# Initiation aux outils Autoconf et Automake


Labo-Unix - http://www.labo-unix.net

2002-2003

# Table des matières

# 1 Introduction

Nous avons déjà vu ensemble le système de compilation Unix nommé ”make” et dont le rôle est de déterminer quelle partie des sources d'un projet nécessite d'être recompilé en se basant sur la date de modification des fichiers. Cet outil est très pratique à été développé avec des objectifs précis :

- l'uniformisation de la compilation : il suffit de taper ”make *cible*” pour que cible soit compilé.
- l'économie de temps cpu : make détermine quels objets il faut recompiler en fonction des dates de modification des fichiers source servant à le construire.

Make est un système très éfficace et ne se limite pas aux systèmes Unix, mais il ne prend pas en compte un certain nombre de difficultés rencontrées par les programmeurs unix. Parmis celles ci on notera :

- Hétérogénéité des systèmes Unix :
  * Architectures matérielles différentes : Taille des variables de certains types, big et little endian, particularitées des compilateurs.
  * Abscence de certaines bibliothèques, fonction de bibliothèques et prototypes différants selon les implémentations.
  * Dispositions des fichiers includes differentes dans l'arborescence : Les fichiers includes ne sont pas toujours dans /usr/include/.
  * Abscence de certaines applications : Applications utilisées lors de la compilation ou de l'exécution du programme.
- Modularité des applications : Certaines applications on une conception modulaire, chaque module apportant une fonctionnalité mais aussi des dépendances vis à vis de bibliothèques (on rejoint le problème de l'hétérogénéité), il est alors indispensable de permettre à celui qui compile l'application de choisir les fonctions qu'il souhaite. On comprend mieux ce problème en prenant l'exemple d'apache : on peut vouloir ”apache” sans pour autant vouloir ”SSL” ou ”WebDav”.
- Complexité et standardisation : Pour certain projets le fichier Makefile est devenu un casse tête pour leur mainteneur et, la synstaxe n'aidant pas, illisible pour les utilisateurs d'autant plus qu'il est devenu nécessaire que des cibles ”standard” existent telles que ”clean” ou ”install”.

Pour palier à ces problèmes deux projets on vu le jour : automake et autoconf dont nous allons voir ici le fonctionnement et l'utilisation du point de vu du programmeur. Gâce à ces outils le Makefile est :

- Construit automatiquement.
- Adapté aux ressources logicel et matérielles.
- Standardisé puisqu'il propose toutes les cibles usuelles.
- Illisible mais efficace

Ces outils sont des projets GNU sous licence GPL depuis 1995 et ont été adoptés par la très grande majorité des grands projets sous Unix libres.

# 2 Présentation du système

## 2.1 Le script ”configure”

### 2.1.1 Qu’est ce que ”configure” ?

Configure est un script shell généré par l’outil autoconf et qui, lorsqu’on l’exécute, génère lui-même les différents Makefiles d’un projet.
Nottez bien que l’on distingue deux type de personnes compilant un logiciel, les développeurs ou mainteneurs et les ”utilisateurs”. Pour chacun de ces types de personnes l’utilisation des outils automake et autoconf, et en particulier configure, sera fondamentalement différente.

Configure a pour bût premier de simplifier la vie de l’utilisateur qui n’aura plus qu’a effectuer quelques commandes pour compiler le projet et ce quel que soit l’environement materiel et logiciel. Le script configure permettra aussi d’indiquer à l’utilisateur si la compilation n’est pas possible sur son système.
Les commandes à exécuter pour l’utilisateur sont simplement :

```
[user@host apache-2.0.43]$ ./configure
[user@host apache-2.0.43]$ make
[root@host apache-2.0.43]$ make install
```

Par contre du point de vue du mainteneur, le script configure est difficile à mêtre en oeuvre en théorie puisque c’est à lui qu’incombe la génération des fichiers Makefile. Mais nous allons voir que le rôle d’autoconf est justement la génération du script ”configure”.

### 2.1.2 Que fait ”configure” ?

Le script configure effectue les opérations suivantes :

- Test des carractéristiques du système : headers, librairies, fonctions, types, exécutables etc.
- Génère un header contenant les résultats des ces tests lorsqu’ils ne rendent pas la compilation impossible : config.h (nom par défaut, modifiable par le mainteneur).
- Génère le ou les fichier(s) ”Makefile”.
- Génère un certain nombre de fichier ”internes” : config.status, config.cache et config.log.

Pour cela il lui faut un certain nombre d’informations dont certaines peuvent lui être données en argument :

- Le fichier config.h.in : décrit le squelette du config.h.
- Un ou plusieurs fichiers Makefile.in : décrit le squelette du/des Makefiles
- Des paramètres au format standardisé tels que : –with-xxx=Y –enable-yyy etc.

### 2.1.3 Ressources nécessaires

Nous l’avons déja vu, le bût de configure est de déterminer les propriétés d’un système de façon à générer les fichiers Makefile correspondants, par conséquent il est indispensable que celui-ci n’ait pas ou peu de dépendances vis à vis du système sur lequel il s’exécute. En réalité configure n’est qu’un ”simple” script shell et ne dépend donc que de la présence de ”/bin/sh”.

Par contre la création de ce script, quasiment illisible et sans doute irréalisable ”à la main”, nécessite différents outils que nous allons détailler et utiliser.

- **autoconf** : génère le script configure.
- **automake** : génère le ou les fichiers Makefile.in.
- **autoheader** : génère le fichier config.h.in.
- **aclocal** : génère le fichier aclocal.m4.
- **libtool** : pour la gestion des librairies.

Ces outils nécessites eux même d'autres logiciels en l'occurence m4 qui est un macro-générateur bien connu des administrateurs de sendmail et perl qui est un langague interprété très répendu.
Nous allons voir successivement chacun de ces outils qui ne sont vraiment utiles que lorsqu'ils sont utilisés ensembles.

