

# Concepts et outils de développement

Frédéric Goualard

Laboratoire d'Informatique de Nantes-Atlantique, UMR CNRS 6241  
Bureau 208



# Objectifs du module

- ▶ Découverte/utilisation des outils de développement :  
Gestion de version, débogueur, profileur, applications de configuration/déploiement, ...
- ▶ Rédaction de documentation
- ▶ Sensibilisation à la sûreté du logiciel
  
- ➡ Etre capable de participer efficacement à un projet de développement collaboratif
- ➡ Rédiger du code de qualité

Enveloppe horaire :

- ▶ Cours magistral

1. Configuration et déploiement
2. Gestion de version
3. Documentation du code source
4. Documentation du logiciel
5. Sécurité du code, tests et débogage

# Configuration et déploiement

## Développement d'une application :

- ▶ Configuration de la compilation
  - ▶ Dépendance p.r. matériel
  - ▶ Dépendance p.r. système d'exploitation (type, version)
  - ▶ Dépendance p.r. logiciels/bibliothèques disponibles
- ▶ Déploiement :
  - ▶ Adaptation aux souhaits de l'utilisateur (bibliothèque à utiliser, fonctionnalités disponibles)
  - ▶ Choix de la méthode d'installation



- ▶ Configuration/déploiement suivant les normes GNU
  - ▶ Langages considérés : C et assimilés
- ▶ Plateformes considérées : Unix/Linux



- ▶ Documents et transparents du cours sur madoc
- ▶ *Autotools Tutorial*, Alexandre Duret-Lutz
- ▶ *GNU coding standards*, R. Stallman et al.
- ▶ *GNU Make : a program for directing recompilation*, R.M. Stallman et R. McGrath
- ▶ *Autoconf : creating automatic configuration scripts*, D. MacKenzie et Ben Elliston
- ▶ *Automake*, D. MacKenzie et Tom Tromey
- ▶ *GNU libtool*, G. Matzigkeit et al.



## Développement et distribution d'un logiciel :

- ▶ Portabilité
  - ▶ Différentes architectures  
*Big endian vs. little endian, 32/64 bits, ...*
  - ▶ Différents systèmes d'exploitation/compilateurs
- ▶ Idiosyncrasies
  - ▶ Présence d'une librairie particulière
  - ▶ Gestion des bogues de certaines versions
- ▶ Gestion des dépendances
- ▶ Optimisations pour certaines architectures



- ▶ Fonctions pas définies partout (e.g., `strtod()`)
- ▶ Comportements différents (e.g., valeur renournée pour `malloc(0)`? `rand()` sur 16 ou 32 bits?)
- ▶ Prototypes différents (e.g., `signal()` prenant en paramètre une fonction renvoyant `int` ou `void`?)
- ▶ Fonctions dans des librairies différentes (e.g., `pow()` dans `libm` ou `libc`?)
- ▶ Fonctions définies dans des en-têtes différents (e.g., `string.h` ou `strings.h`?)



- ▶ Différentes formes d'Unix ( $\approx 70$ ) :
  - ▶ Unix, Bell Labs.
  - ▶ SunOS, SUN MicroSystems
  - ▶ Ultrix, Digital Equipment Corps.
  - ▶ HP-UX, Hewlett-Packard
  - ▶ Linux
  - ▶ ...
- ▶ Développement d'un logiciel « sous Unix »
  - ▶ Détermination de l'architecture ?
  - ▶ Disponibilité d'une fonction/d'un en-tête ?



- ▶ *Compilation conditionnelle basée sur l'OS :*

```
#if (defined(__SYSV__) || defined(SUN))
    extern int *toto();
#else
    extern char *toto();
#endif
```

- ▶ Augmentation des variantes ⇒ impraticable
- ▶ Code parsemé de #ifdef ⇒ illisible
- ▶ SUN : passage BSD à SVR4 ⇒ choix sur l'OS ?



- ▶ *Compilation conditionnelle basée sur les services :*

```
#if HAVE_TOTO
    printf("%d",toto(a));
#endif
```

- ▶ script configure par L. Wall (pour Perl)
- ▶ metaconfig, L. Wall, H. Stenn et R. Manfredi  
Interaction avec l'utilisateur pour l'identification des services disponibles
- ▶ Cygnus configure par K. R. Pixley et GCC configure par R. Stallman
- ▶ **GNU autoconf** par D. MacKenzie



- ▶ Convergence dans autoconf : 1994
- ▶ Autoconf :
  - ▶ Ensemble de macros pour créer un script adaptant un patron de Makefile (`Makefile.in`) à la machine cible
- ▶ Patrons `Makefile.in` :
  - ▶ Larges portions communes
  - ▶ Rigueur pour coller au *standard GNU*
- ▶ automake pour créer `Makefile.in` à partir d'un patron `Makefile.am`



- ▶ Imake (X Windows)  
Informations sur les différents systèmes hard-codées
- ▶ qmake (<http://doc.trolltech.com/3.0/qmake-guide.html>)  
Pas de configuration bas niveau
- ▶ Jam (<http://www.perforce.com/jam/jam.html>)
- ▶ CMake (<http://www.cmake.org/HTML/>)
- ▶ SCons (<http://www.scons.org/>),  
AAP (<http://www.a-a-p.org/>), ...
- ▶ Boost.Build (<http://www.boost.org/boost-build2/>)

Monde Java :

- ▶ Ant / Maven

- ▶ GNU Make
- ▶ Autoconf
- ▶ Automake
- ▶ Libtool
- ▶ GNU Coding Standards
  - ▶ Internationalisation (Gettext)
  - ▶ GNULib

# GNU Make



- ▶ Logiciel composé de nombreux fichiers :
  - ▶ Fichiers sources (.c, .cpp, .h, ...)
  - ▶ Fichiers de documentation
- ▶ Modification d'un fichier
  - ▶ recompiler tous les fichiers qui en dépendent
- ▶ Calcul des dépendances ?
- ▶ Comment recompiler ?



Fichier Makefile : contient la liste explicite des dépendances

*Cible : Dépendances*

*Commande à exécuter n° 1*

*Commande à exécuter n° 2*

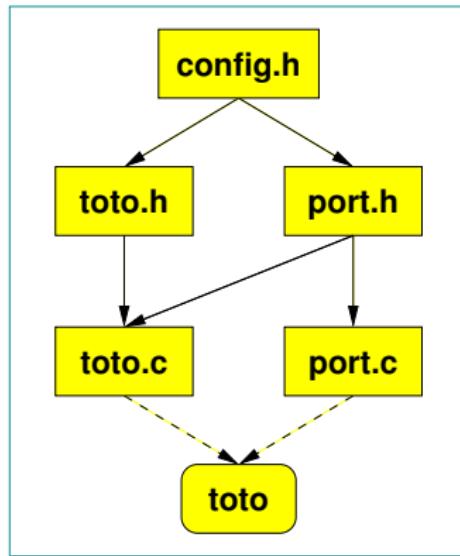
*...*

- ▶ *Cible :*
  - ▶ Fichier à créer (toto.o) ou action (clean)
- ▶ *Dépendances :*
  - ▶ Liste des fichiers nécessaires

Une *cible* est recréée si elle n'existe pas ou si elle est plus ancienne qu'une des *dépendances*



# Make (2)



Dépendances

Makefile:

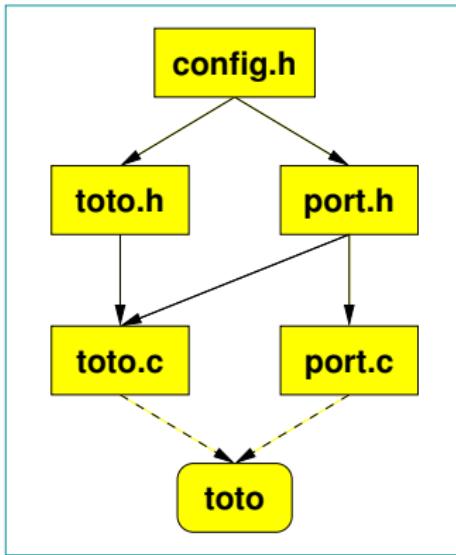
```
toto: toto.o port.o
      gcc -o toto toto.o port.o
toto.o: toto.c toto.h port.h
      gcc -c toto.c
port.o: port.c port.h
      gcc -c port.c
toto.h port.h: config.h
```

Recompilation sélective :

% make



## Make (2)



Dépendances

Makefile:

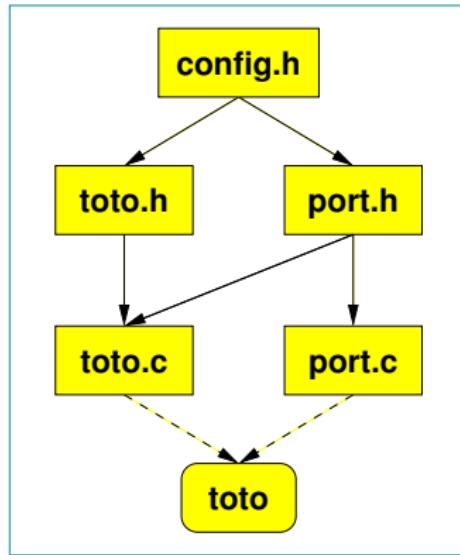
```
toto: toto.o port.o  
      gcc -o toto toto.o port.o  
toto.o: toto.c toto.h port.h  
      gcc -c toto.c  
port.o: port.c port.h  
      gcc -c port.c  
toto.h port.h: config.h  
      @touch toto.h port.h
```

Recompilation sélective :

% make



## Make (2)



Dépendances

Makefile:

```
toto: toto.o port.o  
      gcc -o toto toto.o port.o  
toto.o: toto.c toto.h port.h config.h  
      gcc -c toto.c  
port.o: port.c port.h config.h  
      gcc -c port.c
```

Recompilation sélective :  
% make



Contenu d'un Makefile (GNUmakefile, makefile) :

- ▶ *Définitions de variables*

Association nom/remplacement

- ▶ *règles explicites*

Pour chaque cible : liste des dépendances et méthode d'obtention de la cible

- ▶ *règles implicites*

Méthode d'obtention de cible pour une classe de fichiers

- ▶ *Directives*

Inclusion d'autres Makefiles, conditions, définitions de variables par des patrons

- ▶ Commentaires (introduits par #)

Make ne se limite pas au C/C++ ni même à des langages de programmation



# Variables

- ▶ Affectation :

<i>NomVariable</i> = <i>Valeur</i>	<i>paresseuse</i>
<i>NomVariable</i> ?= <i>Valeur</i>	<i>conditionnelle</i>
<i>NomVariable</i> := <i>Valeur</i>	<i>immédiate</i>
<i>NomVariable</i> += <i>Valeur</i>	<i>concaténation paresseuse</i> <i>ou conditionnelle</i>

- ▶ Utilisation :

```
$(NomVariable)
```

## Exemples :

```
sources = toto.c titi.c
CC = gcc
$(sources): config.h
monprog: $(sources)
        $(CC) -o monprog $(sources)
```



- ▶ Substitution sur des variables :

```
$(var : source=cible)
```

- ▶ Remplacement de la *fin* de chaque mot matchant *source* par *cible*

Exemple :

```
sources = toto.c titi.c
objects = $(sources:.c=.o)
```

- ▶ Substitution sur des commandes :

```
ladate := $(shell date)
all:
    echo $(ladate)
```



*Cible: Dépendances*

*Commande à exécuter n° 1*

*Commande à exécuter n° 2*

...

- ▶ *Cible et dépendances* : liste de noms de fichiers ou d'actions séparés par des espaces
- ▶ *Commande* : *obligatoirement* tabulée
- ▶ Extension sur plusieurs lignes : utilisation du '\' en fin de ligne

```
prog: toto.c tutu.c titi.c \
      tete.c tata.c
      gcc -o prog toto.c tutu.c titi.c tete.c \
            tata.c
```



- .PHONY : cible = action et non fichier

```
.PHONY: clean  
clean:  
        -rm *.o
```

À exécuter même si un fichier du même nom existe déjà

Note :

- -rm : ignorer les erreurs
- @rm : ne pas afficher la commande
- .PRECIOUS : cibles dont dépend .PRECIOUS pas détruites en cas d'arrêt impromptu de make (préservation de fichiers intermédiaires)

```
.PRECIOUS: parser.c  
# parser.c généré par bison à  
# partir de parser.y
```



*Cibles : Patron-cibles : Patron-dépendances*

Commandes

...

Variables automatiques :

\$@	nom du fichier cible (à gauche de ':')
\$<	nom de la première dépendance
\$^	nom de toutes les dépendances
\$?	nom de toutes les dépendances plus récentes que la cible
\$*	Portion du nom de fichier cible <i>matchant</i> un patron

Exemples :

```
objects = toto.o titi.o
$(objects): %.o: %.c
  $(CC) -c $(CFLAGS) $< -o $@
```

```
bigoutput littleoutput: %output: text.g
  generate text.g -$* > $@
```



# Conditions

```
ifeq ( arg1,arg2 )  
    Texte  
else  
    Texte  
endif
```

```
ifdef NomVariable  
    Texte  
else  
    Texte  
endif
```

---

```
ifneq ( arg1,arg2 )  
    Texte  
else  
    Texte  
endif
```

```
ifndef NomVariable  
    Texte  
else  
    Texte  
endif
```



## Conditions : exemples

```
lib_gcc = -lgnu
lib_icc = -lix86

ifeq ($(CC),gcc)
    $(CC) -o toto $(objects) $(lib_gcc)
else
    $(CC) -o toto $(objects) $(lib_icc)
endif
```



- ▶ Règles standards récurrentes  
(formation d'un '.o' à partir d'un '.c')
- ▶ Règles paramétrées par des variables pour adaptation

Exemples :

*Passage d'un .c à un .o :*

```
$(CC) -c $(CPPFLAGS) $(CFLAGS)
```

*Passage d'un .cc à un .o :*

```
$(CXX) -c $(CPPFLAGS) $(CXXFLAGS)
```



# Définition de règles implicites

- ▶ Utilisation de patrons :

*Liste de Préfixe%Suffixe : Liste de Préfixe%Suffixe  
Commandes*

Exemple :

```
%.tab.c %.tab.h: %.y
    bison -d $<
```



- ▶ Pas de facilité avec make
- ▶ Travail laborieux ; risque d'oubli lors des modifications
- ▶ Utilisation des facilités du compilateur :

```
% .d: %.c
$(SHELL) -ec '$(CC) -M $(CPPFLAGS) $< | \
sed '\''s/$*.o/& $@/g'\'' > $@'
```

- ▶ Création d'un fichier toto.d de règle explicite pour chaque fichier toto.c
- ▶ Inclusion dans le fichier Makefile :

```
sources = toto.c titi.c
include $(sources:.c=.d)
```



# Fonctions (1/2)

Appel :

```
$(foncname args)
```

- ▶ Manipulation de chaînes

```
$(subst a,e,une pomme)
```

- ▶ Manipulation de noms de fichiers

```
$(suffix pomme.c text.dat)
```

- ▶ Expansion conditionnelle

```
MLIB := $(if $(findstring -lm,$(LDFLAGS)),,-lm2)
```

- ▶ Appels extérieurs

```
syst := $(shell uname -s)
```

- ▶ Boucles

```
fics := $(foreach n,$(shell seq 5),fic$(n).c)
```



## Fonctions (2/2)

- ▶ Metaprogrammation

```
orig := $(origin LDFLAGS)
```

- ▶ Affichage de messages

```
$(error Fichier $(fic) inaccessible)
$(warning Environnement sous-optimal)
$(info Utilisation des instructions SIMD)
```

- ▶ Évaluation

```
$(eval LDFLAGS += -lm)
```

# Autoconf



`make` :

- ▶ Facilite la compilation
- ▶ Aucun apport en terme de maîtrise de la portabilité

`autoconf` :

- ▶ Langage de macros M4
- ▶ Création d'un script `sh` pour :
  - ▶ Test de l'environnement  
(présence de librairies, architecture, ...)
  - ▶ Adaptation des `makefiles`
  - ▶ Création de fichiers sources spécifiques



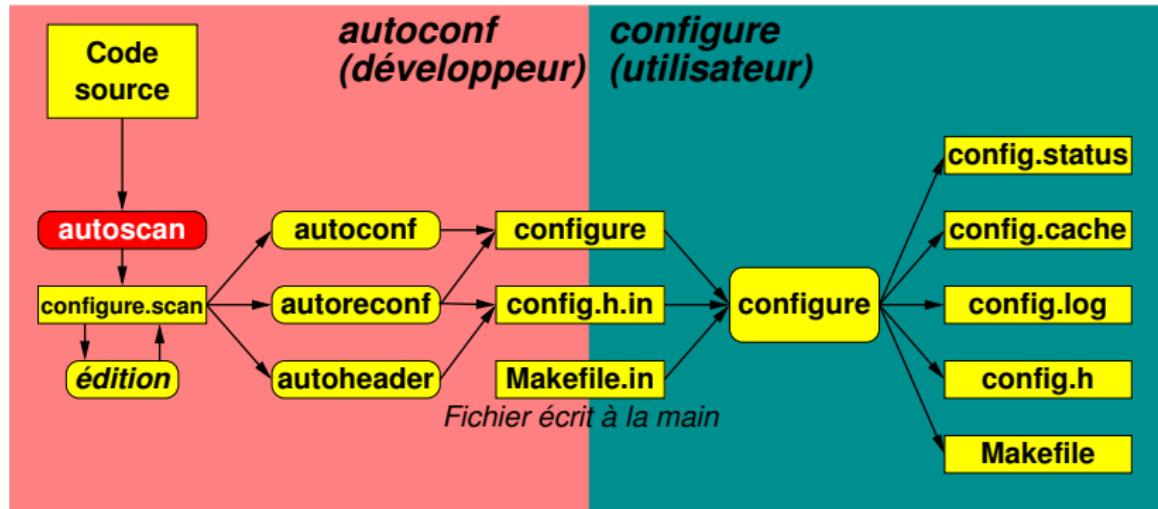
Configuration et installation d'une application/librairie :

```
% ./configure [option]
% make
% [make check]
% make install
```

Procédure standardisée pour tout paquetage GNU



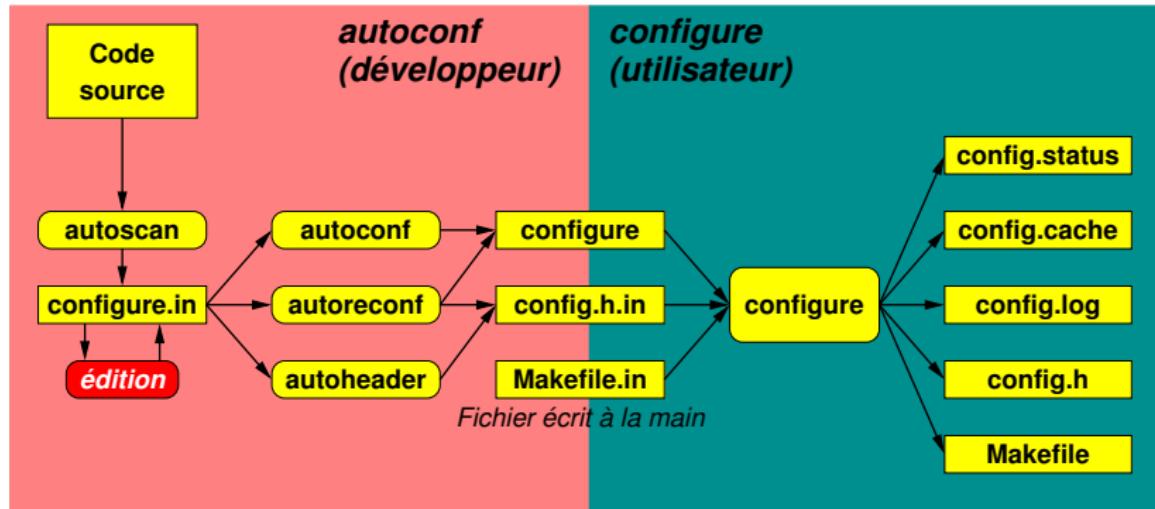
# Chaîne des actions



**autoscans** : examine les fichiers sources et crée un fichier **configure.scan** minimal



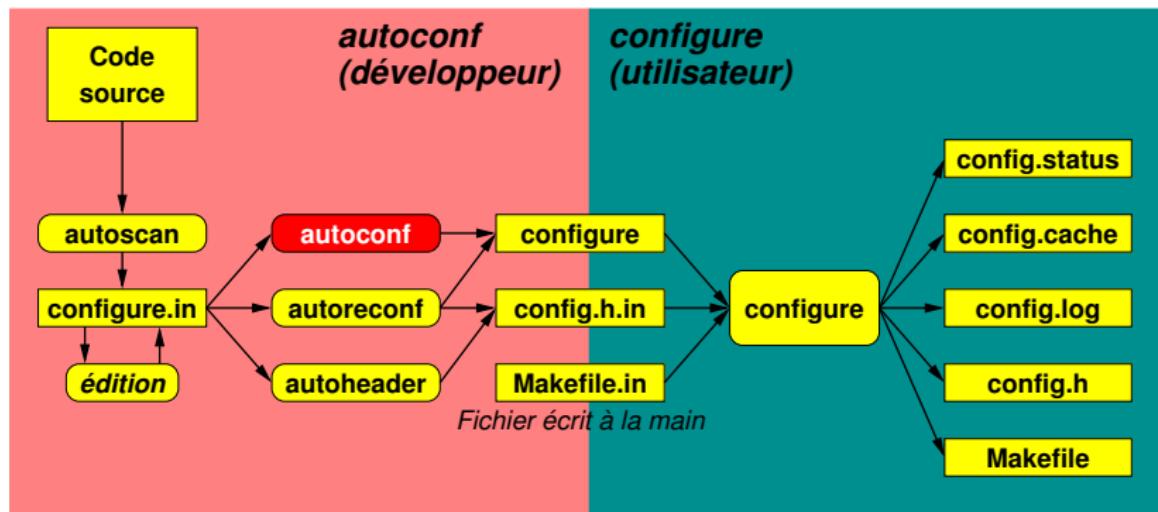
# Chaîne des actions



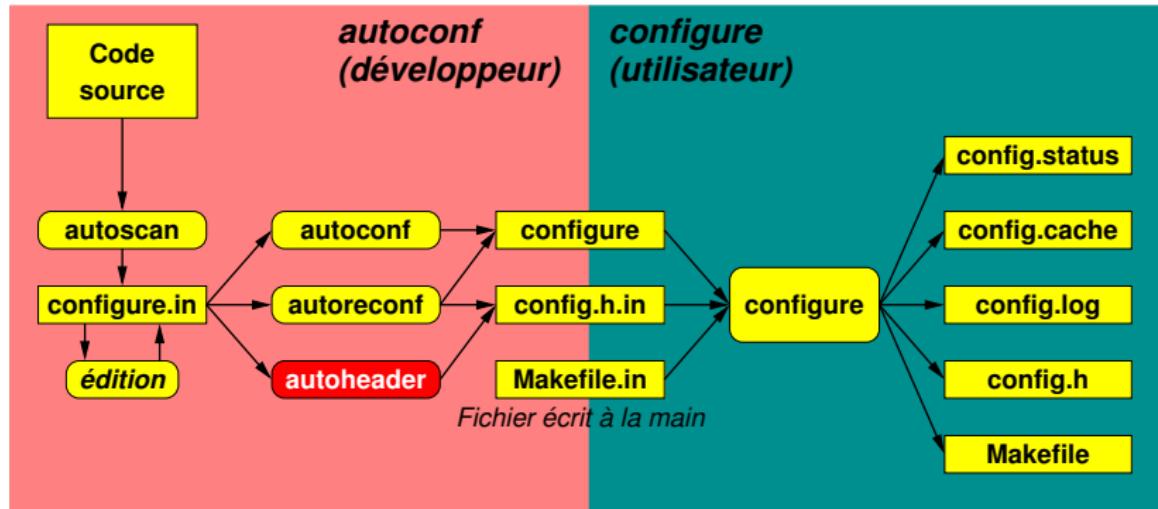
Édition du fichier `configure.scan` pour ajouter des tests, options,  
...⇒ `configure.in`



# Chaîne des actions



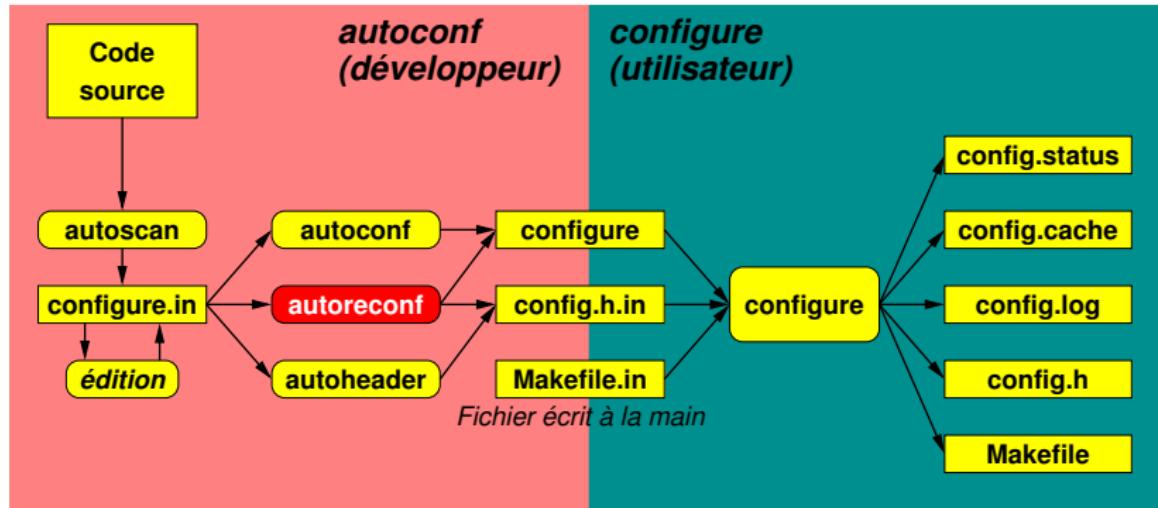
**autoconf** : création d'un fichier de scripts sh à partir de l'expansion des macros M4 de `configure.in` (utilisation du fichier local `aclocal.m4`)



