directement utilisables.

‘c:\aide\src\packages\internet’

Autres bibliothèques

Calcul

Différentes implémentations de FFT existent.

‘c:\aide\src\packages\digital signal processing’

Une bibliothèque implémentant des CRC16 et CRC32 conformes POSIX existe. La conformité POSIX implique un CRC16 conforme CCITT et un CRC32 conforme ANSI/IEEE.

‘c:\aide\src\packages\miscellaneous\crc’

Chaînes de caractères

‘c:\aide\src\packages\miscellaneous\strings’

Chiffrement

Une biliothèque de hachage (haval, md5, ripemd160, sha) existe.

Une implémentation de l’algorithme AES (remplaçant de DES) existe.

Les algorithmes serpent et solitaire sont implémentés.

Une interface à GPG (GNU PG, remplaçant GNU de PGP) existe.

‘c:\aide\src\packages\cryptography’

Listing de la collection

```
* = Racine de la collection de sources, ‘c:\aide\src’ par défaut.

+---books       Sources des examples de plusieurs ouvrages Ada
|   +---Ada95 - Orientation objet, structures de données et algorithmes
|   +---Files structures with Ada
|   +---Object-oriented Software in Ada 95
|   +---Software construction and data structures with Ada 95
+---embedded
|   +---low end micro controllers
|   |   +---atmel avr
|   |   \---motorola 68hc1x
|   \---operating systems
|       +---edu-os
|       +---tiny_lovelace
```

```
|       +---marte
|       +---rtems
|       \---xada
+---end user
|   +---ada terminal emulator
|   +---adabill
|   +---adadmst
|   +---adagio
|   +---adressage
|   +---adump
|   +---AutoCDOrganizer
|   +---bush
|   +---cheddar
|   +---Coldframe
|   +---defendguin
|   +---dtraq
|   +---generic web base reuse library
|   +---gnome ada task monitor
|   +---Klokka
|   +---mastermind
|   +---mine detector
|   +---orbital
|   +---radio link calculator
|   +---sc_timer
|   +---SETI@Home Service
|   +---simulateur de maquettes de train
|   +---smallbench
|   +---test de connaissance ada
|   +---texcad
|   +---win2000
|   \---zephyr base camp
+---packages
|   +---automation
|   |   +---fsm-2.0
|   |   +---genauto-2.0b
|   |   +---irobot
|   |   \---mindstorm
|   +---compress
|   |   +---huffman
|   |   |   +---ptlzhuf
|   |   |   \---transvu
|   |   +---object code compression tool
|   |   +---unzip
|   |   \---zlib
|   +---concurrent and distributed algorithms
|   |   +---ADA
|   |   \---ADAS
|   +---cryptography
|   |   +---adacf
|   |   +---digest
|   |   +---gpgme
|   |   +---rijndael (aes)
|   |   +---serpent
|   |   +---solitaire
|   |   \---validation_cb
|   +---data base management
|   |   +---relational
|   |   |   +---adbm-1.8
|   |   |   +---apq-2.0
|   |   |   +---asset_collection
|   |   |   +---gdbm ada binding
|   |   |   +---gnade
|   |   |   +---mneson-20040601
|   |   |   +---mysql thin binding
|   |   |   +---odbc_client_ada1
```

```
|   |   |   +---odbc_client_ada2
|   |   |   +---oracle binding
|   |   |   \---pgada-0.0.1
|   |   \---sequential
|   |       +---btree
|   |       |   +---generic b plus tree
|   |       |   +---indexed sequential files
|   |       |   +---indexed_io
|   |       |   \---indexed_sequential_io
|   |       +---btree related
|   |       |   +---cs parts
|   |       |   \---mims
|   |       +---dbf
|   |       \---read structured ascii file
|   +---data structures
|   |   +---adasl-1.3
|   |   +---bibSDAda
|   |   +---bintree
|   |   +---data structure
|   |   +---flashsort
|   |   +---scds with ada95
|   |   \---uofs_ds2
|   +---debug & test
|   |   +---aunit-1.01
|   |   \---smallbench
|   +---digital signal processing
|   |   +---adafft
|   |   +---fftwa-1.1a
|   |   +---kalman
|   |   \---libtsp-v5r0
|   +---gautier de montmollin packages
|   |   +---biblio
|   |   +---divers
|   |   +---gnat specific
|   |   +---maths
|   |   \---tests
|   +---internet
|   |   +---ada95_http_hdrs
|   |   +---adasockets-0.1.19
|   |   +---aws-1_3
|   |   +---aws-1_4
|   |   +---aws-2_0
|   |   +---ddn
|   |   |   +---ftp
|   |   |   +---iface
|   |   |   +---smtp
|   |   |   +---tcpip
|   |   |   +---tcpip2
|   |   |   +---telnet