### 2.2 Autoconf

Autoconf est l'outil qui sert à générer le script ”configure”. Pour cela il utilise les définitions du fichier **configure.in** dont l'écriture est à la charge du mainteneur. Nous verrons donc en détail son contennu dans les chapitres suivants. Il utilise aussi un fichier nommé **aclocal.m4** généré par un autre outil nommé **aclocal**.
Le fichier aclocal.m4 contient un certain nombre de définitions de macros utiles à autoconf et automake relatives au système générant le configure (le système du mainteneur et donc pas des utilisateurs). Par exemple on aura au début d'un aclocal.m4 généré par aclocal :

```
# We require 2.13 because we rely on SHELL being computed by configure.
AC_PREREQ([2.13])
```

Ainsi losque le mainteneur exécute autoconf, celui-ci va lire aclocal.m4 et interpréter cette directive ainsi : ”si ta version est inférieure à 2.13 -¿ stop”. Dans ce cas là le mainteneur (développeur) sera informé, via un message d'erreur, qu'il doit mettre à jour autoconf s'il veux générer le script configure.

Ce fichier aclocal.m4 n'est en aucun cas à la charge du mainteneur. Il est généré par aclocal sans aucun paramètre et le mainteneur ne doit pas s'en soucier.

### 2.3 Automake

Automake est l'outil qui va générer les fichiers **Makefile.in** lesquels seront utilisés par configure pour générer les **Makfile**.
Pour cela automake utilise un ou plusieurs fichiers **Makefile.am** dont l'écriture est à la charge du mainteneur et qui défini l'arborescence des sources ainsi que les fichiers qui les composent et les cibles qui en résulte.
De plus automake consulte **aclocal.m4** de la même manière qu'autoconf et le **configure.in**, lequel sert à automake pour certaines informations ne se rapportant pas directement aux fichiers source. Mais nous y reviendrons dans les chapitres traitant du configure.in.
Un exemple tout de même :
Nous verons que tout fichier configure.in doit contenir la déclaration suivante :

```
AM_INIT_AUTOMAKE(sntd, 1.2.3)
```

AM_ signifie que cette macro s'adresses à AutoMake. Le premier argument est le nom du logiciel et le second le numéro de version. Les variables shell PACKAGE et VERSION sont définies en conséquence par automake dans les fichiers qu'il génèrera.

## 2.4 Autoheader, aclocal et autoscan

Autoheader, aclocal et autoscan sont des outils pour le mainteneur integrés au système autoconf et automake mais à la différence de ces deux-ci, ils ne sont réèlement utilisés directement par celui-ci que rarement.
Une des rares exeptions est, par exemple, la première utilisation du système autoconf et automake sur le système d'un développeur.

### 2.4.1 Autoheader

Autoheader sert à générer le fichier **config.h.in** lequel servira au script configure pour générer le fichier **config.h** lequel contiendra les résultats des tests effectués par le script configure. Ces résutat étant sous la forme de "define" du type :

```
/* Define if you have the ANSI C header files.  */
#define STDC_HEADERS 1
```

Ces defines servirons dans le code source du projet pour prendre en charge les résutats du script configure, nous verons cela plus en détail dans le chapitre consacré à l'utilisation de config.h.

Pour générer le config.h.in autoheader consulte le fichier **configure.in** pour déterminer quels tests vont être effectués et donc quels "define" devront être définis. Il consulte aussi le fichier **aclocal.m4** de la même manière qu'autoconf.

### 2.4.2 Aclocal

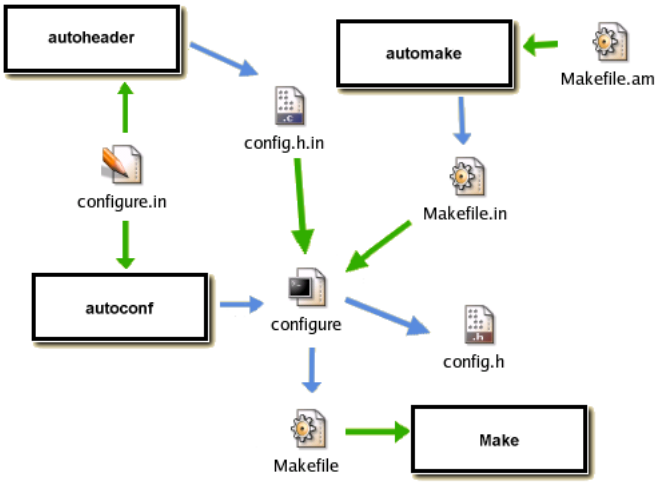
Nous avons déjà parlé de cet utilitaire dans le paragraphe ci-dessus en introduisant autoconf, référez vous y en cas de besoin.

### 2.4.3 Autoscan

Cet outil est une aide au mainteneur destiné uniquement à la mise en place du système autoconf et automake pour un projet existant ou en cours de création, il ne sera pas utilisé par la suite.
Exécuté à la racine de l'arborescence du projet, autoscan va examiner tout les fichiers sources en entrant dans les répèrtoires de manière récursive, et va générer un fichier nommé **configure.scan** qui sera un squelette de fichier configure.in qu'il faudra éditer.
Ceci avait pour bût premier de permettre aux mainteneurs de gros projets de passer plus facilement à automake/autoconf et nous servira pour débuter dans l'édition du **configure.in**.

## 2.5 Récapitulation

Voici un résumé des relations entre les outils que nous avons présenté ici et les différents fichiers générés et exploités :




Nous avons donc les fichiers suivants :

- à la charge du mainteneur :
  - **configure.in** : Fichier central de ce système il peut être concu à partir d'un squelette généré par autoscan (configure.scan) à partir des sources du projet.
  - **Makefile.am** : Fichier décrivant les sources du projet, les cibles, les objets et tout autre fichier devant être construit par l'exécution de *make*. De syntaxe très simple mais entièrement à la charge du mainteneur.
- générés indépendemment :
  - **aclocal.m4** : Généré par aclocal pour servir de source à autoconf et automake. Defini certaines macro en fonction du système du mainteneur.(ne concerne pas l'utilisateur).
- générés par le couple autoconf/automake :
  - **Makefile.in** : Généré par automake pour servir de base au script configure pour la création des fichiers Makefile.
  - **config.h.in** : Généré par autoheader pour servir de base au script configure pour la création du fichier config.h.
  - **configure** : Script effectuant les tests décrits dans le configure.in et générant les fichiers Makefile et le fichier config.h.
- générés par l'exécution du configure :
  - **config.h** : Fichier contenant les résultats des tests sous forme de "define" C.
  - **Makefile** : Un ou plusieurs fichiers utilisés par make pour la compilation.

