

Concepts et outils de développement

Frédéric Goualard

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



- ▶ 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

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 (≈ 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

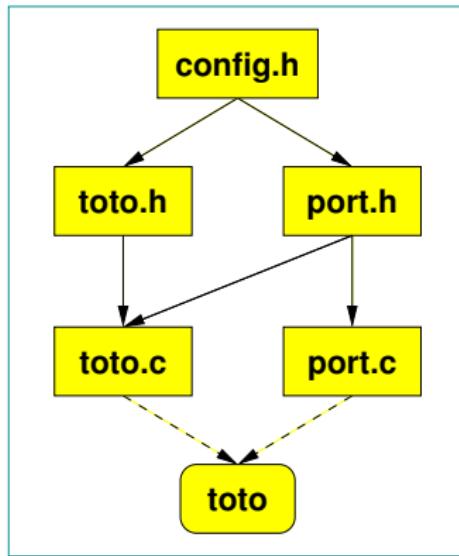
Commande à exécuter n° 2

...

- ▶ *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*
- ▶ Exécution d'un **Makefile** en deux phases (voir plus loin)



Make (2)



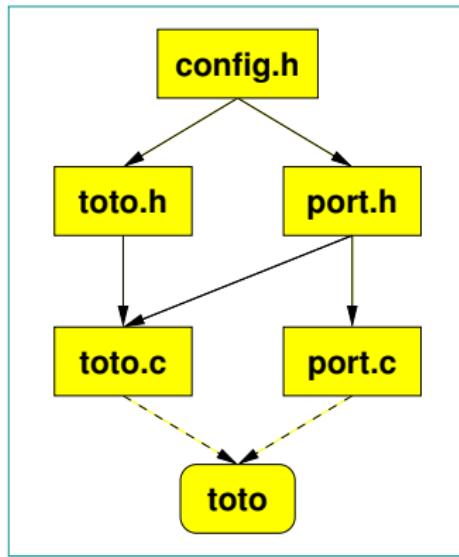
Makefile:

```
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)



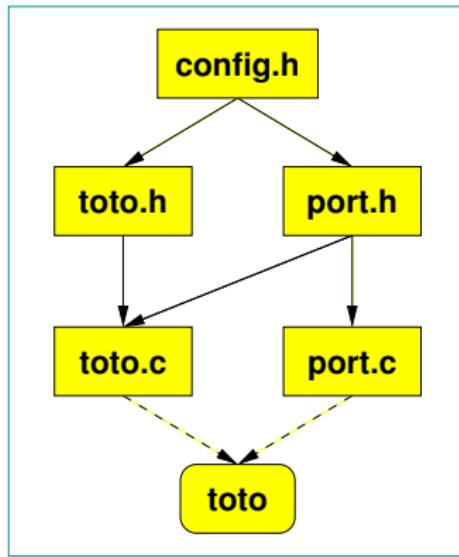
Makefile:

```
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)



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 ou immédiate</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 = -lg.gnu
lib_icc = -lix86

toto: $(objects)
ifeq ($(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)
```



- ▶ 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)
```



- ▶ Metaprogrammation

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

- ▶ Affichage de messages

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

- ▶ Évaluation

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



- ▶ Phase 1 :
 - ▶ Affectations immédiates (`:=`)
 - ▶ Affectations des variables « paresseuses » apparaissant dans un contexte immédiat
 - ▶ Parsing des directives conditionnelles
 - ▶ Expansion des cibles et prérequis des règles
- ▶ Phase 2 :
 - ▶ Affectations différées (`=`)
 - ▶ Lecture de la recette des règles exécutées

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



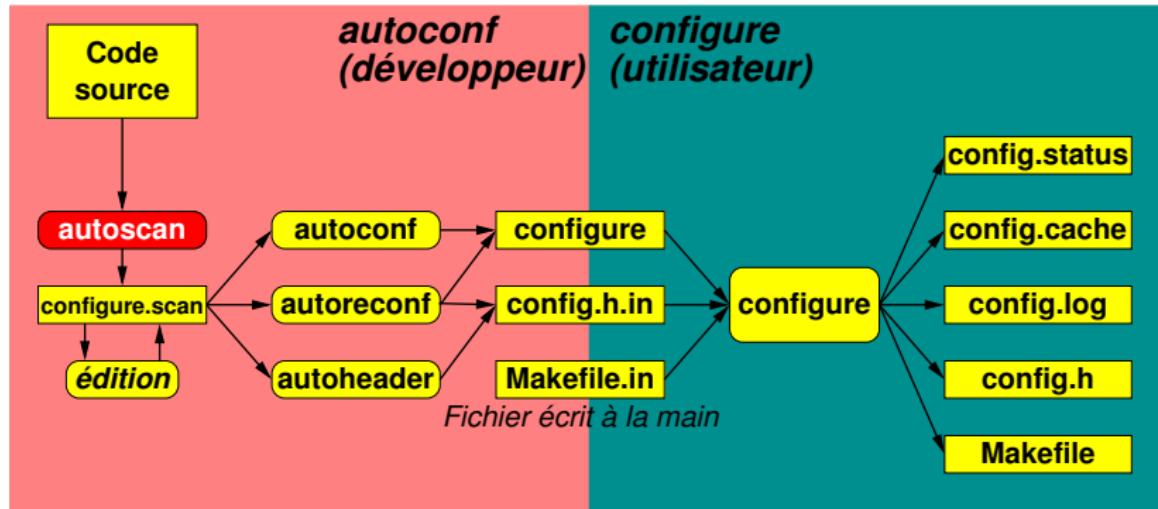
Configuration et installation d'une application/librairie :