**autoheader** : crée un patron de fichier d'en-tête avec les définitions de macros C déclarées dans `configure.in`



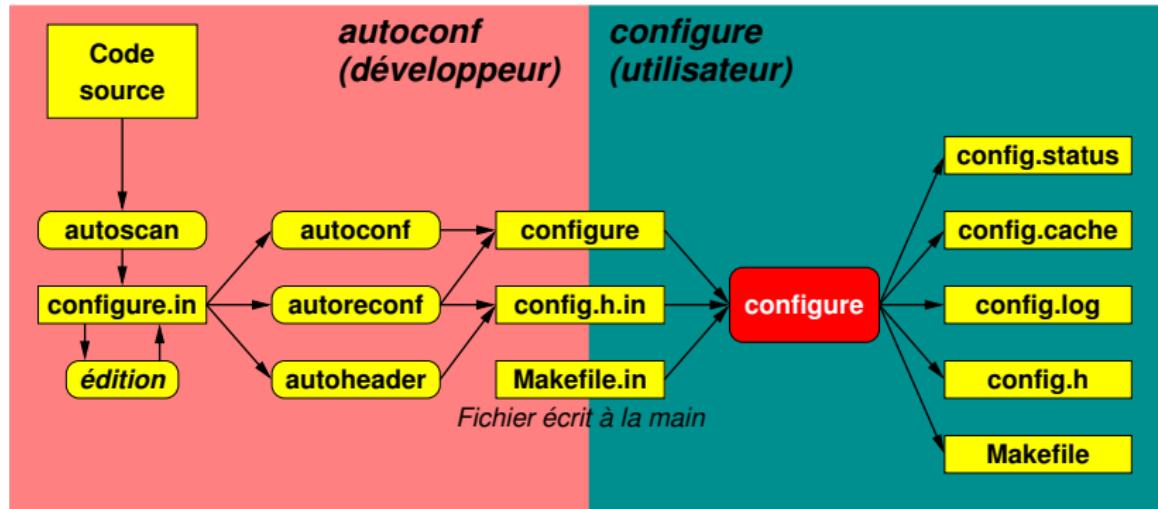
# Chaîne des actions



autoreconf : réinvocation de autoconf et autoheader lorsque nécessaire



# Chaîne des actions



**configure** : crée les fichiers Makefile à partir des patrons  
Makefile.in, config.h à partir de config.h.in, ...



`AC_INIT(application, version, adresse-rapport-bug)`

*Informations sur l'application*

*Recherche de programmes*

*Recherche de librairies*

*Recherche de fichiers d'en-tête*

*Recherche de types*

*Recherche de structures*

*Tests caractéristiques du compilateur*

*Recherche de fonctions dans les librairies*

*Recherche de services de l'OS*

`AC_CONFIG_FILES(Fichiers)`

`AC_OUTPUT`



# Exemple (1)

```
// expo.cpp

#include <iostream>
#include <cmath>
#define IX86 1
#define SPARC 0
#if IX86
    inline void round_downward() {
        asm ("fldcw %0" :
             : "m" (0x163F));
    }
    inline void round_upward() {
        asm ("fldcw %0" :
             : "m" (0x1A3F));
    }
#elif SPARC
    inline void round_downward() {
        int cw;
        asm ("st %%fsr,%0"
             : "=m" (*&cw));
        cw = (cw & 0x3fffffff)
             | 0xC0000000;
        asm ("ld %0,%%fsr" :
             : "m" (*&cw));
    }

```

```
inline void round_upward() {
    int cw;
    asm ("st %%fsr,%0"
         : "=m" (*&cw));
    cw = (cw & 0x3fffffff)
         | 0x80000000;
    asm ("ld %0,%%fsr" :
         : "m" (*&cw));
}
#endif
using std::cout;
using std::endl;
int main() {
    double a_dn, a_up;
    round_downward();
    a_dn = exp(-10.0);
    cout << a_dn << endl;
    round_upward();
    a_up = exp(-10.0);
    cout << a_up << endl;
    return 0;
}
```



## Exemple (2)

- ▶ Définition de fonctions en assembleur
- ▶ Portabilité : utilisateur définit le type de son architecture avec `#define`
- ▶ Passage à autoconf : création de `Makefile.in` et appel de `autoscan`
- ▶ But :
  - ▶ Gestion automatique de l'architecture
  - ▶ Gestion de la librairie mathématique utilisée
  - ▶ Options de compilation (débogage ou non)

```
# Makefile.in
LOADLIBES = @MATHLIB@
all: expo
```



## Exemple (3)

```
# configure.scan                                -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.
AC_PREREQ(2.61)
AC_INIT(FULL-PACKAGE-NAME, VERSION, BUG-REPORT-ADDRESS)
AC_CONFIG_HEADER([config.h])
# Checks for programs.
AC_PROG_CXX
# Checks for libraries.
# Checks for header files.
# Checks for typedefs, structures, and compiler characteristics.
AC_C_INLINE
# Checks for library functions.
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

Appel de autoscan et éventuellement ifnames



## Exemple (3)

```
# configure.ac                                -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.
AC_PREREQ(2.61)
AC_INIT([expo], [1.0.0], [frederic.goualard@univ-nantes.fr])
AC_CONFIG_SRCDIR([expo.cpp])
AC_CONFIG_HEADER([config.h])
AC_CANONICAL_HOST
case "$host" in
i?86-*linux*)
    AC_DEFINE([IX86],1,[Vaut 1 si compilation sur un ix86])
    ;;
sparc*-solaris*)
    AC_DEFINE([SPARC],1,[Vaut 1 si compilation sur Sparc/Solaris])
    ;;
*)
    AC_MSG_ERROR([architecture non supportée])
esac
# Checks for programs.
AC_PROG_CXX
# Checks for libraries.
AC_CHECK_LIB([m],have_exp=yes,have_exp=no)
if test "$have_exp" = "no"; then
    AC_MSG_ERROR([pas de librairie mathématique trouvée ou exp() pas dans libm])
else
    MATHLIB=-lm
fi
# Checks for header files.
# Checks for typedefs, structures, and compiler characteristics.
AC_C_INLINE
# Checks for library functions.
MATHLIB=-lm
AC_SUBST(MATHLIB)
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```



## Exemple (4)

- ▶ Appel de autoheader :

```
/* config.h.in. Generated from configure.ac by autoheader. */
/* Vaut 1 si compilation sur un ix86 */
#undef IX86
/* Define to the address where bug reports for this package should be sent. */
#undef PACKAGE_BUGREPORT
/* Define to the full name of this package. */
#undef PACKAGE_NAME
/* Define to the full name and version of this package. */
#undef PACKAGE_STRING
/* Define to the one symbol short name of this package. */
#undef PACKAGE_TARNAME
/* Define to the version of this package. */
#undef PACKAGE_VERSION
/* Vaut 1 si compilation sur Sparc/Solaris */
#undef SPARC
/* Define to '__inline__' or '__inline' if that's what the C compiler
   calls it, or to nothing if 'inline' is not supported under any name. */
#ifndef __cplusplus
#undef inline
#endif
```

- ▶ Modification de expo.cpp pour inclure le fichier "config.h"
- ▶ Copie locale de install-sh, config.sub, config.guess
- ▶ Appel de autoconf ⇒ configure
- ▶ Appel de ./configure ⇒ (Makefile, config.h)
- ▶ Appel de make ⇒ expo



## Exemple (5)

### Ajout d'un test de bogue sur la librairie mathématique

```
# configure.ac
AC_PREREQ(2.61)
AC_INIT([expo],[1.0.0],[frederic.goualard@univ-nantes.fr])
# ... code retiré ...
AC_LANG_CPLUSPLUS
LIBS=-lm
CXXFLAGS=-O2
AC_MSG_CHECKING(whether we are using a buggy GNU libc exp())
AC_TRY_RUN([double exp(double); int main(){double a=-1.0/0.0;
    exit(exp(a)==0.0);}], buggy_exp=yes, buggy_exp=no,:)
AC_MSG_RESULT(${buggy_exp})
if test "${buggy_exp}" = yes; then
    AC_MSG_RESULT([defining __NO_MATH_INLINES to avoid buggy GNU libc exp function])
    AC_DEFINE([__NO_MATH_INLINES], 1,
              [Define this to 1 if the GNU libc library is buggy.])
fi
# ... code retiré ...
AC_OUTPUT
```



## Exemple (5)

### Ajout d'une option pour le débogage (lancer autoreconf)

```
# configure.ac
AC_PREREQ(2.61)
AC_INIT([expo],[1.0.0],[frederic.goualard@univ-nantes.fr])
# ... code retiré ...
AC_LANG_CPLUSPLUS
LIBS=-lm
CXXFLAGS=-O2
AC_MSG_CHECKING(whether we are using a buggy GNU libc exp())
AC_TRY_RUN([double exp(double); int main(){double a=-1.0/0.0;
    exit(exp(a)==0.0);}],,buggy_exp=yes,buggy_exp=no,:)
AC_MSG_RESULT(${buggy_exp})
if test "${buggy_exp}" = yes; then
    AC_MSG_RESULT([defining __NO_MATH_INLINES to avoid buggy GNU libc exp function])
    AC_DEFINE([__NO_MATH_INLINES], 1,
        [Define this to 1 if the GNU libc library is buggy.])
fi
AC_ARG_ENABLE(debug,[--enable-debug request JAIL embedded debugging facilities],
    debug=${enable_debug},debug=no)
if test $debug = yes; then
    CXXFLAGS="-g2"
else
    CXXFLAGS="-g0"
fi
# ... code retiré ...
AC_OUTPUT
```



## Exemple (6)

- ▶ Modification de Makefile.in :

```
LOADLIBES = @MATHLIB@  
CXXFLAGS = @CXXFLAGS@  
  
all: expo
```

- ▶ Appels possibles de configure :

```
% ./configure --enable-debug  
% ./configure --disable-debug  
% ./configure --enable-debug=no
```



## Exemple (7)

Exemple d'exécution de `configure` :

```
% ./configure --enable-debug
creating cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking for c++... c++
checking whether the C++ compiler (c++) works... yes
checking whether the C++ compiler (c++) is a cross-compiler... no
checking whether we are using GNU C++... yes
checking whether c++ accepts -g... yes
checking whether we are using a buggy GNU libc exp()... no
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
```



# Tests prédéfinis (1)

- ▶ AC\_PROG\_CC : détermination du compilateur C
- ▶ AC\_CHECK\_PROG() : vérifie l'existence d'un programme

```
AC_CHECK_PROG(GPROF, gprof, gprof, prof)
```

- ▶ AC\_CHECK\_LIB() : vérifie l'existence d'une librairie

```
AC_CHECK_LIB(ultim, utan, mathlib_found=yes,  
            mathlib_found=no)
```

- ▶ AC\_CHECK\_FUNCS() : vérifie l'existence de fonctions (définition de la macro HAVE\_FUNCTION associée

```
AC_CHECK_FUNCS(decimal_to_double nextafter clock)
```



## Tests prédéfinis (2)

- ▶ AC\_CHECK\_HEADERS() : vérifie la présence d'en-têtes  
(définition de la macro HAVE\_ENTETE\_H associée)  

AC\_CHECK\_HEADERS(floatingpoint.h)
- ▶ AC\_C\_BIGENDIAN : définit WORDS\_BIGENDIAN si la machine est *big endian*
- ▶ AC\_CHECK\_SIZEOF() : détermine la taille d'un type



## Écrire ses tests (1)

- ▶ AC\_EGREP\_HEADER() : recherche d'une chaîne dans un en-tête

```
AC_EGREP_HEADER("getrusage",sys/resource.h,
  AC_DEFINE(GETRUSAGE_IN_HEADER,1,
  [Define this to 1 if function getrusage() is declared
   in sys/resource.h]),
  AC_DEFINE(GETRUSAGE_IN_HEADER,0,
  [Define this to 1 if function getrusage() is declared
   in sys/resource.h]))
```

- ▶ AC\_TRY\_COMPILE() : essaye de compiler un programme

```
AC_TRY_COMPILE(,bool x=false,has_bool=yes,has_bool=no)
```



## Écrire ses tests (2)

- ▶ AC\_MSG\_CHECKING() : informe de l'exécution d'un test

```
AC_MSG_CHECKING(whether ASM usable for rounding)
```

- ▶ AC\_TRY\_RUN() : Compile un prog. C et l'exécute

```
AC_TRY_RUN([static int _down=5695; static int _up=6719;
           int main(){double x, y, a=1.0, b=10.0;
           asm volatile ("fldcw _down"); x=a/b;
           asm volatile ("fldcw _up"); y=a/b; exit(x==y);}],
           fpu_asm=yes, fpu_asm=no,:)
```

- ▶ AC\_MSG\_RESULT() : affiche le résultat d'un test :

```
AC_MSG_RESULT(${fpu_asm})
```



- ▶ AC\_DEFINE() : définit une macro (à mettre dans config.h)

```
AC_DEFINE([IX86],1,[Vaut 1 si on compile sur un ix86])
```

- ▶ AC\_SUBST() : substitue une variable shell VAR par sa valeur dans tous les fichiers de AC\_OUTPUT où elle apparaît entre '@'

```
CXXFLAGS="-O2 -g0"  
AC_SUBST(CXXFLAGS)
```



- ▶ AC\_ARG\_WITH() : définit l'utilisation ou non d'un paquetage

```
AC_ARG_WITH(mathlib,
[--with-mathlib use MathLib as the mathematical library],
mathlib=${with_mathlib},mathlib=no)
```

- ▶ AC\_ARG\_ENABLE() : utilise ou non une fonctionnalité

```
AC_ARG_ENABLE(exceptions,
[--enable-exceptions raise exceptions instead of simply aborting],
exceptions=${enable_exceptions},exceptions=yes)
```



# Hiérarchie standard des répertoires

<i>prefix</i>	/usr/local
<i>exec-prefix</i>	<i>prefix</i>
<i>bindir</i>	<i>exec-prefix/bin</i>
<i>libdir</i>	<i>exec-prefix/lib</i>
...	
<i>includedir</i>	<i>prefix/include</i>
<i>datarootdir</i>	<i>prefix/share</i>
<i>datadir</i>	<i>datarootdir</i>
<i>mandir</i>	<i>datarootdir/man</i>
<i>infodir</i>	<i>datarootdir/info</i>
...	

```
% ./configure --prefix=/opt --mandir=/local/man
```



# Variables de configuration

Variables déterminées par les macros de autoconf ou automatiquement

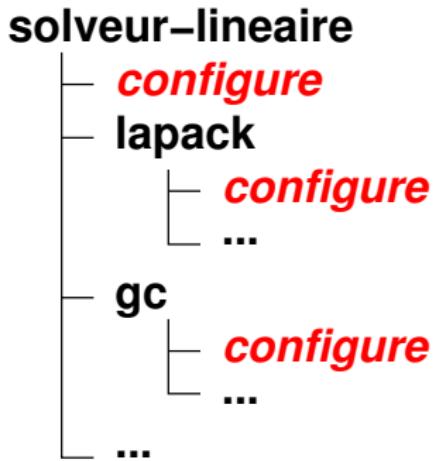
<i>CC</i>	Nom du compilateur C
<i>CFLAGS</i>	Paramètres du compilateur C
<i>CXX</i>	Nom du compilateur C++
<i>CXXFLAGS</i>	Paramètres du compilateur C++
<i>LDFLAGS</i>	Paramètres de l'éditeur de liens
<i>CPPFLAGS</i>	Paramètres du préprocesseur
...	

Variables pouvant être forcées par l'utilisateur à la configuration

```
% ./configure CC=icc
```



- ▶ Application/librairie distribuée avec le source d'autres applications/librairies dont elle dépend
- ▶ Configuration et compilation en cascade





## Atouts :

- ▶ Automatisation des tests (plus d'interaction avec l'utilisateur)
- ▶ Nombreux tests prédéfinis
- ▶ Écriture de nouveaux tests en shell
- ▶ Extension d'autoconf par ajout de macros M4

## Problèmes :

- ▶ GNU standard précis sur les fonctionnalités de `Makefile`
  - ▶ Cibles (`clean`, `distclean`, ...), gestion des variables d'environnement (`CFLAGS`, ...)
- ▶ Écriture d'un `Makefile.in` standard difficile et longue
- ▶ Nombreuses portions en commun avec toutes les applications

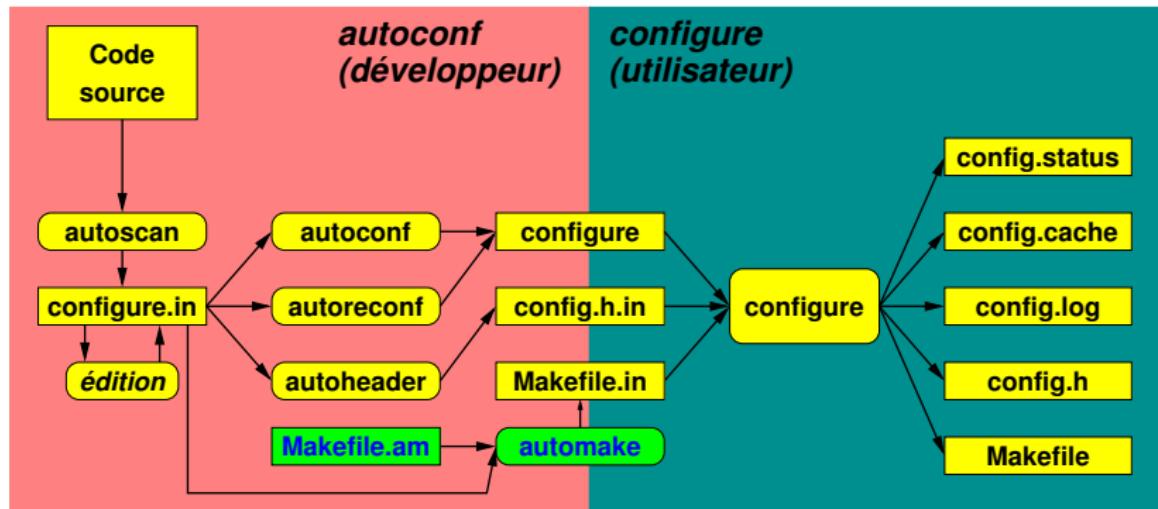
# Automake



- ▶ Factoriser le code commun aux `Makefile.in`
- ▶ Définir automatiquement les cibles standards
- ▶ Remonter encore d'un niveau :
  - ▶ Écriture d'un fichier `Makefile.am` par répertoire
  - ▶ Création automatique de `Makefile.in`
  - ▶ Utilisation de Perl pour la génération
- ▶ Gestion automatique des dépendances entre fichiers
- ▶ Toute variable en `AC_SUBST` est automatiquement une variable `Makefile`



# Chaîne d'exécution





- ▶ Ajout de `AM_INIT_AUTOMAKE()` dans `configure.in`

```
TOTO_VERSION="1.0.0"
AM_INIT_AUTOMAKE([toto], ${TOTO_VERSION})
```

- ▶ Remplacement de `AC_CONFIG_HEADER` par `AM_CONFIG_HEADER`
- ▶ Création de `Makefile.am` de plus haut niveau indiquant les sous-réertoires du logiciel

```
## Makefile.am
EXTRA_DIST = BUGS
SUBDIRS = toto doc check
## Création de l'archive de distribution
${PACKAGE}-${VERSION}.tar.gz: dist
## Création d'un rpm
rpm: ${PACKAGE}-${VERSION}.tar.gz
    rpm -ta -clean ${PACKAGE}-${VERSION}.tar.gz
```



Compilation du programme pig :

```
## Création d'un PROGRAM à installer dans bindir
bin_PROGRAMS = pig
pig_SOURCES = pig_parser.yy pig_lexer.ll pig.cpp \
    pig_internals.cpp \
    pig_common.cpp pig_expression.cpp
pig_LDADD = -ljail -lm
AM_YFLAGS = -d
EXTRA_DIST = pig_parser.h
```

- ▶ Normalisation des noms
- ▶ Interaction automatique avec les options de configure  
(bindir)



Définition de nouvelles règles en fonction d'extensions non standards :

```
SUFFIXES = .ypp
.ypp.cpp:
    $(YACC) $(AM_YFLAGS) $< \
    && mv y.tab.c $*.cpp \
    if test -f y.tab.h; then \
    if cmp -s y.tab.h $*.h; then \
    rm -f y.tab.h; \
    else mv y.tab.h $*.h; fi; \
    else :; fi
```



## Compilation de la librairie cell (utilisation de libtool)

```
EXTRA_DIST = sysdeps/*.* h stamp-h.in
lib_LT_LIBRARIES = libcell.la
MAINTAINER_CLEANFILES = cell_interval_lexer.cc cell_interval_parser.cc \
    cell_interval_parser.h
libcell_la_SOURCES = \
    cell_interval.cpp cell_profile.cpp cell_common.cpp \
    cell_parser.cpp cell_expression.cpp dtoa.c \
    cell_interval_parser.yy cell_interval_lexer.ll cell_port.cpp \
    cell_flt_output.cpp
AM_YFLAGS = -d
celldir=$(includedir)/cell
cell_HEADERS = cell cell.h cell_interval.h cell_fpu.h cell_limits.h \
    cell_profile.h cell_version.h cell_common.h cell_double_op.h \
    cell_parser.h cell_expression.h cell_port.h cell_config.h \
    cell_flt_output.h cell_parameters.h cell_dtoa.h cell_eval_stack.h \
    cell_expr_eval.h cell_expr_visitor.h cell_flags.h cell_interval_parser.h \
    cell_mathlib.h
install-data-local:
    $(mkinstalldirs) $(includedir)/cell/sysdeps
    $(INSTALL_DATA) $(srcdir)/sysdeps/*.* h $(includedir)/cell/sysdeps
```



# Nommage des variables automake

## Primaires

*prefixe – suffixe*

EXTRA	PROGRAMS
nodist	LIBRARIES
dist	SCRIPTS
nobase	DATA
notrans	HEADERS
<i>destdir</i>	MANS
bin	TEXINFOS
sbin	
libexec	
dataroot	
data	
sysconf	
include	
info	
...	

Exemples :

bin\_PROGRAMS = pig  
nodist\_DATA = toto.dat

## Variables derivees

{AM / nom }\_suffixe

SOURCES  
LDADD  
CPPFLAGS  
...

Exemples :  
pig\_SOURCES = pig.cpp  
pig\_CXXFLAGS = -O2  
AM\_YFLAGS = -g

# Libtool

- ▶ Outil facilitant la création et l'utilisation de librairies partagées
- ▶ Appel du compilateur/éditeur de lien = différent d'une machine et d'un OS à l'autre
- ▶ Librairie partagée doit être *installée* pour être utilisée
- ▶ libtool offre une interface uniforme
- ▶ Utilisation transparente avec automake : LTLIBRARIES
- ▶ Définition de quelques variables dans `configure.in`  
+ appel de `AM_PROG_LIBTOOL`
- ▶ Gestion automatique de `--enable-shared` avec autoconf



# libtool et automake

**makefile.am :**

```
lib_LTLIBRARIES = libmatrice.la
# portage.h : en-tête privé
libmatrice_la_SOURCES = matrice_creuse.cpp tableau.cpp \
arithmetic.cpp portage.h matrice.cpp
# matrice.h : en-tête public à installer dans includedir
include_HEADERS = matrice.h
libmatrice_la_LDFLAGS = -version-info $LT_CURRENT:$LT_REVISION:$LT_AGE\
-release $LT_RELEASE

bin_PROGRAM = solver
solver_SOURCES = main.cpp solver.cpp interface.cpp
solver_LDADD = libmatrice.la
```



# libtool et autoconf

```
MATRICE_MAJOR_VERSION=0
MATRICE_MINOR_VERSION=0
MATRICE_MICRO_VERSION=1
MATRICE_INTERFACE_AGE=0
MATRICE_BINARY_AGE=0
MATRICE_VERSION=$MATRICE_MAJOR_VERSION.$MATRICE_MINOR_VERSION.$MATRICE_MICRO_VERSION
AC_SUBST(MATRICE_MAJOR_VERSION)
AC_SUBST(MATRICE_MINOR_VERSION)
AC_SUBST(MATRICE_MICRO_VERSION)
AC_SUBST(MATRICE_INTERFACE_AGE)
AC_SUBST(MATRICE_BINARY_AGE)
AC_SUBST(MATRICE_VERSION)
LT_RELEASE=$MATRICE_MAJOR_VERSION.$MATRICE_MINOR_VERSION
LT_CURRENT='expr $MATRICE_MICRO_VERSION - $MATRICE_INTERFACE_AGE'
LT_REVISION=$MATRICE_INTERFACE_AGE
LT_AGE='expr $MATRICE_BINARY_AGE - $MATRICE_INTERFACE_AGE'
AC_SUBST(LT_RELEASE)
AC_SUBST(LT_CURRENT)
AC_SUBST(LT_REVISION)
AC_SUBST(LT_AGE)
AM_PROG_LIBTOOL
```



Librairie = ensemble d'interfaces (types, variables, fonctions, ...)