# 3 Ecriture d'un configure.in

Comme nous l'avons déjà vu dans le chapitre précédent, configure.in est un des fichiers importants, avec les *Makefile.am* dans le système autoconf/automake puisque c'est lui qui contient les "règles" de création du script configure.

Dans un premier temps il est possible de se servir d'autoscan puis renommer le fichier configure.scan en configure.in ou encore d'utiliser le configure.in d'un autre projet comme model.
Il faudra de toute façon l'éditer pour le faire correspondre aux besoin du projet.
Ce fichier a les propriétés suivantes :

- Les lignes de commentaire commence par 'dnl ' avec au moins un espace après le 'dnl'. Cela signifie que les '#' du fichier configure.scan doivent être remplacés.
- AC_INIT(*file*) doit être la première macro et *file* doit être un path vers un fichier existant et relatif au répèrtoire contenant le configure.in. Ceci premettant à autoconf de vérifier s'il se trouve bien dans le bon répèrtoire.
- AC_OUTPUT(*[file1[,file2[...]]]*) doit être la dernière macro. Elle définie les fichiers devant êtres générés.
- Les autres macros sont **obligatoirement** placées entre AC_INIT et AC_OUTPUT
- AC_* sont les macros autoconf.
- AM_* sont les macros automake.
- Lorsqu'un argument d'une macro est sur plusieures lignes il faut l'encadrer entre [ et ].

L'ordre d'apparition des macros n'a pas d'importance hormis les règles précédentes et les éventuelles relations entres celles-ci (déclaration/utilisation de variables). Toutefois il est préférable d'utiliser, pour plus de lisibilité le shéma suivant :

```
AC_INIT(file)
dnl checks for programs
dnl checks for librairies
dnl checks for headers
dnl checks for typedefs
dnl checks for structures
dnl checks for compiler characteristics
dnl checks for libraries functions
dnl checks for system services
AC_OUTPUT(file)
```

Nous allons voir maintenant les macros les plus utilisées, par type. Bien entendu il en existe beaucoup d'autres et il vous faudra vous référer au manuel d'autoconf/automake pour leur documentation.(cf Annexes).

## 3.1 Macros d'initialisation

Initialisation obligatoires :

- **AC_INIT(*file*)** : doit référencer un fichier unique et constant (non généré). Le path doit être relatif au répertoire où se trouve le configure.in.

– **AC_INIT_AUTOMAKE(*package, version*)** : definit les variables PACKAGE et VERSION.

Initialisations facultatives :

– **AC_PREREQ(*version*)** : Fait échouer autoconf s'il n'est pas de cette version ou suppérieure.
– **AM_CONFIG_HEADER(*header*)** : Fait générer à autoconf le header de configuration "*header*", en général "config.h".
– **AC_CONFIG_AUX_DIR(*dir*)** : Référence un répertoire où seront mis les scripts.
– **AC_PREFIX_DEFAULT(*prefix*)** : Initialise le préfixe "*prefix*", par défault "/usr/local". Il est parfois intéressant de le mettre à "/usr".

### 3.2 Macros de génération

Génération obligatoire :

– **AC_OUTPUT(*[file1[,file2[...]]]*)** : Créer le fichier *file1* à partir de *file1.in* etc. (c'est ainsi que l'on génerera les fichiers 'Makefile').

Générations facultatives :

– **AC_DEFINE(*variable[,valeur,[description]]*)** : Défini *variable* à *valeur* (ou 1 par défaut) dans le config.h. (voir ci-dessous)
– **AC_SUBST(*variable*)** : Active la substitution de *variable* dans les Makefile (voir ci-dessous).

La Macro AC_SUBST substitue toute occurence de "@variable@" dans les fichiers de sortie de configure (Makefile) par la valeur de la variable shell 'variable'. Cette macro n'est utile que si vous ajoutez des variables dans les fichiers Makefile.am et que vous voulez que configure les "set". Il s'agit là d'une macro plutot "avancée".

La macro AC_DEFINE peut être très utile. Elle permet d'ajouter des 'define' dans le config.h (voir le chapitre "utilisation de config.h"). Si *variable* n'est pas standard (c'est à dire que c'est vous qui la créez) il FAUT que vous passiez trois arguments à cette macro.

Exemple :
on veut creer un define DEBUG nous indiquant s'il faut faire des "printf" partout (hum :).
Nous utiliserons une macro d'option (voir le chapitre correpondant) pour savoir si on doit mettre DEBUG à 1 ou à 0. Pour le Mettre à 1 il faudra appeler AC_DEFINE(DEBUG,1,[description]). Ceci nous donnera donc :

```
AC_ARG_ENABLE(debug, [ --enable-debug    set debug in code default=no],
   AC_DEFINE(DEBUG,1,[set to 1 if you want a lot of debug]),)
```

Ce n'est peut être pas très compréhensible pour l'instant mais ca le sera beaucoup plus lorsque nous aurons vu les macros de définition d'options. Sachez juste qu'avec ceci dans le configure.in, lorsque configure sera appelé avec l'option –enable-debug, le config.h aura une ligne supplémentaire :

```
/* set to 1 if you want a lot of debug */
#define DEBUG 1
```

## 3.3 Macros de tests

Les macros de test permettent de verifier si une condition est vérifiée. D'une manière générale soit elles ne prennent pas d'argument soit elles prennent les arguments suivant :

```
"test1[,test2[,...testN,[action-si-ok,[action-si-err]]]]"
```

Ces "tests" pouront porter aussi bien sur des fichiers que sur des programmes ou que sur des fonctions de bibliothèques et "action-si-ok" et "action-si-err" sont des commandes shell.
Les macros de test seront souvant suivies (dans le configure.in) de commandes shell effectuant les actions nécéssaires à l'interprétation du résultat et/ou de macro de message. Par exemple on testera l'existance des headers standards C et "action-si-err" sera d'afficher une erreur et de s'arreter (AC_MSG_ERR(message) vu plus loin).

### 3.3.1 Macros de tests de programmes

Tests de compilateurs :

- **AC_PROG_CC** : Cherche un compilateur C et positionne la variables 'CC'. Teste par ordre de préférence cc puis gcc.
- **AC_PROG_CPP** : Cherche un préprocesseur C et positionne la variable 'CPP'. Si '$CC -E' ne fonctionne pas, prend "/lib/cpp".
- **AC_PROG_CXX** : Cherche un compilateur C++ via la variable CXX, si elle est indéfinie, test c++ puis g++ et positionne la variable 'CXX' en conséquence.