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

Procédure standardisée pour tout paquetage GNU



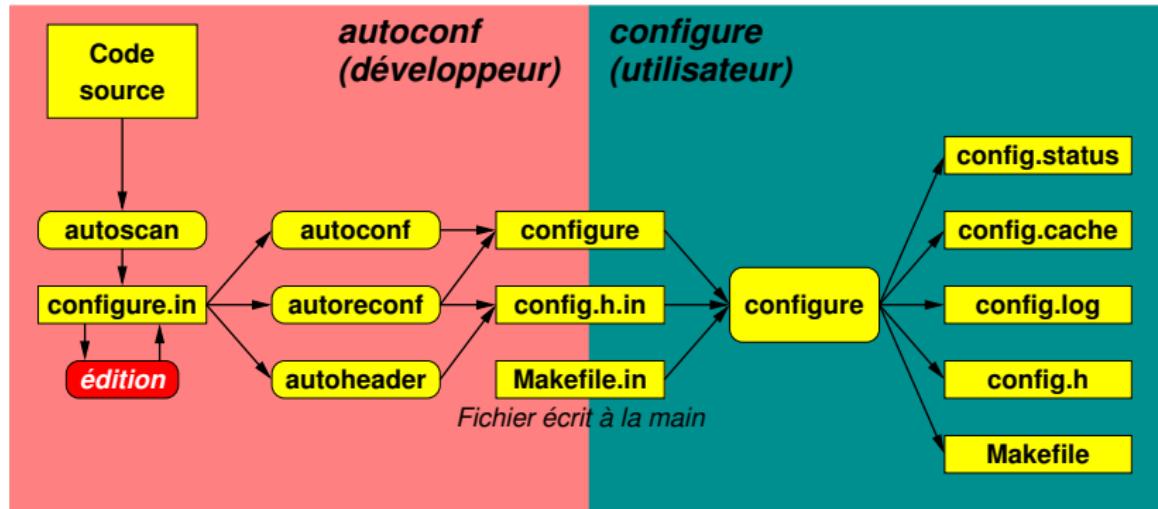
Chaîne des actions



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



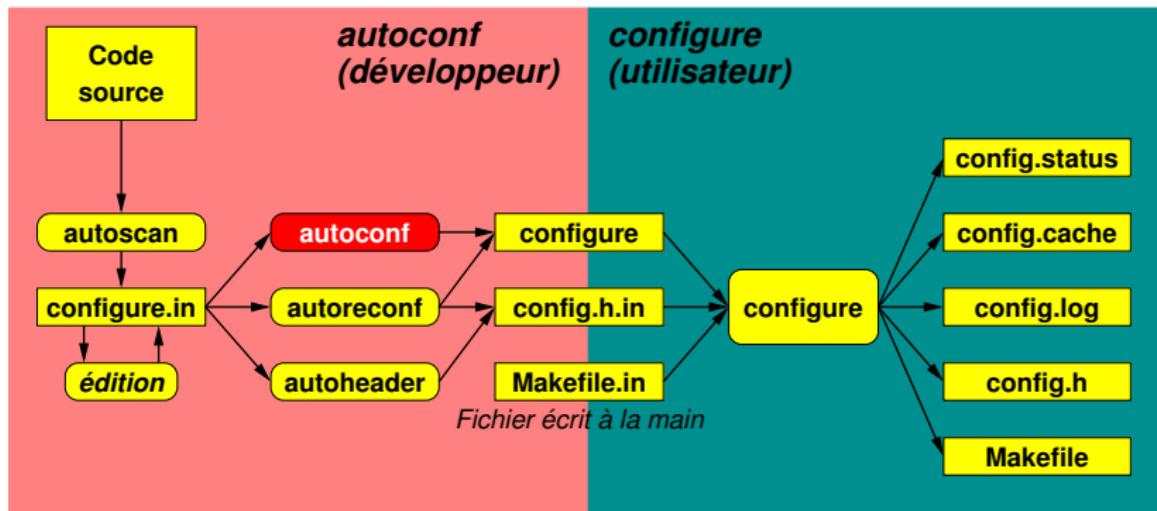
Chaîne des actions



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



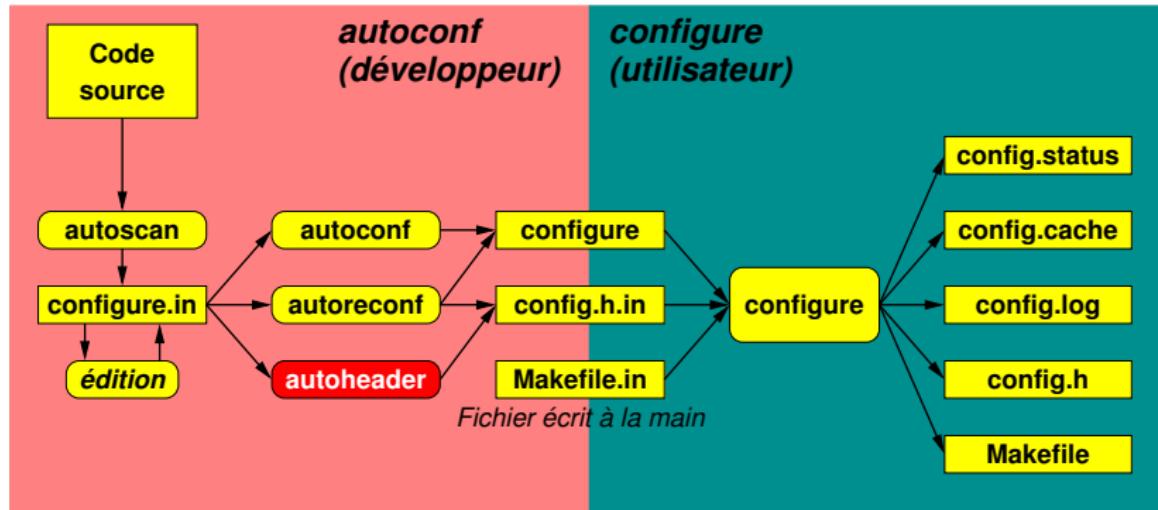
Chaîne des actions



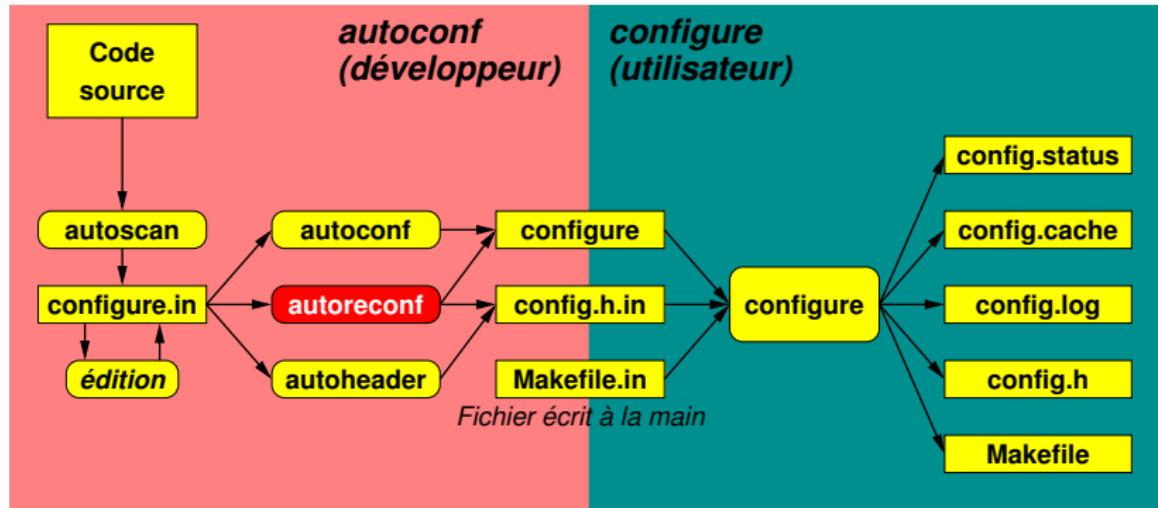
autoconf : création d'un fichier de scripts sh à partir de l'expansion des macros M4 de `configure.in` (utilisation du fichier local `aclocal.m4`)



