Guide de la Compilation de Gcc sous windows avec MinGW

par Philippe Dunski (http://philippe-dunski.developpez.com/)

Date de publication : 22/09/2007

Dernière mise à jour : 05/09/2010

Gcc est une collection de compilateurs pouvant gérer de nombreux langages qui évoluent très rapidement. Si vite que les équipes qui le portent sous Windows n'arrivent pas à suivre le rythme.

C'est la raison pour laquelle cet article vous guidera parmis les étapes à suivre pour compiler la dernière version depuis les sources.

I - Introduction....4
I-A - Gcc, qu'est-ce que c'est?....4
I-B - MinGW, qu'est-ce que c'est?....4
I-C - GNU..., mais c'est linux?....4
I-D - Pourquoi compiler les sources?....4
I-E - Pourquoi encore compiler en 32bits?....5
I-F - Pourquoi MinGW et non cygwin?....5
I-G - mingw-get : le nouvel installateur....6
II - Note sur les versions de développement....6
II-A - Bison....6
II-B - Flex....6
II-C - TeTex....6
II-D - Si vous prenez les versions de développement....6
III - MSYS: un système linux sous windows....7
III-A - Choisir sa méthode d'installation....7
III-A-1 - La liste des archives....7
III-A-2 - L'installation entièrement manuelle....8
III-A-3 - Un installateur "tout en un": mingw-get....8
III-A-3-a - L'installation en ligne de commande....9
III-A-3-a-i - Surveillez la sortie de mingw-get....10
III-A-3-b - L'installation avec l'interface graphique....10
III-A-3-b-i - La page d'accueil....10
III-A-3-b-ii - L'accord de la license....10
III-A-3-b-iii - Choix du dossier de destination....10
III-A-3-b-iv - Créer un raccourci ?....11
III-A-3-b-v - Sélection des parties à installer....11
III-A-3-c - Derniers préparatifs....12
III-B - Premiers contacts....14
III-B-1 - La console sous linux....14
III-B-2 - Bienvenue sous linux....15
III-B-3 - Créer les dossiers de travail....16
III-B-4 - L'arborescence finale....16
III-B-5 - Convention pour les codes....16
III-B-6 - Configurer les chemins d'accès....17
IV - Obtenir les sources....17
IV-A - Les versions testées....18
IV-A-1 - GMP-5.0.1....18
IV-A-2 - MPFR-3.0.0....19
IV-A-3 - MPC-0.8. "Dianthus deltoides"....19
IV-A-4 - Binutils-2.20.51....19
IV-A-5 - GCC-4.6.0....19
IV-B - Décompresser les sources....19
IV-B-1 - Comprendre les commandes....19
IV-B-1-a - Renommer le dossier de sources pour w32api....20
IV-B-1-b - Copier les sources de GMP, MPFR et MPC....20
IV-B-1-b-i - Comprendre la commande....20
IV-B-1-b-ii - Corriger quelques fichiers....21
IV-B-1-b-ii-- - Si vous voulez activer le support de Ada....21
V - Compiler binutils....22
V-A - Configurer la compilation....22
V-B - Comprendre les commandes....23
V-B-1 - Les problèmes posés par l'internationnalisation....23
V-B-2 - Profiter de l'option --with-sysroot....23
V-B-3 - Au sujet des cibles et des hôtes....24
V-C - Lancer la compilation....24
VI - Compiler w32-api et MinGW runtime....25
VII - Compiler Gcc....25
VII-A - Si vous voulez activer le support de Ada....25

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

VII-B - Configurer Gcc....25
VII-B-1 - Comprendre la commande....26
VII-B-2 - Comprendre le processus de compilation....26
VII-C - Compiler Gcc première (selon conditions)....27
VII-C-1 - Comprendre les commandes....29
VII-C-2 - Terminer l'installation de pthreads....29
VII-C-3 - Si vous souhaitez activer le support de Java....29
VII-C-3-a - Installer unzip....30
VII-C-4 - Retourner dans le dossier de compilation de Gcc....31
VII-D - Compiler Gcc deuxième....31
VIII - Recompiler Binutils....31
IX - Finaliser le tout....31
IX-A - Quelques outils supplémentares....31
IX-B - Un peu de ménage....32
X - Liens et remerciements....32
X-A - Trouver les archives....32
X-B - Sources d'inspiration....32
X-C - Liens connexes....32
X-D - Remerciements....33

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - Introduction

La version originale de cette article est parue en 2007 et sert de base à d'autres articles orientés, entres autres, sur la création d'un compilateur croisé.

Elle a en outre été reprise dans certains cours et tutoriels.

Depuis la première version, les techniques ont grandement évolué tant au niveau du développement des outils dont il est question ici que du fait de la généralisation des générations de processeurs 64bits.

S'il peut paraitre "anachronique" de compiler sciemment un compilateur 32 bits, il était cependant impensable de supprimer purement et simplement cet article.

Cet article est donc une mise à jour de la version originelle destiné à prendre cette évolution en compte.

I-A - Gcc, qu'est-ce que c'est?

Gcc est l'acronyme de GNU Compilers Collection. Il s'agit en fait d'une collection qui regroupe des compilateurs supportant de nombreux langages, tels que le C, le C++, l'ObjectiveC, l'ObjectiveC++, le Treelang (un langage expérimental dont la principale raison d'être est de servir de support au développement de GCC), l'Ada, le FORTRAN et même le Java.

Il fournit en outre les différentes bibliothèques standard pour ces langages.

I-B - MinGW, qu'est-ce que c'est?

MinGW est l'acronyme de **Min**imalist **G**cc for **W**indows.

Le format d'exécutables sous Windows (**P**ortable **E**xecutable) est très différent de ceux qui sont utilisés sous linux. Il faut donc ajouter le support de ce format a Gcc pour qu'il soit capable de générer des programmes exécutables sous cet OS et de lier des bibliothèques.

Le but du projet est, tout simplement, de fournir cette adaptation de Gcc sous windows.

I-C - GNU..., mais c'est linux?

Effectivement, GNU (Gnu's Not Unix) est principalement associé à linux. Et c'est sans doute ce qui fait que Gcc évolue si rapidement: En effet, les versions en développement de Gcc apportent déjà le support d'une partie au moins de la prochaine norme de C++ (C++0x) qui n'est pourtant pas attendue, par les plus optimistes, avant 2011 ou 2012.

Ce qui explique aussi dans une certaine mesure que les mainteneurs de MinGW aient autant de mal à garder les versions aussi à jour que ce qui existe sous linux.

I-D - Pourquoi compiler les sources?

Les raisons qui pourraient vous faire décider de compiler les sources de Gcc sont multiples:

Si, au moment d'écrire ces lignes, l'équipe de MinGW propose une version "presque à jour" de Gcc (la version officielle étant 4.5.1 et la version proposée par MinGW étant la 4.5.0), il arrive souvent que l'équipe de MinGW tarde à considérer certaines versions comme stable.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Cela peut aller jusqu'à remarquer des différences de versions primaires, MinGW n'ayant pendant tout un temps proposé que la version 3.4.5 comme version stable alors que Gcc en était déjà aux versions 4.2 et supérieures.

De plus, il est possible de disposer d'un "snapshot" hebdomadaire de la version de développement, et certains pourraient être intéressés par le fait d'en profiter.

En outre, il faut savoir que les versions binaires de MinGW sont généralement compilées de manière à être fonctionnelles sur la plupart des processeurs: "au mieux", sous de vieux Pentium II, "au pire" avec d'antiques 3x86.

Nombreux sont ceux qui pourraient, tout simplement, souhaiter disposer d'un compilateur optimisé pour l'ordinateur dont ils disposent.

Enfin, à l'heure actuelle, le compilateur fait la part belle aux DLL, qui devront fatalement être fournies avec vos exécutables afin de vous assurer de leur présence.

Vous pourriez souhaiter effectuer une liaison statique de vos projets, en étant cependant conscient du "surpoids" que cela peut occasionner pour ceux-ci

I-E - Pourquoi encore compiler en 32bits?

Peut-être vous demandez vous pourquoi il est, encore à l'heure actuelle, nécessaire de compiler son compilateur en 32 bits plutôt que de créer directement son compilateur croisé ou son compilateur 64 bits.

Je vois, personnellement, deux bonnes raisons pour agir de la sorte:

Primo, il y a encore de nombreux ordinateurs 32bits en circulation. Vous apprécierez peut-être de pouvoir utiliser la même version de compilateur sur ces machines "d'un autre temps", plutôt que de devoir revenir à une version de Gcc encore plus ancienne.

Secundo, il est toujours plus facile de compiler un compilateur en limitant le nombre de barrières qu'il faut passer, et en les passant les unes après les autres.

Il vous sera donc plus facile de commencer par mettre à jour votre compilateur plutôt que de vouloir passer directement de la version 3.4.5 de Gcc à sa version 4.6.0 en 64bits, avec le support multilib, ou de compiler la version 4.6.0 comme compilateur croisé pour linux si vous utilisez déjà la dite version 4.6.0.

I-F - Pourquoi MinGW et non cygwin?

Deux raisons m'ont fait choisir une compilation sous MinGW plutôt que sous cygwin:

La première est que cygwin est un émulateur linux qui fournit un compilateur croisé pour windows, et qu'il nécessite donc une optique de travail plus "orientée" vers linux. Vous me direz sans doute que vous devrez de toutes manières passer par un émulateur linux pour le compiler, mais cela pourra rester votre seule expérience sous cet environnement si vous le souhaitez, alors que vous devriez y retourner chaque fois que vous voudriez compiler sous cygwin.

La deuxième tient tout simplement au fait que la bibliothèque dynamique cygwin.dll est fournie sous licence GNU/GPL, et que toute utilisation de cette bibliothèque vous forcerait à... créer des applications sous licence GNU/GPL.

Bien qu'étant personnellement fort attaché à la licence GNU/GPL, je peux concevoir que ce ne soit pas le cas de tout le monde. C'est la raison pour laquelle j'ai décidé de fournir un compilateur natif qui vous permette de choisir votre propre licence selon vos propres goûts en la matière.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I-G - mingw-get : le nouvel installateur

Jusqu'il y a peu, il était relativement facile, en téléchargeant quelques archives à peine, d'obtenir un système minimaliste parfaitement équilibré et fonctionnel.

Pour des raisons qui ne tiennent qu'à elle, l'équipe de développement de MinGW a décidé de ne plus fournir de "paquet global" et de séparer strictement tous les différents outils.

La liste des archives à récupérer pour obtenir quelque chose de fonctionnel est de ce fait devenue particulièrement impressionnante (cf. : la liste des archives).

Afin de permettre de gérer correctement les dépendances, l'équipe a donc mis au point une nouvelle application permettant de récupérer tout ce qu'il faut : mingw-get.