|   |   |   +---usersman
|   |   |   \---wicatmisc
|   |   +---ews-20040425a
|   |   +---nc-sockets-windows
|   |   +---sftp
|   |   \---sntp
|   +---interpreters
|   |   +---adaed
|   |   \---augusta
|   +---miscellaneous
|   |   +---calendar
|   |   +---crc
|   |   +---fuzzy logic
|   |   +---gad
|   |   +---geographic
|   |   |   \---cnes-nacelles pointées
```

```
|   |   +---ini
|   |   |   \---grace-0.51
|   |   +---lua-binding-0.10
|   |   +---pager2
|   |   +---pdf
|   |   +---se_projs
|   |   +---spell2
|   |   +---strings
|   |   |   +---formatted output
|   |   |   +---formatted print
|   |   |   +---pascal obry packages
|   |   |   +---strings_1_4
|   |   |   +---txt tools
|   |   |   \---very fast unbounded strings package
|   |   +---unitrep
|   |   |   \---source
|   |   +---units
|   |   \---xmlAda
|   +---multimedia
|   |   +---AdaSDL_20010505
|   |   +---ADAVOX-0.51
|   |   +---ada_magick-a_01 (g2f_io)
|   |   +---mcc-sounds
|   |   +---opengl
|   |   +---png-2.1a
|   |   +---scan
|   |   +---SDL-1.2.3
|   |   +---simple jpeg lib-1.2
|   |   \---sound thx
|   +---parser lexer
|   |   +---adagoop-1.0
|   |   +---aflex-ayacc
|   |   +---fenillator
|   |   +---format
|   |   \---opentoken-3.0b
|   +---printing
|   |   +---diplot
|   |   +---pager2
|   |   \---psprint
|   +---system & general
|   |   +---ada0y
|   |   +---do while jones packages
|   |   +---gautier de montmollin packages
|   |   +---generics
|   |   +---libra-0.1.3
|   |   +---mw_components acl-2.0
|   |   +---pragmarc
|   |   +---serial
|   |   |   +---serial
|   |   |   +---serial win32 & linux
|   |   |   \---win32_com_ports
|   |   +---system
|   |   |   +---adaplugins-0.43
|   |   |   +---console
|   |   |   |   +---ansi
|   |   |   |   \---win32
|   |   |   +---console vt100
|   |   |   +---console_io & alphabetical
|   |   |   +---jotto
|   |   |   +---os_service
|   |   |   +---pd-curses
|   |   |   |   +---AdaTUI
|   |   |   +---pipes
|   |   |   +---registry
|   |   |   +---smartArgs-0.5
```

```
|   |   |   +---standard_io
|   |   |   \---ufn-2.19
|   |   +---system (platform specific)
|   |   |   +---dos
|   |   |   |   +---conio 02
|   |   |   |   |   \---HANGMAN
|   |   |   |   \---dosports
|   |   |   +---linux
|   |   |   |   +---console
|   |   |   |   |   +---alpha_console
|   |   |   |   |   +---texttools-1_0_3
|   |   |   |   |   \---texttools-2_0_3
|   |   |   |   \---serial
|   |   |   \---windows
|   |   |       +---console
|   |   |       |   +---nt_console
|   |   |       |   +---nt_mouse
|   |   |       |   +---win32-nt_console
|   |   |       |   \---win32-screen
|   |   |       +---CpuMeterV2.0
|   |   |       +---gnatcom
|   |   |       +---gnatdll
|   |   |       +---gnatwin
|   |   |       +---gwindows-beta1b
|   |   |       +---toolhelp
|   |   |       +---win32ada
|   |   |       +---winsock
|   |   |       +---winsock2
|   |   |       +---winsock3
|   |   |       |   +---demo
|   |   |       |   \---doc
|   |   |       \---wintlb
|   |   +---system (posix)
|   |   |   +---florist
|   |   |   |   +---florist-3.15p-src
|   |   |   +---page-1.0
|   |   |   +---posix
|   |   |   |   \---Posix
|   |   |   |       +---OS_SOLARIS
|   |   |   |       +---OS_SUNOS
|   |   |   |       +---OS_WINNT
|   |   |   |       \---test
|   |   |   +---specifications
|   |   |   +---stars
|   |   |   \---win32posix
|   |   +---tasking
|   |   +---unicode
|   |   |   \---Ngeadal-0.0.2.a
|   |   \---util_1.1
|   +---tasking
|   \---unicode
|       \---Ngeadal-0.0.2.a
\---tools
    +---binders
    |   +---adaerl-0.1
    |   +---c2ada-linux
    |   +---cbind4win32
    |   +---ch2ada
    |   +---cpp_ada_class
    |   +---lib2def
    |   \---p2ada
    +---browsers
    |   \---alibrowse-1.7b1
    +---doc generators
    |   +---adabrowse-4.02
```

```
|   +---adadoc-2.02
|   \---arm generator
+---editors
|   \---word processor
+---reformat
|   +---requirements
|   \---tests
\---tests
    +---aunit-1.01
    +---smallbench
    \---tg-3.1
```