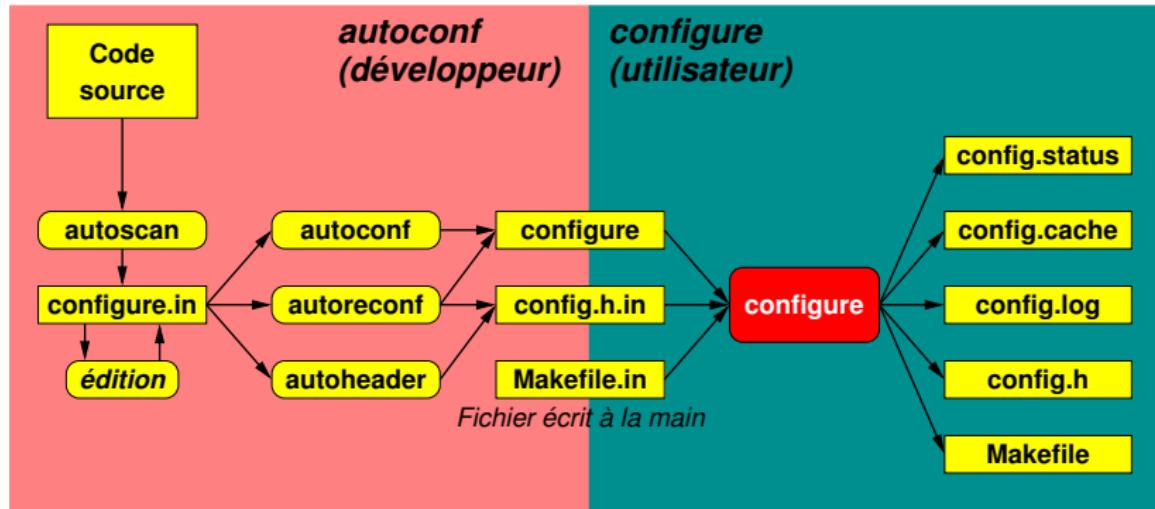
Chaîne des actions



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



autoreconf : réinvocation de autoconf et autoheader lorsque nécessaire



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)
```



- ▶ 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

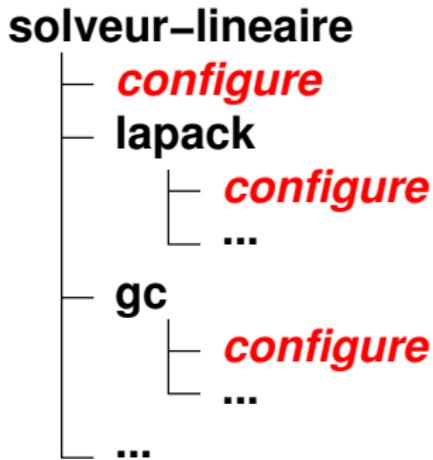
<i>CC</i>	Nom du compilateur C
<i>CFLAGS</i>	Paramètres du compilateur C
<i>CXX</i>	Nom du compilateur C++
<i>CXXFLAGS</i>	Paramètres du compilateur C++
<i>LDFLAGS</i>	Paramètres de l'éditeur de liens
<i>CPPFLAGS</i>	Paramètres du préprocesseur
...	

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

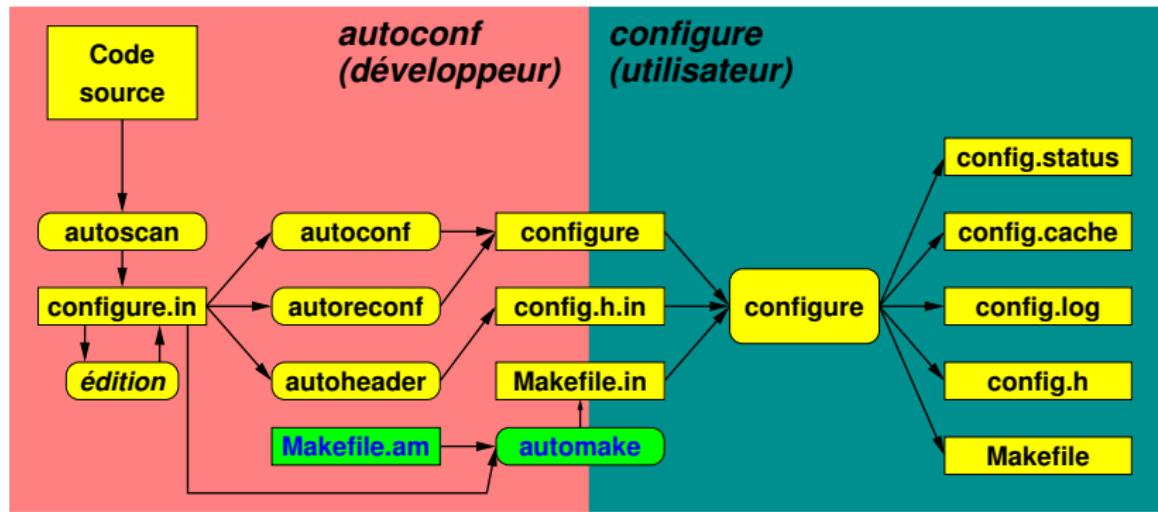
Automake



- ▶ 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-répertoires 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/*.* 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/*.* $(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_LT_LIBRARIES = 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^e 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;
}
```