II - Note sur les versions de développement

Par version de développement, il faut comprendre toute version n'étant pas estampillée "release", à savoir:

- Les révisions disponibles sur le système de gestion concurrentes (CVS, SVN, ...)
- Les *snapshots* hebdomadaires

Ces versions nécessitent de créer quantité de fichiers à l'aide de Bison, Flex ou TeTex.

II-A - Bison

Bison est un générateur universel d'analyseurs lexicaux qui convertit une grammaire sans contexte annotée en analyseur de LALR **(1)** ou de GLR pour cette grammaire.

Il est indispensable pour permettre la compilation de binutils.

II-B - Flex

(F)lex est un analyseur syntaxique souvent utilisé en conjonction avec Bison

II-C - TeTex

TeTex est un outil souvent employé sous unixoïdes pour générer de la documentation en différents formats.

Ce sera bien souvent l'outil le plus compliqué à obtenir à faire fonctionner au travers de MSYS.

II-D - Si vous prenez les versions de développement

Si vous décidez de vous tourner vers les versions de développement, vous ne pourrez très certainement pas vous passer de Bison et de Flex car les fichiers générés par ces outils ne sont purement et simplement pas fournis avec les sources.

Pour ce qui est de l'absence de TeTex, je vous indiquerai en temps utiles comment faire pour éviter que la compilation ne sorte sur une erreur. Ce ne sera bien sur qu'un "pis aller" dans le sens où vous ne disposerez pas de la documentation automatiquement générée.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III - MSYS: un système linux sous windows

GCC est une collection d'outils venant tout droit du monde de linux. A ce titre, il est particulièrement attaché aux outils GNU pour sa compilation.

Je ne prétend pas qu'il est impossible de le compiler sans les outils GNU prévus à cet effet (M4, make, autoconf...) mais je prétend réellement que le plus facile reste quand même d'utiliser ces outils.

L'idéal est donc de disposer d'un système qui puisse émuler linux alors que nous resterons bien au chaud sous windows. L'outil que je vous propose d'utiliser pour ce faire est fourni par l'équipe de MinGW et s'appelle MSYS.

MSYS est l'abréviation de Minimal SYStem, et va vous permettre de disposer de l'ensemble des outils qui vous permettront d'effectuer la compilation.

Vous pouvez le télécharger librement sur le site de MinGW ou sur l'un des miroirs mis à disposition par Sourceforge.

Le plus facile à l'heure actuelle pour se procurer l'ensemble des outils nécessaire est de passer par un outil qui ne fonctionne actuellement qu'en ligne de commande : mingw-get.

III-A - Choisir sa méthode d'installation

Vous disposez de trois possibilité pour installer les outils de base que sont la suite binutils, le compilateur Gcc et MSYS:

1. Le téléchargement et l'installation entièrement manuelle des parties nécessaires
2. l'utilisation de mingw-get en ligne de commandes
3. l'utilisation d'un installateur graphique de mingw-get

Toutes trois auront pour objectif de télécharger et de décompresser une liste assez impressionnante d'archives dans un répertoire donné.

III-A-1 - La liste des archives

Afin d'obtenir le minimum vital pour nous permettre de compiler Gcc par nous même, il nous faudra deux paquetages distinct, composés tous deux d'un grand nombre d'archive:

Le paquetage MinGW qui fournit les outils de compilations indispensables et qui est composé de

- gcc-ada-4.5.0-1-mingw32-bin.tar.lzma (requis uniquement si vous voulez activer le support de Ada)
- gcc-core-4.5.0-1-mingw32-bin.tar.lzma
- libexpat-2.0.1-1-mingw32-dll-1.tar.gz
- libgcc-4.5.0-1-mingw32-dll-1.tar.lzma
- libgmp-5.0.1-1-mingw32-dll-10.tar.lzma
- libgnat-4.5.0-1-mingw32-dll-4.5.tar.lzma(requis uniquement si vous voulez activer le support de Ada)
- libmpc-0.8.1-1-mingw32-dll-2.tar.lzma
- libmpfr-2.4.1-1-mingw32-dll-1.tar.lzma
- libpopt-1.15-2-msys-1.0.13-dll-0.tar.lzma
- libpthread-2.8.0-3-mingw32-dll-2.tar.lzma
- mingwrt-3.18-mingw32-dev.tar.gz
- w32api-3.14-mingw32-dev.tar.gz
- binutils-2.20.51-1-mingw32-bin.tar.lzma

Le paquetage msys fournit quant à lui l'ensemble des outils nécessaires pour disposer d'un système linux minimaliste suffisant pour travailler. Il est composé des archives suivantes:

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- bash-3.1.17-3-msys-1.0.13-bin.tar.lzma
- bzip2-1.0.5-2-msys-1.0.13-bin.tar.lzma
- coreutils-5.97-3-msys-1.0.13-bin.tar.lzma
- cygutils-dos2unix-1.3.4-4-msys-1.0.13-bin.tar.lzma
- diffutils-2.8.7.20071206cvs-3-msys-1.0.13-bin.tar.lzma
- file-5.04-1-msys-1.0.13-bin.tar.lzma
- findutils-4.4.2-2-msys-1.0.13-bin.tar.lzma
- gawk-3.1.7-2-msys-1.0.13-bin.tar.lzma
- grep-2.5.4-2-msys-1.0.13-bin.tar.lzma
- gzip-1.3.12-2-msys-1.0.13-bin.tar.lzma
- less-436-2-msys-1.0.13-bin.tar.lzma
- libiconv-1.13.1-2-msys-1.0.13-dll-2.tar.lzma
- libintl-0.17-2-msys-dll-8.tar.lzma
- liblzma-4.999.9beta_20100401-1-msys-1.0.13-dll-1.tar.gz
- libmagic-5.04-1-msys-1.0.13-dll-1.tar.lzma
- libregex-1.20090805-2-msys-1.0.13-dll-1.tar.lzma
- libssp-4.5.0-1-mingw32-dll-0.tar.lzma
- libtermcap-0.20050421_1-2-msys-1.0.13-dll-0.tar.lzma
- make-3.81-3-msys-1.0.13-bin.tar.lzma
- msysCORE-1.0.15-1-msys-1.0.15-bin.tar.lzma
- msysCORE-1.0.15-1-msys-1.0.15-ext.tar.lzma
- sed-4.2.1-2-msys-1.0.13-bin.tar.lzma
- tar-1.23-1-msys-1.0.13-bin.tar.lzma
- termcap-0.20050421_1-2-msys-1.0.13-bin.tar.lzma
- texinfo-4.13a-2-msys-1.0.13-bin.tar.lzma
- xz-4.999.9beta_20100401-1-msys-1.0.13-bin.tar.gz
- zlib-1.2.3-2-msys-1.0.13-dll.tar.lzma

En effet, l'équipe de MinGW semble avoir décidé de ne plus fournir d'archive globale pour leur système, et il faut donc veiller à récupérer l'ensemble des outils et des dépendances dont nous aurons besoin.

L'ensemble de ces archives est disponible sur le site de MinGW.

Les versions présentées ici sont celles utilisées au moment d'écrire ces lignes.

III-A-2 - L'installation entièrement manuelle

Vous pouvez parfaitement décider de télécharger l'ensemble de ces archives une à une et de les décompresser dans le dossier de votre choix.

Cependant, vous aurez alors besoin d'un gestionnaire d'archive capable de traiter les formats lzma et tar tel que winrar ou 7zip.

Comme la liste est relativement impressionnante et que vous ne disposez peut être pas d'un gestionnaire d'archives approprié, il sera surement plus simple d'utiliser mingw wget.

III-A-3 - Un installateur "tout en un": mingw-get

La manière la plus simple de démarrer est sans doute d'utiliser l'utilitaire mingw-get fourni sur le site de MinGW

Deux versions de celui-ci sont actuellement disponibles:

- Une version en ligne de commande : mingw-get

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- une version constituée d'un installateur classique : mingw-get-inst-YYYYMMDD.exe (YYYYMMDD représentant une date au format Année Mois Jour)

La version en ligne de commande est fournie sous la forme d'une archive "classique", au format .tar.gz et au format .zip.

Vous pouvez sans problème choisir le format .zip qui est naturellement supporté sous windows, le format .tar.gz nécessitant d'être ouvert avec des outils plus perfectionnés (étant donné que nous ne disposons pas encore de MSYS) tel que 7zip, winrar ou autres.

III-A-3-a - L'installation en ligne de commande

Décompressez l'archive dans le dossier de votre choix.

Mingw-get installera le compilateur dans son dossier courent et, comme nous ne souhaitons l'utiliser que de manière temporaire (le temps d'effectuer notre compilation), je vous conseillerais de le placer dans le dossier c:\temp, que ne considérerai comme étant le répertoire utilisé.

Lancez une "invite de commande" en passant par le menu "Démarer->accessoires->Invite de commandes ou par tout raccourcis vers celle-ci dont vous disposeriez.

Entrez dans le dossier dans lequel vous venez de décompresser l'archive de mingw-get et rajoutez le répertoire c:\temp\bin (qui est le dossier dans lequel se trouve effectivement mingw-get.exe) à votre variable PATH

```
c:\Users\philippe>cd \temp
c:\temp set PATH=%PATH%;c:\temp\bin
```

Demandez à mingw-get d'installer gcc-core et la base nécessaire pour MSYS, et éventuellement de Ada.

Nous demandons également l'installation de quelque paquetages propres à msys qui seront nécessaire pour le processus de compilation des différents outils: flex, bison et texinfo.

```
c:\temp\bin>mingw-get install gcc-core ada msys-bison msys-texinfo msys-flex
```

A partir d'ici, vous ne devriez plus avoir besoin de l'invite de commandes de windows.

Le terme ada n'est, bien évidemment, nécessaire que si vous souhaitez compiler Gcc avec le support de ce langage activé, car le compilateur Ada est écrit en Ada, alors que tous les autres sont écrits en C.
Autrement, vous pouvez vous contenter de lancer la commande
c:\temp\bin>mingw-get install gcc-core msys-base

Vous pourriez également décider d'installer d'autres langages tels que c++, objc ou fortran, mais ils ne sont pas indispensables pour l'usage que nous prévoyons d'en faire.

Vous ne pourrez pas lancer mingw-get sans lui fournir d'arguments: Toute tentative en ce sens aurait pour résultat d'essayer de lancer une interface graphique qui n'est pas fournie pour le moment.

Mingw-get téléchargera dans un premier temps une série de fichiers xml décrivant les différents paquets et leurs dépendances, puis les paquets demandés (et leurs dépendances), et terminera en les décompressant dans son dossier racine.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Mingw-get récupérera également quelques archives supplémentaires qui ne sont pas absolument nécessaire comme celles relatives à la licence d'utilisation ou à la documentation.

III-A-3-a-i - Surveillez la sortie de mingw-get

Il n'est pas impossible que mingw-get échoue dans sa tentative d'obtenir l'un ou l'autre fichier xml ou l'une ou l'autre archive.

Il émettra alors un message d'erreur, mais continuera à travailler comme si de rien n'était.