Annexe B Le cheminement vers Ada

Afin de motiver les choix proposés par AIDE, il peut être utile de présenter le cheminement ayant entrainé l'adoption de Ada.

Origine

AIDE résulte d'une étude initiée en 1993, au sein d'une société de service, lorsque l'apparition d'environnements graphiques tels *Presentation Manager* pour OS/2, *Windows 3* pour MS-DOS ou *XWindows* pour Unix, rendirent obsolètes les environnements en mode texte.

Cette étude devait mener à la création d'un environnement graphique dénommé V93, selon la nomenclature de cette société, en remplacement de l'environnement V90, fondé MS-DOS en mode texte avec menu déroulants et souris *à la Turbo C* et spooler d'impression en tâche de fond, basé sur Nantucket Clipper Summer 87, Microsoft C 5.10 et Microsoft macro-assembleur 5.00.

V90 avait nécessité plusieurs mois de développements et comptait plusieurs dizaines de milliers de lignes dans ces trois langages. V90 permettait de réaliser des applications industrielles ou de gestion avec une excellente productivité et une remarquable fiabilité.

Cette société de service décida qu'il ne serait plus possible de refaire un tel effort à chaque évolution technologique et qu'il serait souhaitable de créer un environnement plus universel.

1993 : Une tentative avortée

A cette époque, OS/2, fiable et multi-tâche, perdait déjà du terrain devant Windows 3, instable et mono-tâche, et les premières solutions Unix graphiques commençaient à se répandre, accroissant une offre que l'on devinait encore appellée à s'étendre.

Dans le même temps, la société Nantucket, éditeur de Clipper, était rachetée par Computer Associates, amenant un problème de pérennité, alors que la nouvelle version du langage Clipper présentait des problèmes de fiabilité inacceptables, tout en n'étant pas adaptée aux environnements graphiques émergents.

Cette situation amena la société de service à poser les critères suivants :

- Coût de l'environnement de développement complet inférieur à 50 KF (1993) ;
- Langage unique, de haut niveau, et disponible chez différents éditeurs ;
- Solution multi-plateforme avec les sources accessibles.

La notion de solution multi-plateforme sous-entendait la sélection d'un ensemble de composants multi-plateforme. L'accès aux sources devait permettre de s'affranchir de la disparition ou le rachat de tel ou tel éditeur :

- Interface homme-machine : bibliothèque graphique multi-plateforme ;
- Interface base de données : bibliothèque d'accès aux standards du marché ;
- Bibliothèques d'exécution : accès aux services des différents systèmes.

L'étude dura près d'un an. Toutes les solutions disponibles aux États-Unis et en Europe furent évaluées. Le principal résultat fut la sélection du C++ comme seul langage permettant de respecter tous les critères ci-dessus.

Ensuite, des essais de développement furent menés pour évaluer la validité de la solution. Malheureusement, ils ne furent pas concluants, le C++ est rapidement apparu comme un langage peu lisible, nécessitant des efforts très importants pour obtenir un programme fiable, et pour tout dire terriblement improductif par rapport à la solution mixte Clipper & C alors utilisée.

Les essais furent alors stoppés et il fut décidé d'attendre l'apparition de solutions multi-plateforme plus évoluées : V93 fut abandonné et remplacé par V95, limité à Windows et fondé

sur Delphi, environnement *propriétaire* très bien conçu et issu du langage Pascal, paradigme agréable et lisible.