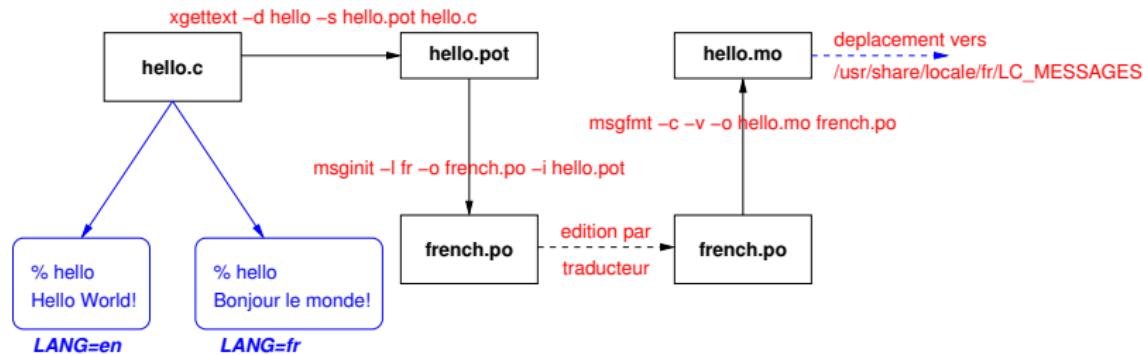
Utilisation de gettext()

```
#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/ras.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, svnserve, ...)

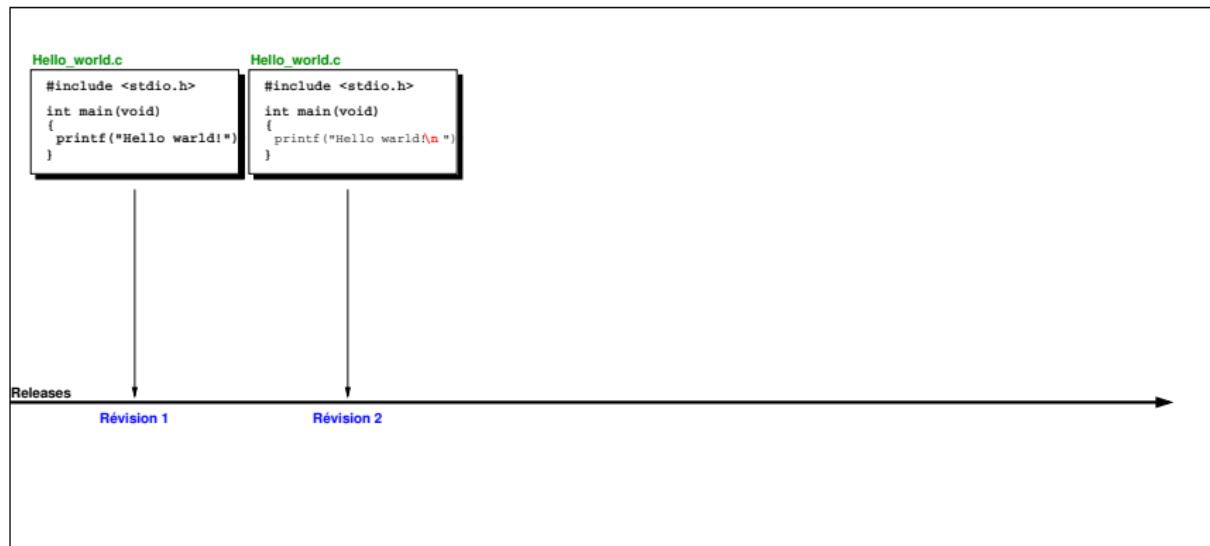


Scenario 1 : gestion des « versions »



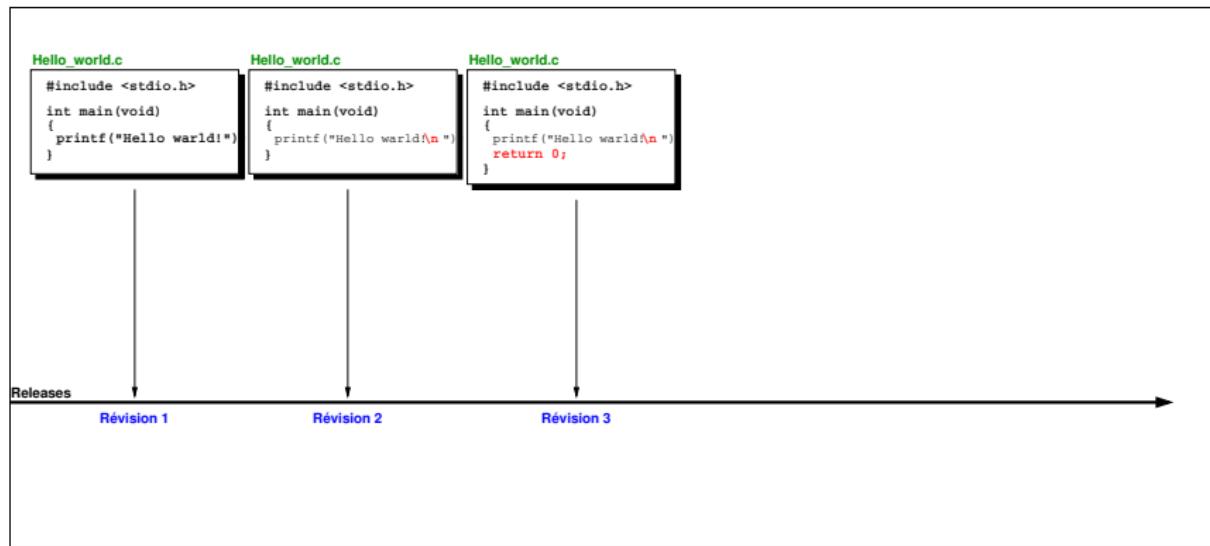