Laissez le simplement continuer son travail jusqu'au bout, et relancez la commande par la suite.

Ne tentez en aucun cas d'annuler la commande (avec CTRL + C) si vous obtenez une telle erreur: Cela aurait comme résultat le fait de renommer l'exécutable mingw-get.exe se trouvant dans le dossier c:\temp\bin en mingw-get.exe~ et la DLL mingw-get.dll se trouvant dans le dossier c:\temp\libexec\mingw-get en mingw-get.dll~, avec comme conséquence directe l'impossibilité de relancer la commande par la suite avant d'avoir rendu leur nom d'origine à ces fichiers.

III-A-3-b - L'installation avec l'interface graphique

Au moment d'écrire ces lignes, la version graphique de l'installateur est nommée mingw-get-inst-20100831.exe.

Il suffit, après l'avoir téléchargé, de le lancer de la manière habituelle.

III-A-3-b-i - La page d'accueil

Lorsque nous lancerons l'installateur, il vous présentera une page d'accueil finalement tout ce qu'il y a de plus classique.

Cliquons sur next sans nous en inquiéter.

III-A-3-b-ii - L'accord de la license

Nous arrivons sur la page relative à l'acceptation de la license d'utilisation.

Le texte affiché est celui de la GNU/GPL (version 3 au moment d'écrire ces lignes)

Nous ne pourrons pas aller plus loin avant d'avoir sélectionné le bouton "I accept the agrement".

III-A-3-b-iii - Choix du dossier de destination

L'installateur nous demandera dans quel dossier vous souhaitez installer mingw-get. Il propose de placer mingw-get dans le dossier c:\MinGW par défaut.

Comme mingw-get placera les outils de compilation dans le dossier dans lequel il se trouve, et que nous ne souhaitons pas garder ceux-ci plus que nécessaire, je vous conseillerais de changer le dossier d'installation pour c:\temp, avant de cliquer sur suivant

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-A-3-b-iv - Créer un raccourci ?

Vous aurez ensuite l'occasion de choisir de créer ou non un raccourcis dans le menu démarrer.

Vous êtes bien évidemment libre de choisir de le faire ou non, mais pensez bien qu'il nous faudra lancer MSYS.

III-A-3-b-v - Sélection des parties à installer

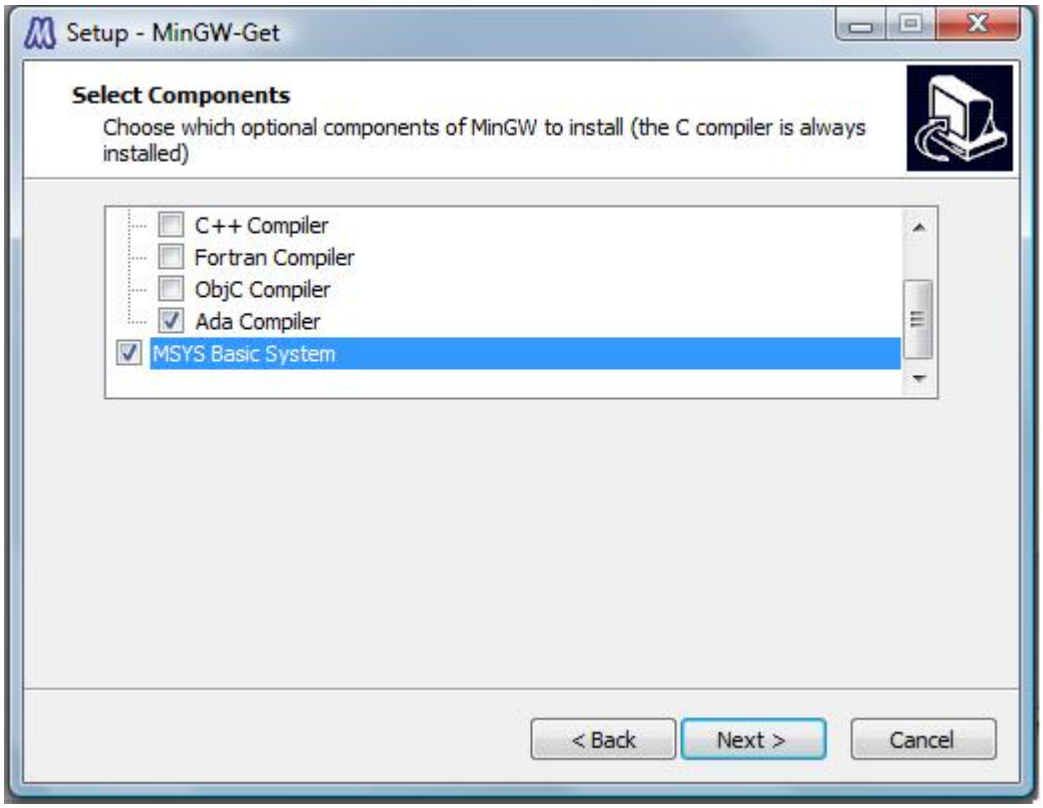
Il est maintenant temps de sélectionner les parties que nous souhaitons installer.

Par défaut, la seule partie sélectionnée, que vous ne pouvez au demeurant pas désélectionner, est gcc-core, qui correspond au compilateur C.

Comme nous voulons utiliser MSYS pour compiler Gcc, il faudra descendre tout en bas et veiller à sélectionner MSYS Base.

Si vous envisagez de demander le support de Ada lors de la compilation, vous devrez également sélectionner cette partie car Ada ne peut être compilé... que par lui-même.

La fenêtre présentant la sélection minimale que vous devrez avoir (support d'Ada inclus) ressemblera alors à



sélection des parties à installer

Lorsque vous cliquerez sur suivant, l'installateur lancera automatiquement mingw-get en lui passant les options relatives aux parties que vous aurez séletionnées.

Mingw-get téléchargera alors l'ensemble des archives nécessaires et les décompressera dans son dossier d'installation.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-A-3-c - Derniers préparatifs

A ce stade, vous disposez de l'ensemble des outils nécessaires pour envisager la compilation de Gcc.

Cependant, mingw-get aura tout installé dans son répertoire racine, ce qui vous donnera une arborescence proche de

```
temp
|     |->bin                        Gcc et binutils se trouvent ici
|     |->include                    Les fichiers d'en-tête de gcc se trouvent ici
|     |->lib                        Les bibliothèques statiques
|     |->msys
|     |     |->0.1                  dossier connu pour être / sous MSYS
|     |     |     |->home
|     |     |     |     |->votre nom   dossier connu pour être ~ sous MSYS
|     |                             il s'agit de votre "dossier personnel" sous
|     |                             linux
|     |->var            dossier contenant essentiellement les fichiers utilisés
                        et récupérés par mingw-get
```

Cela peut être tout à fait satisfaisant si vous ne souhaitez pas garder msys après avoir compilé Gcc (entre autres si vous n'envisager pas d'utiliser les autotools pour générer des scripts de configuration et de compilaton).

Lorsque la compilation sera terminée, vous pourrez en effet décider de supprimer purement et simplement l'ensemble du contenu de temp sans vous poser de question.

Cependant, si vous souhaitez garder msys "sous la main" après avoir supprimé cette version temporaire de MinGW, je vous conseille de faire remonter le dossier msys d'un cran (pour le placer en racine de votre disque dur).

Si vous avez une ligne de commandes ouvertes (par exemple celle que vous venez d'utiliser pour lancer mingw-get), vous pouvez le faire rapidement en lançant la commande.

```
c:\temp move c:\temp\msys c:\
```

Si vous effectuez ce déplacement et que vous souhaitez par la suite invoquer à nouveau mingw-get pour ajouter des outils supplémentaires à MSYS, ce dernier les placera dans le dossier qu'il considère comme étant le dossiser d'installation de MSYS, en l'occurrence c:\temp\msys. Vous devrez donc veiller à déplacer ces fichiers dans le dossier réel dans lequel se trouvent les fichiers relatifs à MSYS.

Je considérerai par la suite que le dossier msys se trouve bel et bien à la racine du disque C

Si vous avez utilisé l'installateur graphique, le raccourcis placé dans votre menu démarrer deviendra invalide du fait du déplacement que nous venons d'effectuer. N'oubliez pas de mettre la cible de celui-ci à jour cliquant droit dessus puis en sélectionnant "propriétés"

Si vous n'avez pas utilisé l'installateur graphique mais que vous avez effectué une installation complètement manuelle ou que vous avez utilisé mingw-get en ligne de commande, vous pouvez créer un raccourci vers c:\msys\0.1\msys.bat sur le bureau en cliquant droit dessus puis en sélectionnant "envoyer vers-> Bureau (créer un raccourcis)"

Vous pouvez bien entendu également utiliser l'explorateur windows si vous le préférez.

Avant de lancer MSYS, nous allons nous assurer une dernière fois que tout est prêt:

Si vous n'avez pas encore créé un dossier MinGW à la racine de c:\, c'est le moment de le faire.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Linux utilise un fichier nommé fstab (**F**ile **S**ystem **TAB**le) qui nous sera très utile.

dans le dossier c:/msys/1.0/etc, vous en trouverez un exemple nommé fstab.sample. Nous allons nous en servir.

Comme il s'agit d'un simple fichier texte, vous pouvez demander à l'ouvrir avec n'importe quel éditeur de texte que vous appréciez. Gardez cependant en tête le fait que c'est un fichier créé sous linux... Cela signifie, entre autre, que les caractères de retour à la ligne sont représentés sous la forme CR, alors que sous windows, on utilise généralement la forme CRLF... Le résultat en est que, si votre éditeur de texte ne comprend pas cette représentation, vous verrez l'ensemble du texte sur une seule ligne, avec une série de carrés qui auraient dû être considérés comme des sauts de ligne.

Au pire, vous pouvez toujours décider d'utiliser WordPad (l'éditeur de texte RTF installé d'origine sous windows), qui vous le présentera de manière correcte.

contenu de fstab.sample

```
#fstab.sample
#This is a sample file for /etc/fstab.
#Currently /etc/fstab is only read during dll initialization.
#I will eventually watch the directory for changes and reread the file.
#The line format is simple in that you give the Win32 path, followed by one or
#more space or tab delimiter, followed by the mount point.  Mount points in
#typical UNIX environments must be a physical name on a drive before it can
#actually be used as a mount point.  In this implementation the "must exist"
#requirement isn't enforced, however, it will be an aide to such programs as
#find and readline's tab completion if it does exist.

#You can use a # as the first character on the line as a comment indicator.
#Blank lines are ignored.

#Win32_Path  Mount_Point
c:/mingw  /mingw
c:/ActiveState/perl /perl
```

Comme vous l'aurez surement remarqué, les lignes commençant par un # sont considérées comme des commentaires.

La ligne qui nous importe surtout est la ligne

```
c:/mingw  /mingw
```

Il faut veiller à ce que la première partie (c:/mingw) corresponde au dossier que vous avez prévu pour l'installation de la version compilée par vos soins.