Version de librairie :

- ▶ Ensemble (consécutif) des interfaces implémentées par une librairie + position dans la série des implémentations de cet ensemble d'interfaces
- ▶ Trois nombres : *current:revision:age*
- ▶ *revision* édition d'une librairie implémentant les interfaces  
*current – age* ... *current*

Exemple :

- ▶ `libmatrice.la 7:2:4` :  
2<sup>e</sup> révision de la librairie implémentant les interfaces  $7 - 4 = 3$  à 7

*Release* : numéro de version d'une application  
(indépendant de la version de la librairie)

# GNU Coding Standards



- ▶ Définition des options devant être reconnues  
(e.g., `--help`)
- ▶ Établissement de bonnes pratiques de programmation pour assurer la portabilité/maintenabilité
- ▶ Formatage des messages d'erreurs
- ▶ Standard pour les interfaces utilisateurs
- ▶ Formatage du code
- ▶ Internationalisation
- ▶ Documentation avec *Texinfo*
- ▶ ...



## Makefile : cibles standards

**make all** Compilation des programmes, bibliothèques,  
documentations, ... (cible par défaut)

**make install** Installation de la distribution en place

**make install-strip** Installation de la distribution avec élimination des  
symboles de débogage

**make uninstall** Désinstallation

**make clean** Effacement de tous les fichiers compilés par `make all`

**make distclean** `make clean` + effacement de tous les fichiers créés  
lors de la configuration

**make check** Exécution de la suite de tests

...



## Utilisation de gettext()

```
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```



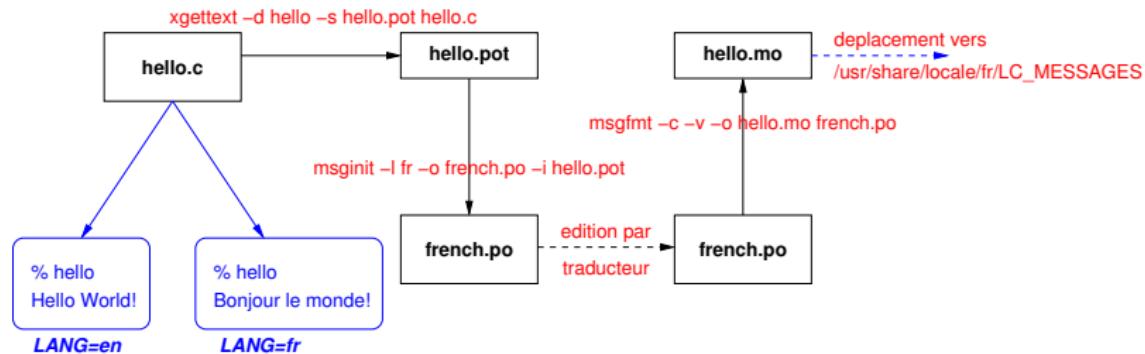
## Utilisation de gettext()

```
#include <stdio.h>
#include <libintl.h>
#include <locale.h>
#include <stdlib.h>

int main(void)
{
    setlocale( LC_ALL, "" );
    bindtextdomain( "hello", "/usr/share/locale" );
    textdomain( "hello" );
    printf(gettext("Hello, world!\n"));
    return 0;
}
```



# Internationalisation (2/3)





# Internationalisation (3/3)

## french.po

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-01-04 21:38+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: hello.c:13
#, c-format
msgid "Hello, world!\n"
msgstr ""
```



# Internationalisation (3/3)

## french.po

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-01-04 21:38+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: hello.c:13
#, c-format
msgid "Hello, world!\n"
msgstr "Bonjour le monde\n"
```

- ▶ Bibliothèque de *code source* de routines
- ▶ Copie des modules de GNULib utilisés directement dans l'arborescence source d'une application (`gnulib-tool`)
- ▶ But : portabilité et réutilisabilité (factorisation de code souvent utilisé)
- ▶ Exemples de modules :
  - ▶ Allocation mémoire `alloca`
  - ▶ Expressions régulières
  - ▶ Listes chaînées

# Gestion de version



- ▶ Transparents et documents utiles sur *madoc*
- ▶ La page Subversion (svn) :  
<http://subversion.tigris.org/>
- ▶ Version Control with Subversion :  
<http://svnbook.red-bean.com/>



- ▶ Gestion centralisée des fichiers d'une application :
  - ▶ Fichiers sources
  - ▶ Fichiers de tests
  - ▶ Documentation
  - ▶ ...
- ▷ Gestion des différentes « versions » des fichiers
- ▷ Gestion des différentes « branches » de développement
- ▷ Gestion du travail en collaboration



# Pourquoi utiliser la gestion de version

- ▶ Retour en arrière : *une modification apportée au code est mauvaise. Comment revenir à une version précédente sans le bug ?*
- ▶ Traçage des modifications : *Qui a modifié quoi ? Quand ? Pourquoi ?*
- ▶ Branches : *Comment gérer des versions très différentes d'une application ? Comment permettre à un programmeur de modifier massivement du code sans impact sur le travail des autres ? Comment fusionner deux versions différentes ?*
- ▶ Tags : *Comment retrouver l'état du code lors de la dernière release publique alors qu'il a beaucoup changé depuis ?*



# Les « concurrents » de Subversion

**SCCS/GNU CSSC.** *Source Code Control System/Compatibly Stupid Source Control.*  
Précurseur. Contrôle par verrouillage des fichiers.

**RCS/GNU RCS.** *Revision Control System.* Contrôle par verrouillage des fichiers.  
(<http://www.gnu.org/software/rcs/rcs.html>)

**CVS.** *Concurrent Version System* (<http://ximbiot.com/cvs/wiki/>)

**Perforce.** Application commerciale de gestion de code. Rapide (utilisation d'une BD) (<http://www.perforce.com>)

**Microsoft VSS.** Application commerciale de gestion de code. Rapide (utilisation d'une BD)

**Bitkeeper.** Système propriétaire distribué de gestion de version  
(<http://www.bitkeeper.com/>)

**Git.** Système distribué de gestion de version (<http://git-scm.com/>).  
Ponts avec Subversion.

**Mercurial.** Système distribué de gestion de version  
(<http://mercurial.selenic.com/>)

**Bazaar.** Système distribué de gestion de version  
(<http://bazaar.canonical.com/en/>)



# Pourquoi Subversion ?

- ▶ Gère correctement les fichiers binaires (cf. CVS)
- ▶ Gestion simple des branches
- ▶ Puissant : *propriétés, tags, changesets, ...*
- ▶ Indépendant du protocole d'accès au dépôt (local, http, svnserv, ...)

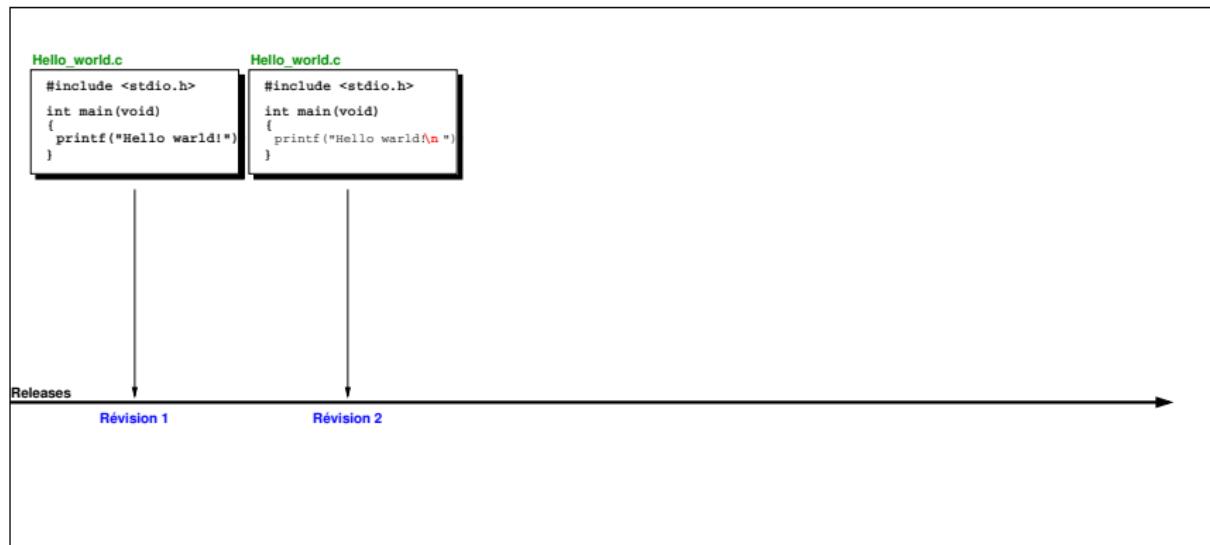


# Scenario 1 : gestion des « versions »



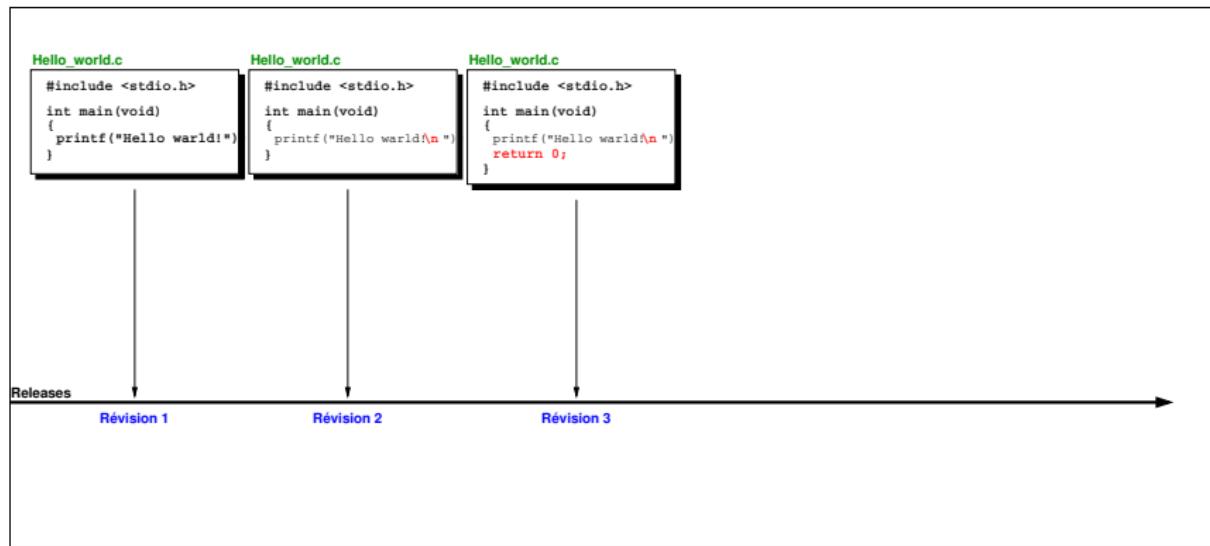


# Scenario 1 : gestion des « versions »



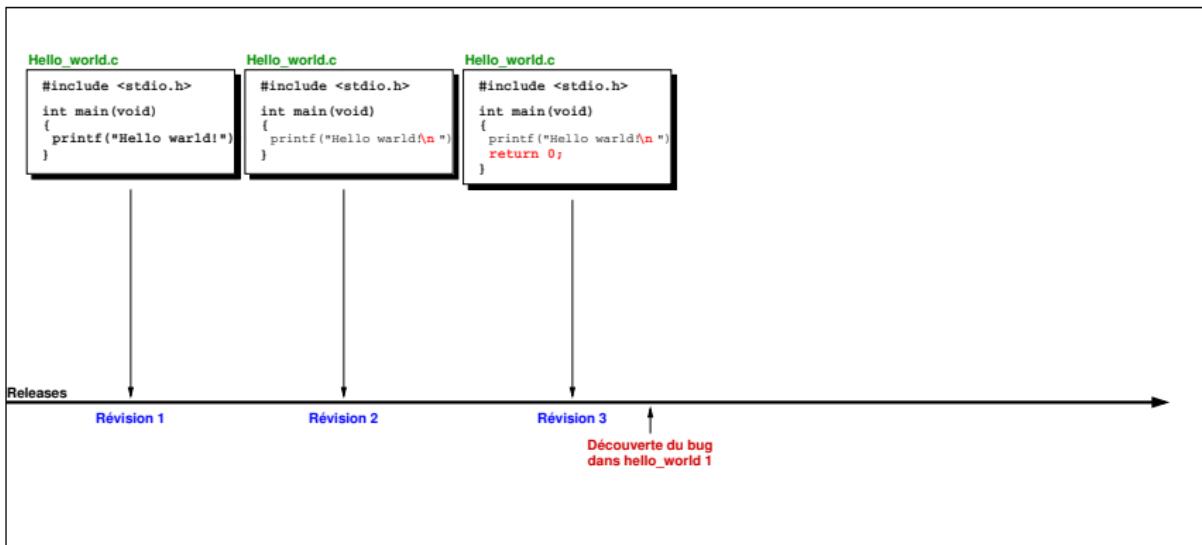


# Scenario 1 : gestion des « versions »



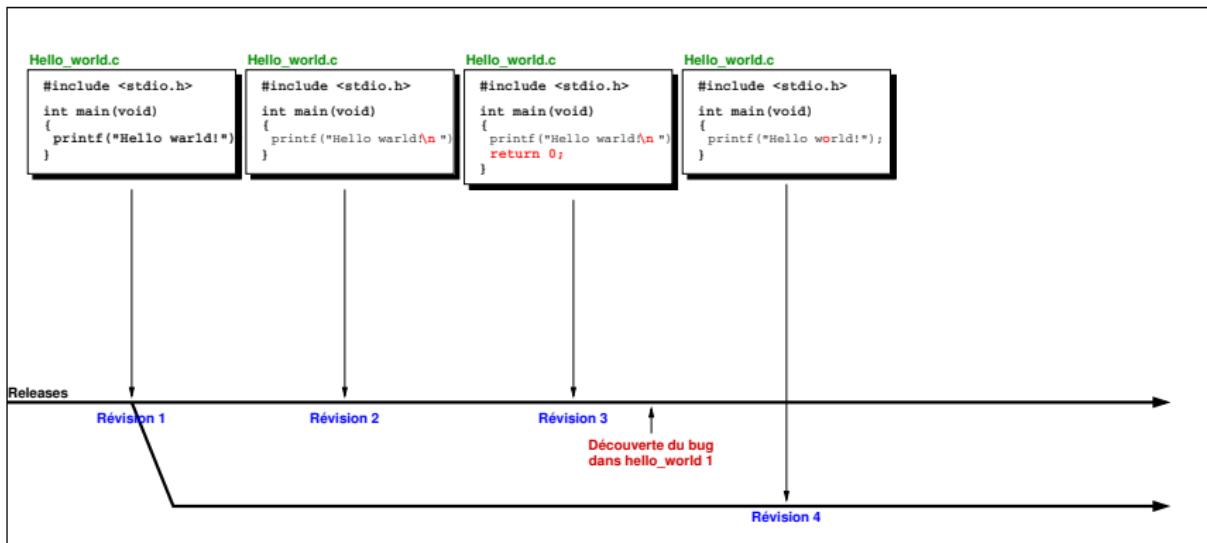


# Scenario 1 : gestion des « versions »



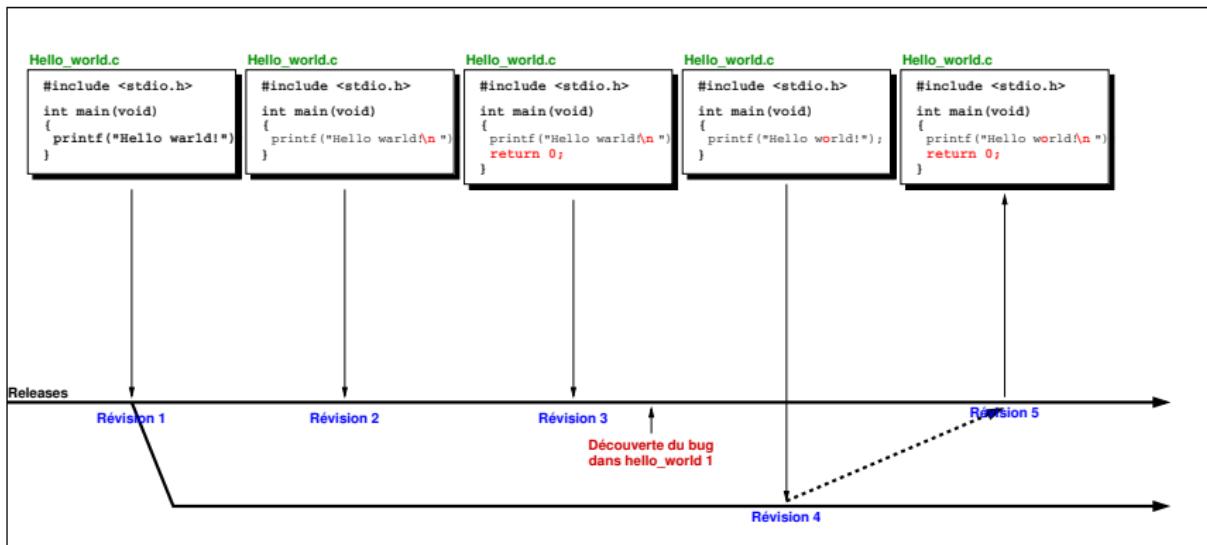


# Scenario 1 : gestion des « versions »





# Scenario 1 : gestion des « versions »





# Version vs. révision

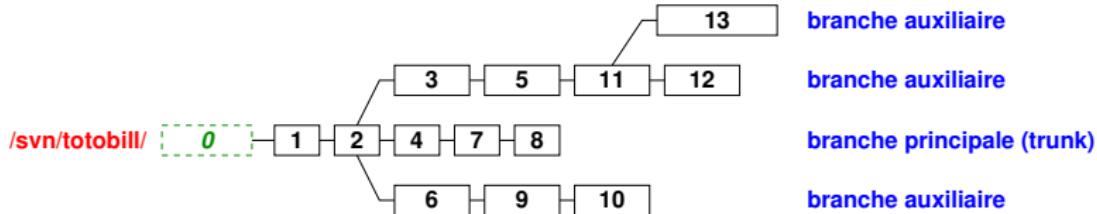
**Version.** Indice attribué à une application/librairie lors de sa *release*



**Majeur.** Change lorsqu'une *release* présente des incompatibilités avec la précédente

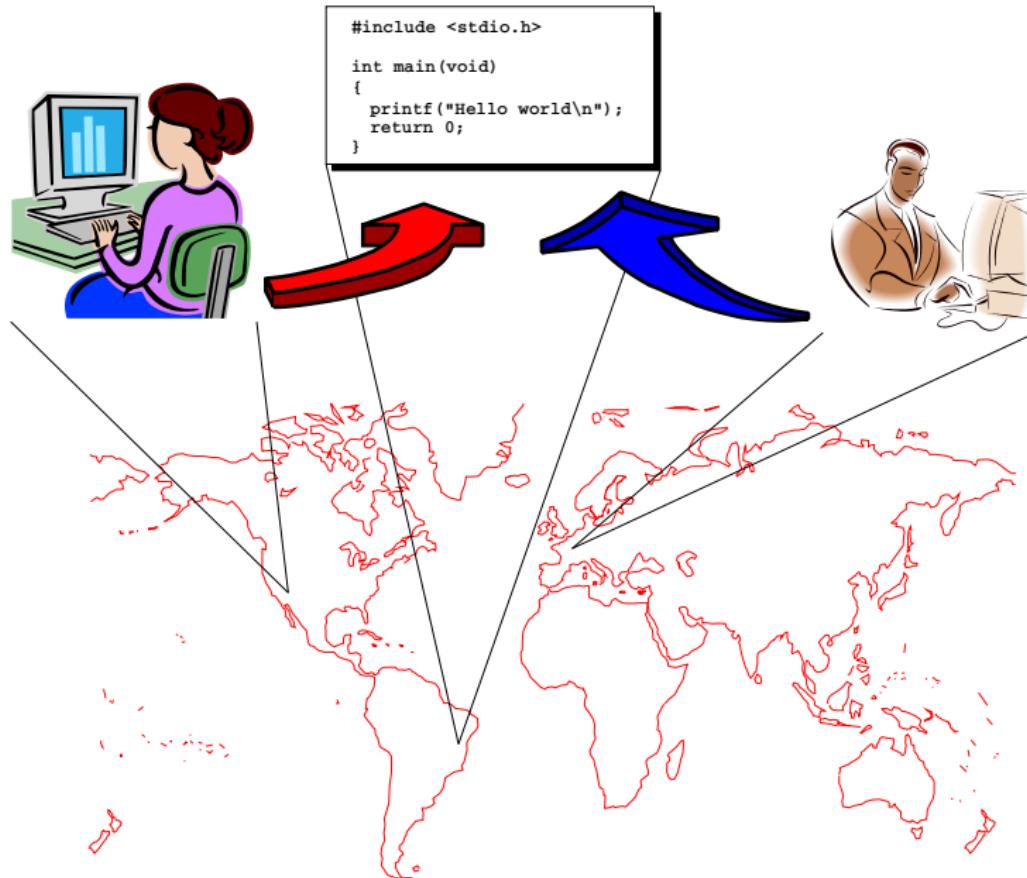
**Mineur.** Change lorsque les modifications apportées n'affectent pas la compatibilité

**Révision.** Indice attribué à l'arborescence d'une application dans son ensemble lors d'une modification prise en compte par svn





## Scenario 2 : le travail collaboratif



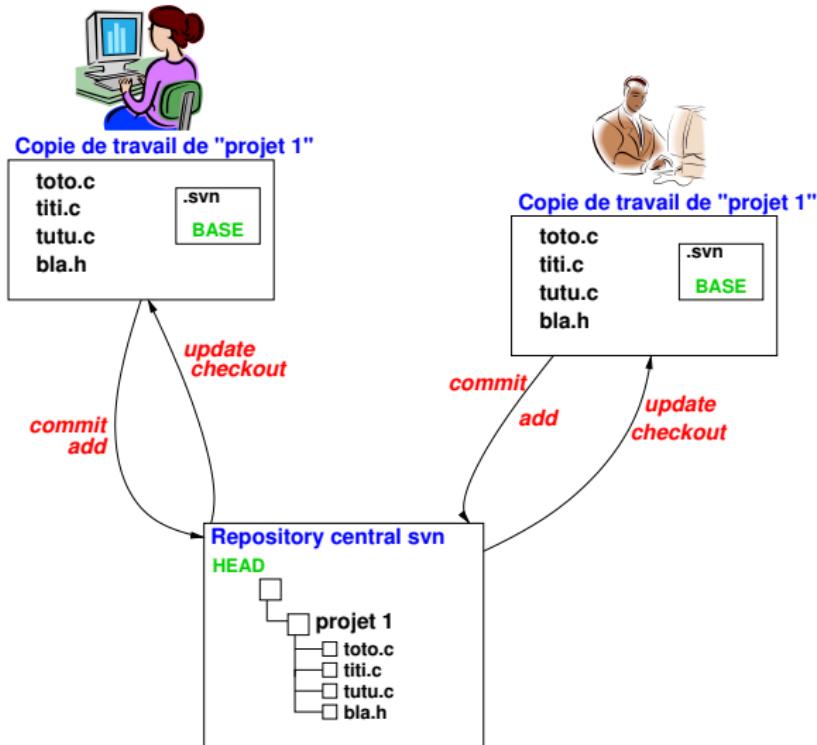


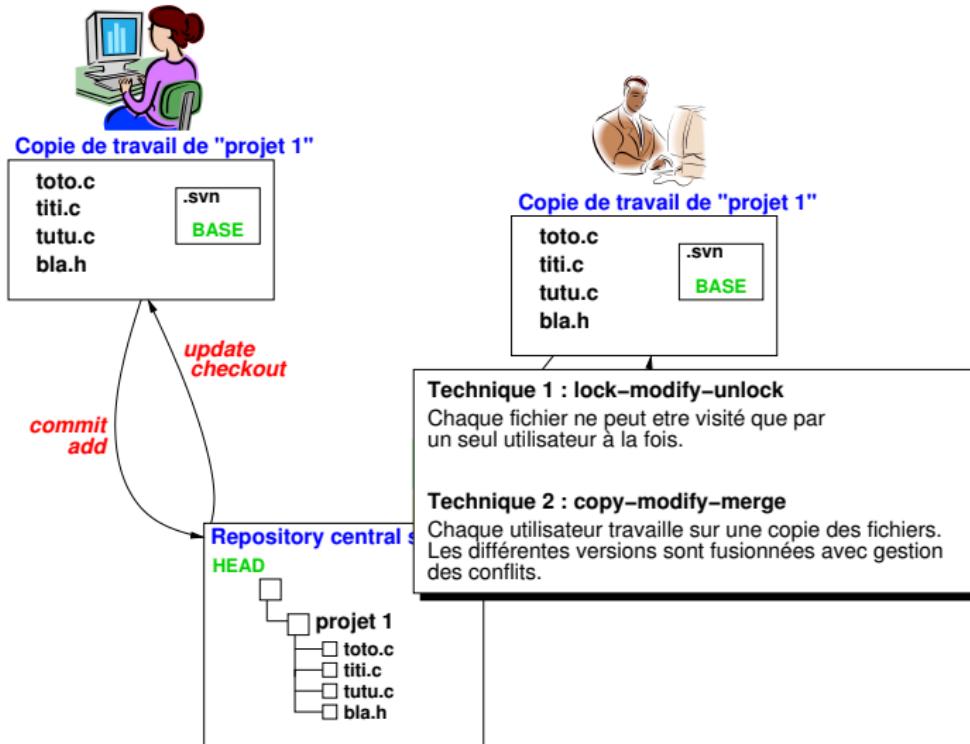
## Scenario 2 : le travail collaboratif





# Travail collaboratif







**Dépôt.** Répertoire sur le serveur contenant la base de donnée de gestion de version d'un projet

**Révision.** Numéro associé à un état d'un projet

**Copie de travail.** Copie du projet sur l'ordinateur client

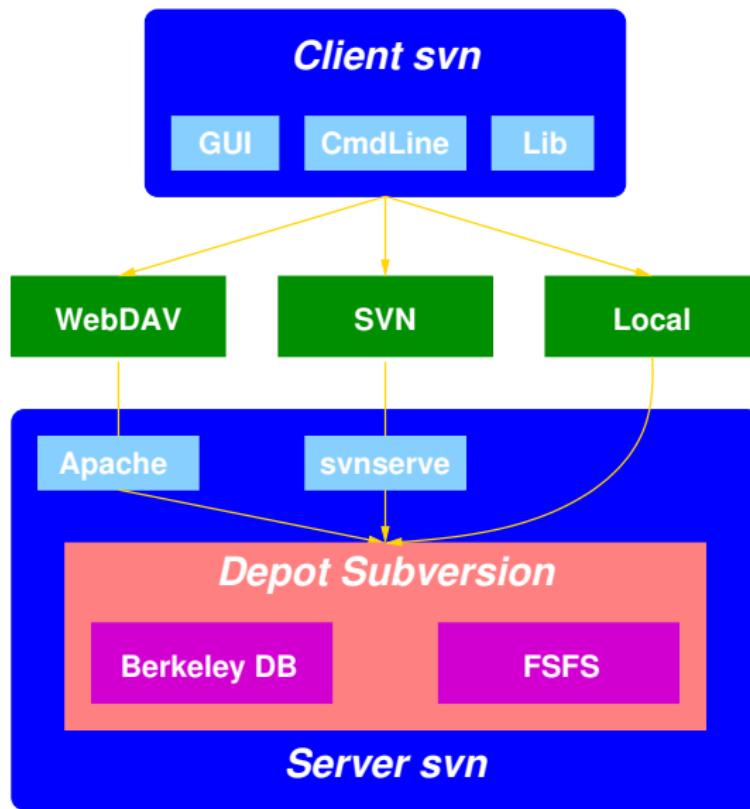
**HEAD.** Révision la plus récente dans le dépôt

**BASE.** Révision dans le répertoire .svn sur l'ordinateur client

**COMMITTED.** Révision la plus récente (inférieure ou égale à BASE) à laquelle un fichier a été modifié



# Architecture de Subversion





# Aide sur les commandes