Scenario 1 : gestion des « versions »



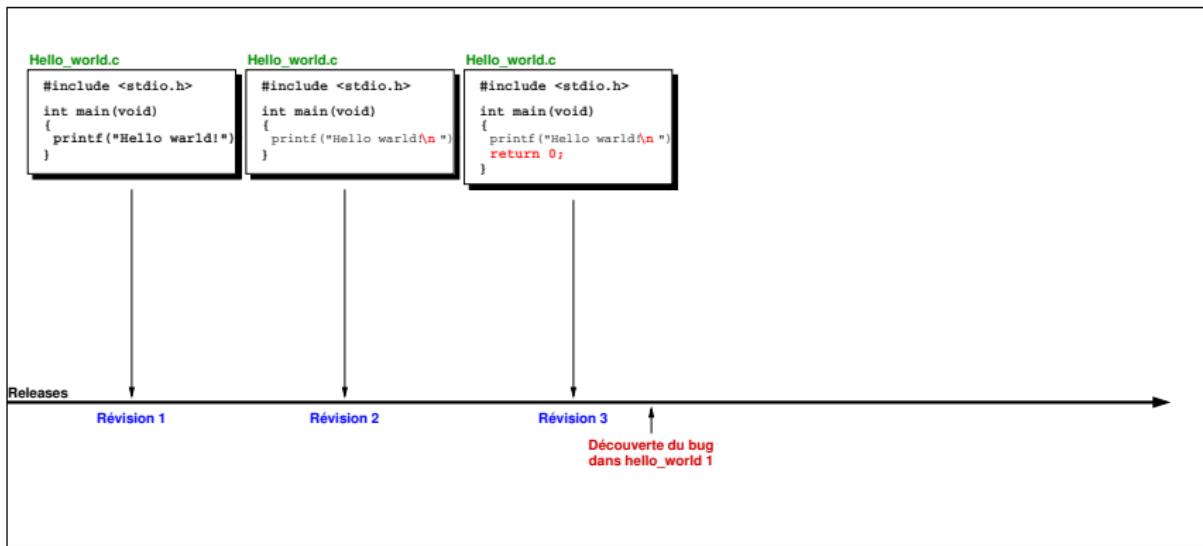


Scenario 1 : gestion des « versions »



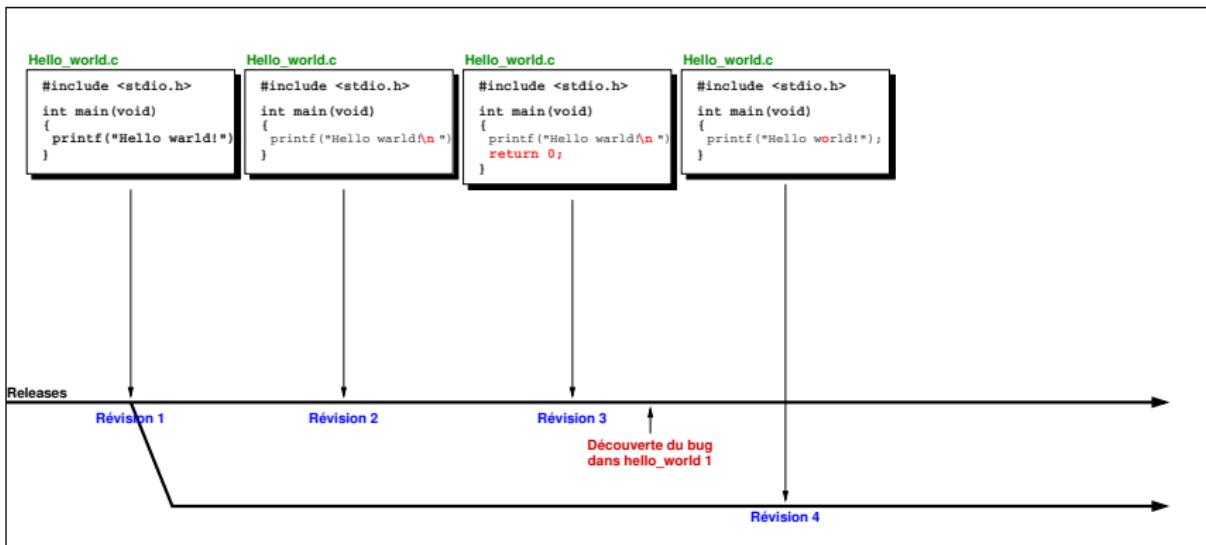


Scenario 1 : gestion des « versions »



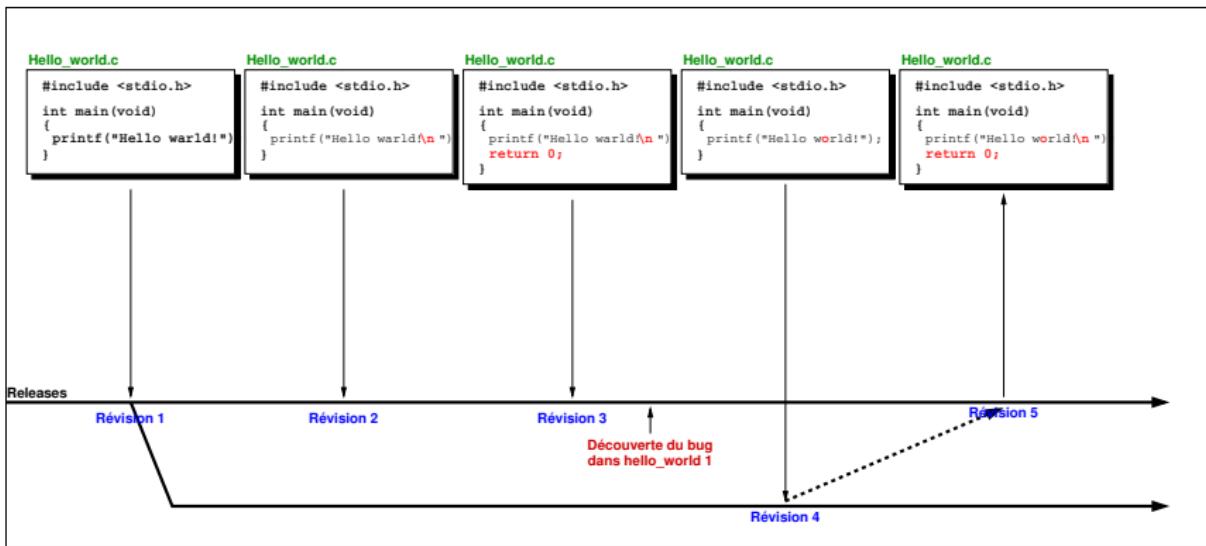


Scenario 1 : gestion des « versions »





Scenario 1 : gestion des « versions »





Version vs. révision