Si vous disposez de perl, vous pouvez modifier la dernière ligne du fichier (celle qui commence par c:/ActivePerl/perl) pour lui permettre de refléter le dossier dans lequel perl est installé. Si vous n'avez pas perl, car ce n'est pas obligatoire pour notre usage, vous pouvez commenter la ligne en plaçant un # à son début.

Il ne vous reste plus qu'à enregistrer les modifications en renommant le fichier de fstab.sample en fstab (dans le dossier c:\msys\1.0\etc).

Vérifiez bien que windows ne vous rajoute pas l'extension .txt, car, sinon, il ne sera pas pris en compte
Si c'est le cas, renommez le en supprimant l'extension et acceptez l'avertissement que windows ne manquera pas de vous donner.

Si vous avez utilisé l'installateur graphique, mingw-get aura automatiquement créé le fichier fstab, mais en utilisant son dossier d'installation par défaut.
Il est préférable, pour la facilité, de modifier le fichier fstab comme indiqué plus haut.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Vérifiez enfin la présence d'un dossier nommé mingw dans c:\msys\1.0 et créez le en cas de besoin.

Ce dossier permettra à MSYS de faire effectivement le lien entre ce qu'il identifiera comme /mingw et le dossier réel de destination: c:\MinGW

III-B - Premiers contacts

III-B-1 - La console sous linux

Ceux qui se sont déjà frottés à linux, ou au monde Unix en général peuvent passer cette sous-section sans crainte, rien de nouveau n'apparaîtra dans ces lignes.

Pour les autres, ne vous en faîtes pas, il n'est nullement dans mes intentions de vous donner un cours complet sur linux - il faudrait plusieurs tutoriaux pour mener ce projet à terme - et ce n'est sûrement pas le but de celui-ci.

Le but de cette sous-section est, tout simplement, de vous signaler quelques différence notables entre ce que vous connaissez des commandes "DOS" et leur équivalents linux.

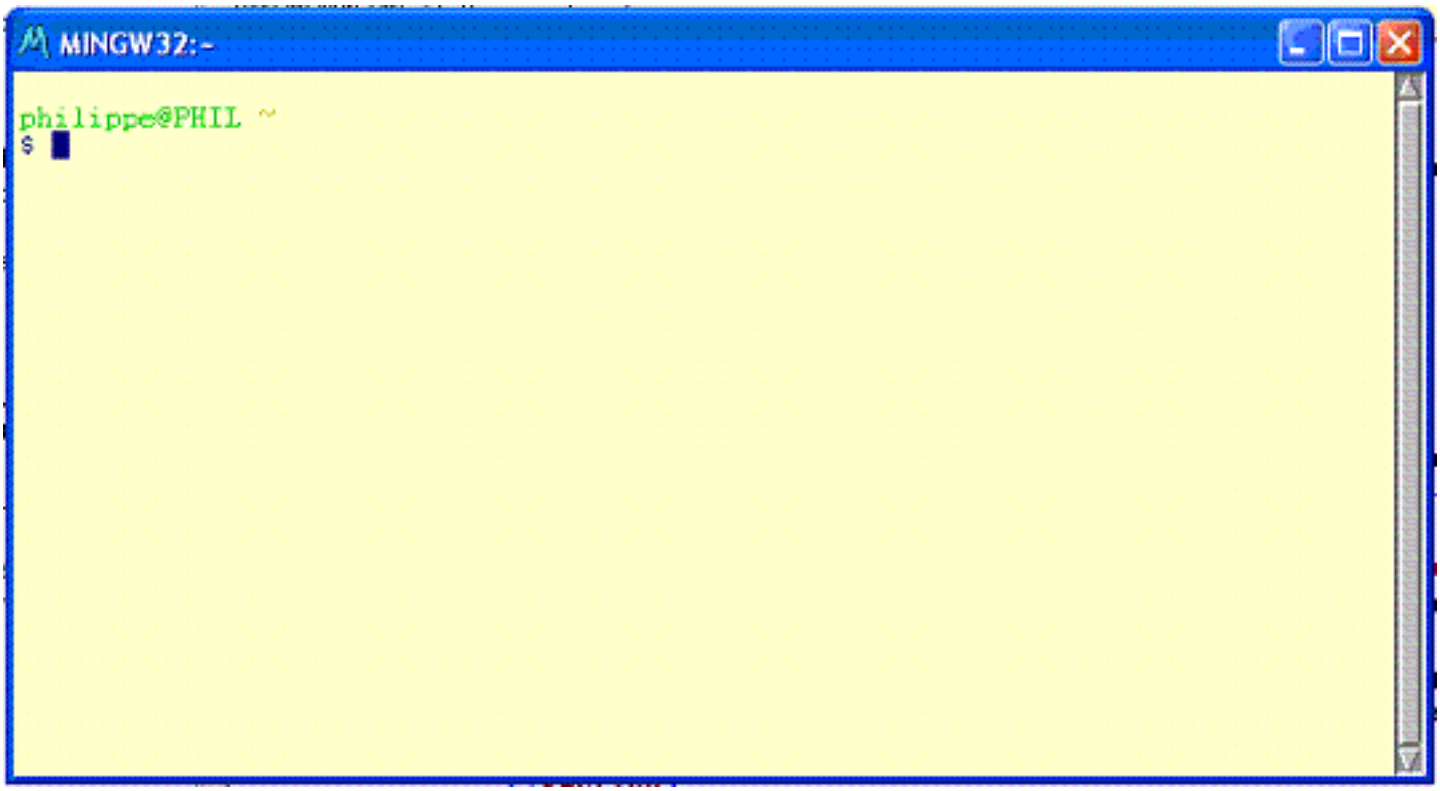
- Ainsi, les paramètres passés au commandes sous DOS sont précédées du /, alors qu'ils sont précédés d'un trait d'union - pour les versions courtes ou de deux traits d'union -- pour les versions longues.
- La très grosse majorité des commandes contient une aide en ligne. Pour y arriver, il suffit de faire suivre le nom de la commande par --help (ou -h si vous êtes fainéants)
- La console linux dispose d'un système d'auto-complétion très efficace; il permet au choix de compléter le nom d'un dossier, d'un fichier, ou d'une commande. Pour en profiter, il suffit d'appuyer sur la touche <tabulation>.
  Le premier appui dessus fera compléter le mot jusqu'à ce qu'il y ait "conflit": dans le cas de deux mots commençant par les mêmes lettres, la complétion s'arrêtera juste avant la première lettre qui n'est plus identique. Le deuxième appui affichera la liste des (dossiers, fichiers, instructions) qui correspond à ce qui a déjà été introduit.
- Linux utilise le "slash"(/) comme séparateur entre les dossiers et les fichiers et supporte "relativement mal" les espaces dans les noms. Il est donc "de bon ton" d'essayer autant que possible de les éviter. Pour accéder à un dossier ou à un fichier contenant des espaces, il faut faire précéder chaque espace d'un "back-slash" (\).
  Ceci dit, l'auto-complétion le rajoute automatiquement.
- La notion de "disque logique" est inconnue sous linux.
  Les partitions sont tout simplement rattachées à des dossiers qui se trouvent dans l'arborescence partant de la racine du système.
  Et la racine du système ne se trouve pas forcément... à la racine d'une partition.
  Ainsi, quand vous aurez lancé MSYS, la racine du système (/) se trouvera en réalité dans c:\msys\1.0, et si vous voulez quelque chose qui se trouve "au dessus" de c:\msys\1.0, il faudra y accéder au départ de / (ex: /c/mingw, /c/downloads)
- Alors que windows se base sur l'extension d'un fichier pour déterminer si c'est un exécutable ou non, linux regarde un bit particulier qui lui indique qu'il s'agit d'un exécutable. Cependant, étant donné que MSYS doit fonctionner sous windows, il est également sensible aux extensions exe et autres.
- Linux est "sensible à la casse" en ce qui concerne les noms de dossiers, de fichiers et de commandes. Ainsi, si vous souhaitez accéder au dossier c:\Program Files, il faudra écrire un P majuscule et non un p minuscule, autrement, même l'auto-complétion ne trouvera pas le dossier.
- Il y a une série d'instructions qui diffèrent sous DOS et sous linux.
  Il y a d'ailleurs une série d'instructions qui sont propres à linux. La liste des différences étant relativement longue, et comme vous n'aurez sans doute pas besoin d'une bonne partie d'entre elles, je me contenterai de vous expliquer les commandes nécessaires en temps utiles.
- Linux dispose d'un historique des commandes particulièrement efficace. Vous pouvez naviguer dans cette historique à l'aide des fleches de directions "haut" et "bas".
  Il n'y a aucune limite quant au nombre d'instructions de cette historique, et quand bien même vous quitteriez la console, l'historique est maintenue au prochain lancement.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-B-2 - Bienvenue sous linux

Il est maintenant temps de lancer MSYS et d'entrer pour la première fois sous cet environnement

Vous obtiendrez une console ressemblant à



Cette console vous fournit une quantité non négligeable d'informations qui sous la forme de

```
 UserName@server ~
$
```

- **Username** représente votre nom d'utilisateur
- **server** représente le nom du serveur sur lequel vous êtes connecté, et qui est typiquement, ici, le nom de l'ordinateur
- **~** représente le dossier actuel (ce symbole indique que vous êtes dans votre dossier personnel, qui sera dans le dossier <répertoire d'installation>/1.0/home/<votre nom>)
- **$** indique que vous êtes logué en tant qu'utilisateur simple (si vous aviez été logué en tant qu'administrateur, cela aurait été un #). Le fait que vous *travailliez éventuellement avec une session administrateur sous windows n'intervient absolument pas ici, mais, comme les droits accordés à l'utilisateur simple nous suffisent, il n'y a pas lieu de s'inquiéter*

MSYS ne connait pas la commande su, et ne dispose d'ailleurs d'aucun des fichiers de gestion de droits que l'on retrouve habituellement sous linux (/etc/passwd, /etc/group...). Il est donc totalement impossible d'envisager de se loguer en tant que "root", mais on dispose malgré tout de l'ensemble des droits. (quand on vous dit que c'est un système minimal :D)

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-B-3 - Créer les dossiers de travail

Par facilité, nous allons créer un dossier dans lequel se feront toutes les compilations et les différents dossiers de compilation avec la commande

```
$ mkdir -p build/binutils build/w32api build/mingwrt build/gcc
```

La commande mkdir permet, tout comme avec l'invite de commandes sous windows, de créer les dossiers dont les noms sont donnés. L'option p force à la création des dossiers parents si ceux-ci n'existent pas.

En effet, la plupart des outils supportent difficilement d'être compilés directement dans le dossier dans lequel se trouvent leurs sources. Cela a aussi l'avantage de laisser le dossier dans lequel se trouvent les sources "en l'état", et donc d'éviter des manipulations hasardeuses si on veut reprendre les choses depuis le début.