2003 : transformation de l'essai

En dix ans, les développements logiciels ont beaucoup évolué. Les environnements de développement aussi, en grande partie grâce aux logiciels libres. Cette évolution notable a déclenché une nouvelle étude, à titre personnel cette fois.

De nouveaux langages interprétés, tels Java, Python ou Harbour, ont été exclus d'entrée de jeu, pour des raisons de performances et de fiabilité pour Java et d'ingénierie inverse pour les trois[1].

Les critères de 1993 furent repris et légèrement modifiés, compte tenu de l'évolution :

- Les outils de développement et les langages *propriétaires*, coûteux et non open source (à défaut d'être libre), sont de plus en plus souvent perçus comme une solution dangereuse par les développeurs avisés ;
- Les outils de développement libres et multi-plateformes gagnent en audience par leur qualité et leur pérenité garantie. Il est à noter que la gratuité d'un logiciel libre n'est pas ici un critère essentiel.

Les critères suivants ont été utilisés :

- Solution libre multi-plateforme ;
- Langage unique, de haut niveau, productif, fiable et compilé ;
- Interface homme-machine : bibliothèque graphique multi-plateforme ;
- Interface base de données : bibliothèque d'accès aux standards du marché (SQL) ;
- Bibliothèques d'exécution : accès aux services des différents systèmes (POSIX) ;
- Bibliothèques annexes : accès aux formats courants (XML, JPG, PDF, ZIP), aux services internet standards, etc...

Ada s'est alors imposé comme une évidence.

[1] L'ingénierie inverse est le nom donné, dans le domaine du logiciel, à la *décompilation*, c'est à dire à la re-création du code source à partir du programme exécutable. L'ingénierie inverse, dans se cas, est achevée par la re-création de l'analyse à partir de la lecture du code source re-créé. Utiliser un langage interprété pour écrire un produit commercial peut donc, dans certains cas, se révéler un choix particulièrement douloureux.

Annexe C Références

Bibliographie

Toute sélection est arbitraire, aussi il n'est pas inutile de préciser les critères utilisés ici : Soit les ouvrages cités ont été particulièrement appréciés (sérieux, consistance, lisibilité) ou bien les communications ou des interventions de l'auteur cité en font penser le plus grand bien !

Les avis complémentaires sont les bienvenus.

Programmation séquentielle avec Ada95

Pierre Bréguet & Luigi Zaffalon
Presses polytechniques et universitaires romandes
ISBN 2-88074-404-0
http://ppur.epfl.ch

Mon premier ouvrage Ada. Clair et efficace, c'est un livre de référence, véritablement indispensable.

Programmation concurrente et temps réel avec Ada95

Pierre Bréguet & Luigi Zaffalon
Presses polytechniques et universitaires romandes
ISBN 2-88074-408-3
http://ppur.epfl.ch

Indispensable pour les programmeurs TR, les développeurs matériels et logiciels, etc... Ouvrage conçu dans le même esprit que l'ouvrage précédent.

Programmer en Ada 95

John Barnes
Vuibert
ISBN 2-7117-8851-X
http://ppur.epfl.ch

La référence la plus connue. Très bien rédigé. Se lit comme un livre.

Ada95 : Orientation objet, structures de données et algorithmes

Philippe Gabrini
De Boeck Université
ISBN 2-8041-3790-2
http://grosmax.si.uqam.ca/Professeurs/Gabrini/Exemples

Ce livre de Philippe Gabrini est excellent. Tous les algorithmes sont présentés sous forme de pseudo-code. A posséder absolument. Les sources des exemples de l'ouvrage sont disponibles, avec la permission de l'auteur, dans '`aide-doc-1.03.tar.bz2`'.

Object-oriented Software in Ada 95

Michael A. Smith
School of Computing
University of brighton
http://http://www.it.brighton.ac.uk/~mas

Ce livre est une extraordinaire référence et il est disponible gratuitement en téléchargement ! Son contenu est exceptionnel et les sources des exemples de l'ouvrage sont disponibles dans '`aide-doc-1.03.tar.bz2`'.

Les ouvrages de Jean-Pierre Rosen

Adalog
Jean-Pierre Rosen
http://www.adalog.fr

Jean-Pierre Rosen est un *gourou* de HOOD et de Ada et a écrit plusieurs ouvrages traitant de la conception logicielle, dont l'un spécifique à HOOD. Tous les écrits de cet auteur sont extrèmement intéressants, voire incontournables et ses conférences sont de grands moments "d'apprentissage jubilatoire" qui rassemblent une compétence rare et un plaisir d'enseigner évident.