Tests de programmes :

- **AC_PROG_INSTALL** : Recherche *install-sh* (il est conseillé de l'inclure dans la distribution du projet).
- **AC_PROG_AWK** : Teste la présence de mawk, gawk, nawk, awk dans cet ordre.
- **AC_PROG_LEX** : Teste la présence de flex, lex dans cet ordre.
- **AC_PROG_YACC** : Teste la présence de bison,yacc dans cet ordre.
- **AC_PROG_RANLIB** : Test la présence de ranlib.
- **AC_PROG_LN_S** : Test si ln -s fonctionne (lien symbolique).

Il est bien entendu possible de tester la présence de programmes pour lesquels il n'existe pas de macro particulière :

- **AC_CHECK_PROG(*var,prog,ok[,err]*)** : Teste si *prog* est dans le "PATH" si oui, var prend la valeur *ok* sinon, prend la valeur *err*.
- **AC_PATH_PROG(*var,prog[,err]*)** : Idem que AC_CHECK_PROG sauf que *var* prend la valeur du path de prog s'il est trouvé, *err* sinon.

### 3.3.2 Macros de tests de bibliothèques et de fonctions

Test de présence d'un bibliothèque :

**AC_CHECK_LIB(*lib,fonction,action-si-ok[,action-sinon]*)** :

Teste si un programme C compile correctement lorsqu'il est linké avec l'option -l*lib*.
*action-si-ok* est une liste de commandes à executer si la bibliothèque est trouvée.
*action-sinon* est une liste de commandes à executer si la bibliothèque n'est pas trouvée (erreur lors de la compilation du programme de test).
En général, si le test réussi, il faut :

- Ajouter -l*lib* à la variable 'LIB'
- Définir la variable HAVE_LIB*LIB*.

Exemple :

```
AC_CHECK_LIB(crypto, des_crypt,
             LIBS="$LIBS -lcrypto"; AC_DEFINE(HAVE_LIBCRYPTO) ,
             AC_MSG_ERROR([ libcrypto not found ! ]) )
```

Ou bien encore :

```
AC_CHECK_LIB(snmp, snmp_free_pdu,,,)
if test $ac_cv_lib_snmp_snmp_free_pdu = yes ; then
    LIBS="$LIBS -lsnmp"
    AC_DEFINE(HAVE_LIBSNMP)
else
    AC_MSG_ERROR([ libsnmp not found ! ])
fi
```

Test de fonctions particulières :

Il existe un grand nombre de macros testant des fonctions particulières, nous ne prendrons donc qu'un exemple, elle sont toutes faites sur le même modèle et si vous cherchez une macro pour une fonction particulière, la documentation d'autoconf est exhaustive.(voir annexes pour l'url de la documentation).

Exemple :

**AC_FUNC_WAIT3** : A pour effet de définir HAVE_WAIT3 à 1 dans le config.h.

Tests génériques de fonction :

Comme pour les bibliothèques, il est possible de tester des fonctions pour lesquelles il n'existe pas de macro particulière :

**AC_CHECK_FUNC(*fonction, action si OK[,action si non]*)**
**AC_CHECK_FUNCS(*fonctions ....,action si OK[,action si non]*)**

Note : si on cherche des fonctions dans une librairie il faut avoir positionné 'LIBS' correctement avant, en testant la présence de la bibliotheque avant par exemple. Dans le cas contraire la compilation du programme de test échourait même si la bibliothèque dispose de la fonction : configure ne différenciera pas l'erreur de link "bibliothèque non trouvée" et "fonction non trouvée".

Exemple :

```
AC_CHECK_FUNCS(getuid gettimeofday inet_pton strncasecmp strdup strerror drand48)
```

#### 3.3.3 Macros de test de headers

Comme pour les fonctions et les bibliothèques il existe des macros de test pour des headers particuliers, par exemple :

**AC_HEADER_STDC** :

Test si les includes ”stdlib.h”, ”stdarg.h”, ”string.h” et ”float.h” sont présent et défini la variable STDC_HEADER en conséquence.
De même, il existe une macro générique pour les tests de headers :
**AC_CHECK_HEADERS(*header ..., action si ok[, action sinon]*)** :

Défini pour chaque header trouvé la variable HAVE_*HEADER_H*.

#### 3.3.4 Macros de test de structures et typedefs

Tests de structures :

Un certain nombre de problèmes existent vis à vis des structures utilisées pour la représentation du temps : il s'agit, en effet, d'un des points ou les implémentations UNIX peuvent varier énormément. Autoconf propose donc des macros tres spécialisées permettant de résoudre ces problèmes mais nous ne les verrons pas toutes ici. Pour de plus amples détails référez-vous à la documentation d'autoconf.

**AC_STRUCT_TM** :

Teste si ”time.h” définit bien la structure *tm*. Dans le cas contraire, TM_IN_SYS_TIME est défini et indique que le header ”sys/time.h” doit définir la structure *tm*. (si ce n'était pas le cas le système ne disposerais pas de la librairie standard C ce qui serait de toute façon fatal à toute tentative de compilation d'un quelconque programme C).

**AC_STRUCT_TIMEZONE** :

Teste la méthode d'obtention de la ”timezone”.
Si la structure *tm* posède un membre *tm_zone* alors HAVE_TM_ZONE est défini. Si le tableau externe *tzname* est trouvé alors HAVE_TZNAME est défini. Enfin si ces deux test échouent il s'agit comme précédement d'une erreur ”fatale”.

Tests de Typedefs particuliers :

**AC_TYPE_SIGNAL** :

Certains UNIX définissent la fonction signal comme retournant un pointeur vers une fonction retournant un ”void”, d'autre un ”int”. Pour palier à ce problème, cette macro définira RETSIGTYPE à ”void” ou à ”int” selon le cas. Ainsi pour écrire un code ”portable” on utilisera cette macro dans le configure.in, dans le fichier source ou sera déclaré une fonction de gestion de signal on incluera ”config.h” et la fonction de gestion de signal sera alors déclarée ainsi :

```
RETSIGTYPE hup_handler (...param...)
{
  ...(code de la fonction)...
}
```

**AC_TYPE_PID_T** : Si le type pid_t n'existe pas il est définit comme int dans config.h.
**AC_TYPE_UID_T** : Si le type uid_t n'existe pas uid_t et gid_t sont définis comme int.
**AC_TYPE_SIZE_T** : Si le type n'existe pas il est défini comme unsigned int.
**AC_TYPE_OFF_T** : Si le type n'existe pas il est défini comme long int.