```
% svn help diff
diff (di): Display the differences between two revisions or paths.
usage: 1. diff [-c M | -r N[:M]] [TARGET[@REV]...]
        2. diff [-r N[:M]] --old=OLD-TGT[@OLDREV] [--new=NEW-TGT[@NEWREV]]
                  [PATH...]
        3. diff OLD-URL[@OLDREV] NEW-URL[@NEWREV]

1. Display the changes made to TARGETs as they are seen in REV between
two revisions. TARGETs may be all working copy paths or all URLs.
If TARGETs are working copy paths, N defaults to BASE and M to the
working copy; if URLs, N must be specified and M defaults to HEAD.
The '-c M' option is equivalent to '-r N:M' where N = M-1.
Using '-c -M' does the reverse: '-r M:N' where N = M-1.
[...]
```



Mise sous Subversion de l'application *hello* :

```
% tree hello
hello
|--- Makefile
|--- README
|--- doc
|   |--- Makefile
|   '-- hello.texi
`--- hello.c
```



## Étape 0 : préparation de l'import

On ajoute les répertoires trunk, tags, branches (optionnel mais recommandé)

```
% mkdir hello/trunk && mv hello/* hello/trunk
mv: cannot move 'hello/trunk' to a subdirectory of itself,
  'hello/trunk/trunk'
% mkdir hello/tags hello/branches && tree hello
hello
|--- branches
|--- tags
`--- trunk
    |--- Makefile
    |--- README
    |--- doc
        |--- Makefile
        |--- hello.texi
    '-- hello.c

4 directories, 5 files
```



## Étape 1 : création du dépôt

- ▶ Création sur le serveur d'un répertoire contenant tous les dépôts svn futurs (optionnel)

```
% mkdir /home/goualard svnroot
```

- ▶ Création d'un dépôt svn pour hello sur le serveur

```
% svnadmin create /home/goualard svnroot hello
```

Position du dépôt : chemin d'accès, *pas* URI

- ▶ Édition des droits d'accès au dépôt  
(dépendant de la méthode d'accès : WebDAV, svnserve, ...)



## Étape 2 : import dans le dépôt

### ► Import de hello/ du client dans le dépôt svn

```
% svn import hello svn://almighty/hello -m "Import dans svn"
Adding          hello/trunk
Adding          hello/trunk/hello.c
Adding          hello/trunk/doc
Adding          hello/trunk/doc/hello.texi
Adding          hello/trunk/doc/Makefile
Adding          hello/trunk/Makefile
Adding          hello/trunk/README
Adding          hello/branches
Adding          hello/tags

Committed revision 1.
```

URIs possibles :

- file:///, http:///, https:///, svn:///, svn+ssh://



## Étape 3 : Mise en place (1/2)

- ▶ Récupération d'une version de hello gérée par svn :

```
% rm -fr hello
% svn checkout svn://almighty/hello/trunk hello
A    hello/hello.c
A    hello/doc
A    hello/doc/hello.texi
A    hello/doc/Makefile
A    hello/Makefile
A    hello/README
Checked out revision 1.
```



## Étape 3 : Mise en place (1/2)

```
% tree -a hello
hello
|-- .svn
|   |-- entries
|   |-- format
|   |-- prop-base
|   |-- props
|   |-- text-base
|       |-- Makefile.svn-base
|       |-- README.svn-base
|       '-- hello.c.svn-base
`-- tmp
    |-- prop-base
    |-- props
    '-- text-base
|-- Makefile
`-- README
```

```
|--- doc
|   |-- .svn
|   |   |-- entries
|   |   |-- format
|   |   |-- prop-base
|   |   |-- props
|   |   |-- text-base
|   |       |-- Makefile.svn-base
|   |       '-- hello.texi.svn-base
|   '-- tmp
|       |-- prop-base
|       |-- props
|       '-- text-base
|           |-- Makefile
|           '-- hello.texi
`-- hello.c

17 directories, 14 files
```



- ▶ Un répertoire .svn dans chaque sous-répertoire contrôlé par svn
- ▶ Pour chaque fichier  $f$  :
  - ▶ Révision de  $f$  dans la copie de travail
  - ▶ Date de la dernière mise à jour à partir du dépôt

⇒ Quatre états possibles :

E1	<i>Non modifié localement et courant</i>	Identique à sa copie dans le dépôt
E2	<i>Modifié localement et courant</i>	Révision identique à sa copie dans le dépôt mais modifié localement
E3	<i>Non modifié mais non-sync</i>	Pas modifié localement mais sa révision est inférieure à celle du dépôt
E4	<i>Modifié localement et non-sync</i>	Modifié localement et sa révision est inférieure à celle du dépôt



# update vs. commit

Contrairement à CVS :

- ▶ update n'implique pas commit
  - ▶ commit n'implique pas update
- Les fichiers d'une copie locale n'ont pas tous la même révision

Limites :

- ▶ Impossible de détruire un répertoire dans un état  $\neq$  E3 ou E4
- ▶ Impossible de modifier les méta-données d'un répertoire dans un état  $\neq$  E3 ou E4

Destruction accidentelle de .svn ?

- ▶ Détruire le répertoire correspondant + update



# Cycle des commandes

1. Mise à jour de la copie de travail

```
svn update
```

2. Modifications

```
svn add / svn delete / svn copy / svn move
```

3. Examen des modifications

```
svn status / svn diff
```

4. Retour arrière (*undo*)

```
svn revert
```

5. Résolution de conflits

```
svn update / svn resolve
```

6. Prise en compte des changements

```
svn commit
```



# Mise à jour de la copie de travail

Avant toute reprise de travail sur la copie locale :

```
% svn update  
U      hello.c  
Updated to revision 3.
```

Informations :

Colonne	1 Fichier	2 Propriété du fichier	3 : "B" Vol/destruction de verrou
---------	--------------	------------------------------	---

Lettres possibles :

A	Ajout	(Added)
D	Destruction	(Deleted)
U	Mise à jour	(Updated)
C	Conflit	(Conflict)
M	Fusion	(Merged)



# Modifications (1/3)

- ▶ Modification de fichiers (déttection automatique)
- ▶ Modification de l'arbre des fichiers (ajout/destruction de fichiers/répertoires)

```
% ls
doc/ hello.c Makefile README
% mkdir check; touch check/Makefile; svn add check
A      check
A      check/Makefile
% svn mkdir examples
A      examples
% ls
check/ doc/ examples/ hello.c Makefile README
% svn remove doc
D      doc/hello.texi
D      doc/Makefile
D      doc
```



# Modifications (2/3)

Prise en compte des modifications seulement après un commit :

```
% ls
check/ doc/ examples/ hello.c Makefile README
% svn commit -m "Destruction de doc/ car la doc ne sert à rien"
Adding check
Adding check/Makefile
Deleting doc
Adding examples
Transmitting file data .
Committed revision 4.
% ls
check/ examples/ hello.c Makefile README
% svn copy README README.TXT
A README.TXT
% svn commit -m "copie de README plutôt que lien (MS windows)"
Adding README.TXT

Committed revision 6.
```



## Modifications (3/3)

Possibilité de travailler directement sur le dépôt sans copie locale (commit automatique !) :

```
% svn move svn://almighty/hello/trunk/examples \
>           svn://almighty/hello/trunk/example \
>           -m "Normalisation des noms de répertoire"

Committed revision 5.
% svn update
D    examples
A    example
Updated to revision 5.
```



# Examen des modifications

- ▶ Examen des changements à la copie locale avant de faire un commit
- ▶ Comparaison avec version dans .svn ➡ Pas d'accès au dépôt

```
% emacs hello.c
% svn status
M      hello.c
% svn diff
Index: hello.c
=====
--- hello.c      (revision 5)
+++ hello.c      (working copy)
@@ -2,6 +2,6 @@
 
     int main(void)
    {
-    printf("Hello Warld\n");
+    printf("Hello World\n");
        return 0;
    }
```



- ▶ Six colonnes d'un caractère :
  - ▶ Colonne 1 : modification sur fichier/répertoire  
Exemple : A (*Ajout*), ? (inconnu de svn), ! (non présent), ...
  - ▶ Colonne 2 : modification sur propriétés de fichier/répertoire
  - ▶ Colonne 3 : copie locale verrouillée
  - ▶ ...



- ▶ Affiche les différences (*unified diff format*)
- ▶ Sortie de `svn diff` peut servir de *patch*

```
% svn diff > patchfile.txt
```

- ▶ Utilisation de `patchfile.txt` avec la commande `patch`

## Unified diff format :

- ▶ Change hunks (chunks) : en-tête + lignes ajoutées/supprimées + contexte
- ▶ En-tête : @@ -R, +R @@ avec R de la forme d,l  
(d : ligne de début ; l : nombre de lignes affectées)
- ▶ Lignes retirées précédées de -
- ▶ Lignes ajoutées précédées de +
- ▶ Lignes contextuelles précédées d'un espace



## Retour arrière (*undo*)

- ▶ Retour à l'état dans .svn d'un fichier modifié
- ▶ Abandon des modifications prévues

```
% svn delete example
D          example
% svn revert example hello.c
Reverted 'example'
Reverted 'hello.c'
```



# Résolution de conflits (1/5)

```
% svn update
At revision 6.
% cat hello.c
#include <stdio.h>
int main(void)
{
    printf("Hello Warld\n");
    return 0;
}
% emacs hello.c
[On réécrit "Hello World"]
[Tartempion ajoute un point d'exclamation après "Warld"]
% svn update
C    hello.c
% ls
check/    hello.c      hello.c.r6  Makefile  README.TXT
example/  hello.c.mine  hello.c.r8  README
```



## Résolution de conflits (2/5)

- ▶ *file.mine* : fichier original de la copie locale
- ▶ *file.r<sup>OLDREV</sup>* : fichier de BASE
- ▶ *file.r<sup>NEWREV</sup>* : fichier de HEAD
- ▶ *file* : fichier avec marquage des conflits

```
% cat hello.c
#include <stdio.h>
int main(void)
{
<<<<< .mine
    printf("Hello World\n");
=====
    printf("Hello Warld!\n");
>>>>> .r8
    return 0;
}
```



## Résolution des conflits (3/5)

- ▶ Pas de commit possible si conflit

```
% svn commit
svn: Commit failed (details follow):
svn: Aborting commit: '/home/goualard/temp/tmp/hello/hello.c'
      remains in conflict
```



# Résolution des conflits (4/5)

Subversion V. 1.5 :

- ▶ Résolution du conflit :
  - ▶ Choix de la version de BASE :

```
% svn resolve --accept base hello.c
```
  - ▶ Choix de la version de travail  

```
% svn resolve --accept mine-full hello.c
```
  - ▶ Choix de la version du dépôt :  

```
% svn resolve --accept theirs-full hello.c
```
  - ▶ Édition de *file* pour sélection des changements et élimination des marqueurs :  

```
% svn resolve --accept working hello.c
```



L'élimination des conflits se fait par le dialogue entre programmeurs



## Résolution des conflits (5/5)

```
% emacs hello.c
% cat hello.c
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
% svn resolve --accept working hello.c
Sending      hello.c
Transmitting file data .
Committed revision 9.
```

Si version de Subversion antérieure à 1.5 : effacement manuel des fichiers, puis `svn commit`



## Qu'est-ce qu'un conflit ?

- ▶ Pour des fichiers textes :
  - ▶ Modifications à des positions “proches”
- ▶ Pour des fichiers binaires :
  - ▶ Fichiers différents

Pas de notion *sémantique* des modifications

- ➡ Certains conflits peuvent passer inaperçus
- ➡ Faux positifs possibles



# Prise en compte des changements

```
[Modification de hello.c]
% emacs hello.c
% svn commit -m "Préparation pour future extension"
Sending      hello.c
Transmitting file data .
Committed revision 10.
```

- ▶ Composition d'un *log* pour chaque commit
  - ▶ En ligne : paramètre “*-m*”
  - ▶ En prenant le contenu d'un fichier : “*svn commit -F fic*”
  - ▶ avec un éditeur de texte (variable SVNEDITOR ou EDITOR)



## Historique des modifications (1/5)

svn log : visualisation des messages donnés lors des *commits*

```
% svn log -r HEAD:8 hello.c
-----
r10 | (no author) | 2009-01-07 22:46:48 +0100 (Wed, 07 Jan 2009) | 1 line
Préparation pour future extension
-----
r9 | (no author) | 2009-01-07 22:37:02 +0100 (Wed, 07 Jan 2009) | 1 line
-----
r8 | (no author) | 2009-01-07 22:05:08 +0100 (Wed, 07 Jan 2009) | 1 line
Un peu de dynamisme dans le message
```



# Historique des modifications (2/5)

svn diff :

- ▶ Étude de changements locaux
- ▶ Comparaison de la copie de travail au dépôt
- ▶ Comparaison de révisions

```
% svn diff hello.c
[hello.c est sync avec BASE]
% svn diff -r 9 hello.c
Index: hello.c
=====
--- hello.c      (revision 9)
+++ hello.c      (working copy)
@@ -1,7 +1,6 @@
 #include <stdio.h>
-int main(void)
+int main(int argc, char *argv[])
{
    printf("Hello World!\n");
-
    return 0;
}
```



## Historique des modifications (3/5)

```
% svn diff -r 3:7 hello.c
Index: hello.c
=====
--- hello.c      (revision 3)
+++ hello.c      (revision 7)
@@ -2,6 +2,6 @@
 
     int main(void)
    {
-    printf("Hello Warld\n");
+    printf("Hello Warld!\n");
        return 0;
    }
```



## Historique des modifications (4/5)

svn cat : affichage d'une révision d'un fichier

```
% svn cat -r 5 hello.c
#include <stdio.h>

int main(void)
{
    printf("Hello Warld\n");
    return 0;
}
```



## Historique des modifications (5/5)

svn list : affichage du contenu d'un répertoire dans le dépôt

```
% svn list
Makefile
README
README.TXT
check/
example/
hello.c
% svn list svn://almighty/hello/trunk/doc
svn: URL 'svn://almighty/hello/trunk/doc' non-existent in that revision
% svn list svn://almighty/hello/trunk/doc@1
Makefile
hello.texi
```



## Récupération d'une révision particulière d'un projet :

```
% svn checkout -r 2 svn://almighty/hello/trunk hello-rev2
A    hello-rev2/hello.c
A    hello-rev2/doc
A    hello-rev2/doc/hello.texi
A    hello-rev2/doc/Makefile
A    hello-rev2/Makefile
A    hello-rev2/README
Checked out revision 2.
```



## Export de la copie locale sans .svn :

```
% svn export -r 4 svn://almighty/hello/trunk hello-release-rev4
A    hello-release-rev4
A    hello-release-rev4/hello.c
A    hello-release-rev4/Makefile
A    hello-release-rev4/README
A    hello-release-rev4/check
A    hello-release-rev4/check/Makefile
A    hello-release-rev4/examples
Exported revision 4.
% ls
hello/  hello-release-rev4/
```



- ▶ Noms symboliques
  - ▶ HEAD, BASE
- ▶ Par date (ISO 8601)

```
% svn checkout -r {"2009-12-07 22:30"} \
>           svn://almighty/hello/trunk old-hello
A   old-hello/hello.c
A   old-hello/README.TXT
A   old-hello/example
A   old-hello/Makefile
A   old-hello/README
A   old-hello/check
A   old-hello/check/Makefile
Checked out revision 10.
```

Retourne la révision la plus récente à la date donnée



## Les propriétés (1/2)

- ▶ Métdonnées attachées aux :
  - ▶ Fichiers (*versioning*)
  - ▶ Répertoires (*versioning*)
  - ▶ Révisions (pas de *versioning*)
- ▶ Propriété = couple (nom/valeur)
- ▶ Propriétés « svn:\* » réservées par svn

```
% svn propset licence -F ../lgpl-3.0.txt hello.c
property 'licence' set on 'hello.c'
% svn propget licence hello.c
GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
[...]
```



## Les propriétés (2/2)

```
% svn proplist hello.c
Properties on 'hello.c':
  licence
% svn propdel licence hello.c
property 'licence' deleted from 'hello.c'.
```

- ▶ Propriétés utilisées par Subversion pour stocker *log* et caractéristiques :

```
% svn propget svn:log -rHEAD --revprop
Préparation pour future extension (dans le repository)
% svn add lengl-3.0.pdf
A (bin) lengl-3.0.pdf
% svn proplist lengl-3.0.pdf
Properties on 'lengl-3.0.pdf':
  svn:mime-type
% svn propget svn:mime-type lengl-3.0.pdf
application/octet-stream
```



# Substitution de mots-clés

Mots-clés remplacés automatiquement par Subversion dans les fichiers :

```
% emacs hello.c
% cat hello.c
/* $Date$ */
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World!\n");
    return 0;
}
% svn propset svn:keywords "Date" hello.c
property 'svn:keywords' set on 'hello.c'
% svn commit -m "Ajout d'info de versioning"
Sending      hello.c
Transmitting file data .
Committed revision 11.
% cat hello.c
/* $Date: 2009-01-08 00:05:14 +0100 (Thu, 08 Jan 2009) $ */
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World!\n");
    return 0;
}
```



# Substitution de mots-clés

Mots-clés remplacés automatiquement par Subversion dans les fichiers :

```
% emacs hello.c
% cat hello.c
/* $Date$ */
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World!\n");
    return 0;
}
% svn propset svn:keywords "Date" hello.c
property 'svn:keywords' set on 'hello.c'
% svn commit -m "Ajout d'info de versioning"
Sending      hello.c
Transmitting file data .
Committed revision 11.
% cat hello.c
/* $Date: 2009-01-08 00:05:14 +0100 (Thu, 08 Jan 2009) $ */
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

## Mots-clés:

Date	Author
Revision	HeadURL
Id	



Deux modèles :

- ▶ *lock-modify-unlock* (LMU)
- ▶ *copy-modify-merge* (CMM)
- ▶ CMM plus efficace que LMU (meilleure parallélisation des tâches)
- ▶ MAIS : *Merge* pour des fichiers binaires ?
- ➡ Subversion offre la possibilité du modèle LMU



## Lock-Modify-Unlock

```
% svn add logo-lina.jpg
A (bin)  logo-lina.jpg
% svn lock logo-lina.jpg -m "Changement de la couleur de fond"
'logo-lina.jpg' locked by user 'goualard'.
% svn status
K logo-lina.jpg
% svn unlock logo-lina.jpg
'logo-lina.jpg' unlocked
```

- ▶ Déverrouillage automatique en cas de *commit* sur le fichier par le propriétaire du verrou
- ▶ Verrous pas inviolables (peuvent être détruits ou volés)
  - ➡ servent de rappel seulement
- ▶ Attacher `svn:needs-lock` aux fichiers non fusionnables



# Les *changelists* [SVN 1.5] (1/2)

- ▶ Changelists = moyen d'organiser des *paniers*

```
% svn copy hello.c hallo.c
A          hallo.c
% svn copy hello.c bonjour.c
A          bonjour.c
% svn copy hello.c hola.c
A          hola.c
% svn commit
Adding      bonjour.c
Adding      hallo.c
Adding      hola.c
% svn changelist langues-romanes bonjour.c hola.c
Path 'bonjour.c' is now member of changelist 'langues-romanes'
Path 'hola.c' is now member of changelist 'langues-romanes'
% svn status

--- Changelist 'langues-romanes':
M          bonjour.c
M          hola.c
```



## Les *changelists* [SVN 1.5] (2/2)

- ▶ Possibilité de faire référence à une *changelist* dans les opérations

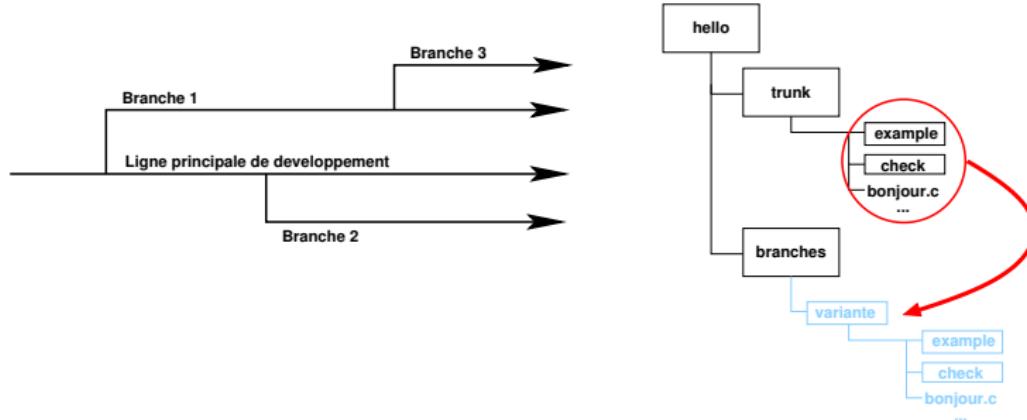
```
% svn diff --changelist langues-romanes
Index: bonjour.c
=====
--- bonjour.c      (revision 13)
+++ bonjour.c      (working copy)
[...]
Index: hola.c
=====
--- hola.c        (revision 15)
+++ hola.c        (working copy)
[...]
```

- ▶ Changelists associés à une copie de travail (pas propagables au dépôt)



# Les branches (1/3)

- ▶ Maintenance de spécialisations d'un projet
- ▶ Branchement si les modifications souhaitées vont bouleverser le code pour garder un trunk opérationnel
- ▶ ...



```
% svn copy \
>     svn://almighty/hello/trunk svn://almighty/hello/branches/variante \
>             -m "Creation d'une variante de hello"
Committed revision 17.
```



## Les branches (2/3)

- ▶ Branche = copie (peu coûteuse en temps/espace)
- ▶ Utilisation de la branche par checkout :

```
% svn checkout svn://almighty/hello/branches/variante new-hello
A    new-hello/bonjour.c
A    new-hello/hallo.c
A    new-hello/logo-lina.jpg
A    new-hello/hello.c
A    new-hello/hola.c
A    new-hello/hej.c
A    new-hello/README.TXT
A    new-hello/example
A    new-hello/Makefile
A    new-hello/README
A    new-hello/check
A    new-hello/check/Makefile
Checked out revision 17.
```