III-B-4 - L'arborescence finale

A ce stade, je considérerai que vous disposez d'une arborescence ressemblant à ceci:

```
c:
|    |->downloads                        <- dossier de téléchargements
|    |->MinGW                            <- dossier de destination finale
|    |->msys                             <-dossiers d'installation de MSYS
|    |    |->1.0                         <- dossier connu comme étant "/" sous MSYS
|    |    |    |->bin
|    |    |    |->etc
|    |    |    |->lib
|    |    |    ...
|    |    |    |->home
|    |    |    |    |->votre nom         <- votre dossier personnel "~" sous MSYS
|    |    |    |    |    |->build
|    |    |    |    |        |->binutils <- dossier de compilation pour binutils
|    |    |    |    |        |->gcc      <- dossier de compilation pour Gcc
|    |    |    |    |        |->mingwrt  <- dossier de compilation pour MinGW Runtime
|    |    |    |    |        |->w32api   <- dossier de compilation pour W32-api
|    |->temp                             <- dossier de l'installation temporaire
|    |    |->bin
|    |    |->etc
|    |    |->lib
|    |    |->...
```

III-B-5 - Convention pour les codes

Une série d'instructions seront présentées dans les lignes suivantes.

Afin d'éviter toute confusion, il me semble intéressant de présenter la convention que j'ai utilisée :

Une commande commence par $. Ce $ représente l'invite de commande et ne doit pas être écrit.

```
$ une ligne de commande
```

Si une commande est écrite sur plusieurs lignes, chaque ligne termine par un \ qui permet à la console de savoir que la ligne continue plus loin. La ligne suivante commence alors par un >.

```
$ une commande \
> sur plusieurs lignes
```


Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Vous pouvez décider de ne pas tenir compte de ce signe, et continuer sur la même ligne, l'interpréteur de commande passera automatiquement à la ligne en cas de besoin.

Cependant, si vous décidez de passer à la ligne (par appuis sur la touche <Enter>) vous **devez** mettre ce symbole avant le passage à la ligne.

En cas de commande multiligne, vous ne pourrez vous déplacer que sur la dernière ligne de celle-ci.

III-B-6 - Configurer les chemins d'accès

Afin de pouvoir utiliser notre installation temporaire de GCC, il y a lieu de configurer c:\temp\bin comme étant un chemin dans lequel aller chercher les exécutables.

Sous linux, la variable qui permet à la console de retrouver les dossiers dans lesquels se trouvent les exécutable est la variable PATH.

Nous pouvons donc rajouter n'importe quel dossier à cette variable avec la commande

```
$ PATH=/temp/bin:$PATH
```

Ceux qui ont l'habitude de l'invite de commande sous windows auront remarqué que le nom de la variable représentant les dossiers dans lesquels le système va chercher les exécutables est identique. Les différences essentielles sont que, sous windows, les noms de variables sont entourés de symbole '%' alors qu'ils sont, simplement, précédés du symbole '$' sous linux et dans le fait que le séparateur pour cette variable est le point virgule ';' sous windows et les deux points ':' sous linux

Vous pourrez ensuite vérifier si vous ne vous êtes pas trompé en invoquant la commande

```
$ gcc -v
```

qui devrait vous sortir quelque chose ressemblant à

```
Using build-in specs.
COLLECT_GCC=c:\temp\bin\gcc.exe
COLLECT_LTO_WRAPPER=/temp/bin/../libexec/gcc/mingw32/3.4.5/lto-wrapper.exe
Target: mingw32
Configured with: ../gcc-4.5.0 --enable-languages:c,c++,ada,fortran,objc,obj-c++
--disable-sjlj-exceptions --with-dwarf2 --enable-shared --enable-libgomp
--disable-win32-regstry --enable-libstdcxx-debug
--enable-version-specific-runtime-libs --disable-werror --build=mingw32
--prefix=/mingw
Thread model: win32
gcc version 4.5.0 (GCC)
```

IV - Obtenir les sources

Nous devrons compiler un certain nombre de paquets afin d'obtenir une version fonctionnelle de Gcc, dans l'ordre:

- binutils : une collections d'outils qui fournit tout ce qui est nécessaire à la gestion et à l'utilisation correctes de fichiers binaires (exécutables, bibliothèques dynamiques et statiques...).
- w32-api une adaptation des headers de Microsoft pour être utilisables avec GCC et fournit également certaines bibliothèques fréquemment utilisées telles que OpenGl ou DirectX.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- mingw-runtime : l'implémentation, basée sur les appels windows au lieu des appels posix/linux, de la bibliothèque standard du C.
- GMP (requis pour Gcc) : une bibliothèque de calcul multi-précision.
- MPFR (requis pour Gcc, dépend de GMP) : une bibliothèque de calcul multi-précision en virgule flottante avec arrondi correct.
- MPC (requis pour Gcc, dépend de MPFR) : une bibliothèque dédiée à l'arithmétique des nombres complexes avec un arrondi correct du résultat.
- Gcc : le compilateur en lui-même.
- pthreads : le support des threads "posix" sous windows (uniquement si vous souhaitez activer la compilation de libgomp, ou si vous souhaitez pouvoir manipuler les threads posix dans vos développements)

Pour cela, nous devons donc veiller à obtenir les sources relatives à ces différents paquetages.

Les sources des paquets propres à MinGW (mingw-runtime, w32-api et le cas échéant pthreads) seront téléchargées directement sur le site de MinGW.

Pthread n'est pas à proprement parler indispensable pour la compilation de Gcc. Cependant, elle sera requise si vous souhaitez activer le support de libgomp, qui est une bibliothèque de base supportant OpenMP permettant la programmation parallèle.

Pour les autres, paquets, vous pourrez décider de récupérer les sources directement sur le site de MinGW, mais, si votre idée est de profiter des dernière versions, l'idéal est d'aller les chercher directement sur le site dédié :

- binutils : vous pouvez obtenir la liste des mirroirs à l'adresse **http://www.gnu.org/software/binutils/** .
- GMP est disponible à l'adresse **http://gmplib.org/**
- MPFR est disponible à l'adresse **http://www.mpfr.org/**
- MPC est disponible à l'adresse **http://www.multiprecision.org/**
- GCC est disponible à l'adresse **http://gcc.gnu.org/**

IV-A - Les versions testées

Ainsi que je l'ai signalé, les numéros de version sont susceptibles d'évoluer très rapidement, ne serait-ce que du côté des "snapshot" étant donné qu'il en sort un par semaine.

S'il n'est pas impossible que la version 4.7 de GCC ne sorte qu'après la finalisation de la norme C++0x, bien que je ne puisse le jurer, il est donc aussi tout à fait possible que la version stable numérotée 4.5 évolue encore d'ici la stabilisation de la version 4.6.

Les numéros de version et les instructions de compilation que je fournis dans ce tutorial sont donc à prendre dans le sens où c'est ce qui fonctionnait au moment de la rédaction.

IV-A-1 - GMP-5.0.1

Il s'agit de la dernière version à l'heure d'écrire ces lignes

Cette version apporte, outre les améliorations propres, d'énormes facilités en terme de compilation :

Il est, avec cette version, possible de compiler GMP directement lors de la compilation de Gcc.

Nous pourrons donc nous contenter de copier l'intégralité des sources de GMP dans un sous dossier des sources de Gcc nommé gmp

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

IV-A-2 - MPFR-3.0.0

Il s'agit de la dernière version à l'heure d'écrire ces lignes

La possibilité de compiler GMP directement en compilant Gcc permet d'assurer les dépendances des différentes bibliothèques qui en dépendent, dont MPFR lorsque l'on tente de les compiler en même temps que Gcc.

Nous pourrons donc nous contenter de copier le dossier source de MPFR dans un sous dossier des sources de Gcc nommé mpfr

IV-A-3 - MPC-0.8. "Dianthus deltoides"

Il s'agit de la dernière version à l'heure d'écrire ces lignes

Cette bibliothèque pourra également être compilée directement lors de la compilation de Gcc, grâce au fait que ses dépendances sont assurées.

Nous pourrons donc nous contenter de copier le dossier sources de MPC dans un sous dossier des sources de Gcc nommé mpc

IV-A-4 - Binutils-2.20.51

La version reconnue comme stable de binutils par l'équipe de MinGW au moment d'écrire ces lignes est la version 2.20.51. J'ai compilé le "snapshot du 5 septembre 2010 de cette version.

IV-A-5 - GCC-4.6.0

La dernière version stable de GCC au moment d'écrire ces lignes est la version 4.5.1, cependant, j'ai réussi à compiler la version en cours de développement: la version 4.6.0

IV-B - Décompresser les sources

Commençons par décompresser l'ensemble des sources en lançant les commandes

```
$ tar -vxjf /c/download/binutils-2.20.51.tar.bz2
$ tar -vxjf /c/downloead/gcc-4.6.20100904.tar.bz2
$ tar -vxjf /c/download/gmp-5.0.1.tar.bz2
$ tar -vxjf /c/download/mpfr-3.0.0.tar.bz2
$ tar -vxzf /c/download/mpc-0.8.2.tar.gz
$ lzma -d /c/download/w32api-3.15-1-mingw32-src.tar.lzma
$ tar -vxf /c/download/w32api-3.15-1-mingw32-src.tar
$tar -vxzf /c/download/mingwrt-3.18-mingw32.tar.gz
```

IV-B-1 - Comprendre les commandes

L'utilitaire tar est un utilitaire capable de gérer plusieurs formats d'archive, parmi lesquels les archives compressées au format bz2 ou au format gzip.

Outre le nom du fichier à décompresser, les arguments passés à la ligne de commande sont:

- **v** : **v**erbose : demande d'afficher la liste des fichiers extraits
- **x** : e**x**tract : indique qu'il s'agit d'extraire les fichiers, et non de les placer dans l'archive

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- **z** : g**z**ip : indique que l'archive est compressée au format gzip
- **j** : indique que l'archive est compressée au format bz2 (fourni par bzip)
- **f** : **f**orce oblige tar à remplacer d'éventuels fichiers ou préexistants par ceux contenus dans l'archive

Vous pouvez évidemment placer les arguments dans l'ordre qu'il vous plait, les "coller" les uns aux autres comme je le montre ici, ou les séparer clairement, en n'oubliant pas de rajouter les tirait avant chaque argument, sous une forme proche de tar -v -x -z -f <nom de l'archive>

lzma est un utiltaire permettant de gérer un autre format d'archive, généralement considéré comme plus efficace que bzip ou gzip.

Outre le nom du fichier à décompresser, l'argument passé est **d** : **d**ecompress : indique que nous souhaitons extraire le contenu de l'archive

IV-B-1-a - Renommer le dossier de sources pour w32api

Lors de la compilation de mingw-runtime, le système recherchera un dossier nommé w32api dans le dossier parent à celui dans lequel se trouvent les sources.