Enseigner Ada

Daniel Feneuille
http://d.feneuille.free.fr/enseignerada.htm

À l'initiative de l'association Ada-France, ce texte est une réflexion et une synthèse sur les qualités pédagogiques et professionnelles du langage de programmation Ada.

Daniel Feneuille enseigne Ada depuis près de 15 ans. Passionné par ce langage, ses étudiants l'ont baptisé *l'Adatollah*. Il faut croire qu'il a bien su communiquer cet enthousiasme car il me semble qu'au moins un de ses anciens élèves est un développeur bien connu d'AdaCore !

File structures with Ada

Nancy E. Miller & Charles G. Petersen
Alan Apt
ISBN 0-8053-0440-1
http://www2.netdoor.com/~petersen/nembooks www.aw-bc.com

Ce livre est la référence pour implémenter ISAM ou VSAM grâce à des algorithmes de type Btree ou B+Tree. Ce livre est une perle rare.

Les sources des exemples de l'ouvrage sont disponibles, avec la permission des auteurs, dans '`aide-doc-1.03.tar.bz2`'.

Liens internet

Sites Ada

Ada Core Technologies	http://www.adacore.com
GNAT compiler repository	ftp://cs.nyu.edu/pub/gnat
Ada Libre software	http://libre.act-europe.fr
AdaWorld	http://www.adaworld.com
AdaPower	http://www.adapower.com
AdaHome	http://www.adahome.com
Ada IC	http://www.adaic.com
Adalog	http://www.adalog.fr
Académie de l'US Air Force	http://www.usafa.af.mil
École d'ingénieurs de Genève	http://www.eig.unige.ch
Ludovic Brenta	http://users.skynet.be/ludovic.brenta
Gautier de Montmollin	http://homepage.sunrise.ch/mysunrise/gdm/
Pascal Obry	http://perso.wanadoo.fr/pascal.obry
Thomas Quinot	http://www.infres.enst.fr/~quinot/
Samuel Tardieu	http://www.rfc1149.net/sam

Ada scripting lib	http://adacl.sourceforge.net
Gnade	http://gnade.sourceforge.net
MaRTE OS	http://marte.unican.es

Mailing lists et Newsgroups Ada

Mailing list Ada-France (fr)	http://www.ada-france.org/mailman/listinfo/ada-france
Mailing lists Act (en)	http://libre.act-europe.fr
Mailing list Aide (fr)	aide@rochebrune.org
Newsgroup Ada (fr)	news://fr.comp.lang.ada
Newsgroup Ada (en)	news://comp.lang.ada

Outils bureautiques

The Gimp	http://gimp.org
Dia	http://www.lysator.liu.se/~alla/dia
Ghostscript	http://www.cs.wisc.edu/~ghost
Texlive	http://www.texlive.org
Texlive (fr)	http://www.tug.org/texlive
Texlive windows	http://www.ctan.org/tex-archive/systems/win32/fptex
OpenOffice Writer et Draw	http://www.openoffice.org
Mozilla	http://www.mozilla.org

Cryptographie

Généralités	http://lambda.eu.org
Sources	http://www.ussrback.com/crypto/tree.html

Divers

Open source for Windows	http://osswin.sourceforge.net

Standards

ECMA	http://www.ecma.ch
ETSI	http://www.etsi.org
IEC	http://www.iec.ch
ITU	http://www.itu.int
NCS	http://www.ncs.gov/ncs/html/library.html
US DOD online (mil std)	http://www.dsp.dla.mil/onlinedocs.htm

Annexe D Contributions, versions & évolutions

Contributions

Afsys

La société Bordelaise Afsys, conseil en organisation des systèmes informatiques, a hébergé gracieusement AIDE 1.02 (75 Mo) et tous ses composants (450Mo) de 2004 à 2005.

Adaworld

Stéphane Richard, webmaster de AdaWorld, http://www.adaworld.com, a participé à la traduction du manuel en anglais.

Daniel Feneuille

Daniel Feneuille a participé à la correction orthographique et grammaticale de la version française du manuel et a également contribué, par ses conseils pédagogiques, à l'amélioration générale de ce dernier.