## Les branches (3/3)

Utilisation de la branche par versement de la copie locale :

```
% svn info
Path: .
URL: svn://almighty/hello/trunk
Repository Root: svn://almighty/hello
Repository UUID: 26c53331-ef0e-4609-9dc7-f796365a2311
Revision: 12
Node Kind: directory
Schedule: normal
Last Changed Rev: 12
Last Changed Date: 2009-01-08 00:21:21 +0100 (Thu, 08 Jan 2009)
% svn switch svn://almighty/hello/branches/variante
At revision 17.
% svn info
Path: .
URL: svn://almighty/hello/branches/variante
Repository Root: svn://almighty/hello
Repository UUID: 26c53331-ef0e-4609-9dc7-f796365a2311
Revision: 17
Node Kind: directory
Schedule: normal
Last Changed Rev: 17
Last Changed Date: 2009-01-08 01:10:36 +0100 (Thu, 08 Jan 2009)
```



# Fusion de branches

```
% svn cat svn://almighty/hello/trunk/hello.c
/* $Date: 2009-01-08 01:24:17 +0100 (Thu, 08 Jan 2009) $ */
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
    printf("Hello World from process %i!\n",getpid());
    return 0;
}
% svn cat svn://almighty/hello/branches/variante/hello.c
/* $Date: 2009-01-08 01:24:21 +0100 (Thu, 08 Jan 2009) $ */
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World! Great Day!\n");
    return 0;
}
% svn merge svn://almighty/hello/branches/variante \
>     svn://almighty/hello/trunk
U     hello.c
```

- ▶ On peut continuer à travailler sur un branche même après fusion.
- ▶ Destruction d'une branche :

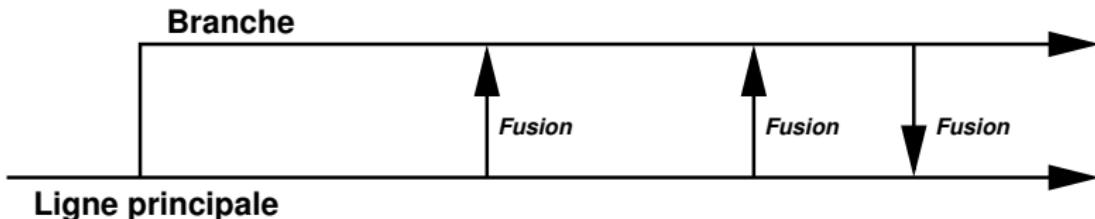
```
% svn delete svn://almighty/hello/branches/variante \
>     -m "Destruction de 'variante'"
```

```
Committed revision 20.
```



# Fusion de branches

```
% svn cat svn://almighty/hello/trunk/hello.c
/* $Date: 2009-01-08 01:24:17 +0100 (Thu, 08 Jan 2009) $ */
#include <stdio.h>
#include <sys/types.h>
```



```
return 0;
}
% svn merge svn://almighty/hello/branches/variante \
>     svn://almighty/hello/trunk
U     hello.c
```

- ▶ On peut continuer à travailler sur un branche même après fusion.
- ▶ Destruction d'une branche :

```
% svn delete svn://almighty/hello/branches/variante \
>     -m "Destruction de 'variante'"
```

```
Committed revision 20.
```



Récupération de fichiers/répertoires n'existant plus dans la copie de travail :

```
% ls
bonjour.c  example  hej.c    hola.c      Makefile  README.TXT
check      hallo.c  hello.c   logo-lina.jpg README
% svn log
[...]
r4 | (no author) | 2009-01-07 21:11:12 +0100 (Wed, 07 Jan 2009) | 1 line
Destruction de doc/ car la doc ne sert à rien
[...]
% svn merge -c -4 svn://almighty/hello/trunk
D      check/Makefile
D      check
Skipped missing target: 'examples'
A      doc
A      doc/hello.texi
A      doc/Makefile
% svn commit -m "Retour de doc/"
Deleting      check
Adding       doc
Adding       doc/Makefile
Adding       doc/hello.texi

Committed revision 21.
```



- ▶ Tags = instantané de l'état du dépôt à un instant donné
- ▶ Utilité : release
- ▶ Identique à une branche (=copie du dépôt)
- ▶ Différence avec branche : convention d'utilisation
- ▶ Contrôle d'accès possible par script

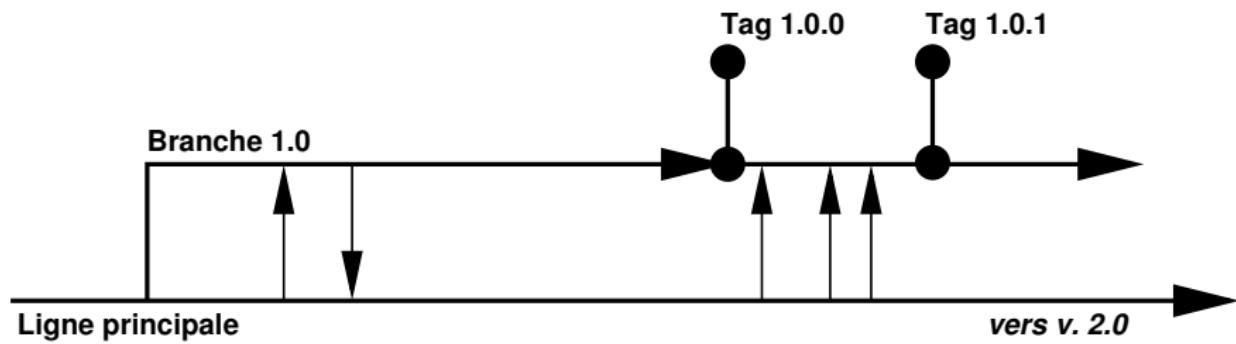
```
% svn copy svn://almighty/hello/trunk  
%           svn://almighty/hello/tags/release-1-0\  
%           -m "Première livraison du produit"  
  
Committed revision 22.
```

- ▶ Destruction de tag comme pour une branche :

```
% svn delete svn://almighty/hello/tags/release-1-0 \  
%           -m "Destruction du snapshot de la première livraison"  
  
Committed revision 23.
```



## Cycle de développement





```
% ls home/goualard svnroot/hello/
conf  dav  db   format  hooks  locks  README.txt
% ls ~/svnroot/hello/conf/
authz  passwd  svnserve.conf
% ls ~/svnroot/hello/hooks/
post-commit.tmpl      post-unlock.tmpl  pre-revprop-change.tmpl
post-lock.tmpl        pre-commit.tmpl   pre-unlock.tmpl
post-revprop-change.tmpl  pre-lock.tmpl  start-commit.tmpl
```

**conf** : Fichiers de configuration (y compris *svnserve*)

**db** : Base de données contenant les données sous svn

**hooks** : patrons de scripts à exécuter avant/après une opération



# Configuration du serveur SVN

- ▶ **svnserv** : à lancer sur la machine serveur

```
% svnserv -d -r /home/goualard svnroot
```

- ▶ Authentification avec conf/svnserv.conf
- ▶ Accès anonyme possible

- ▶ Apache+WebDAV :

- ▶ utilisation des modules mod\_dav et mod\_dav\_svn
- ▶ Modification du httpd.conf pour rendre le dépôt svn accessible
- ▶ Bonus : visualisation du contenu du dépôt en http
- ▶ Utilisation des facilités d'Apache pour l'authentification

Possibilité d'accéder à un dépôt par Apache et svnserv simultanément



**scripts.** Nombreux scripts (*hooks*) prédéfinis sur le site de subversion

**tortoiseSVN.** Interface SVN/Windows Explorer.  
(<http://tortoisesvn.tigris.org/>)

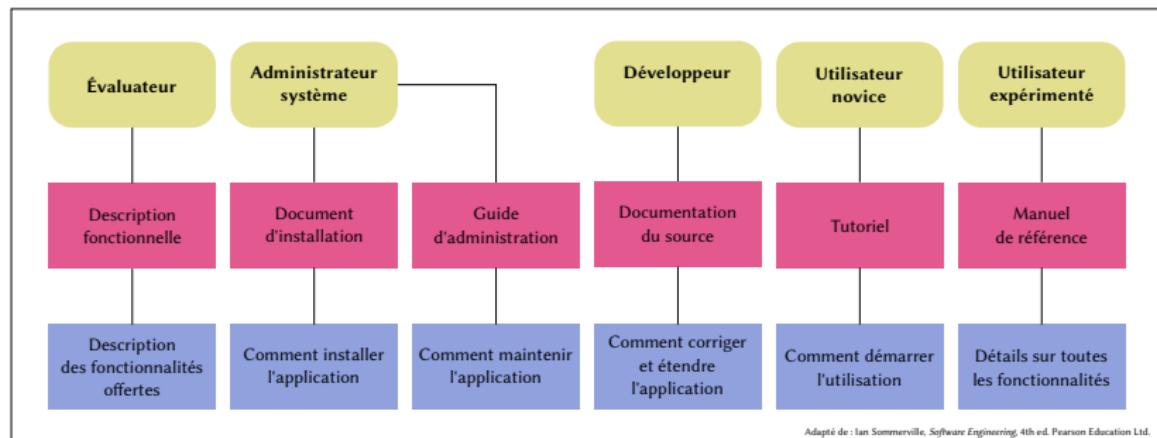
**rapidSVN.** Interface cross-plateforme écrite en C++ avec wxWidgets (<http://rapidsvn.tigris.org/>)

**webSVN.** Accès au dépôt SVN par le web  
(<http://www.websvn.info/>)

# Documentation



# Différentes documentations



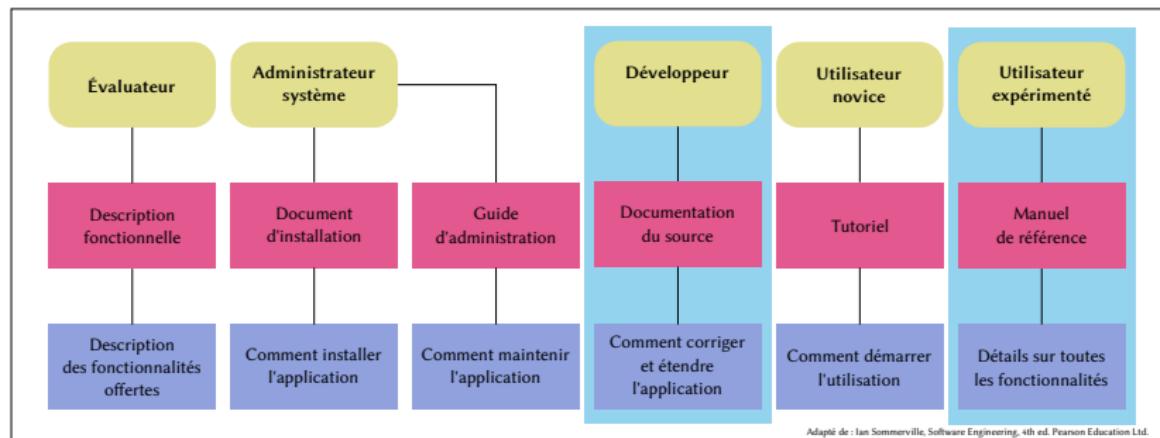
## Documentation du code source

*Programs must be written for people to read, and  
only incidentally for machines to execute.*

— H. Abelson & G. J. Sussman, The Structure  
and Interpretation of Computer Programs



# Documentation du code



Adapté de : Ian Sommerville, Software Engineering, 4th ed. Pearson Education Ltd.



Pourquoi :

- ▶ Maintenance  
(corrections, modifications, extensibilité)
- ▶ Augmenter la qualité du code à *la production*

Comment :

- ▶ Du code dans un texte descriptif : *literate programming*
- ▶ Du texte descriptif en commentaires dans du code : javadoc, doxygen, *docstrings* en Python et LISP, ...
- ▶ Adaptation de la langue à la cible (anglais si développement international)
- ▶ Le code peut/doit être sa propre documentation



```
#include <iostream>
#include <cmath>
using std::cin; using std::cout; using std::endl; using std::sqrt; double
f(double p1, double p2, double p3){return p2*p2-4*p1*p3;}
int main(void){double v1,v2,v3;cout<<"Valeurs ? ";cin>>v1>>v2>>v3;
double x=f(v1,v2,v3);if(x<0){cout<<"Pas de solution"<<endl;}else{
double y=2*v1;if(x>0){double x2=sqrt(x);cout<<"Solutions : "<<(-v2-x2)/y
<<" et "<<(-v2+x2)/y<<endl;}else{cout<<"Solution : "<<-v2/y<<endl;
}};return 0;}
```

- ▶ Sens et maintenabilité du programme?



# Motivation

```
#include <iostream>
#include <cmath>

using std::cin;
using std::cout;
using std::endl;
using std::sqrt;

double f(double p1,
          double p2, double p3) {
    return p2*p2 - 4*p1*p3;
}

int main(void) {
    double v1, v2, v3;

    cout << "Valeurs ? ";
    cin >> v1 >> v2 >> v3;

    double x = f(v1, v2, v3);
```

```
if (x < 0) {
    cout << "Pas de solution" << endl;
} else {
    double y = 2*v1;
    if (x > 0) {
        double x2 = sqrt(x);
        cout << "Solutions : "
            << (-v2-x2)/y
            << " et "
            << (-v2+x2)/y
            << endl;
    } else {
        cout << "Solution : " << -v2/y
            << endl;
    }
}
return 0;
```

- ▶ Sens et maintenabilité du programme?
- ▶ Première documentation : *l'indentation*



# Motivation

```
#include <iostream>
#include <cmath>

using std::cin;
using std::cout;
using std::endl;
using std::sqrt;

double discriminant(double a,
                     double b, double c) {
    return b*b - 4*a*c;
}

int main(void) {
    double a, b, c;

    cout << "Valeurs a b c ? ";
    cin >> a >> b >> c;
}

double delta = discriminant(a, b, c);

if (delta < 0) {
    cout << "Pas de solution" << endl;
} else {
    double twice_a = 2*a;
    if (delta > 0) {
        double sqrt_delta = sqrt(delta);
        cout << "Solutions : "
            << (-b-sqrt_delta)/twice_a
            << " et "
            << (-b+sqrt_delta)/twice_a
            << endl;
    } else {
        cout << "Solution : " << -b/twice_a
            << endl;
    }
}
return 0;
```

- ▶ Sens et maintenabilité du programme?
- ▶ Première documentation : *l'indentation*
- ▶ Deuxième documentation : *choix des identificateurs*



```
#include <iostream>
#include <cmath>

using std::cin;
using std::cout;
using std::endl;
using std::sqrt;

double discriminant(double a,
                     double b, double c) {
    return b*b - 4*a*c;
}

int main(void) {
    double a, b, c;

    cout << "Valeurs a b c ? ";
    cin >> a >> b >> c;

    double delta = discriminant(a, b, c); }

    if (delta < 0) {
        cout << "Pas de solution" << endl;
    } else {
        double twice_a = 2*a;
        if (delta > 0) {
            double z = -.5*(b
                            + copysign(1.0, b)
                            * sqrt(delta));
            cout << "Solutions : "
                            << z/a
                            << " et "
                            << c/z
                            << endl;
        } else {
            cout << "Solution : " << -b/twice_a
                            << endl;
        }
    }
    return 0;
}
```

- ▶ Sens et maintenabilité du programme?
- ▶ Première documentation : *l'indentation*
- ▶ Deuxième documentation : *choix des identificateurs*
- ▶ Nouveau sens du programme?



```
#include <iostream>
#include <cmath>

using std::cin;
using std::cout;
using std::endl;
using std::sqrt;

double discriminant(double a,
                     double b, double c) {
    return b*b - 4*a*c;
}

int main(void) {
    double a, b, c;

    cout << "Valeurs a b c ? ";
    cin >> a >> b >> c;

    double delta = discriminant(a, b, c);
    if (delta < 0) {
        cout << "Pas de solution" << endl;
    } else {
        double twice_a = 2*a;
        if (delta > 0) {
            /* Version garantissant la robustesse
               des calculs
               (voir: "Numerical Recipes in C",
               section 5.6) */
            double z = -.5*(b
                            + copysign(1.0,b)
                            * sqrt(delta));
            cout << "Solutions : "
                            << z/a
                            << " et "
                            << c/z
                            << endl;
        } else {
            cout << "Solution : " << -b/twice_a
                            << endl;
        }
    }
    return 0;
}
```

- ▶ Sens et maintenabilité du programme?
- ▶ Première documentation : *l'indentation*
- ▶ Deuxième documentation : *choix des identificateurs*
- ▶ Nouveau sens du programme?
- ▶ « *Code = comment, commentaires = pourquoi* »



- ▶ Nombreuses formes d'indentation  
(e.g., en C : K&R, GNU, ...)
- ▶ Standards définis au sein d'une organisation
- ▶ Sans standard, l'important est la cohérence de style
- ▶ L'indentation clarifie le sens... ou l'obscurcit :

```
if (x != 0)
    y = 4;
    z = 5;
t = 3;
```

- ▶ Archétype : Python



# Choix des identificateurs

Noms de fonctions, variables, constantes, classes...

- ▶ Choix pertinent et non sujet à mauvaise interprétation

```
// Ambigus
int pi = 17;
double matrix = 3.1;
// Apportent du sens
double delta = b*b - 4*a*c;
double energie_cinetique = .5*masse*vitesse*vitesse;
double Ec = .5*m*v*v;
```

- ▶ Nombreuses conventions :

- ▶ int an\_identifier ;
- ▶ int anotherIdentifier ;
- ▶ ...

Le choix dépend de la communauté de travail et du langage

- ▶ Bon choix de noms d'identificateurs  $\implies$  moins de commentaires



## Les commentaires (1/2)

*Besides a mathematical inclination, an exceptionally good mastery of one's native tongue is the most vital asset of a competent programmer.*  
— Edsger Dijkstra

- ▶ Un bon commentaire ne paraphrase pas le code

```
int i = i+1; // incrémentation de i
```

- ▶ Un bon commentaire est à jour

```
// On ne considère que les 8 premières valeurs  
double somme = accumulate(T,T+5,0.0);
```

- ▶ Rechercher la pertinence plutôt que le volume

- ▶ Plus il y a de commentaires et plus le risque de désynchronisation est important



## Les commentaires (2/2)

- ▶ Un bon commentaire décrit succinctement le « pourquoi » du code

```
/* On utilise l'algorithme de Moore-Skelboe pour
   trouver la valeur minimale de la fonction
   (voir S. Skelboe, BIT 1974, vol 14, p 87—95) */
for (int i=0; i < nbboxes; ++i) {
    [...]
```

- ▶ Un commentaire doit être non ambigu
- ▶ Un commentaire doit être écrit correctement et professionnellement (humour, ...)

```
// Cette variables sont amphet des csts
double pi = 3.14;
double e = 2.71;
[...]
/**
 * If you don't understand this code, you should be
 * flipping burgers instead.
*/
[...]
```

- ▶ Style des commentaires dépend du langage et de la communauté



*Good code is its own best documentation. As you're about to add a comment, ask yourself, 'How can I improve the code so that this comment isn't needed?' Improve the code and then document it to make it even clearer.*

— Steve McConnell, *Code Complete*.

*There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.*

— C. A. R. Hoare

- ▶ **Simplicité** du code (*do it right [then do it fast]*)



- ▶ Présentation des éléments du code
- ▶ Explication des liens entre éléments
- ▶ Commentaire sur les éléments et leurs liens
- ▶ Doit être écrite en même temps que le code (dire ce que l'on va faire en langage courant, puis en langage formel)
- ▶ Autorise la maintenance aisée du code
- ▶ Aide à la rédaction du manuel de référence
- ▶ Recensement des liens et création d'index : automatisable
- ▶ Explication du code : non automatisable (à fournir par les développeurs)
- ▶ Synchronisation code/documentation : fusion code/documentation en un seul fichier (documentation en commentaires)



## Collecte d'informations automatisable

gaol (Just Another Interval Library): gaol\_assert.h File Reference - Mozilla Firefox

Ble Edit View History Bookmarks ScrapBook Tools Help

file:///home/goualard/projects/gaol/doc/html/gaol\_assert\_bh.html

Google Dics Amazon Madoc Repositories P1788 Math Reviews Pages web Projects Python Aigain DB Bib. Treillères Brother HL-5380DN

Main Page Namespace Classes Files

## gaol\_assert.h File Reference

Assertions: [More...](#)

```
#include "gaol/gaol_config.h"
Include dependency graph for gaol_assert.h
```

```

graph TD
    gaol_assert_bh[gaol_assert.h] --> gaol_gaol_config_h[gaol/gaol_config.h]
    gaol_gaol_config_h --> gaol_gaol_configuration_h[gaol/gaol_configuration.h]
    
    gaol_gaol_configuration_h --> gaol_double_op_h[gaol_double_op.h]
    gaol_double_op_h --> gaol_double_op_apmath_h[gaol_double_op_apmath.h]
    gaol_double_op_h --> gaol_double_op_citbm_h[gaol_double_op_citbm.h]
    gaol_double_op_h --> gaol_double_op_m_h[gaol_double_op_m.h]
    
    gaol_double_op_apmath_h --> gaol_interval_h[gaol/interval.h]
    gaol_double_op_citbm_h --> gaol_interval_h
    gaol_double_op_m_h --> gaol_interval_h
    
    gaol_interval_h --> gaol_gaol_h[gaol.h]
    gaol_gaol_h --> gaol_interval_parser_cpp[gaol/interval_parser.cpp]
    
    gaol_gaol_h --> gaol_expression_h[gaol/expression.h]
    gaol_expression_h --> gaol_common_cpp[gaol/common.cpp]
    gaol_common_cpp --> gaol_expr_evalh[gaol/expr_eval.h]
    gaol_expr_evalh --> gaol_parser_cpp[gaol/parser.cpp]
    gaol_parser_cpp --> gaol_interval_lever_cpp[gaol/interval_lever.cpp]
    gaol_interval_lever_cpp --> gaol_expression_cpp[gaol/expression.cpp]
    gaol_expression_cpp --> gaol_common_cpp
  
```

This graph shows which files directly or indirectly include this file:



# Literate Programming

*Most programs are written to be executed,  
a few are written to be maintained, but  
almost no program are written to be read.*

— J. Bentley

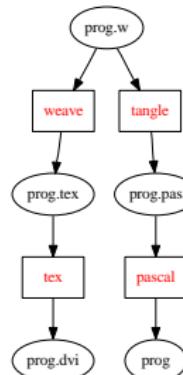
Donald Knuth, 1981 : un programme doit pouvoir être de la « littérature »

Exemple :

The `\tt treeprint` routine takes one option, `"-p"`, which tells it to use the printer's line-drawing set, rather than the terminal's.

```
0c
0<Global definitions@>@;
0<Global include files@>@;
0<Global declarations@>@;

0#
main( argc, argv )
    int argc;
    char **argv;
{
0<|main| variable declarations@>;
0<Search for options and set special characters on |"-p"|@>;
0<Read output from find and enter into tree@>;
0<Write tree on standard output@>@;
    exit (0);
}
[...]
```





Approche « alternative » au *literate programming* :

- ▶ Ajout de commentaires avec un formatage spécial dans le code
- ▶ Nombreux outils disponibles pour différents langages :
  - ▶ Javadoc : Java
  - ▶ Doxygen : C/C++, Python, VHDL
  - ▶ Epydoc/Epytext : Python ...
- ▶ Point positif : code et documentation fortement couplés



# Commentaires formatés

```
/*
 \brief Format to use for the output of intervals.

The supported formats so far are the following:
— bounds: the interval is output in the form "[l, r]" where l and r
are respectively its left and right bounds
— center: the interval is output as a single value, its center.
— hexa: same as "bounds" except that bounds are printed as hexadecimal
values to avoid problems due to round-off errors when translating
the floats into decimal
— agreeing: the interval is output in the form "r [l, r]" where
r is the number containing all the digits that are the same in both
left and right bounds, and where l and r are the disagreeing
remaining digits. See "Factored Notation for Interval I/O", Maarten
Herman van Emden, CoRR index=cs.NA/0102023.

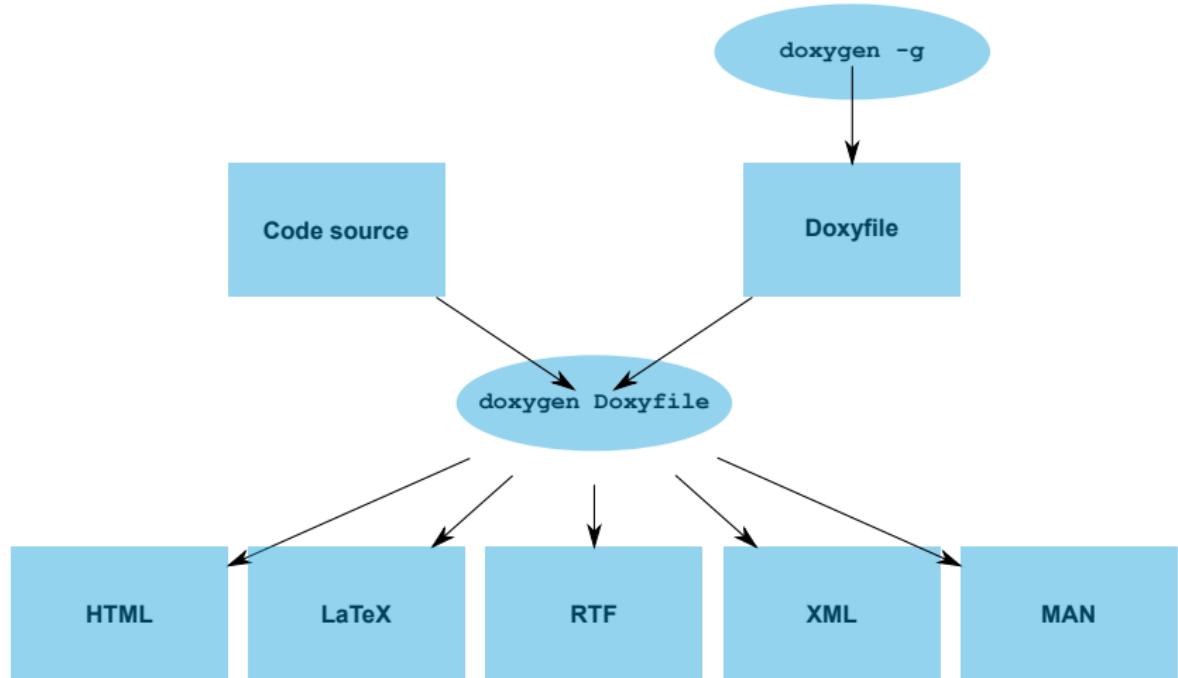
*/
struct __GAOL_PUBLIC__ interval_format {
    enum format_t {
        bounds,
        center,
        hexa,
        agreeing
    };
};

/*
 \brief test for evenness
 \warning d should not be \f$ \pm \infty \f$
 */
bool feven(double d)
{
    // FIXME: check wether floor is in std namespace beforehand
    return (std::floor(0.5*d)*2.0 == d);
}
```