Afin qu'il soit en mesure de les trouver, il est préférable de directement renommer le dossier w32api-3.15-1-mingw32 en w32api. Cela se fait à l'aide de la commande

```
$ mv w32api-3.15-1-mingw32 w32api
```

IV-B-1-b - Copier les sources de GMP, MPFR et MPC

Il est possible de faire en sorte que les bibliothèques GMP, MPFR et MPC dont dépend Gcc soient compilées directement pendant le processus de compilation de Gcc.

Il suffit pour ce faire de copier les sources de ces bibliothèques dans des dossiers au nom clairement établi : le nom de la bibliothèque sans précision de la version.

Plutôt que de commencer à compiler ces bibliothèques séparément, copions donc les dossiers qui nous intéressent avec les commandes

```
$ cp -r gmp-5.0.1 gcc-4.6-20100904/gmp
$ cp -r mpfr-3.0.0 gcc-4.6-20100904/mpfr
$ cp -r mpc-0.8.2 gcc-4.6-20100904/mpc
```

Cela permettra entre autres de faire en sorte que ces bibliothèques soient compilées à chaque stage de la compilation de Gcc.

IV-B-1-b-i - Comprendre la commande

La commande permettant de copier un fichier ou un répertoire sous linux est la commande **cp**.

Comme il s'agit de faire copier l'ensemble d'un répertoire, il faut ajouter le paramètre **-r** pour ***recursive***.

Nous indiquons enfin le nom du dossier à copier ainsi que le nom du dossier de destination.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

IV-B-1-b-ii - Corriger quelques fichiers

Avec les sources de Gcc que j'utilise pour cet article, je me suis retrouvé confronté à à l'impossibilié de créer une partie de la documentation, suite à l'absence de TeTex, dont résulte une erreur de compilation.

Pour contourner ce problème, il m'a fallu modifier le fichier ~/gcc-4.6.20100904/gcc/Makefile.in, aux alentours de la ligne 3731 en remplaçant le false; qui se trouve à cette ligne par true;

Le fichier original prend en effet la forme de

```
(entre les lignes 3717 et 3732 environ)
$(RUN_GEN) build/genhooks$(build_exeext) \
  $(srcdir)/doc/tm.texi.in > tmp-tm.texi
$(SHELL) $(srcdir)/../move-if-change tmp-tm.texi tm.texi
@if cmp -s $(srcdir)/doc/tm.texi tm.texi; then \
  $(STAMP) $@; \
elif test $(srcdir)/doc/tm.texi -nt $(srcdir)/doc/tm.texi.in \
  && test $(srcdir)/doc/tm.texi -nt $(srcdir)/doc/target.def; then \
  echo >&2 ; \
  echo You should edit $(srcdir)/doc/tm.texi.in rather than $(srcdir)/doc/tm.texi . >&2 ; \
  false; \
else \
  echo >&2 ; \
  echo Verify that you have permission to grant a GFDL license for all >&2 ; \
  echo new text in tm.texi, then copy it to $(srcdir)/doc/tm.texi. >&2 ; \
  false; \
fi
```

Et le fichier modifié devrait prendre la forme de

```
$(RUN_GEN) build/genhooks$(build_exeext) \
  $(srcdir)/doc/tm.texi.in > tmp-tm.texi
$(SHELL) $(srcdir)/../move-if-change tmp-tm.texi tm.texi
@if cmp -s $(srcdir)/doc/tm.texi tm.texi; then \
  $(STAMP) $@; \
elif test $(srcdir)/doc/tm.texi -nt $(srcdir)/doc/tm.texi.in \
  && test $(srcdir)/doc/tm.texi -nt $(srcdir)/doc/target.def; then \
  echo >&2 ; \
  echo You should edit $(srcdir)/doc/tm.texi.in rather than $(srcdir)/doc/tm.texi . >&2 ; \
  false; \
else \
  echo >&2 ; \
  echo Verify that you have permission to grant a GFDL license for all >&2 ; \
  echo new text in tm.texi, then copy it to $(srcdir)/doc/tm.texi. >&2 ; \
  true; \
fi
```

Les corrections suivantes ne sont nécessaires que si vous souhaitez activer le support de Ada.

IV-B-1-b-ii-- - Si vous voulez activer le support de Ada

L'un des principaux problèmes que l'on rencontrera au cours de la compilation est que la création de liens symboliques échoue sous MSYS, lors de la compilation de libada.

Pour résoudre ce problème, il suffira de modifier une seule variable.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Ainsi, il faudra rechercher la variable **LN_S**, qui est définie avec une valeur de **@LN_S@**, dans le fichier c:\msys\1.0\<votre nom>\src\gcc-4.2.1\gcc\libada\Makefile.in aux alentours de la ligne 43 et la modifier pour lui donner la valeur cp -r de manière à forcer la copie des fichiers plutôt que la création de liens symboliques.

Assez bizarrement les autres bibliothèques ne souffrent pas de ce problème.

De plus, avec les sources que j'ai récupérées, j'ai un problème pour la compilation de Ada dans le sens où un fichier propre à MinGW provoque une erreur:

Il s'agit du fichier g-socthi-mingw.adb se trouvant dans <dossier_source>/gcc/ada.

Entre les lignes 272 et 280, vous trouverez le code:

```
function C_Recvmsg
  (S     : C.int;
   Msg   : System.Address;
   Flags : C.int) return System.CRTL.ssize_t
is
   use type C.size_t;

   Fill  : constant Boolean :=
             (C.unsigned (Flags) and SOSC.MSG_WAITALL) /= 0;
   --  Is the MSG_WAITALL flag set? If so we need to fully fill all vectors
```

Le fait est que MSG_WAITALL est défini à -1 sous MinGW car il s'agit d'une valeur inexistante et que cette valeur ne peut décidément pas être considérée comme une valeur non signée.

La compilation échouera donc sous prétexte que la valeur est hors des limites admises.

Il faut donc modifier le code pour qu'il ressemble à

```
function C_Recvmsg
  (S     : C.int;
   Msg   : System.Address;
   Flags : C.int) return System.CRTL.ssize_t
is
   use type C.size_t;

   Fill  : constant Boolean :=
             SOSC.MSG_WAITALL /= -1
             and then(C.unsigned (Flags) and SOSC.MSG_WAITALL) /= 0;
   --  Is the MSG_WAITALL flag set? If so we need to fully fill all vectors
```

Ada ne supporte pas l'indentation sous forme de tabulation. Vous devrez donc veiller à faire en sorte d'indenter le code à l'aide d'espaces uniquement.

V - Compiler binutils

Il est maintenant temps de passer aux choses sérieuses en commençant par compiler binutils.

V-A - Configurer la compilation

La première étape consiste à créer un répertoire de compilation séparée et à configurer la compilation à l'aide des commandes:

```
$ cd build/binutils
```


Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

```
$ ../../binutils-2.20.51/configure --prefix=/mingw --with-sysroot=/mingw \
--disable-nls --disable-shared --enable-auto-import
```

V-B - Comprendre les commandes

Nous indiquons enfin les répertoires à créer. Comme il est **indispensable** de compiler gcc et binutils dans des dossiers autres que le répertoire dans lequel se trouvent leur sources, et qu'il est généralement conseillé de faire pareil pour les autres projets que l'on compile, nous créons donc directement un répertoire de compilation pour chacun des projets que nous allons copier.

L'option --prefix=<chemin_d'installation> nous permet d'indiquer le répertoire dans lequel sera installé le projet que l'on s'apprête à compiler.

Dans le cas présent, nous demandons à ce qu'il soit installé dans /mingw qui fait un lien direct avec le dossier c:\MinGW.

L'option --disable-nls nous permet d'indiquer que nous ne souhaitons pas profiter de l'internationalisation du projet.

L'option --disable-shared indique que nous ne souhaitons que les exécutables utilisent la version statique des bibliothèques dont elles dépendent au lieu d'une version dynamique (DLL).

Enfin, l'option --enable-auto-import permet à l'éditeur de liens "ld" d'importer automatiquement le symboles depuis les DLL, en nous évitant dés lors un avertissement pour autant que le symbole soit trouvé. Cette option n'empêche nullement une sortie sur erreur si un symbole nécessaire n'est pas trouvé.

V-B-1 - Les problèmes posés par l'internationnalisation

Si vous vous demandez pourquoi je désactive l'internationnalisation, il faut vous rappeler que l'ensemble des outils que nous nous apprêtons à compiler ont à la base été développés pour et sous linux.

Le problème est que l'encodage utilisé par la console linux et celui utilisé par la ligne de commande windows sont différents.

Il s'en suit que certains caractères, essentiellement les caractères accentués, seront interprétés différemment selon l'environnement dans sous lequel ils sont utilisés.

Si nous activons l'internationnalisation, nous ne pourrons pas éviter le recours aux caractères accentués, tant ils sont utilisés intensivement dans la langue française.

Comme ils seront mal interprétés sous windows, ils seront remplacés par des symboles plus ou moins cabalistiques qui ne feront que rendre la lecture des messages envoyés par les outils plus difficile.

L'anglais utilisé étant malgré tout technique et, finalement, relativement simple, il me semble préférable de forcer l'utilisation de cette langue qui a la particularité de n'utiliser aucun caractère accentué.

Vous restez bien entendus libre de supprimer cette option, mais vous serez prévenus...

V-B-2 - Profiter de l'option --with-sysroot

Il est possible, depuis quelques versions, lors de la compilation de binutils et de Gcc, de préciser l'option --with-sysroot=<chemin_vers_dossier> de manière à signaler à tous les outils qu'ils doivent parcourir le chemin indiqué afin de rechercher ce dont ils ont besoin.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Cette option en remplace une série d'autres, Cependant, nous devrons prendre quelques précautions lors de la compilation de Gcc.

Elle aura en outre l'énorme intérêt de configurer les différents outils de sortent qu'ils fonctionnent même si l'on vient à déplacer le dossier d'installation.

V-B-3 - Au sujet des cibles et des hôtes

Il est possible, lors de la compilation de nombreux projets, de déterminer un système cible, un système hôte ou un système "général" pour laquelle on souhaite effectuer la compilation.

La cible permet de déterminer sur quel système les exécutables créés par le compilateur seront exécutés. Cela nous permet, par exemple, d'envisager la création d'un "compilateur croisé" : fonctionnant sous windows mais créant des exécutables prévus pour fonctionner sous linux, ou inversement.

L'hôte quant à lui permet de déterminer sur quel système nous allons compiler nos projet

Le système général enfin défini une cible et un hôte identique.