Test de typedef générique :

**AC_CHECK_TYPE(*type,defaut*)** : Si *type* n'existe pas il est défini comme le type de base *defaut*.

#### 3.3.5 Autres macros de test

**AC_C_BIGENDIAN** : défini WORDS_BIGENDIAN si c'est le cas.
**AC_C_CONST** : si le compilateur ne supporte pas le mot clef "const", defini const comme vide.
**AC_C_INLINE** : idem pour le mot clef "inline".
**AC_C_CHAR_UNSIGNED** : définit __CHAR_UNSIGNED__ si c'est le cas.
**AC_C_LONG_DOUBLE** : definit HAVE_LONG_DOUBLE si ce type est supporté.
**AC_CHECK_SIZEOF(*type*)** : definit SIZEOF_uctype au nombre d'octet du type ou 0 si le type est inconnu. uctype est le nom du type en majuscule avec les espaces et points remplacés par des _ et les '*' par des 'p'.
Exemple :
AC_CHECK_SIZEOF(*unsigned char **) définira SIZEOF_UNSIGNED_CHAR_P à N octets.

### 3.4 Macros de messages

Le script configure affiche des messages à l'intension de l'utilisateurs. Comme le mainteneur va construire ses propres tests, il existe de macro permetant un uniformité dans les messages produits par le script configure.

**AC_MSG_CHECKING(*feature*)** : affiche "Cheking *feature* ..." sans retour à la ligne.
**AC_CHECKING(*feature*)** : idem mais avec un retour à la ligne.
**AC_MSG_RESULT(*résultat*)** : complète la fin d'un 'checking feature...' habituellement "yes" ou "no".
**AC_MSG_WARN(*description*)** : notifie l'utilisateur d'un possible problème en affichant *description*.
**AC_MSG_ERROR(*description*)** : notifie l'utilisateur d'une erreur fatale en affichant "*description*" et quitte configure (erreur "fatale").

### 3.5 Macros de définition d'options

On distinguera deux types d'options que l'utilisateur poura passer aux script configure, les options relative à des packages exterieurs, qui s'activent avec "–with-*package*" et se désactivent avec "–without-*package*". Et les options relatives aux options du package (le projet lui-même), qui s'activent avec "–enable-*fontion*" et se désactivent avec "–disable-*fonction*. Autoconf propose les macros permettant de proposer ses options mais il est à la charge du mainteneur d'en écrire l'action en cas d'activation ou de desactivation.

#### 3.5.1 Options du package

Les options du package permettent au mainteneur d'offrire un certain choix à l'utilisateur, en lui donnant la possibilité d'inclure ou exclure des fonctionnalités optionnelles.

L'utilisateur n'aura qu'à executer la commande "configure –help" pour connaitre les options disponibles. Et pourra activer ou desactiver une option en lancant le script configure avec une option du type :

```
--enable-feature[=arg]
--disable-feature
```

Pour pouvoir proposer une fontion optionnelle, le mainteneur doit utiliser la macro suivante :
**AC_ARG_ENABLE(*feature, description, action_si_choisi[, action_si_non_choisi]*)**
Cette macro défini une variable shell "enableval" qui contient "yes" ou "no" selon l'activation de cette option.

Exemple :

```
AC_ARG_ENABLE(optimize, [ --enable-optimize    allow to optimize code [default=no]],
              enable_optimize=$enableval, enable_optimize=no)
if test "$enable_optimize" = "yes"; then
   CFLAGS="-Wall -ffast-math -fexpensive-optimizations $CFLAGS"
fi
```

Comme vous le voyez c'est au mainteneur de décider si une option est activé ou désactivé par défaut, selon l'action de "*action_si_non_choisi*"

#### 3.5.2 Options de packages exterieurs

De la même manière que les options du package, les options de package exterieur permettent au mainteneur de donner la posibiliter à l'utilisateur de spécifier où sont situés des packages extérieurs, et d'éventuellement désactiver leur usage. Ainsi l'utilisateur poura spécifier ou désactiver un package exterieur avec des arguments au script configure :

```
--with-package[=arg]
--without-package
```

Pour proposer ce type d'options le mainteneur doit utiliser la macro autoconf suivante :
**AC_ARG_WITH(*package, description, action_si_choisi[, action_si_non_choisi]*)**
Cette macro défini la variable "withval" à la valeur passée en argument ou bien "yes" ou "no".

Exemple :

```
AC_ARG_WITH(mysql, [ --with-mysql[=dir]    allow use of mysql [default=no]],
            with_mysql=$withval, with_mysql=no)
if test "$with_mysql" != "no"; then
  if test "$with_mysql" != "yes"; then
    LDFLAGS="$LDFLAGS -L${with_mysql}"
  fi
  AC_CHECK_LIB(mysqlclient, mysql_query,,,)
  if test $ac_cv_lib_mysqlclient_mysql_query = yes ; then
    ...
  fi
fi
```

### 3.6 Exemples de configure.in

### 3.6.1 Configure.in minimal

```
dnl gives an existing source file
AC_INIT(src/helloworld.c)

dnl package + version
AM_INIT_AUTOMAKE(helloworld, .0)

dnl Make Makefiles
AC_OUTPUT([Makefile
          src/Makefile
          doc/Malefile])
```

Comme vous le voyez on a juste besoin d'initialiser autoconf et automake puis de générer les fichiers Makefile.