Versions

GNAT 3.13p (2002)

0.10 - 01/12/01	commun	Version initiale 0.x (GNAT 3.13p, GtkAda 1.3 et Emacs/Glide).
0.20 - 22/12/01	commun ide	Simplification des chemins Mise à jour Emacs et Gvd
0.30 - 23/12/01	ide	Suppression des dépendances registry et modifications d'Adagide.

0.40 - 30/12/01	commun gnat	Amélioration de la documentation et de l'arborescence. Suppression de l'anomalie crt2.o, validation compilation C.
0.50 - 05/01/02	commun doc	Réorganisation de l'arborescence, finalisation de la distribution. Amélioration de la documentation
0.51 - 13/01/02	doc	Amélioration du lisezmoi.txt (Daniel Feneuille).
0.52 - 11/02/02	ide doc	Nouvelle version d'ada-mode compatible avec Emacs v21 Corrections du lisezmoi.txt et de l'installation.txt (François Bergeret). Intégration de la traduction du tutoriel de Glide (Daniel Feneuille).
0.60 - 18/03/02	commun	Validation NT4, 2K, 9x, validation sur différentes arborescences et différents disques. Procédure d'installation pour windows 9x.
	doc	Amélioration du lisezmoi.txt et de installation.txt

GCC 3.2 (2003)

0.70 - 23/02/03	commun	Amélioration de setada.cmd pour Windows 2K et Xp. Validation expérimentale de GtkAda 2 sous Windows 2K avec Gcc 3.2
	doc	Amélioration de installation.txt.
0.74 - 12/04/03	ide doc	Intégration d'Emacs/Glide avec GVD. Traduction de installation.txt en anglais.

GNAT 3.15p (2004)

1.00 - 01/02/04	commun	Version initiale 1.00 (GNAT 3.15p, GtkAda 2.2, Glide 3.15p avec Emacs 21.1.1 et GPS 1.4).
1.02 - 08/07/04	commun doc web	Correction d'anomalies mineures. Amélioration de la documentation des exemples. Mise à jour du site Web et du site de téléchargement.

GNAT 3.15p (2005)

1.03 - 10/08/05	commun	Un programme d'installation multi-volumes a été créé, avec installation des icônes sur le bureau. Amélioration de la procédure de lancement de AIDE : il n'existe désormais plus qu'un seul fichier de configuration pour GPS, Glide et MSys. Importante simplification du packaging : AIDE est désormais réduit à deux ensembles : AIDE lui-même et AIDE-SRC pour la collection de sources. Nouvelle version 2.1 de GPS. Un jeu d'utilitaires pour graver des CD-R et CD-RW a été ajoûté (cdrecord, mkisofs, etc...). Un jeu d'utilitaires basé sur AFIO a été ajoûté pour effectuer des sauvegardes. Adabrowse, doublon d'Adadoc, a été supprimé de la distribution.

lib	Opengl a été inclus dans AIDE avec des exemples variés et testés.
doc	Création d'un chapitre documentation dans le manuel, la procédure d'installation de Tex a été mieux documentée après de nombreux tests. Les exemples de programme et les exemples de documentations ont été mis à jour. Mise à jour de l'url pour obtenir fpTeX. Mise à jour de recode (v3.4 vers v3.6). Amélioration de l'encodage Latin-9 (ISO-8859-15) dans Glide. Amélioration du fichier .emacs et de l'impression. La traduction en anglais du manuel a commencé. Le jeu de manuels de référence Ada95 a été ajouté. Une section contribution a été créée. La section version a été améliorée. Remplacement de ACT-Europe par AdaCore.
texinfo	Un module Texinfo a été ajouté au générateur de documentation AdaDoc.

Évolutions

Les évolutions prioritaires sont en gras.

Module	Libellé
Aide	Créer une version LIVE-AIDE, totalement opérationnelle sur CD-ROM, sans installation sur le disque dur. Seul le répertoire de l'utilisateur est créé sur le disque, ne nécessitant que quelques Ko).
GtkAda	Améliorer le support (Glide & compilation).
Glide	Documenter un exemple d'utilisation des fichiers projets. Implémenter ispell (ou aspell) pour la correction orthographique. Aujourd'hui, ispell ne fonctionne que dans une fenêtre CMD, ni à l'intérieur de EMACS, ni dans RXVT.
Doc	**Traduire ce document en anglais.** Renseigner les programmes d'initiation. Symétriser "questions courantes" et "outils utilisés".
V04	**Continuer le développement de V04**. Documenter V04 via AdaDoc.