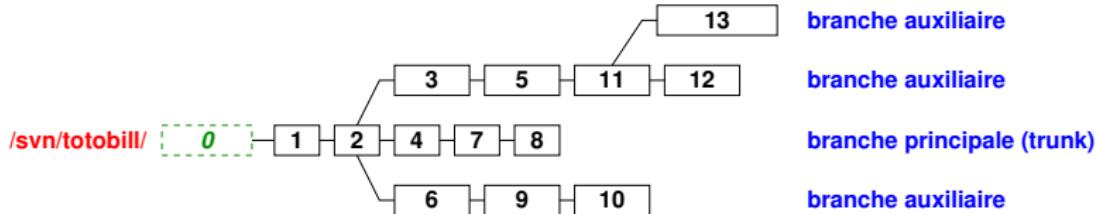
Version. Indice attribué à une application/librairie lors de sa *release*



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

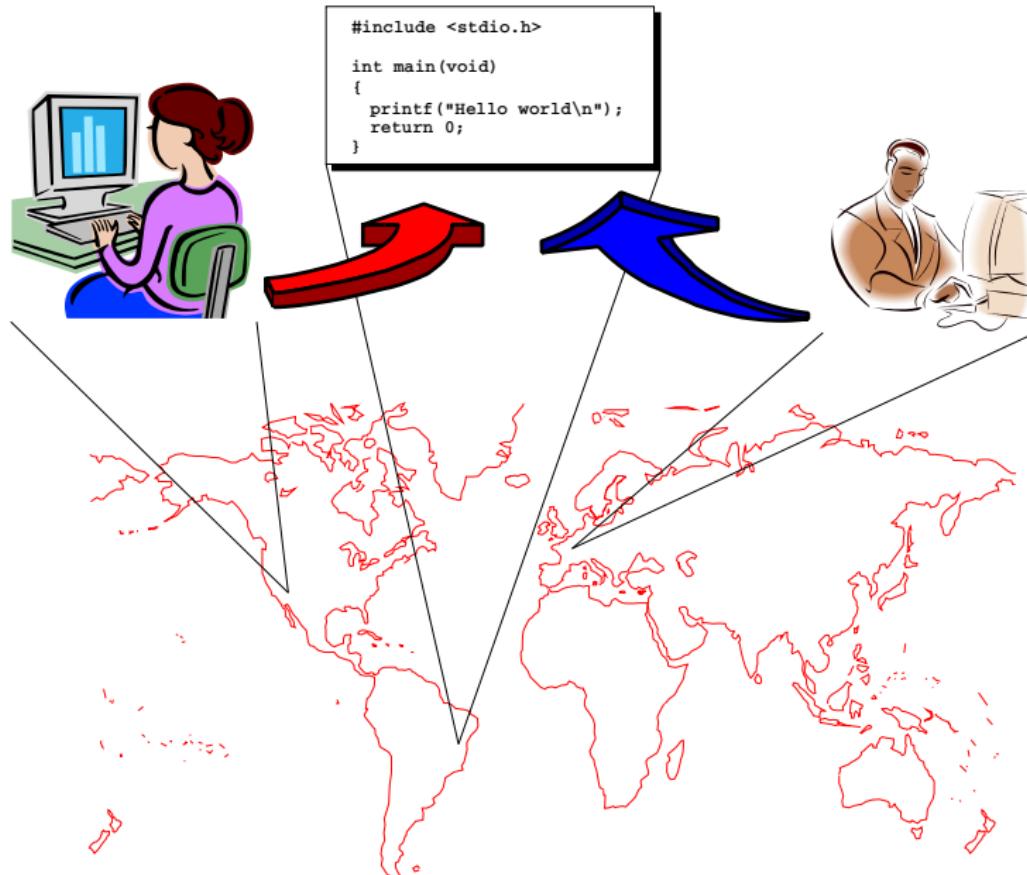
Mineur. Change lorsque les modifications apportées n'affectent pas la compatibilité

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