### 3.6.2 Configure.in évolué

Voici un exemple pour un projet qui a besoin de :

- Un compilateur C++.
- La bibliothèque TCL/TK.
- La bibliothèque pthread.
- Les bibliothèques POSIX.

```
dnl ********* INITIALISATION ***********

AC_INIT(src/main.cpp)
AM_CONFIG_HEADER(config.h)

dnl ***** on utilise automake *****
AM_INIT_AUTOMAKE(myproj,1.0)

dnl Checks for programs.

AC_PROG_CC
AC_PROG_CXX
AC_PROG_INSTALL

dnl Checks for libraries.

AC_CHECK_LIB(tcl, Tcl_Init)
AC_CHECK_LIB(tk, Tk_Init)
AC_CHECK_LIB(X11, XOpenDisplay)
AC_CHECK_LIB(nsl, gethostbyname)
AC_CHECK_LIB(socket, connect)
AC_CHECK_LIB(pthread, pthread_create)

dnl Checks for header files.

AC_HEADER_STDC
AC_HEADER_SYS_WAIT
AC_HEADER_TIME

dnl Checks for typedefs, structures, and compiler characteristics.
```

```
AC_C_BIGENDIAN
AC_TYPE_UID_T
AC_TYPE_PID_T
AC_TYPE_SIZE_T

AC_CHECK_HEADERS([unistd.h fcntl.h strings.h pthread.h
                  bits/socket.h netinet/igmp.h netinet/igmp_var.h])
AC_MSG_CHECKING(for usleep)
AC_EGREP_HEADER(usleep, unistd.h, is_usleep=yes, is_usleep=no)
if test $is_usleep = yes; then
  AC_DEFINE(HAVE_USLEEP)
  AC_MSG_RESULT(yes)
else
  AC_MSG_RESULT(no)
fi

dnl check Multicast support

AC_MSG_CHECKING(checking if Multicast is supported)
AC_EGREP_HEADER(IP_ADD_MEMBERSHIP, netinet/in.h, is_mc=yes, is_mc=no)
if test $is_mc = yes; then
  AC_MSG_RESULT(yes)
  AC_DEFINE(HAVE_MULTICAST)
else
  if test $ac_cv_header_netinet_igmp_h = yes -o $ac_cv_header_netinet_igmp_var_h = yes;
  then
    AC_MSG_RESULT(yes)
    AC_DEFINE(HAVE_MULTICAST)
  else
    AC_MSG_RESULT(no)
  fi
fi

dnl Checks for library functions.

AC_FUNC_VPRINTF
AC_CHECK_FUNCS([getuid getpid gethostname gettimeofday
                select socket waitpid strchr strdup strerror
                memcpy uname])
if test x$ac_cv_func_strerror = xno; then
dnl Replace 'main' with a function in -liberty:
  AC_CHECK_LIB(iberty, main)
fi

dnl Make Makefiles
AC_OUTPUT(Makefile support/Makefile src/Makefile html/Makefile man/Makefile)
```

# 4 Ecriture d'un Makefile.am

Le ou les fichiers Makefile.am définissent les règles de génération des fichiers Makefile (en réalité des Makefile.in qui, eux, serviront à générer les fichiers Makefile). Leur Syntaxe est assez simple et permettra au mainteneur de rajouter facilement des sources et des cibles.
Il existe plusieurs façon de répartir les sources d'un programme sur le disque :

- **flat** : Toutes les sources dans le répertoire de base du projet. Dans ce cas il n'y a qu'un Makefile.am dans ce même répertoire.
- **deep** : Les sources sont réparties dans des sous-répertoires. Dans ce cas il y a un Makefile.am dans le répertoire de base du projet qui utilise (uniquement) la directive "SUBDIRS = DIR1 DIR2 ..." pour définir les sous-répertoires. Il y a aussi un Makefile.am dans chaque sous-répertoire concerné, le quel peut aussi utiliser "SUBDIRS" pour définir ses sous-répertoires.
- **shallow** : Certaines sources sont dans le répertoire de base du projet d'autres dans des sous répertoires. Dans ce cas là il y a un Makefile.am par répertoire et une directive "SUBDIRS" à chaque fois que des sources sont placées dans des sous-répertoires du répertoire contenant le Makefile.am.

Le choix de cette organisation appartien au mainteneur, il est toutefois conseiller d'utiliser la méthode "deep" qui a le mérite d'etre la plus claire.
Automake et un outil GNU et a été concu comme le gestionaire de Makefile des projet GNU, par conséquence il peut, et il le fait par défaut, vérifier que le projet est conforme à l'organisation des projet GNU. Cette organisation conserne en particulier la présence de certains fichiers dans le répertoire de base. Il est possible de dire à automake quel type de vérification il doit faire via une option :

- **–foreign** : laxiste : ne rien vérifier (recommandé pour les projets non GNU).
- **–gnu** : conforme au standard GNU (par défaut).
- **–gnits** : conforme au standard GNITS.

Nous verrons dans le chapitre "Installation et integration" que cela a une incidence sur les fichier qui seront integrés à la distribution et à quel moment il faudra lancer automake avec une de ces options.

la sévérité "foreign" fait qu'automake ne vérifiera rien.

la sévérité "gnu" provoque l'arrêt si l'un de ces fichiers sont manquants :

- **INSTALL** : Documentation de l'installation.
- **NEWS** : Changements depuis la dernière release.
- **README** : Document à lire en premier.
- **COPYING** : license du logiciel.
- **AUTHORS** : Auteurs du logiciel.
- **ChangeLog** : Liste des modification au cours des release.

Nous verrons qu'il est possible de dire à automake de creer ces fichiers avec "–add-missing". De plus le mode "gnu" empeche l'utilisation des options (de configure) "no-installman" et "no-installinfo", qui permettent respectivement de ne pas installer les pages de manuel et d'info.

La sévérité "gnits" effectue les mêmes vérifications que "gnu" mais ajoute d'autres restrictions. Nous vous renvoyons vers la documentation d'automake pour plus de précision.

D'une manière généralle il est préférable d'utiliser "–foreign" ou bien si votre projet est GNU/GPL "–gnu –add-missing".

### 4.1 Syntaxe générale

La syntaxe des Makefile.am est tres simple, il s'agit toujours de :
**DIRECTIVE = ARG1 ARG2**
Attention à respecter les espaces !
Autre remarque : si les arguments ne tiennent pas sur une ligne, il est possible de les mettre sur plusieur à condition d'échaper le retour à la ligne avec un backslash (que je ne sais pas faire en latex :) et que la ligne suivante débute par au moins 8 espaces. Attention des espaces, pas des tabulations !.
Ce qui nous donne :

```
DIRECTIVE = arg1 arg2 arg3 \
        arg4 arg5
```

Nous allons voir quelles sont ces fameuses "directives" dans les chapitre suivant ("SUBDIRS" en est une).
Il est aussi possible d'inclure dans le Makefile.am des commandes de Makefile qui seront alors integées dans le Makefile.in puis dans le Makefile.