- ▶ Programme analogue à Javadoc principalement pour C/C++
- ▶ Développé par Dimitri van Heesch et distribué en GPL
- ▶ Documentation produite en HTML, L<sup>A</sup>T<sub>E</sub>X, RTF, PostScript, PDF, XML, Man pages
- ▶ Nombreux tags disponibles  
(multiples notations, e.g. : \author, @author)



# Utilisation de Doxygen



## Doxyfile :

```
# Doxyfile 1.6.3
# This file describes the settings to be used by the documentation system
# doxygen (www.doxygen.org) for a project
#
# All text after a hash (#) is considered a comment and will be ignored
# The format is:
#     TAG = value [value, ...]
# For lists items can also be appended using:
#     TAG += value [value, ...]
# Values that contain spaces should be placed between quotes (" ")

# The PROJECT_NAME tag is a single word (or a sequence of words surrounded
# by quotes) that should identify the project.

PROJECT_NAME      =

# The OUTPUT_DIRECTORY tag is used to specify the (relative or absolute)
# base path where the generated documentation will be put.
# If a relative path is entered, it will be relative to the location
# where doxygen was started. If left blank the current directory will be used.

OUTPUT_DIRECTORY   =

# If the EXTRACT_ALL tag is set to YES doxygen will assume all entities in
# documentation are documented, even if no documentation was available.
# Private class members and static file members will be hidden unless
# the EXTRACT_PRIVATE and EXTRACT_STATIC tags are set to YES

EXTRACT_ALL        = NO
[...]
```



Contenu des commentaires :

- ▶ Une phrase courte de résumé (terminée par un point final)
- ▶ Une description plus détaillée (au présent, à la troisième personne du singulier)
- ▶ Des tags spéciaux (commençant une ligne)

Fichier HTML généré pour une classe :

- ▶ Description générale de la classe
- ▶ Index des méthodes et constructeurs
  - ▶ Liens linkables
  - ▶ Une phrase de description par entrée
- ▶ Description détaillée des méthodes et constructeurs
- ▶ Description des paramètres des méthodes et constructeurs



## Commentaires Doxygen (2/2)

- ▶ Commentaires dans le code pour les classes, variables, structures, constantes, ...
- ▶ Commentaires à part pour les fichiers (*structural command*)
- ▶ Commentaires reconnus par Doxygen :

```
/*!  
   Commentaire long  
*/
```

```
/** Commentaire long  
  
/// Commentaire court  
  
///! Commentaire court
```

```
/*
 * \brief Pretty nice class.
 * \details This class is used to demonstrate a number of section commands.
 * \author John Doe
 * \author Jan Doe
 * \version 4.1a
 * \date 1990-2011
 * \pre First initialize the system.
 * \bug Not all memory is freed when deleting an object of this class.
 * \warning Improper use can crash your application
 * \copyright GNU Public License.
 */
class SomeNiceClass {};

/*
Copies bytes from a source memory area to a destination memory area,
where both areas may not overlap.
@param[out] dest The memory area to copy to.
@param[in] src The memory area to copy from.
@param[in] n The number of bytes to copy
@exception None
@return Nothing
*/
void memcpy(void *dest, const void *src, size_t n);
```



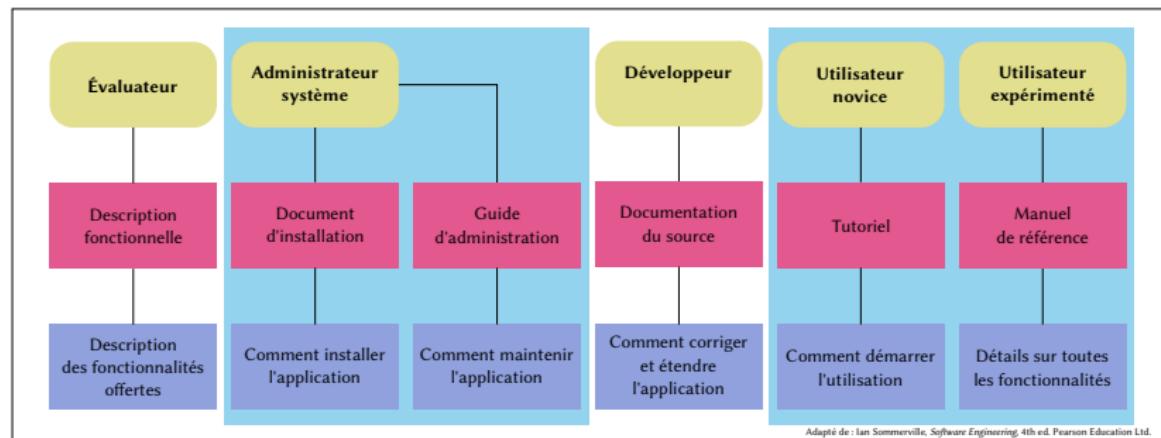
# Limites des outils du type « Doxygen »

- ▶ Deux sortes de commentaires :
  - ▶ À destination des seuls programmeurs
  - ▶ À destination des utilisateurs (*via* Doxygen)
- ▶ Floutage de la frontière développeur/utilisateur
- ▶ Documentation Doxygen : suit l'organisation du code et non l'organisation logique de l'application
  - ▶ Adéquat pour une documentation technique à destination des développeurs
  - ▶ Utilisable pour le manuel de référence
  - ▶ Inadapté pour la rédaction du manuel d'utilisation et du tutoriel

## **Documentation du logiciel**



# Documentation du logiciel



- ▶ Documenter un logiciel
- ▶ Production d'une documentation :  $\text{\LaTeX}$ 
  - ▶ Avant-propos : pourquoi  $\text{\LaTeX}$  ?
  - ▶ Liens et références utiles
  - ▶ Historique de  $\text{\TeX}$  et  $\text{\LaTeX}$
  - ▶ Possibilités de  $\text{\LaTeX}$  : une visite guidée
  - ▶ Bases de  $\text{\LaTeX}$
  - ▶ Bibliographie et Bib $\text{\TeX}$
  - ▶ Makeindex
  - ▶ Pour aller plus loin avec  $\text{\LaTeX}$
- ▶  $\text{\TeX}info$



But :

- ▶ Fournir à l'utilisateur *lambda*/confirmé un moyen d'utiliser le programme ou la librairie
- ▶ Cas des librairies : confusion possible documentation technique/documentation utilisateur

Découplage code source / manuel : l'architecture du programme ne détermine pas l'organisation du manuel

Différentes organisations :

- ▶ Tutoriel
- ▶ Manuel thématique
- ▶ Manuel de référence

*GNU Standards* : une bonne documentation se lit suivant les trois organisations



- ▶ Attentes de l'application en terme d'architecture machine et de logiciels pré-installés
- ▶ Comment compiler/installer l'application
- ▶ Limitations de l'application (tailles maximum des fichiers d'entrée, ...)
- ▶ Comment utiliser l'application
  - ▶ Fonctionnalités de l'application (avec exemples)
  - ▶ Options de la ligne de commande
  - ▶ Description des messages d'erreurs
  - ▶ Glossaire de termes spécifiques
  - ▶ ...



## Écrire une bonne documentation utilisateur (1/2)

- ▶ Écriture d'une bonne documentation par imitation (voir documentation outils GNU)
- ▶ Organiser la documentation suivant les concepts et les questions que se pose l'utilisateur (à appliquer à tous les niveaux : ordre des paragraphes jusqu'à l'ordre des chapitres)
- ▶ Éviter une liste de fonctionnalités (organisation logique par concepts)
- ▶ Décrire les fonctionnalités avec leur applicabilité et leurs limites (exemples)
- ▶ Décrire les bonnes pratiques d'utilisation et ce qui doit être évité
- ▶ Bon manuel = tutoriel et référence lisible de bout en bout avec accès direct à la section pertinente pour résoudre un problème précis

- ▶ Fournir une information sur la documentation elle-même (quels chapitres lire en fonction des attentes du lecteur)
- ▶ Prévoir plusieurs niveaux de lecture (débutant/confirmé)
- ▶ Prévoir un index des concepts
- ▶ Écrire à la forme active plutôt que passive :
- ▶ Utiliser des phrases courtes ; chaque phrase présente un seul fait
- ▶ Utiliser des paragraphes courts (*la perfection est atteinte non lorsqu'il n'y a plus rien à ajouter mais lorsqu'il n'y a plus rien à retrancher*)
- ▶ Expliquer sous différentes formes les faits complexes

Rédaction de document : un métier en soi

## Production d'une documentation : **LATEX**



# Avant-propos : pourquoi L<sup>A</sup>T<sub>E</sub>X ? (1)

Document = forme + fond

- ▶ Outils de traitement de texte WYSIWYG :
  - ▶ Entrelacement
    - ▶ « écriture du contenu »
    - ▶ « mise en forme »
  - ▶ Rédacteur  $\neq$  composeur/typographe
  - ▶ Vision locale (la ligne)
    - ⇒ composition de mauvaise qualité
  - ▶ Rétro-compatibilité non assurée



# Avant-propos : pourquoi L<sup>A</sup>T<sub>E</sub>X ? (2)

- ▶ Outils de composition (troff, lout, T<sub>E</sub>X) WYMIWYG :
  - ▶ Rédaction du contenu + ajout de *tags* pour la structure
  - ▶ Description du « *quois* », pas du « *comment* »
  - ▶ Vision globale (la page)  
⇒ composition équilibrée digne des typographes humains
  - ▶ Source = fichier ASCII (rétro-compatibilité assurée)
- ▶ L<sup>A</sup>T<sub>E</sub>X :
  - ▶ Sur-couche de T<sub>E</sub>X simplifiant la rédaction
  - ▶ Outil très largement répandu



**CTAN.** *Comprehensive TeX Archive Network.* <http://www.ctan.org>

**GUTenberg.** Association GUTenberg. <http://www.gutenberg.eu.org/>

**Projet LATEX.** <http://www.latex-project.org/>

**Newsgroups.** comp.text.tex et fr.comp.text.tex

**Lamport 94.** *LATEX : a document preparation system.* L. Lamport. Addison-Wesley

**Goossens et al. 94.** *The LATEX companion.* M. Goossens, F. Mittelbach, A. Samarin. Addison-Wesley

**Oetiker 01.** *The not so short introduction to LATEX2<sub>E</sub>.* T. Oetiker et al. Disponible sur le CTAN

**Kopka & Daly 2004.** *Guide to LATEX.* 4th ed. H. Kopka et P. W. Daly Addison-Wesley, 2004.

**Dario Taraborelli.** The Beauty of LATEX. <http://dartar.free.fr/w/?vakka=latex>

**Gerben Wierda.** The TeX showcase. <http://www.tug.org/texshowcase/>



- ▶ 1978 : Knuth horrifié par la qualité de production de TAOCP
- ▶ Définition du *méilleur* logiciel de composition... présent et à venir
- ▶ Création de  $\text{\TeX}$  ( $\tau_{\varepsilon}\chi$ ) :
  - ▶ Langage complet de programmation orienté « composition »
  - ▶ Fonctions du langage = *macros*
  - ▶ Précision de placement  
 $1/100$  longueur d'onde de la lumière visible



- ▶  $\text{\TeX}$  :
  - ▶ Programme très robuste  
(erreur dans  $\text{\TeX} = \$ 327,68$ )
  - ▶ Puissant mais difficile à manipuler
- ▶ 1985 :  $\text{\LaTeX}2.09$  par Leslie Lamport
  - ▶ Ensemble de macros simplifiant l'utilisation de  $\text{\TeX}$
- ▶ 1994 :  $\text{\LaTeX}2_{\varepsilon}$
- ▶ 1995 : début du projet  $\text{\LaTeX}3$

**Point important** : compatibilité ascendante garantie

# Les possibilités de $\text{\LaTeX}$

## *Visite guidée*

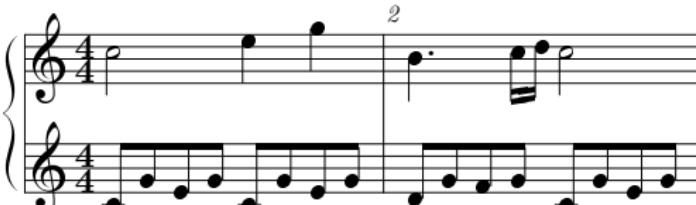
- ▶ Livres
  - ▶ Rapports
  - ▶ Articles de recherche
  - ▶ Poésies
  - ▶ Calligrammes
  - ▶ ...

8164062  
795028841971693  
832795028841971693  
9937510582097494592  
9937510582097494592  
P

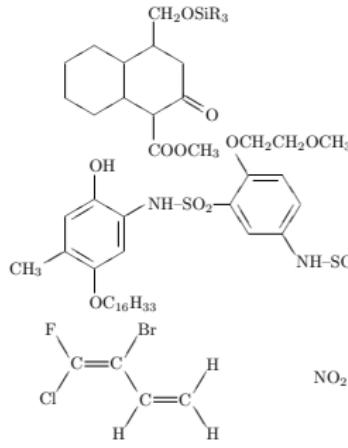
## Écriture de partitions de musiques :

```
\begin{music}
\parindent 1cm
\def\nbinstruments{1}\relax
\def\instrumenti{Piano}%
\nbporteesi=2\relax
\generalmeter{\meterfrac{4}{4}}\relax
\debutextrait
\normal
\tempo\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\tempo\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\ql l\sk\ql n\enotes
\barre
\Notes\ibu0f0\qh0{dgf}\qlp i\enotes
\notes\tbu0\qh0g|\ibbl1j3\qb1j\tbl1\qb1k\enotes
\tempo\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\fineextract
\end{music}
```

Piano

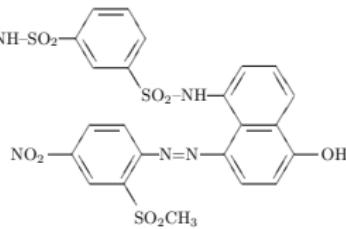


### ► Lilypond



```
\decalinev{1B==CH$_{2}$OSiR$_{3}$;3D==0;4A==COOCH$_{3}$;%  
0FB==CH$_{3}$;OGA==H}  
  
\bzdrv{1==OH;5==CH$_{3}$;4==OC$_{16}$H$_{33}$;%  
2==\ryl(4==NH$-$SO$_{2}$){4==\bzdrh{1==(y1)};%  
2==OCH$_{2}$CH$_{2}$OCH$_{3}$;%  
5==\ryl(2==NH$-$SO$_{2}$){4==\bzdrh{1==(y1)};%  
5==\ryl(2==SO$_{2}$-$-$NH){4==\naphdrh{1==(y1);5==OH;%  
8==\lyl(4==N=N){4==\bzdrh{4==(y1);1==NO$_{2}$;%  
5==SO$_{2}$CH$_{3}$}}}}}}}}
```

```
\Ethyleneh{1==C; 2==C}{1==F; 2==Cl; 4==Br; %
3==\Ethyleneh{1==C; 2==C}{1===(yl); 2==H; 3==H; 4==H}}}
```





Avec MS Equation 3.0 :

$$G(z) = e^{\ln G(z)} = \exp\left(\sum_{k \geq 1} \frac{S_k z^k}{k}\right) = \prod_{k \geq 1} e^{S_k z^k / k}$$

Avec L<sup>A</sup>T<sub>E</sub>X :

$$G(z) = e^{\ln G(z)} = \exp\left(\sum_{k \geq 1} \frac{S_k z^k}{k}\right) = \prod_{k \geq 1} e^{S_k z^k / k}$$

```
\begin{equation*}
G(z) = e^{\ln G(z)} = \exp\biggl(\sum_{k \geq 1} \frac{S_k z^k}{k}\biggr) =
\prod_{k \geq 1} e^{S_k z^k / k}
\end{equation*}
```



La dérivée de  $f(x) : x \mapsto \cos(x)^2 + \sin(\ln(x))$  est  $-2 \sin(x) \cos(x) + \frac{\cos(\ln(x))}{x}$ .

```
\documentclass{article}
\usepackage{sagetex}

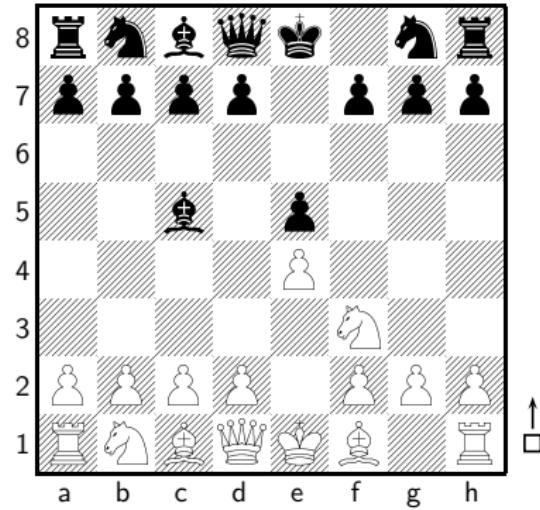
\begin{document}
\begin{sagesilent}
var('x')
f(x)=sin(log(x))+cos(x)^2
\end{sagesilent}
La dérivée de $f(x)\colon sage{f}$$ est $\sage{diff(f(x),x)}$.
\end{document}
```

- ▶ **Pythontex** (Python dans \LaTeX)
- ▶ **plasTeX** (\LaTeX dans Python)



Et bien plus encore...

1 e4 e5 2 Nf3 Bc5

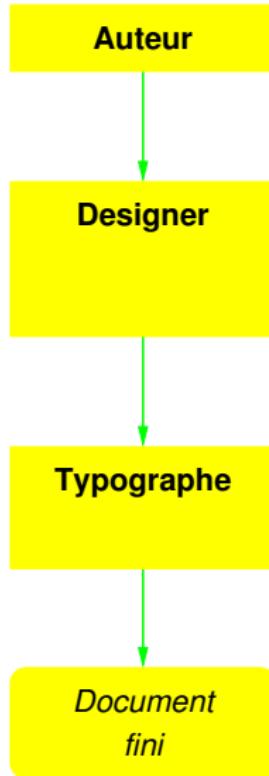


```
\newgame  
\mainline{1. e4 e5 2. Nf3 Bc5}  
\[\showboard\]
```

## Bases de $\text{\LaTeX}$



# Principe de L<sup>A</sup>T<sub>E</sub>X

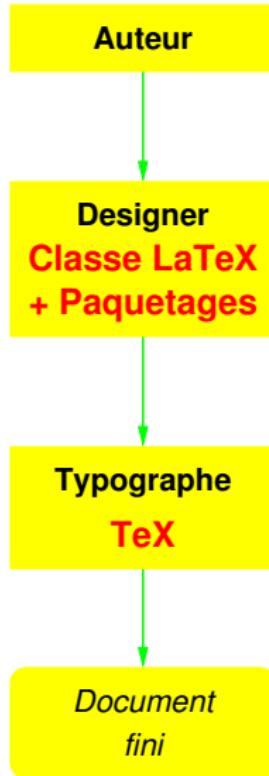


L<sup>A</sup>T<sub>E</sub>X :

- ▶ Choix d'une *classe* en fonction du type de document
  - ▶ `report` (rapport)
  - ▶ `article` (article de recherche)
  - ▶ `letter` (lettre)
  - ▶ ...
- ▶ Choix de *paquetages* pour utiliser des fonctionnalités additionnelles
  - ▶ `graphicx` (inclusion d'images)
  - ▶ `amsmath` (extensions mathématiques)
  - ▶ Très nombreux paquetages sur le **CTAN**



# Principe de L<sup>A</sup>T<sub>E</sub>X



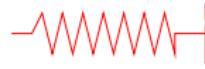
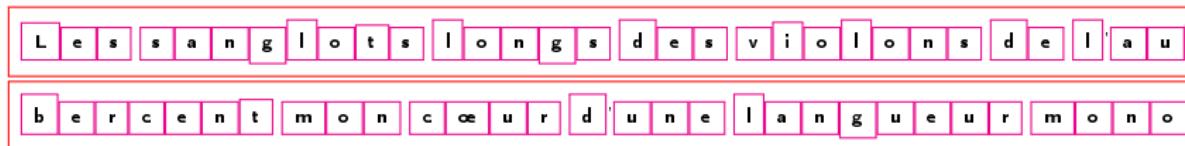
L<sup>A</sup>T<sub>E</sub>X :

- ▶ Choix d'une *classe* en fonction du type de document
  - ▶ report (rapport)
  - ▶ article (article de recherche)
  - ▶ letter (lettre)
  - ▶ ...
- ▶ Choix de *paquetages* pour utiliser des fonctionnalités additionnelles
  - ▶ graphicx (inclusion d'images)
  - ▶ amsmath (extensions mathématiques)
  - ▶ Très nombreux paquetages sur le **CTAN**



# Principe de TEX

- TEX = boîtes + ressorts



- Boîtes horizontales (lettres, lignes) et verticales (paragraphes, pages)



# Contenu d'un fichier L<sup>A</sup>T<sub>E</sub>X (1)

- ▶ Fichier ASCII
  - ▶ Formattage du texte :
    - ▶  $n$  espaces  $\Rightarrow$  1 espace
    - ▶ Ligne vide : sépare deux paragraphes
- 

Longtemps, je me suis couché de bonne  
heure. Parfois, à peine ma bougie  
éteinte...  
mes yeux se fermaient si vite...

Longtemps, je me suis couché  
de bonne heure. Parfois,  
à peine ma bougie éteinte...

mes yeux se fermaient si vite...

---



## Contenu d'un fichier L<sup>A</sup>T<sub>E</sub>X (1)

- ▶ Commentaires : introduit par '%', jusqu'à la fin d'une ligne
- ▶ Caractères spéciaux :

# \$ % ^ & \_ { } ~ \

- ▶ Commandes (*macros*) : identificateur (seulement des lettres) précédé de '\'

---

```
Adieu veaux, vaches, cochons, \dots %<- ellipse
```

```
Adieu veaux, vaches, cochons, ...
```

---



# Caractères spéciaux

- ▶ *em dash* et *en dash* :

Lisez les pages 34--45 --- du moins je le crois ---

Lisez les pages 34-45 — du moins je le crois —

- ▶ Espace insécable '~~'

M. ~Jean Dupont

M. Jean Dupont

- ▶ Ellipse :

Des pommes, des poires\ldots

Des pommes, des poires...



- ▶ Combinaison accent + lettre :

H<sup>^</sup>otel, NO<sup>"</sup>EL, na<sup>\i{}</sup>ve, sm{\o}rrebr{\o}d,  
Stra{\ss}e, {\OE}uf, {\AA}rhus

Hôtel, NOËL, naïve, smørrebrød, Straße, Øuf, Århus

- ▶ Écriture directe avec 'é', 'ï', ...