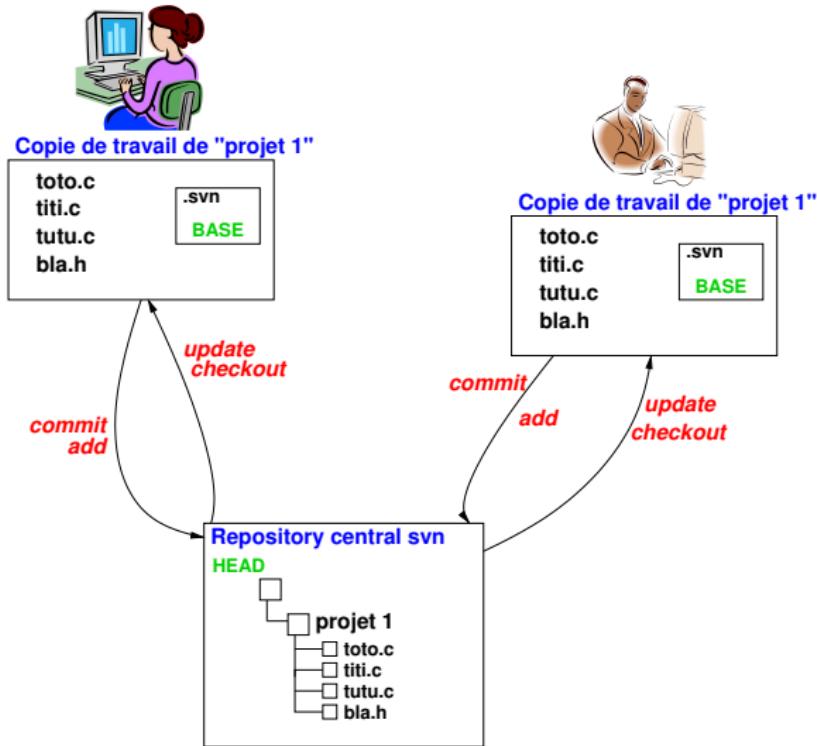


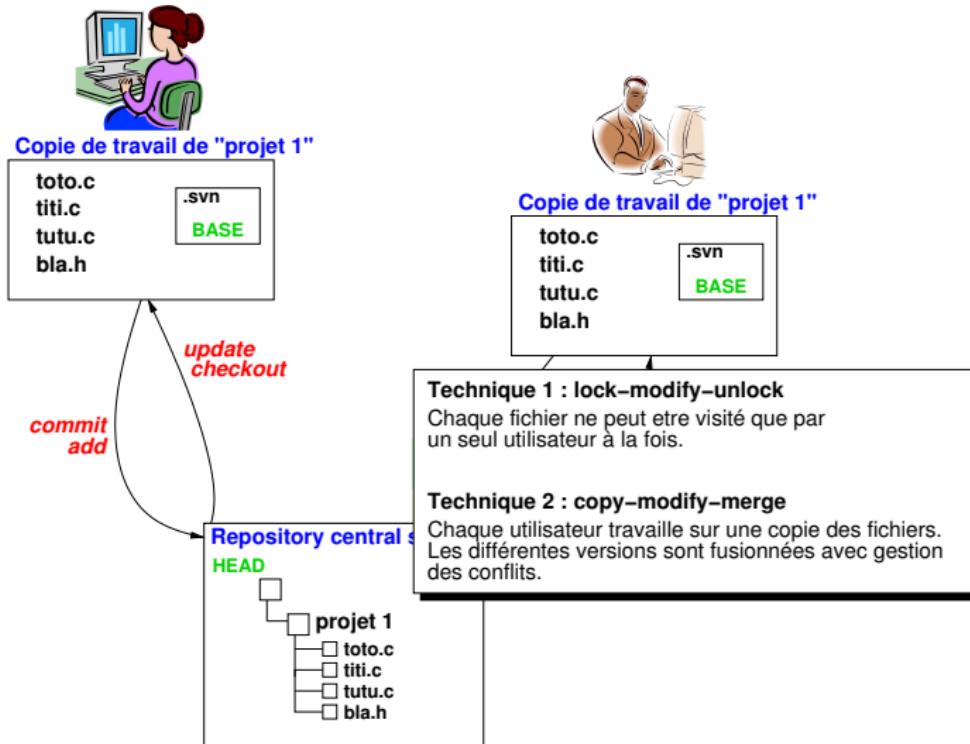
Scenario 2 : le travail collaboratif





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

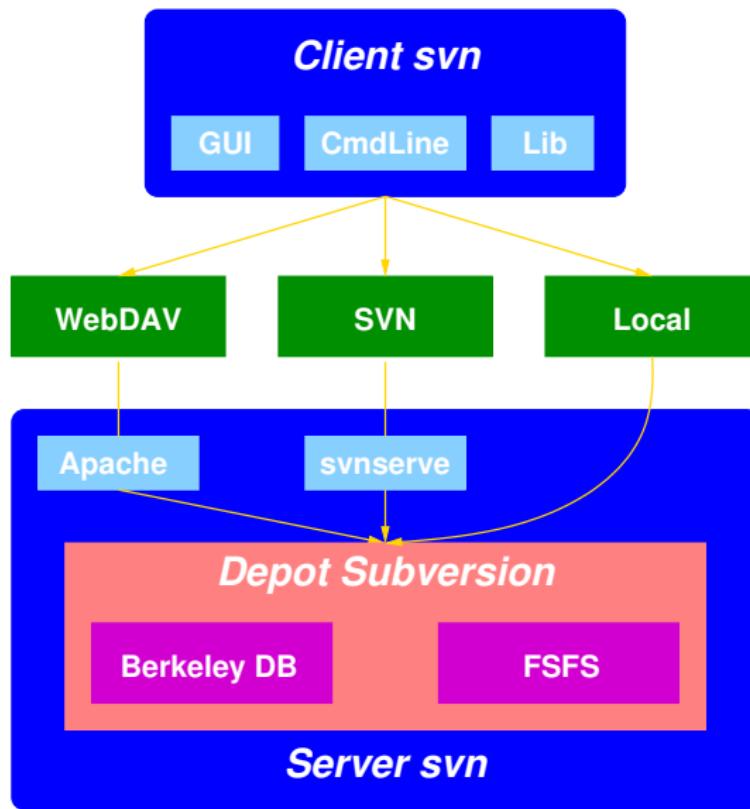
HEAD. Révision la plus récente dans le dépôt

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
```