Les options à passer pour définir la cible, l'hôte et le système général sont respectivement

- --target=<architecture-type_d_orddinateur-système> pour le système cible
- --host=<architecture-type_d_orddinateur-système> pour le système hôte
- --build=<architecture-type_d_orddinateur-système> lorsque la cible et l'hôte sont identique

Comme il ne s'agit ici que de compiler pour des PC, sans autre précision, et pour des systèmes utilisant MinGW, les options prendraient donc la forme de *-pc-mingw32, où l'étoile pourrait être remplacée par

- i386 : pour les processeurs de type 3x86
- i486 : pour les processeurs de type 4x68
- i586 : pour les processeurs de type Pentium I (et équivalents)
- i686 : pour les processeurs de type Pentium II (et équivalents)
- i786 : pour les processeurs de type supérieurs

Je ne présente ici que les types de processeurs dits "32 bits", car il est préférable de partir d'un autre projet (mingw-w64) si vous souhaitez obtenir une version de MinGW compatible avec le 64 bits.

A l'heure actuelle, MSYS considère par défaut que le système est de type "i686-pc-mingw32", aussi bien pour la cible que pour l'hôte. C'est la raison pour laquelle je ne précise ni l'un ni l'autre, car cela permet de générer malgré tout des exécutables pour des systèmes qui, bien qu'ils deviennent relativement vieux, peuvent toujours être utilisés.

Vous trouverez la liste complètes des spécifications d'hôte et de cible ***directement sur le site de Gcc***

V-C - Lancer la compilation

Il ne nous reste plus qu'à lancer la compilation avec la commande

```
make
```

Une fois la compilation terminée avec succès, il ne nous restera plus qu'à lancer l'installation avec la commande

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

```
make install
```

Nous pouvons maintenant retourner dans le dossier racine de compilation (~/build) avec la commande

```
cd ..
```

VI - Compiler w32-api et MinGW runtime

Ces deux bibliothèques sont des dépendances capitales pour le processus de compilation de Gcc. Il faut donc veiller à ce qu'il en dispose dés le départ.

Nous allons donc les compiler directement à l'aide des commandes

```
$ cd w32api
$ ../../w32api/configure --prefix=/mingw/i686-pc-mingw32
$ make && make install
$ cd mingwrt
$ ../../mingwrt-3.18-mingw32/configure --prefix=/mingw/i686-pc-mingw32
$ make && make install
$ cp -r /mingw/i686-pc-mingw32 /mingw/mingw
```

La dernière commande (cp -r /mingw/i686-pc-mingw32 /mingw/mingw) a pour but de faire en sorte de copier l'intégralité du contenu du dossier /mingw/i686-pc-mingw32 dans le dossier /mingw/i686-pc-mingw32

Vous l'aurez constaté, je demande à ce que l'installation se fasse dans le sous dossier nommé i686-pc-mingw32 par rapport au dossier racine dans lequel je prévois de faire mon installation.

La raison de cette manière de faire est que l'option --with-sysroot présente cette particularité, sous windows, d'impliquer que le processus de compilation cherchera ce dont il a besoin dans un dossier nommé mingw se trouvant dans le dossier indiqué par cette option

Mais il n'y a pas lieu de préciser l'hôte pour la compilation, le processus d'installation copiera les fichier à installer exactement dans le dossier indiqué par l'option --prefix.

Le problème sera alors double: D'une part, le processus de compilation de Gcc ne trouvera pas les fichiers fournis par w32-api et par mingw-runtime lorsqu'il en aura besoin, et, d'autre part, il sera impossible de faire cohabiter plusieurs version ces deux paquets, afin de permettre la compilation vers des cibles multiples.

Agir comme je le fait permet de résoudre ces deux problèmes en même temps.

VII - Compiler Gcc

VII-A - Si vous voulez activer le support de Ada

VII-B - Configurer Gcc

Il est temps de configurer la compilation de Gcc. Cela se fait à l'aide des commandes

```
$ cd gcc
$ ../../gcc-4.6.20100904/configure --prefix=/mingw --with-sysroot=/mingw \
> --enable-libssp --enable-libgomp --enable-libstdcxx-debug \
> --enable-version-specific-runtime-libs --enable-fully-dynamic-string \
```


Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

```
> --enable-shared --enable-languages=c,c++,ada,fortran,objc,obj-c++,java \
> --with-dwarf2 --disable-nls --disable-win32-registry \
> --disable-werror
```

VII-B-1 - Comprendre la commande

Après être entré dans le dossier dans lequel nous souhaitons que la compilation s'effectue, nous configurons Gcc en lui passant (seuls les options n'ayant pas encore été vues sont reprises)

- --enable-libssp : provoque la compilation de la bibliothèque SSP, (Stack Smashing Protection) une bibliothèque permettant de se protéger un peu des problèmes liés à la corruption de la pile
- --enable-libgomp : provoque la compilation de la bibliothèque GOMP, une bibliothèque implémentant la norme OpenMP pour le travail parallèle
- --enable-libstdcxx-debug : provoque la compilation d'une version de la biblitothèque standard de C++ avec le support des informations de débogage
- --enable-specific-runtime-libs : fait en sorte que les différentes bibliothèques soient placées dans des dossiers particuliers en fonctions de la cible et de la version du compilateur, de manière à permettre à plusieurs versions de cohabiter
- --enable-fully-dynamic-string : fait en sorte que les chaines de caractères soient d'office représentées par des chaines de caractères de taille dynamique, pour éviter les éventuels problèmes avec certaines bibliothèques
- --with-dwarf2 : indique que les informations de débogage seront par défaut au format Dwarf2
- --disable-sjlj-exceptions : désactive explicitement le support de SJLJ comme gestionnaire d'exception.
- --disable-win32-registry : fait en sorte que Gcc n'aille pas écrire d'informations particulières dans le registre de windows
- --disable-werror : désactive le flag "-Werror" lors des deuxièmes et troisièmes stages de compilation
- --enable-languages:c,c++,ada,fortran,objc,obj-c++,java : indique les langages dont nous souhaitons activer le support

J'ai demandé à ce que les différentes bibliothèques soient fournies sous leur forme dynamique, quand elle est disponible
Vous pouvez parfaitement demander à ce que la version dynamique ne soit pas créée en remplaçant l'option --enable-shared par l'option inverse: --disable-shared

J'ai demandé à ce que la compilation fournisse le support pour l'ensemble des langages
Vous pouvez bien entendu sélectionner les langages qui vous intéressent.
Pensez cependant au fait que C est absolument obligatoire.

Vous pourrez en apprendre d'avantage sur les options de configurations de Gcc directement ***sur le site de Gcc***
Vous pouvez également utiliser l'option --help ou l'option --help=recursive du script de configuration pour obtenir une aide en ligne.

N'oubliez pas que vous ne pourrez demander à ce que le compilateur Ada soit compilé que ***si vous disposez déjà*** *d'un compilateur pour ce langage.*

L'ordre dans lequel vous transmettrez les options ou demanderez le support des langages n'a pas d'importance. Il faut cependant veiller à ne pas oublier une option qui pourrait vous intéresser.

VII-B-2 - Comprendre le processus de compilation

Lorsque la cible (l'option --target et l'hôte (l'option --host) pour lequel vous demandez la compilation de Gcc sont identiques, le processus de compilation s'effectuera trois fois:

- La première fois pour créer un compilateur susceptible de fournir des exécutables utilisable par la cible

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- La deuxième fois pour créer un compilateur utilisable sur la cible (qui est également l'hôte)
- La troisième fois pour s'assurer que le compilateur obtenu soit capable de se compiler lui-même. On se base en effet sur le principe que, si un compilateur est capable de se compiler lui-même, il sera capable de compiler n'importe quoi.

Ce processus s'appelle le ***bootstrap***

Il *paraitrait* que le compilateur sera d'autant mieux optimisé qu'il aura été compilé de nombreuses fois, mais cela n'est peut-être que légende urbaine.

La première passe se contentera généralement de compiler un stricte minimum, alors que les étapes suivantes compileront Gcc dans son ensemble (en fonction de la configuration que vous aurez choisie).

Ces deux dernières étapes utiliseront, par défaut, le drapeau "-Werror" lors de la compilation, qui a pour objectif de faire en sorte que le compilateur traite les avertissements exactement comme s'il s'agissait d'erreur, en quittant le processus de compilation.

Comme il reste malgré tout de nombreux points dans le code source où des avertissements sont lancés, alors que l'auteur du code savait parfaitement ce qu'il faisait, il est important de passer l'option --disable-werror lors de la configuration, afin d'éviter que la compilation n'échoue "au beau milieu" du processus suite à un "simple" avertissement.

VII-C - Compiler Gcc première (selon conditions)

Vous pouvez passer ce chapitre si vous n'avez pas demandé la compilation de libgomp au lors de la configuration

Afin de pouvoir compiler la bibliothèque libgomp, nous aurons besoin d'une version utilisable de la bibliothèque pthreads qui fournit un support partiel des threads de type "posix".

Libgomp et pthreads ne sont pas **indispensables** pour la compilation de Gcc, mais libgomp ne peut absoluement pas être compilé si l'on ne dispose pas du support des threads posix.

Si vous ne voulez pas installer pthreads, vous ne pourrez pas utiliser l'option --enable-libgomp.

Si vous avez passé l'option --enable-libgomp au script de configuration, il faudra veiller à assurer la présence de la bibliothèque pthead, dont libgomp dépend.

Tant qu'à faire, il est intéressant de faire en sorte que pthread sera compilée en utilisant la version du compilateur que nous sommes sur le point de compiler.

Il s'agira alors de compiler et d'installer un stricte minimum pour être en mesure de créer les fichiers objets, à savoir le compilateur C et la bibliothèque qui va avec avec les commande

```
$ make all-gcc && make install-gcc && \
> make target-libgcc && make install-target-libgcc
```

Vous l'aurez constaté, il faut en réalité enchainer quatre commandes, car il faut au minimum que le compilaeur C ait été compilé avant de pouvoir compiler libgcc.

Comme les deux devront de toutes manières être installés, autant les installer directement, même si l'installation de libgcc semble également installer le compilateur C.

Vous pouvez obtenir les sources de pthreads directement sur le site de MinGW. Ma version actuelle est la 2.8.0-3

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Après avoir téléchargé les sources, décompressez les avec les commande

```
$ cd /~
lzma -d /c/download/pthreads-w32-2.0.8-3-mingw32-src.tar.lzma
tar -vxf /c/download/pthreads-w32-2.0.8-3-mingw32-src.tar
tar -vxzf pthreads-w32-2.0.8.-3-release.tar.gz
```

La commande cd ~ permet de retourner dans son dossier personnel, qui est le dossier dans lequel nous avons à la base décompressé les sources.

Il semblerait que l'équipe de MinGW aie décidé de fournir une archive des sources de pthread à l'intérieur d'une autre, au format tar.lzma. C'est la raison pour laquelle vous constatez les trois commandes d'extraction.

Les sources de pthreads ne sont pas fournies avec le script de configuration habituel, mais directement avec un Makefile qui s'occupe de gérer la compilation.

Afin de nous assurer que ce soit effectivement le compilateur que nous sommes occupés à compiler qui sera utilisé, nous pouvons demander l'affichage de la versin de gcc avec la commande

```
gcc -v
```

Si vous obtenez, entre autre, l'affichage

```
gcc versin 4.6.0 20100904 (experimental)
```

vous aurez votre certitude.