⇒ Bon choix d'encodage (ou Xe<sup>L</sup>T<sub>E</sub>X)



# Les macros (1)

- ▶ Macros sans paramètres

```
\dots \hrulefill \par
```

- ▶ Macros avec paramètres (encadrés par des accolades)

```
Mot \emph{accentué}.\n\GenericWarning{Bla bla}{Bli}
```

- ▶ Macros avec un paramètre optionnel (entre crochets)

```
\marginpar[Gauche]{Droite}
```



## Les macros (2)

- ▶ Une macro sans paramètre détruit l'espace qui suit :

---

\LaTeX est un langage merveilleux

\LaTeX est un langage merveilleux

---

⇒ Rajouter une paire d'accolades après la macro



- ▶ Un environnement exerce son influence sur une portion de texte :

```
\begin{toto}
```

```
...
```

```
Influence de l'environnement toto
```

```
...
```

```
\end{toto}
```



## Les groupes

- ▶ Un groupe est délimité par des accolades
- ▶ Un environnement constitue un groupe  
{un groupe {et un groupe dans un groupe}}
- ▶ Une macro exerce son influence à l'intérieur du groupe où elle apparaît

Du texte normal, {\huge du gros texte} et du normal de nouveau

Du texte normal, **du gros texte** et du normal de nouveau



# Organisation d'un fichier L<sup>A</sup>T<sub>E</sub>X

```
\documentclass[Options]{NomDeClasse}
\usepackage{NomDePaquetage}
\usepackage{NomDePaquetage}
...
Définitions et appels de macros : le préambule
\begin{document}
    Texte
\end{document}
```

Classes standards :

- ▶ article, report, book, frames

Paquetages utiles :

- ▶ amsmath, amssymb, graphicx, pstricks, ...
- ▶ inputenc



## Exemple : un article

```
\documentclass[a4paper,12pt]{article}
\usepackage{graphicx}
\title{De la prolifération des couleuvres en Bas-Morvan}
\author{Jean Dupont}
\begin{document}
\maketitle
\section{Introduction}
Ceci est l'introduction...
\section{Conclusion}
Ceci est la conclusion.
\bibliographystyle{plain}
\bibliography{ma_biblio}
\end{document}
```



- ▶ Sections (disponibilité dépendant de la classe) :

```
\part{}      \chapter{}  
\section{}   \subsection{}    \subsubsection{}  
\paragraph{} \ subparagraph{}
```

- ▶ Création d'une table des matières : appel de  
`\tableofcontents`

```
\begin{document}  
  \maketitle  
  \tableofcontents  
  ...  
\end{document}
```



# Référencement (1)

- ▶ Marquage d'une position par un *label* pour référence ultérieure
- ▶ Indépendance vis à vis des déplacements ultérieurs

Nous découvrirons dans la section~\ref{sec:couleuvre} la vie passionnante de la couleuvre à collier.

...

```
\section{La couleuvre à collier}
\label{sec:couleuvre}
```

...

Nous découvrirons dans la section 2 la vie passionnante de la couleuvre à collier.

...

2. La couleuvre à collier

...



- ▶ Pose d'une étiquette :
  - ▶ `\label{etiquette}`
- ▶ Référence à une étiquette :
  - ▶ `\ref{etiquette}`
  - ▶ `\eqref{etiquette}` (référence dans une équation)
  - ▶ `\pageref{etiquette}` (page où apparaît l'étiquette)



- ▶ Mise **en gras** : `\textbf{texte}`
- ▶ Mise *en italique* : `\textit{texte}`
- ▶ Mise en sans-sérif : `\textsf{texte}`
- ▶ Soulignement d'un point *important* : `\emph{texte}`  
`\emph{}` indique « *quoi* », pas « *comment* »

Les tailles :

- ▶ Normal : `{\normalsize texte}`
- ▶ Grand : `{\large texte}`
- ▶ Très grand : `{\Large texte}`
- ▶ Énorme : `{\huge texte}`
- ▶ On a aussi `\small`, `\footnotesize`, `\scriptsize`, `\tiny`  
`\fontsize{}{} \selectfont`



## ► Listes enumerate, itemize, description

```
\begin{enumerate}
\item Pomme
\item Poire
\item Banane
\end{enumerate}
```

```
\begin{itemize}
\item Pomme
\item Poire
\item Banane
\end{itemize}
```

```
\begin{description}
\item[Pomme.] Un fruit
\item[Poire.] Heu\ldots
\item[Banane.] Ben\ldots
\end{description}
```

1. Pomme
2. Poire
3. Banane

- ▶ Pomme
- ▶ Poire
- ▶ Banane

- Pomme. Un fruit  
Poire. Heu...  
Banane. Ben...



```
\begin{center}  
    Ceci est centré  
\end{center}
```

Ceci est centré



- ▶ Affichage du texte tel quel sans prise en compte des caractères actifs dans L<sup>A</sup>T<sub>E</sub>X
- ▶ Les espaces et les retours à la ligne sont respectés

```
\begin{verbatim}
```

```
\end{verbatim}
```

- ▶ Texte *verbatim* sur une ligne : \verb+tralala+



## ► Environnement tabular :

```
\begin{tabular}{|l c | p{4cm} r| }
\hline
34 & pomme & tralala pouet & droite \\
\cline{2-3}
Youpi & aglaglagla & 45.5 & yam\\
\hline
\end{tabular}
```

34	pomme	tralala pouet	tout à droite
Youpi	aglaglagla	45.5	yam



## ► Environnement tabbing

```
\begin{tabbing}
    il était une fois\= \hspace{4cm}\=\kill
    pomme \> poire \> 4\\
    \pushtabs
    12\=12\=12\=12\=\kill
    A\>B\>C\>D\>E\\
    \> F\>G\>H\>I\\
    \poptabs
    ananas \> oglala \> 12346578\\
\end{tabbing}
```

---

pomme	poire	4
A B C D E		
F G H I		
ananas	oglala	12346578



- ▶ Figures, tableaux, ... ne pouvant être découpés
- ▶ N'apparaissent pas dans le corps du texte
- ▶ Ont une légende et un *label*

```
\begin{figure}[htbp]
    \includegraphics{figure.eps}
    \caption{La légende de la figure}
    \label{fig:maFigure}
\end{figure}
```



## Éléments flottants (2)

Exemple de Figure :



Figure 1: Le logo du LINA

---



Une table :

```
\begin{table}[!htbp]
  \caption{Une jolie table}
  \label{tab:maTable}
```

Ici la jolie table  
\end{table}

► Afficher la liste des tables : \listoftables

► Afficher la liste des figures : \listoffigures

► **Attention :**

Dans tous les cas, \caption{} *avant* \label{}



- ▶ Équation dans le texte : `$...$`
- ▶ Équations hors-texte : environnement `equation`

```
\begin{equation*}
\sum_{i=0}^n x_i y_i = \sqrt{3 * \frac{z_i^3}{12\pi}}
\end{equation*}
```

$$\sum_{i=0}^n x_i y_i = \sqrt{3 * \frac{z_i^3}{12\pi}}$$

---

La valeur de  $x^{n+1}$  n'est pas celle de  $x^{n+1}$

La valeur de  $x^n + 1$  n'est pas celle de  $x^{n+1}$

- ▶ Charger `amsmath` pour avoir `equation*` (et plein d'autres environnements)



- ▶ Environnement array (seulement en mode mathématique)

```
\begin{equation*}
x=\left\{\begin{array}{ll}
12 & \text{si } y \text{ est pair} \\
9 & \text{sinon}
\end{array}\right.
\end{equation*}
```

$$x = \begin{cases} 12 & \text{si } y \text{ est pair} \\ 9 & \text{sinon} \end{cases}$$



- ▶ Définition d'un environnement à partir de la macro `\newtheorem`

```
\newtheorem{loi}{Loi}
```

```
\begin{loi}[Loi de Murphy]
```

De  $n$  possibilités censément équiprobables, c'est toujours la pire qui arrive.

```
\end{loi}
```

## Loi (Loi de Murphy)

*De  $n$  possibilités censément équiprobables, c'est toujours la pire qui arrive.*



# Symboles mathématiques (1)

$\alpha$	<code>\alpha</code>	$\theta$	<code>\theta</code>	$\circ$	<code>\circ</code>	$\tau$	<code>\tau</code>
$\beta$	<code>\beta</code>	$\vartheta$	<code>\vartheta</code>	$\pi$	<code>\pi</code>	$\upsilon$	<code>\upsilon</code>
$\gamma$	<code>\gamma</code>	$\iota$	<code>\iota</code>	$\varpi$	<code>\varpi</code>	$\phi$	<code>\phi</code>
$\delta$	<code>\delta</code>	$\kappa$	<code>\kappa</code>	$\rho$	<code>\rho</code>	$\varphi$	<code>\varphi</code>
$\epsilon$	<code>\epsilon</code>	$\lambda$	<code>\lambda</code>	$\varrho$	<code>\varrho</code>	$\chi$	<code>\chi</code>
$\varepsilon$	<code>\varepsilon</code>	$\mu$	<code>\mu</code>	$\sigma$	<code>\sigma</code>	$\psi$	<code>\psi</code>
$\zeta$	<code>\zeta</code>	$\nu$	<code>\nu</code>	$\varsigma$	<code>\varsigma</code>	$\omega$	<code>\omega</code>
$\eta$	<code>\eta</code>	$\xi$	<code>\xi</code>				
$\Gamma$	<code>\Gamma</code>	$\Lambda$	<code>\Lambda</code>	$\Sigma$	<code>\Sigma</code>	$\Psi$	<code>\Psi</code>
$\Delta$	<code>\Delta</code>	$\Xi$	<code>\Xi</code>	$\Upsilon$	<code>\Upsilon</code>	$\Omega$	<code>\Omega</code>
$\Theta$	<code>\Theta</code>	$\Pi$	<code>\Pi</code>	$\Phi$	<code>\Phi</code>		



# Symboles mathématiques (2)

$\pm$	<code>\pm</code>	$\cap$	<code>\cap</code>	$\diamond$	<code>\diamond</code>	$\oplus$	<code>\oplus</code>
$\mp$	<code>\mp</code>	$\cup$	<code>\cup</code>	$\triangle$	<code>\bigtriangleup</code>	$\ominus$	<code>\ominus</code>
$\times$	<code>\times</code>	$\oplus$	<code>\uplus</code>	$\triangledown$	<code>\bigtriangledown</code>	$\otimes$	<code>\otimes</code>
$\div$	<code>\div</code>	$\sqcap$	<code>\sqcap</code>	$\triangleleft$	<code>\triangleleft</code>	$\oslash$	<code>\oslash</code>
$*$	<code>\ast</code>	$\sqcup$	<code>\sqcup</code>	$\triangleright$	<code>\triangleright</code>	$\odot$	<code>\odot</code>
$\star$	<code>\star</code>	$\vee$	<code>\vee</code>	$\lhd^*$	<code>\lhd^*</code>	$\bigcirc$	<code>\bigcirc</code>
$\circ$	<code>\circ</code>	$\wedge$	<code>\wedge</code>	$\rhd^*$	<code>\rhd^*</code>	$\dagger$	<code>\dagger</code>
$\bullet$	<code>\bullet</code>	$\setminus$	<code>\setminus</code>	$\unlhd^*$	<code>\unlhd^*</code>	$\ddagger$	<code>\ddagger</code>
$\cdot$	<code>\cdot</code>	$\wr$	<code>\wr</code>	$\unrhd^*$	<code>\unrhd^*</code>	$\amalg$	<code>\amalg</code>
$+$	<code>+</code>	$-$	<code>-</code>				

\* présents seulement dans les paquetages `latexsym`, `amsfonts` ou `amssymb`.

Liste des symboles : [The Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol List](#)



## Symboles mathématiques (3)

\leq	\geq	\equiv	\equiv	\models
\prec	\succ	\sim	\simeq	\perp
\preceq	\succeq	\simeq	\asymp	\mid
\ll	\gg	\approx	\approx	\parallel
\subset	\supset	\approx	\approx	\bowtie
\subseteq	\supseteq	\cong	\cong	\Join^*
\sqsubset^*	\sqsupset^*	\neq	\neq	\smile
\sqsubseteq^*	\sqsupseteq	\doteq	\doteq	\frown
\in	\ni	\propto	\propto	=
\vdash	\dashv	<	\vee	>
:				

\* présents seulement dans les paquetages `latexsym`, `amsfonts` ou `amssymb`.



## Symboles mathématiques (4)

Grands opérateurs :

$\sum$	<code>\sum</code>	$\bigcap$	<code>\bigcap</code>	$\odot$	<code>\bigodot</code>
$\prod$	<code>\prod</code>	$\bigcup$	<code>\bigcup</code>	$\otimes$	<code>\bigotimes</code>
$\coprod$	<code>\coprod</code>	$\bigsqcup$	<code>\bigsqcup</code>	$\oplus$	<code>\bigoplus</code>
$\int$	<code>\int</code>	$\bigvee$	<code>\bigvee</code>	$\uplus$	<code>\biguplus</code>
$\oint$	<code>\oint</code>	$\bigwedge$	<code>\bigwedge</code>		

---

```
\sum_{i=1}^n x_i = x_1 + \cdots + x_n
```

$$\sum_{i=1}^n x_i = x_1 + \cdots + x_n$$



- ▶ Utilisation du paquetage babel avec l'option french

```
\usepackage[frenchb]{babel}
```

- ▶ Césures
- ▶ Ponctuation active
- ▶ Encodage des fontes accentuées :

```
\usepackage[latin1]{inputenc}
```

ou

```
\usepackage[utf8]{inputenc}
```



- ▶ Inclusion d'images sous forme PostScript :

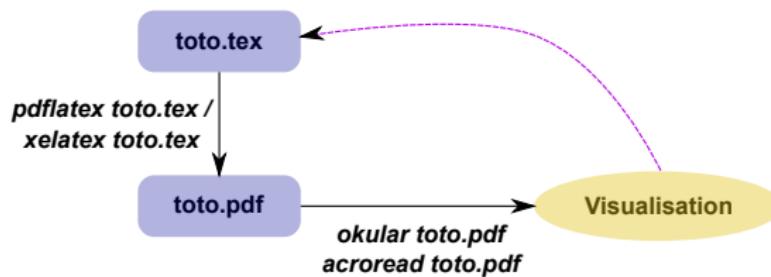
```
\includegraphics [width=5cm, height=64pt]{mon_dessin}
```

- ▶ Utilisation du paquetage graphicx
- ▶ Possibilité de rotation (option angle), mise à l'échelle, ...
- ▶ Extension du fichier : « .pdf », « .png », ...



# Compilation (via PDF)

- ▶ pdflatex
  - ▶ “special” postscript inutilisable (e.g., pstricks)
- ▶ xelatex
  - ▶ Unicode par défaut
  - ▶ Utilisation de fontes vectorielles plus aisée (OpenType, AAT)





# Les erreurs

```
! Undefined control sequence.  
1.254 \rut  
          (8,-2.3){\includegraphics{echecs.ps}}
```

?

---

```
! LaTeX Error: \begin{raggedright} on input line 278 ended by \end{toto}.
```

```
See the LaTeX manual or LaTeX Companion for explanation.  
Type H <return> for immediate help.  
...
```

```
1.279 \end{toto}
```

?



# Les avertissements (1)

```
Overfull \hbox (34.13577pt too wide) in paragraph at lines 215--215
[] \OT1/pcr/m/n/6 3=\Ethyleneh{1==C;2==C}{1==(y1);2==H;3==H;4==H} []
<chimie.ps> [11] <formule-doc.ps> <formule-tex.ps>
Overfull \hbox (5.94807pt too wide) in paragraph at lines 240--240
[] \OT1/pcr/m/n/8 G(z) =
\biggr)=[]
[12] <echecs.ps> [13] [14]
Overfull \hbox (20.34808pt too wide) in paragraph at lines 286--286
[] \OT1/pcr/m/n/8 ! LaTeX Error: \begin{titi} on input line 278 ended by
\end{toto}. []
```



## Les avertissements (2)

```
\documentclass{article}

\begin{document}
~\\
\end{document}
```

---

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(toto.tex
LaTeX2e <2000/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman, i
talian, portuges, spanish, swedish, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
No file toto.aux.
```

Underfull \hbox (badness 10000) in paragraph at lines 4--5

```
[1] (toto.aux) )
(see the transcript file for additional information)
Output written on toto.dvi (1 page, 212 bytes).
Transcript written on toto.log.
This is dvips(k) 5.86e Copyright 2001 Radical Eye Software (www.radicaleye.com)
' TeX output 2002.09.26:1805' -> toto.ps
<tex.pro><alt-rule.pro><texc.pro><texps.pro>. <cmr10.pfb>[1]
```

## Bibliographie et BibT<sub>E</sub>X



# Ajout d'une bibliographie

- ▶ Utilisation d'un style bibliographique

```
\bibliographystyle{plain}
```

- ▶ Inclusion d'un fichier BibT<sub>E</sub>X contenant la bibliographie

```
\bibliography{ma_biblio}
```

- ▶ Citation d'une entrée bibliographique par sa clé

```
Comme le montre Baumann~\cite{Baumann:88}, il  
apparait évident...
```



# Contenu du fichier BibT<sub>E</sub>X

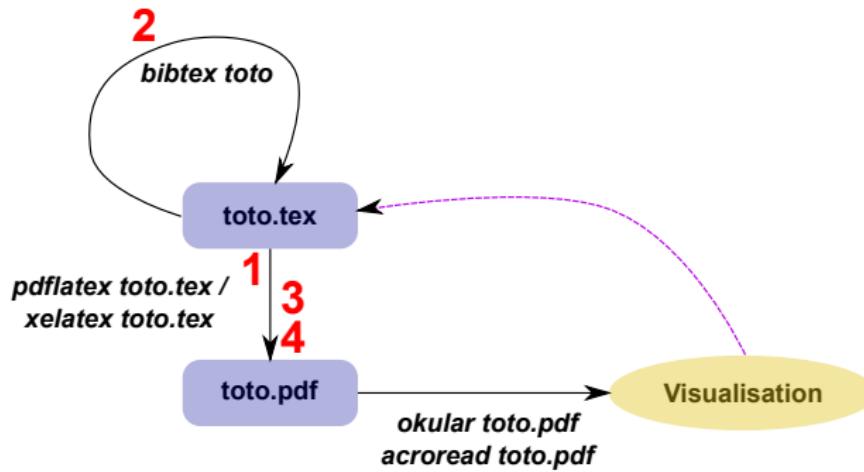
## ► ma\_biblio.bib :

```
@Book{Aho:89,
  author = {A.V. Aho and R. Sethi and J.D. Ullman},
  title = {Compilateurs : Principes, techniques et outils},
  publisher = {InterEditions},
  year = {1989},
}
@Article{Baumann:88,
  author = "Eckart Baumann",
  title = "Optimal centered forms",
  journal = "BIT",
  volume = "28",
  number = "1",
  pages = "80-87",
  year = "1988",
  month = jan
}
```

- ▶ Mise en forme des champs suivant le style bibliographique utilisé
- ▶ Entrées : article, book, booklet, conference, inbook, incollection, inproceedings, manual, masterthesis, misc, phdthesis, proceedings, techreport, unpublished
- ▶ Champs : address, annote, author, booktitle, chapter, crossref, edition, editor, howpublished, institution, journal, key, month, note, number, organization, page, publisher, school, series, title, volume, year



Compilation si utilisation de Bib $\text{\TeX}$  :



## Makeindex



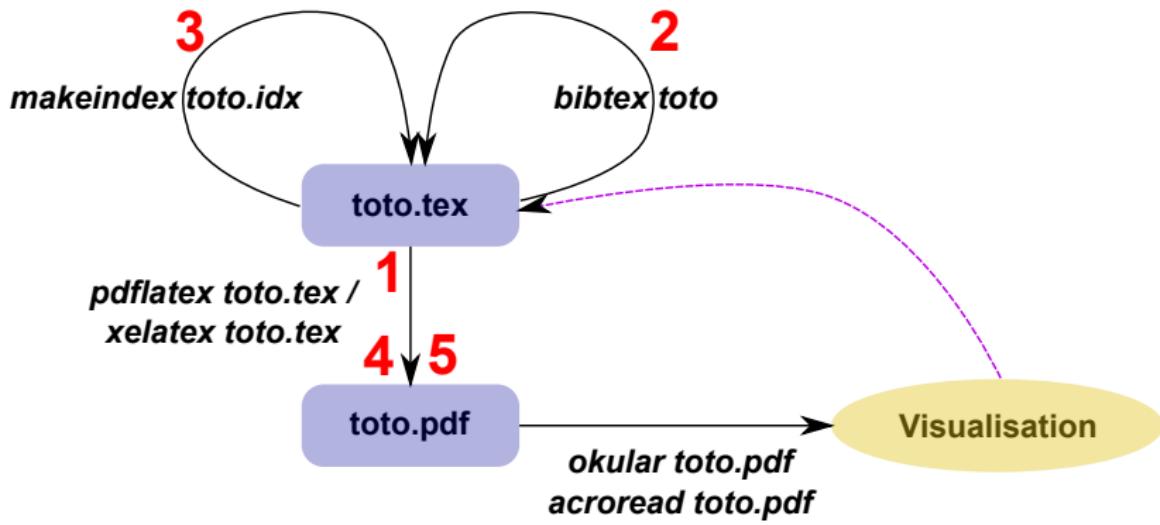
# Création d'un index

- ▶ Utilisation du paquetage `makeidx`
- ▶ Ajout de `\makeindex` dans le *préambule*
- ▶ Appel de la macro `\printindex` à l'endroit où afficher l'index
- ▶ Utilisation de la macro `\index{}` pour ajouter une entrée
- ▶ Création automatique d'un fichier `.idx`

Exemple	Entrée d'index	Commentaire
<code>\index{hello}</code>	hello, 1	Entrée normale
<code>\index{hello!Peter}</code>	Peter, 3	Sous-entrée de « hello »
<code>\index[Sam@\textsl{Sam}]</code>	<i>Sam</i> , 2	Entrée formatée
<code>\index[Jenny textbf]</code>	Jenny, 3	N° page formaté



Compilation si utilisation de makeindex et BibTEX :

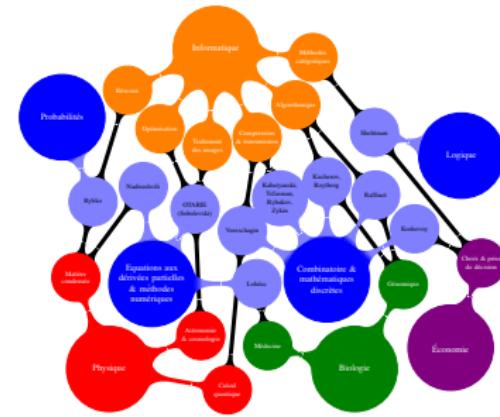
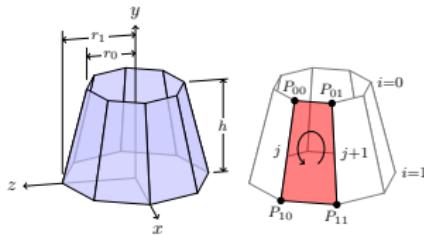


## Aller plus loin avec L<sup>A</sup>T<sub>E</sub>X

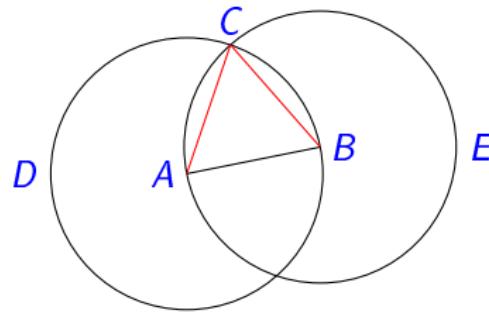


- ▶ Seminar
- ▶ Prosper
- ▶ Beamer

- ▶ Outils extérieurs (`xfig`, `inkscape`, `asymptote`)
  - ▶ Classes (`pstricks`, `pgf/tikz`)



```
\begin{tikzpicture}[every node/.style={font=\small\color{blue}}]
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\node (D) [name path=D,draw,circle through=(B),label=left:$D$] at (A) {};
\node (E) [name path=E,draw,circle through=(A),label=right:$E$] at (B) {};
\path [name intersections={of=D and E}];
\coordinate [label=above:$C$] (C) at (intersection-1);
\draw [red] (A) -- (C);
\draw [red] (B) -- (C);
\end{tikzpicture}
```





# Définition d'une macro

- ▶ Définition de macros avec ou sans arguments

```
\newcommand{nom}{NbArgs}{Définition}
```

```
\newcommand{\valeurAbsolue}[1]{\ensuremath{| #1 |}}  
\newcommand{\itv}[2]{\ensuremath{[#1 \mathrel{\ldots} #2]}}
```

Appel :

```
La valeur absolue \valeurAbsolue{x} de $x$ vaut...  
Considérons l'intervalle \itv{-3}{4}...
```



# Définition d'un environnement

```
\newenvironment{Nom}[nbArgs]{Avant}{Après}
```

- ▶ La valeur des paramètres n'est utilisable que dans *Avant*

Exemple :

```
\newenvironment{exemple}[1]{%
  \bgroup\par\noindent%
  \textbf{Exemple} (#1). \itshape{%
  \par\hrulefill\egroup}}
```

```
\begin{exemple}{Pomme}
Un joli exemple
\end{exemple}
```



## Créer son paquetage

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{macrosAMoi}
  %[05/12/2000 -- Mes p'tites macros]

\RequirePackage{graphicx}

\newcommand{\bla}{...}

\endinput
```

- ▶ À sauver dans un fichier `macrosAMoi.sty`
- ▶ Peut être utilisé dans un fichier `.tex` par

```
\usepackage{macrosAMoi}
```



# Les catcodes (1)

- ▶ Catégorie d'un caractère :

Catégorie	Signification	Exemple
0	début de macro	\
5	fin de ligne	<CR>
10	espace	<ESPACE>
11	lettre	A, B, C, ...
13	caractère actif	~
14	commentaire	%

- ▶ Changement de catégorie :

```
\catcode`Caractère=Catégorie
```



## Les catcodes (2)

---

```
\catcode`:\active  
\def:{!!!}
```

Hello:

```
\catcode`\%12
```

---

Remplacement de ':' par '!!!'