Les lignes du Makefile.am commencant par "##" ne sont pas interprértées et ne sont pas recopiées dans le Makefile.in.

### 4.2 Définition des sources

Une des données que doit contenir le Makefile.am est la liste des programmes, de leurs dépendances et des fichiers sources. Le format est simple :
***cible*_SOURCES = f1.c f2.c f3.c**
On peut aussi utiliser nos propres variables comme par exemple :

```
CLI_SRCS = client.c gui.c net.c common.c
CLI_INCL = client.h gui.h net.h common.h
SRV_SRCS = server.c data.c
SRV_INCL = server.h data.h

client_SOURCES = $(CLI_SRCS) $(CLI_INCL)
server_SOURCES = $(SRV_SRCS) $(SRV_INCL)
```

Il est possible aussi de définir des pramètres d'include avec la directive "INCLUDE" ou de librairie en utilisant le prefix de "cible" ainsi :

```
INCLUDES = -I.. -I../includes

cible1_LDADD = $(SRCDIR)/mylib/mylib.a $(XLIBS) -lm
cible2_LDADD = $(SRCDIR)/mylib/mylib.a
```

Nous verrons comment définire une ”cible” dans le chapitre prochain. Pour ensuite prendre un exemple concret.

### 4.3 Définition de cibles

Les cibles sont les programmes ou librairies que vous voulez construire ainsi que certaines cibles standard du type ”install”, ”noinst”, ”all” etc.
Il Faut définire vos cibles c’est à dire au moins un programme. Ces définitions suivent des règles simples :
Pour les programmes :

- La partie gauche est constituée d’un préfixe suivi de ’_’, suivie d’un primaire en capitales.
- Le préfixe représente le répertoire d’installation : bin = $prefix/$bindir
- bin_PROGRAMS = cible1 cible2 : les programmes à installer (et donc à construire).
- noinst_PROGRAMS = cible3 : programme à ne pas installer (mais à construire quand même)
- bin_SCRIPTS = script : script à installer
- Les primaires possibles sont ”PROGRAMS”, ”LIBRARIES”, ”SCRIPTS”, ”HEADERS”, ”DATA” et ”MANS”

Exemple :

```
bin_PROGRAMS = client server

noinst_PROGRAMS = test_server
bin_SCRIPTS  = client.sh
```

Pour les bibliothèques, c’est à peu près identique sauf que l’on utilise pas la primaire ”PROGRAMS” mais ”LIBRARIES” et ”LTLIBRARIES” selon si la biblothèque est statique ou partagée :

```
xxx_SOURCES =  x1.c x2.c x3.c x4.c
lib_LIBRARIES =  libxxx.a          ## bibliothèque statique à installer
yyy_SOURCES = y1.c y2.c y3.c y4.c
noinst_LIBRARIES = libyyy.a        ## bibliothèque statique à ne pas installer
zzz_SOURCES = z1.c z2.c z3.c z4.c
lib_LTLIBRARIES = libzzz.la        ## bibliothèque dynamique partageable à installer
```

### 4.4 Exemple complet

Nous allons prendre un exemple le plus ”standard” possible, il vous est conseiller de procéder comme dans cet exemple.
Supposons que nous ayons un projet organisé selon le model ”deep” et contenant l’arborescence suivante :

```
.
|-- Makefile.am
|-- configure.in
```

```
|-- src
|   |-- Makefile.am
|   |-- compute.c
|   |-- compute.h
|   |-- main.c
|   |-- main.h
|   |-- parse_func.c
|   `-- parse_func.h
`-- test
    |-- Makefile.am
    `-- compute-test.c
```

Le Makefile.am de la ”racine” du projet doit contenir ceci :

```
DIRS = src test
```

Le Makefile.am de ”test/” contiendra ceci :

```
bin_PROGRAMS = compute-stress-test
compute_stress_test_SOURCES = compute-test.c
```

Le Makefile.am de ”src/” contiendra ceci :

```
AM_CXXFLAGS = -Wall -D_REENTRANT
CXXFLAGS = -g -O0
INCLUDES = -I$(top_builddir) -I$(top_srcdir)
bin_PROGRAMS = myprog
myprog_SOURCES = compute.c compute.h main.c main.h \
                 parse_func.c parse_func.h
```

Ainsi les makefiles permettant de compiler les differents programmes seront générés. Les options à passer au compilateur et au linker auront été définies soit dans le configure.in avec les variables ”LIBS”, ”CPPFLAGS” (ou ”CFLAGS”)..., soit dans le Makefile.am avec les variables ”CXXFLAGS” ”INCLUDE” ...

### 4.5 Makefile conditionel

Il est possible de faire des ”if ... else ...” dans les Makefiles.am, ce qui peut être utile dans certains cas, comme la gestion d’options de programmes externes vu précedement. Ainsi il faut avoir mis dans le configure.in une directive du type :
**AM_CONDITIONAL(*condition, test*)** : où ”condition” est positionnée au résultat de ”test”.
Il est alors possible d’utiliser ”condition” dans le Makefile.am ainsi :

```
Dans le configure.in :
-----------------------------------------------------------------
AC_ARG_ENABLE(debug,
[  --enable-debug    Turn on debugging],
[case "${enableval}" in
  yes) debug=true ;;
  no)  debug=false ;;
  *) AC_MSG_ERROR(bad value ${enableval} for --enable-debug) ;;
esac],[debug=false])
AM_CONDITIONAL(DEBUG, test x$debug = xtrue)
-----------------------------------------------------------------
Dans le Makefile.am :
```

```
----------------------------------------------------------------------
if DEBUG
  DBG = debug
else
  DBG =
endif
noinst_PROGRAMS = $(DBG)
----------------------------------------------------------------------
```

# 5 Utilisation du config.h

Nous avons vu précédement que le fichier config.h était généré par configure et contenait les résultats des tests qu'il avait effectué.
Malheureusement la seule présence du fichier config.h à la racine du projet ne suffit pas à rendre celui-ci portable.
Il faut en effet prendre en compte les "define" qu'il contient dans le code source du projet.