Autrement, vous devrez modifier la ligne 56 du fichier GNUmakefile qui ressemble à

```
CC    = $(CROSS)gcc
```

pour qu'elle corresponde à

```
CC    = /mingw/bin/gcc
```

Une fois que nous serons sur du compilateur utilisé, nous pourons compiler pthreads en allant directement dans le répertoire des sources avec les commandes

```
cd pthreads-w32-2-8-0-release
make clean GC
```

Le makefile fourni ne permet pas l'installation automatique, il faut donc copier un certain nombre de fichiers manuellement avec les commandes

```
cp pthreadGC2.dll /mingw/bin
cp pthreadGC2.dll /mingw/i686-pc-mingw32/lib/libpthread.a
cp pthreadGC2.dll /mingw/i686-pc-mingw32/lib/libpthreadGC2.a
cp pthreads.h semaphore.h sched.h /mingw/i686-pc-mingw32/include
cp config.h /mingw/i686-pc-mingw32/include/pthreads_config.h
```


Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

VII-C-1 - Comprendre les commandes

La première instruction copie, tout simplement, la bibliothèque dans le dossier dans lequel se trouvent tous les exécutables.

La bibliothèque d'importation (libpthreadGC.a) ne semble pas convenir, ce qui justifie que nous copions la DLL elle-même en la renommant au nom de la biliothèque d'importation.

Comme de nombreux système ne font pas la différence entre les différentes versions envisageables de pthreads, faisons de même en donnant un nom sans information de version : libpthread.a

Nous copions ensuite les fichiers d'en-tête indispensables dans le dossier include: pthreads.h, sched.h et semaphore.h

Enfin, nous copions le fichier config.h dans le dossier include en le renommant au passage en config_pthreads.h afin d'éviter qu'il n'interfère avec d'autres fichiers config.h lors de processus de compilation d'autres projets.

VII-C-2 - Terminer l'installation de pthreads

Il ne nous reste plus que deux choses à faire au sujet de pthreads: Nous assurer que pthreads_config.h sera bel et bien inclus dans pthreads.h

```
modifier

#if HAVE_CONFIG_H
#include "config.h"
#endif /* HAVE_CONFIG_H */

en


//#if HAVE_CONFIG_H
#include "pthreads_config.h"
//#endif
```

Puis veiller à ce que le dossier /mingw/mingw contiennent exactement la même chose que le dossier /mingw/i686-pc-mingw32, avec les commandes

```
$ rm -rf /mingw/mingw
$ cp /mingw/i686-pc-mingw /mingw/mingw
```

Vous pouvez bien évidemment utiliser l'explortateur windows pour ce faire, mais, tant qu'à avoir la ligne de commande MSYS sous la main...

Vous pouvez attendre d'avoir installé unzip si vous souhaitez également activer le support pour Java. Il ne sert pas à grand chose de le faire deux fois alors qu'une seule serait suffisante.

VII-C-3 - Si vous souhaitez activer le support de Java

Vous le savez certainement, java utilise un format particulier d'archives: le JAR.

Tels qu'elles sont présentées classiquement, les sources de Gcc ne permettent pas de gérer de telles archives.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Vous devrez donc veiller à installer une bibliothèque supplémentaire: unzip, dont vous trouverez les sources sur **le site de sourceforge qui leur est dédié**

Cette bibliothèque ne fournit pas de script de configuration, mais un certain nombre de Makefile "prêts à l'emploi".

Après avoir téléchargé les sources, décompressez les archives dans votre dossier personnel (c:\msys\1.0\<votre nom>)

VII-C-3-a - Installer unzip

Entrez dans le dossier contenant les sources de zip et demandez la compilation de l'ensemble avec la commande

```
$ make -fwin32/Makefile.gcc NOASM=1 all
```

Copiez les fichiers qui nous intéressent avec les commandes

```
cp libunzip32.a /mingw/i686-pc-mingw32/lib
cp libunzip32.a /mingw/i686-pc-mingw32/lib/libunzip.a
cp zip.h unzip.h unzpriv.h /mingw/i686-pc-mingw32/include
cp globals.h /ming/i686-pc-mingw32/include/globals_zip.h
```

Modifiez le fichier unzpriv.h à auteur de la ligne 2320 pour qu'il inclue globals_zip.h au lieu de globals.h (vu que c'est le nouveau nom donné au fichier).

A l'origine, ce fichier ressemble à

```
/*************/
/*  Globals  */
/*************/

#if (defined(OS2) && !defined(FUNZIP))
#  include "os2/os2data.h"
#endif

#include "globals.h"
```

La ligne en question étant la dernière. Modifiez cette dernière ligne en

```
/*************/
/*  Globals  */
/*************/

#if (defined(OS2) && !defined(FUNZIP))
#  include "os2/os2data.h"
#endif

#include "globals_zip.h"
```

La raison d'être de ce changement de noms est que nous courrons le risque que d'autres projets essayent d'ajouter leur propre fichier "globals.h" au fichier d'en-tête, ce qui aurait pour résultat de remplacer celui de unzip.

Il ne reste plus qu'à veiller à ce que le dossier /mingw/mingw contiennent exactement la même chose que le dossier /mingw/i686-pc-mingw32, avec les commandes

```
$ rm -rf /mingw/mingw
```


Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

```
$ cp /mingw/i686-pc-mingw /mingw/mingw
```

Vous pouvez bien évidemment utiliser l'explortateur windows pour ce faire, mais, tant qu'à avoir la ligne de commande MSYS sous la main...

VII-C-4 - Retourner dans le dossier de compilation de Gcc

Une fois que

```
$ cd /home/<votre nom>/build/gcc
```

VII-D - Compiler Gcc deuxième

Nous pouvons maintenant lancer le processus de compilation dans son intégralité, puis installer Gcc à l'aide des commandes

```
$ make && make install
```

Après avoir lancé cette commande, vous pourrez passer à autre chose le temps que tout se fasse, car le processus de bootstrap prendra un temps certain (trois à quatre heures sur un Athlon 1800 XP, un peu plus de deux heures sur un Intel quad core I7 sans le recours à la compilation parallèle).

VIII - Recompiler Binutils

De par le processus de bootstrap de Gcc, nous sommes surs que la version de ce dernier que nous avons a été compilée avec un compilateur de version strictement identique.

Par contre, il est temps de nous rappeler que Binutils a été compilé bien avant que l'on ne compile Gcc.

Tant que nous resterons dans le domaine des compilateurs 32 bits, il n'y aura pas forcément de problèmes, mais il serait malgré tout intéressant de veiller à ce que Binutils soit, lui aussi, compilé avec la version de Gcc que nous venons de compiler.

La solution la plus simple consiste à quitter MSYS et à le relancer, de manière à être sur de recommencer avec une variable PATH dans laquelle le dossier d'installation temporaire n'apparait pas, de supprimer l'intégralité du contenu du dossier de compilation de binutils avec la commande rm -rf build/binutils/* puis de recommencer l'ensemble du processus de compilation de binutils tel qu'il est décrit plus haut.

IX - Finaliser le tout

IX-A - Quelques outils supplémentares

Vous souhaiterez peut être disposer de certains outils supplémentaires utilisables avec votre "tout nouveau" compilateur, comme un système de compilation automatique basé sur les Makefile (make) ou un débuggeur (GDB).

Ces deux outils (et d'autres) sont disponibles, sous forme de sources et compilées) sur le site de MinGW.

Si vous souhaitez les compiler personnellement avec votre nouveau compilateur, vous devriez pouvoir vous en sortir avec les commandes

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

```
$ &lt;chemin-vers-les-sources&gt;/configure --prefix=/mingw
$ make
$ make install
```

Si vous décidez d'installer make et de garder MSYS, il est préférable de renommer le fichier make.exe se trouvant dans le dossier c:\MinGW\bin en make-mingw32.exe, de manière à évter tout conflit avec la version propre à msys de cet outil. Certains projets (Qt, par exemple) recherchent d'ailleurs ce nom lorsqu'on décide de les compiler sous MinGW.

IX-B - Un peu de ménage

Entre le compilateur "temporaire" que l'on a utilisé, les fichiers générés lors de la compilation des différentes parties, le compilateur "final" obtenu,les archives téléchargées contenant les sources et leur extraction, l'ensemble du processus prendra environ entre 4 et 5 Gb d'espace disque, alors que le seul compilateur ne nécessite "que" +/- 600 Mb si on active l'ensemble de ses possibilités.

A moins que cela ne vouspose pas de problème ou que vous n'ayiez d'autres projets pour tous les fichiers que cela représente, il est intéressant de faire de l'espace en supprimant au minimum tout ce qui ne servira plus, à savoir:

- le dossier c:\MinGW\mingw qui n'avait d'utilité que pour le processus de compilation de Gcc
- L'ensemble du contenu du dossier de compilation (build) (c:\msys\1.0\<votre nom>\build
- L'ensemble des sources: binutils, Gcc, GMP, MinGW runtime, MPC, MPFR, W32-api libunzip, pthreads et éventuellement de make et de GDB ainsi que des archives associées
- l'ensemble du compilateur "temporaire" (c:\temp\bin, à l'exception éventuelle de mingw-get, c:\temp\doc, c:\temp\include, c:\temp\lib, c:\temp\libexec\gcc c:\temp\mingw32, c:\temp\share)
- Eventuellement l'ensemble des fichiers relatifs à mingw-get (c:\temp\var, c:\temp\libexec\mingw-get)
- Si vous ne souhaitez pas garder de système linux "minimaliste", l'ensemble du contenu de c:\msys

X - Liens et remerciements

X-A - Trouver les archives

- **Site qui s'occupe de la bibliothèque GMP**
- **site qui s'occupe de la bibliothèque MPFR**
- **Section téléchargements du site MinGW**
- **liste des miroirs Gcc**
- **le projet info-zip**

Vous pourrez en outre, une fois que vous avez choisi votre miroir pour Gcc, accéder à l'ensemble des sources du projet GNU en cliquant sur le lien "vers un rép. de plus haut niveau".

X-B - Sources d'inspiration

- **How to compile GCC 4.1 sur le wiki de MinGW**
- **delorie's web site: GNU Compiler Collection (GCC) Internals**
- **Installing GCC sur gcc.gnu.org**

X-C - Liens connexes

- **Introduction aux makefile**

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- **Mode d'emploi de GCC**
- **Test de l'EDI code::blocks**
- **Entrée de la FAQ concernant la prochaine norme**

X-D - Remerciements

Je tiens à remercier **Alp**, **Frank.H**, **Millie** et **Aurelien.Regat-Barrel** pour leur relecture attentive et leurs commentaires éclairés lors de la rédaction de cet article.

Je veux aussi faire une dédicace spéciale à **Nono40** pour avoir fourni l'outil sans lequel cet article n'aurait pas eu une telle qualité de présentation.

Copyright © - Philippe Dunski. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.