## Environnements pour $\text{\LaTeX}$

URL : <http://www.lyx.org>

LyX: example.lyx (Changed)

Edit Layout Insert Math Options Documents Help

Standard 



Figure: This is a picture of a platypus [fig platy](#)

We can now refer back to the picture as Figure [Ref: fig platy](#). Let's now add a small table:

rocks	minerals
Granite	Mica
Sandstone	Quartz

Now we come to one of LyX's real strengths: mathematical equations. The most beautiful equation in mathematics is  $e^{i\pi} + 1 = 0$ , according to some mathematicians – I'm just a dumb scientist.

$e^{i\pi} + 1 = 0$ . Uglier equations such as the integral of  $1/x$  can be written as

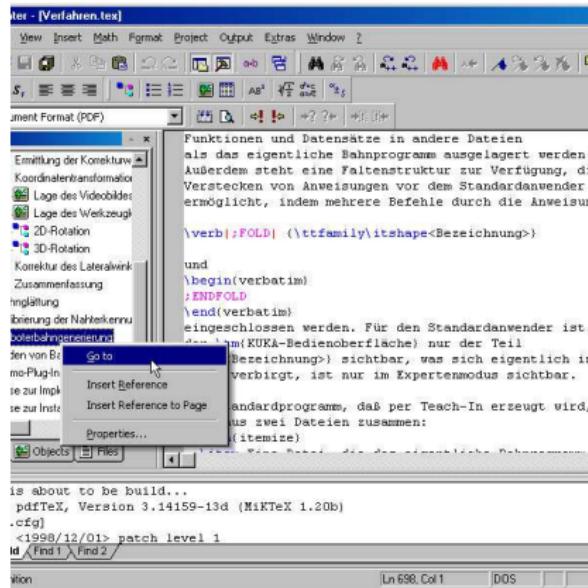
$$\int \frac{dx}{x} = \ln|x| + C$$

/example.lyx (Changed)



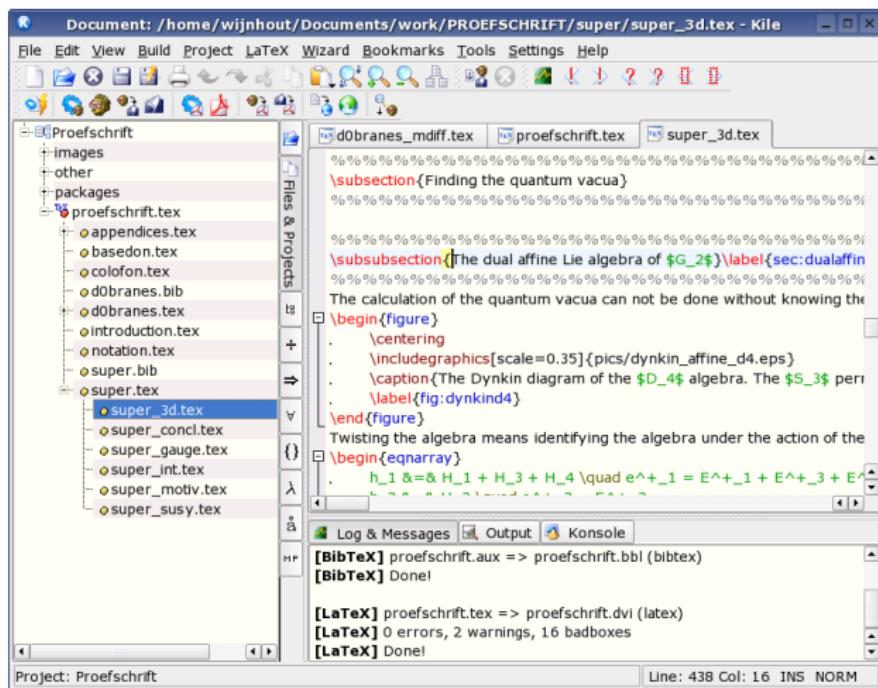
URL : <http://www.texniccenter.org/>

Environnement de travail sous MS Windows :



URL : <http://kile.sf.net/>

Environnement de travail sous KDE :



## Ensemble de macros lisp pour (X)Emacs

- ▶ Complétion sur les noms de macros/environnements
- ▶ Indentation automatique du code
- ▶ Exécution de L<sup>A</sup>T<sub>E</sub>X dans (X)Emacs avec déplacement sur la ligne d'erreur
- ▶ ...

TeXinfo

- ▶ Langage pour la rédaction des documentations GNU (documentation utilisateur)
- ▶ Ensemble de macros au-dessus de  $\text{\TeX}$
- ▶ Contient de nombreuses facilités pour la rédaction :
  - ▶ Environnements standards (exemple, ...)
  - ▶ Macros de définition de fonction, ...
- ▶ Compilation du fichier source en différents formats :
  - ▶ man
  - ▶ HTML
  - ▶ PDF
  - ▶ ...
- ▶ Utilisation standard avec les *autotools*



# Exemple

```
\input texinfo  @c -*-texinfo-*-
@c %**start of header
@setfilename sample.info
@settitle Sample Document
@c %**end of header
@setchapternewpage odd
@ifinfo
This is a short example of a complete Texinfo file.
Copyright 1990 Free Software Foundation, Inc.
@end ifinfo
@titlepage
@sp 10
@comment The title is printed in a large font.
@center @titlefont{Sample Title}
@c The following two commands start the copyright page.
@page
@vskip Opt plus ifilll
Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end titlepage
@node Top, First Chapter, , (dir)
@comment node-name, next, previous, up
@menu
* First Chapter:: The first chapter is the
only chapter in this sample.
* Concept Index:: This index has two entries.
@end menu
@end menu
@node First Chapter, Concept Index, Top, Top
@comment node-name, next, previous, up
@chapter First Chapter
@index Sample index entry
This is the contents of the first chapter.
@index Another sample index entry
Here is a numbered list.
@enumerate
@item
This is the first item.
@item
This is the second item.
@end enumerate
The @code{makeinfo} and @code{texinfo-format-buffer}
commands transform a Texinfo file such as this into
an Info file; and @TeX{} typesets it for a printed
manual.
@node Concept Index, , First Chapter, Top
@comment node-name, next, previous, up
@unnumbered Concept Index
@printindex cp
@contents
@bye
```

## Sécurité, tests et débogage

*The most likely way for the world to be destroyed, most experts agree, is by accident. That's where we come in; we're computer professionals. We cause accidents.* — Nathaniel Borenstein

## Sécurité du logiciel

*The user is going to pick dancing pigs over security every time.* — B. Schneier



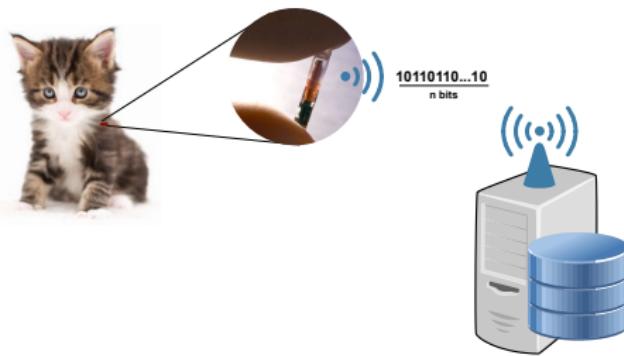
# Is Your Cat Infected with a Computer Virus?



<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>



# Is Your Cat Infected with a Computer Virus?



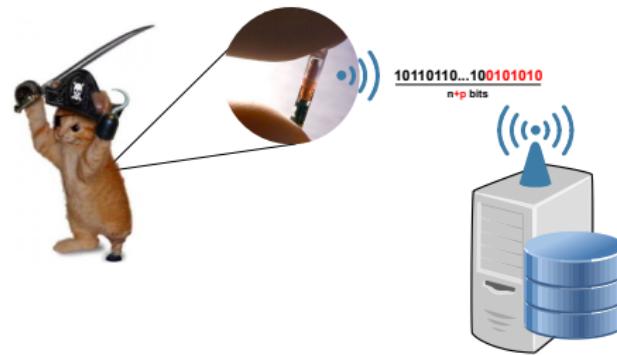
<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>



# Is Your Cat Infected with a Computer Virus?



<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>

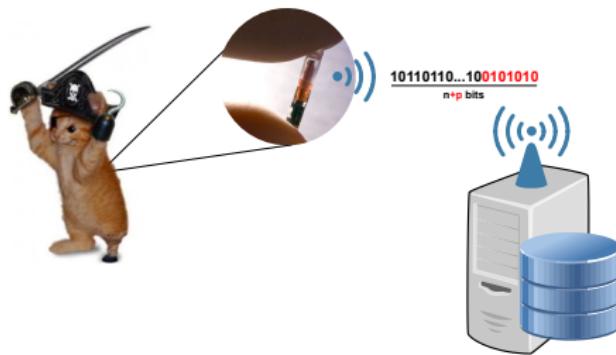




# Is Your Cat Infected with a Computer Virus?



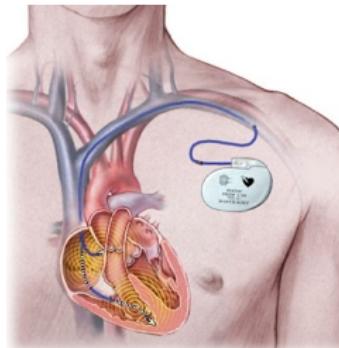
<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>



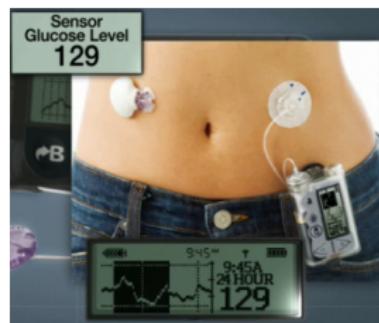
*Is Your Cat Infected with a Computer Virus?* M. R. Rieback, B. Crispo, A. S. Tanenbaum. PERCOM '06 Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, 2006.



# Le crime parfait



Pacemaker



Pompe à insuline



Pacemaker



Pompe à insuline

*Pacemakers and Implantable Cardiac Defibrillators : Software Radio Attacks and Zero-Power Defenses.* D. Halperin et al., Proceedings of the IEEE Symposium on Security and Privacy, 2008.

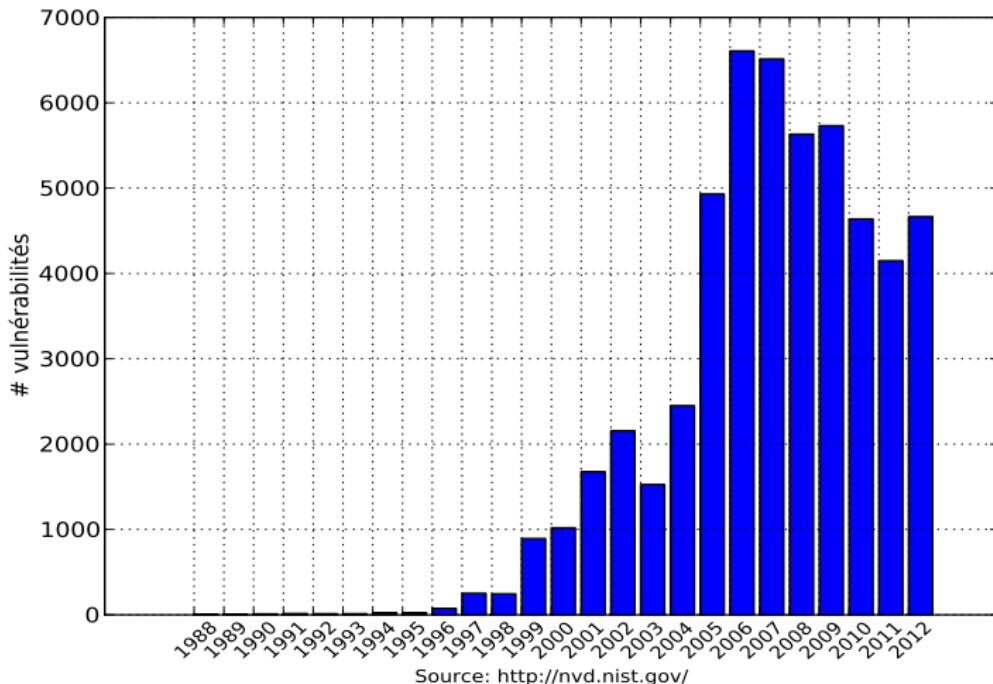


Les programmes informatiques contrôlent tous les aspects critiques de la vie :

- ▶ Banques : DAB/GAB, sites web
- ▶ Industrie nucléaire (*Stuxnet*)
- ▶ Automobiles (*Self-Driving Cars*)
- ▶ Aviation (*Fly-by-wire*)
- ▶ Médecine (*pacemakers*, pompes à insuline—**bug Hospira Symbiq**—, ...)
- ▶ Téléphonie (*Greek Watergate*)
- ▶ Machines à voter (*Hursti Hack*)
- ▶ ...



# Progression des rapports de « vulnérabilités »



Source: <http://nvd.nist.gov/>

- CERT (*Computer emergency response team*)
- NVD/NIST (*National Vulnerability Database*)



- ▶ Virus
- ▶ Vers (*standalone*)
- ▶ Chevaux de Troie
- ▶ Keyloggers
- ▶ Rootkits (dont *firmware rootkits*)
- ▶ Spyware

Exploitation de « vulnérabilités » des logiciels/systèmes  
d'exploitation



*Companies spend millions of dollars on firewalls, encryption and secure access devices, and it's money wasted, because none of these measures address the weakest link in the security chain.*

— Kevin Mitnick

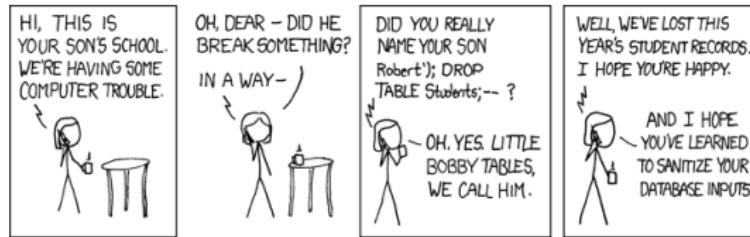
*If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.*

— Bruce Schneier

- ▶ Antivirus
- ▶ Pare-feux
- ▶ Analyseurs statiques de code (*lint*, option `-Wall` du compilateur, ...)
- ▶ Éducation
  - ▶ Développeurs
  - ▶ Utilisateurs



# Danger 1 : les interactions avec l'utilisateur



- ▶ Valeur non prévue (crash, ...)
- ▶ Valeurs malveillantes



# Exemple

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
    bool mdp_ok = false;
    char mot_de_passe[15];

    cout << "Mot de passe ? ";
    cin >> mot_de_passe;

    if (!strcmp(mot_de_passe, "Sesame")) {
        mdp_ok = true;
    }

    // [...]

    if (mdp_ok) {
        cout << "Mot de passe correct" << endl;
    } else {
        cout << "Mot de passe incorrect" << endl;
    }
}
```

Mot de passe ? avoine  
Mot de passe incorrect

Mot de passe ? Sesame  
Mot de passe correct



## Exemple

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
    bool mdp_ok = false;
    char mot_de_passe[15];

    cout << "Mot de passe ? ";
    cin >> mot_de_passe;

    if (!strcmp(mot_de_passe, "Sesame")) {
        mdp_ok = true;
    }

    // [...]

    if (mdp_ok) {
        cout << "Mot de passe correct" << endl;
    } else {
        cout << "Mot de passe incorrect" << endl;
    }
}
```

Mot de passe ? avoine  
Mot de passe incorrect

Mot de passe ? Sesame  
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAAA  
Mot de passe correct



# Exemple

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
    bool mdp_ok = false;
    char mot_de_passe[15];

    cout << "Mot de passe ? ";
    cin >> mot_de_passe;

    if (!strcmp(mot_de_passe, "Sesame")) {
        mdp_ok = true;
    }

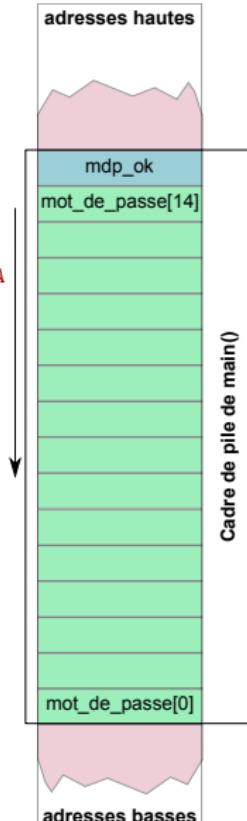
    // [...]

    if (mdp_ok) {
        cout << "Mot de passe correct" << endl;
    } else {
        cout << "Mot de passe incorrect" << endl;
    }
}
```

Mot de passe ? avoine  
Mot de passe incorrect

Mot de passe ? Sesame  
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAAA  
Mot de passe correct



Problème : Buffer overflow

Pile d'appels



## Exemple de *buffer overflow* (2)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *str = "Une jolie chaîne constante";
    char str1[128];
    char str2[16];

    strncpy(str2,str,16); // off-by-one
    // [...]
}
```



## Exemple de *buffer overflow* (2)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *str = "Une jolie chaîne constante";
    char str1[128];
    char str2[16];

    strncpy(str2,str,16); // off-by-one
    // [...]
}
```

- ▶ Oubli de la place prise par le terminateur '\0'



## Exemple de *buffer overflow* (2)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *str = "Une jolie chaîne constante";
    char str1[128];
    char str2[16];

    strncpy(str2,str,15);
    // [...]
}
```



## Exemple de *buffer overflow* (2)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *str = "Une jolie chaîne constante";
    char str1[128];
    char str2[16];

    strncpy(str2,str,15);
    // [...]
}
```

- ▶ Pas de terminateur '\0' ajouté à str2 par strncpy()
- ▶ Toute manipulation en lecture de str2 débordera sur str1



## Exemple de *buffer overflow* (3)

```
#include <stdio.h>
#include <string.h>

void pbOV(char *bigbuf)
{
    short int szbig = strlen(bigbuf);
    char tinybuf[100];

    if (szbig < 100) {
        strcpy(tinybuf, bigbuf);
    }
    // [...]
}

int main(void)
{
    char bigbuf[40000];
    // Lecture dans un fichier
    // et stockage dans 'bigbuf'
    pbOV(bigbuf);
}
```



## Exemple de *buffer overflow* (3)

```
#include <stdio.h>
#include <string.h>

void pbOV(char *bigbuf)
{
    short int szbig = strlen(bigbuf);
    char tinybuf[100];

    if (szbig < 100) {
        strcpy(tinybuf, bigbuf);
    }
    // [...]
}
```

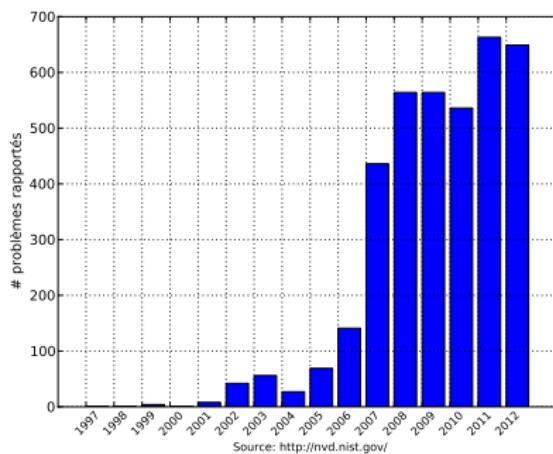
```
int main(void)
{
    char bigbuf[40000];
    // Lecture dans un fichier
    // et stockage dans 'bigbuf'
    pbOV(bigbuf);
}
```

- ▶ Possibilité d'un *integer overflow* dans `szbig`
- ▶ Copie dans `tinybuf` avec *buffer overflow*



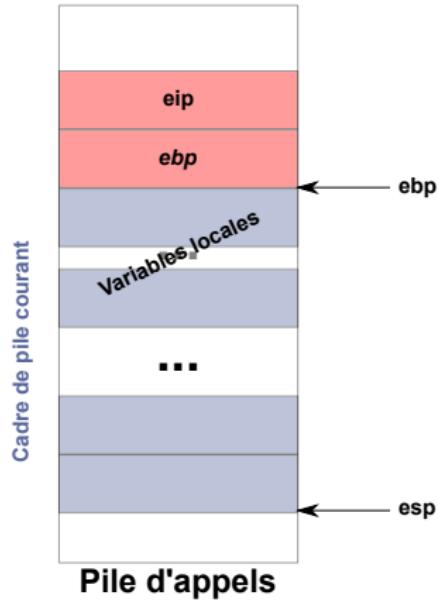
## Buffer overflow

- ▶ Trou de sécurité courant
- ▶ Ecriture en mémoire au-delà de l'espace spécifiquement alloué
- ▶ Risques :
  - ▶ Crash du programme
  - ▶ Exécution inattendue de code
  - ▶ Prise de contrôle du programme





# Exploitations du *buffer overflow*



- ▶ Modification de eip pour
  - ▶ pointer sur du code dans la pile
  - ▶ *return-to-libc*
  - ▶ Saut inattendu dans le code
  - ▶ Crash
- ▶ Lecture d'une valeur dans la pile
- ▶ Modification d'une valeur dans la pile



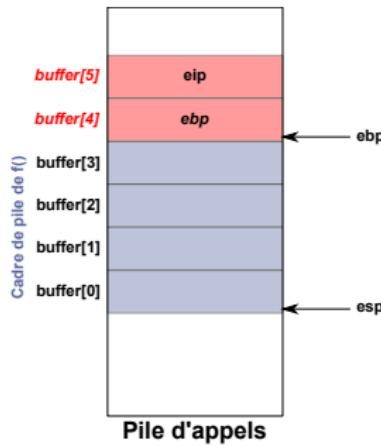
# Exemple de *buffer overflow* (4)

```
(gdb) disas main
Dump of assembler code for function main:
0x0804841d <+0>:    55                      push   ebp
0x0804841e <+1>:    89 e5                   mov    ebp,esp
0x08048420 <+3>:    83 e4 f0                 and    esp,0xffffffff
0x08048423 <+6>:    83 ec 20                 sub    esp,0x20
0x08048426 <+9>:    c7 44 24 1c 00 00 00 00  mov    DWORD PTR [esp+0xic],0x0
0x0804842e <+17>:   e8 d9 ff ff ff           call   0x804840c <f>
0x08048433 <+22>:   c7 44 24 1c 01 00 00 00  mov    DWORD PTR [esp+0xic],0x1
0x0804843b <+30>:   8b 44 24 1c                 mov    eax,DWORD PTR [esp+0xic]
0x0804843f <+34>:   89 44 24 04                 mov    DWORD PTR [esp+0x4],eax
0x08048443 <+38>:   c7 04 24 f8 84 04 08  mov    DWORD PTR [esp],0x80484f8
0x0804844a <+45>:   e8 a1 fe ff ff           call   0x80482f0 <printf@plt>
0x0804844f <+50>:   b8 00 00 00 00           mov    eax,0x0
0x08048454 <+55>:   c9                      leave 
0x08048455 <+56>:   c3                      ret    
End of assembler dump.
```



# Exemple de *buffer overflow* (4)

```
(gdb) disas main
Dump of assembler code for function main:
0x0804841d <+0>:    55                      push   ebp
0x0804841e <+1>:    89 e5                   mov    ebp,esp
0x08048420 <+3>:    83 e4 f0                   and    esp,0xffffffff
0x08048423 <+6>:    83 ec 20                   sub    esp,0x20
0x08048426 <+9>:    c7 44 24 1c 00 00 00 00  mov    DWORD PTR [esp+0xic],0x0
0x0804842e <+17>:   e8 d9 ff ff ff           call   0x804840c <f>
0x08048433 <+22>:   c7 44 24 1c 01 00 00 00  mov    DWORD PTR [esp+0xic],0x1
0x0804843b <+30>:   8b 44 24 1c                   mov    eax,DWORD PTR [esp+0xic]
0x0804843f <+34>:   89 44 24 04                   mov    DWORD PTR [esp+0x4],eax
0x08048443 <+38>:   c7 04 24 f8 84 04 08  mov    DWORD PTR [esp],0x80484f8
0x0804844a <+45>:   e8 a1 fe ff ff           call   0x80482f0 <printf@plt>
0x0804844f <+50>:   b8 00 00 00 00           mov    eax,0x0
0x08048454 <+55>:   c9                      leave
0x08048455 <+56>:   c3                      ret
End of assembler dump.
```



- ▶ Affiche « 0 »
- ▶ Manipulation du registre eip



- ▶ Prudence dans les calculs de taille de chaînes
- ▶ Utilisation de types de haut niveau dès que possible (e.g., `std::string` vs. « `char []` »)
- ▶ Nettoyage des entrées de l'utilisateur
- ▶ Test des programmes (*fuzzer*)
- ▶ Aide du compilateur (canaris, ...)



# Entrée inattendue

*Any fool can use a computer. Many do.*

— Ted Nelson

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int abandon;
    printf("Abandonner le formatage du disque dur (0:Non, 1:Oui) ? ");
    scanf("%d",&abandon);
    if (!abandon) {
        // Formatage
    }

    return EXIT_SUCCESS;
}
```



*Any fool can use a computer. Many do.*

— Ted Nelson

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int abandon;
    printf("Abandonner le formatage du disque dur (0:Non, 1:Oui) ? ");
    scanf("%d",&abandon);
    if (!abandon) {
        // Formatage
    }

    return EXIT_SUCCESS;
}
```

Et si l'utilisateur ne lit pas correctement la consigne ?

- ▶ « o » formate le disque
- ▶ « n » formate le disque