Exemple :

On utilise *strerror_r* pour récupérer le message d'erreur dont le code est dans errno. Il s'agit de la version thread_safe de *strerror* voir "man strerror". Or en lisant la page de man on apprend que certaines implémentations la déclare comme ceci :

**char *strerror_r(int errnum, char *buf, size_t n) ; (DGUX et HPUX)**

et d'autres ainsi :

**int strerror_r(int errnum, char *buf, size_t n) ; (linux glibc2+ et le reste)**

Par conséquent on a ajouté au configure.in la macro de test prévue à cet effet :
**AC_FUNC_STRERROR_R**
elle a pour effet de définire : **HAVE_STRERROR_R** si strerror_r est déclaré dans un header et **STRERROR_R_CHAR_P** si elle retourne un "char *".
Pour en prendre compte il faudra dans les fichiers utilisant cette fonction, include "config.h" de cette façon :

```
#if HAVE_CONFIG_H
  #include "config.h"
#endif
```

Puis prendre en compte les "define" dans le code. Ce qui donnera un code resemblant à cela :

```
[...]
/* déclaration de variables */
#ifdef HAVE_STRERROR_R
  char strerr[STRERR_BUFFLEN];
  #ifdef STRERROR_R_CHAR_P
    char * StrErrRet=NULL;
  #else
    int StrErrRet=0;
  #endif // STRERROR_R_CHAR_P -> Type de retour de strerr_r
#endif // HAVE_STRERROR_R
[...]
/* appel de strerr_r */
#ifdef HAVE_STRERROR_R
      StrErrRet=strerror_r(Error,strerr,STRERR_BUFFLEN-1);
#endif
[...]
```

Vous en conluerez donc deux choses :

- Sans Automake/Autoconf il aurait été très difficile de prendre en charge cela sans une intervention de l'"utilisateur".
- La portabilité est un problème complexe qui peut très rapidement couter un temps considérable au développeur.

# 6 Distribution et integration

## 6.1 Distribution

Automake prévoit une cible automatique nommé *dist* qui permet de constrire le ”tar.gz” distribuable de votre logiciel. Pour cela il suffit d'avoir effectivement déclaré VERSION et PACKAGE dans le configure.in via la macro ”AM_INIT_AUTOMAKE(package, version)”.
Les fichiers inclus dans le ”tar.gz” sont ceux déclarés dans les Makefile.am hormis les ”cibles” et ceux déclarés ”noinst”.
Les fichiers suivants sont inclus systématiquement s'il existent : ”configure.in”, ”configure”, ”aclocal.m4”, ”acconfig.h”, ”config.h.in”, ”Makefile.am”, ”Makefile.in”, ”README”, ”INSTALL”, ”ChangeLog”, ”NEWS”, ”COPYING”, ”VERSION”, ”TODO”, ”AUTHORS”, ”THANKS”.
Il est possible d'inclure n'importe quel fichier en le déclarant dans le Makefile.am du répertoire ou il se trouve ainsi :
**EXTRA_DIST = fichier1 fichier2**

Une autre cible est définie automatiquement : *distcheck*
elle permet de verifier le bon fonctionnement de la cible ”dist” : elle créé le tar.gz puis le vérifie.

Ainsi pour créer une distribution de votre projet il vous suffit dans le répertoire de base de celui-ci de taper :
*make distcheck*
et si tout ce passe bien :
*make dist*
Et vous aurez allors un nouveau fichier nommé *PACKAGE-VERSION*.tar.gz qui contiendra les sources distribuables de votre projet.

## 6.2 Integration

Par integration on entend integration au sein de CVS.
En utilisant Automake/Autoconf, une partie des fichiers de l'arborescence de votre projet sont générés par d'autres, de plus ils sont pour partie dépendants du système sur lequelle s'effectue la compilation. Il convient donc de :

- **Ne surtout PAS** integrer les fichiers générés au CVS.
- Prevoir un script d'appel à automake/autoconf pour les développeurs.

Le script d'appel à autoconf/automake pourait être celui ci (bootstrap.sh placé dans le répertoire de base du projet) :

```
#!/bin/sh
set -x
aclocal
autoheader
automake --add-missing
```

```
autoconf
```

Ainsi les développeurs après un *checkout*, n'auraient qu'a lancer ce script pour regénerer les fichiers.
Notez bien qu'en cas de modification des fichiers Makefile.am, un "make" le détecte et relance le script "configure".
Par contre il faudra relancer "bootstrap.sh" à chaque modification du configure.in.

Ainsi, pour ne pas integrer les fichiers générés, vous devriez avoir un fichier ".cvsignore" contenant :

```
*.cache
Makefile.in
Makefile
configure
config.h.in
aclocal.m4
stamp-h.in
config.h
config.log
config.status
stamp-h1
depcomp
install-sh
missing
mkinstalldirs
```

Note : selon les versions d'autoconf et d'automake que vous utiliserez ces fichiers peuvent changer. Pensez à ajouter au ".cvsignore" les fichiers qui sont générés.
Note2 : Oui ce n'est pas une erreur il ne faut PAS ajouter les fichiers "Makefile" au CVS.
Note3 : Il vous faut un ".cvsignore" par répertoire géré par cvs.
Note4 : C'est dans le script bootstrap.sh que vous pouriez ajouter les options "–foreign" ou "–gnits".

# 7 Annexes

## 7.1 Pointeurs

Homepage Automake :
http ://www.gnu.org/software/automake/

Documentation Automake :
http ://www.gnu.org/manual/automake/index.html

Homepage Autoconf :
http ://www.gnu.org/software/autoconf/autoconf.html

Documentation Autoconf :
http ://www.gnu.org/manual/autoconf/index.html

Homepage libtool :
http ://www.gnu.org/software/libtool/

Documentation libtool :
http ://www.gnu.org/software/libtool/manual.html

Un tutoriel intéressant :
http ://seul.org/docs/autotut/

Un document un peu obscure de chez redaht :
http ://sources.redhat.com/autobook/autobook/autobook_toc.html

# 8 GNU Free Documentation License

Version 1.1, March 2000



**Preamble**

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom : to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation : a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals ; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 8.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

### 8.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 8.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts : Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material

on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 8.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version :

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.

– Preserve the section entitled “History”, and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
– Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
– In any section entitled “Acknowledgements” or “Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
– Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
– Delete any section entitled “Endorsements”. Such a section may not be included in the Modified Version.
– Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another ; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

### 8.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you in-

clude in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

### 8.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

### 8.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

### 8.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the

original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

### 8.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

### 8.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http ://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### ADDENDUM : How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page :

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation ; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST" ; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.