Les répertoires .svn

- ▶ Un répertoire « .svn » dans chaque sous-répertoire contrôlé par svn (mais, un seul « .svn » à partir de v. 1.7)
- ▶ 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



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 (détection 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^{OLDREV}* : fichier de BASE
- ▶ *file.r^{NEWREV}* : 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



```
[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)

- ▶ Métadonnées 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

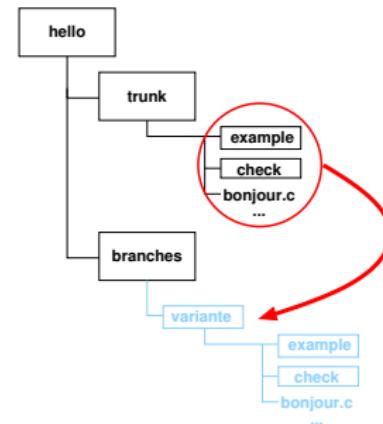
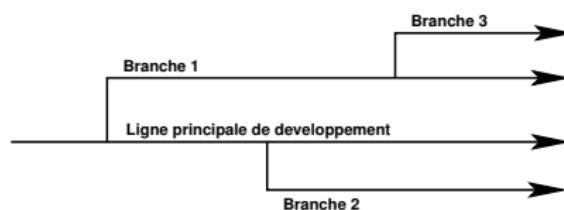
```
% svn diff --changelist langues-romanes
Index: bonjour.c
=====
--- bonjour.c      (revision 13)
+++ bonjour.c      (working copy)
[...]
Index: hola.c
=====
--- hola.c        (revision 15)
+++ hola.c        (working copy)
[...]
```

- ▶ 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 :

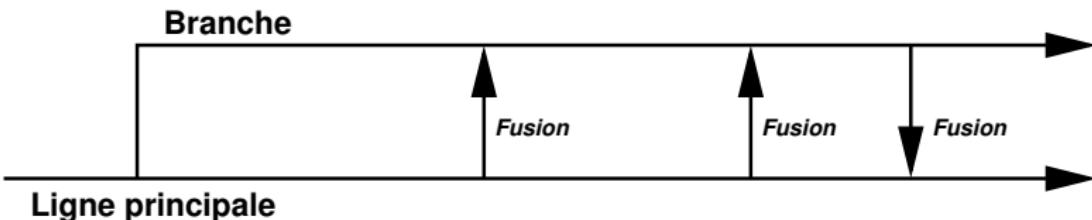
```
% svn delete svn://almighty/hello/branches/variante \
>     -m "Destruction de 'variante'"
```

```
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

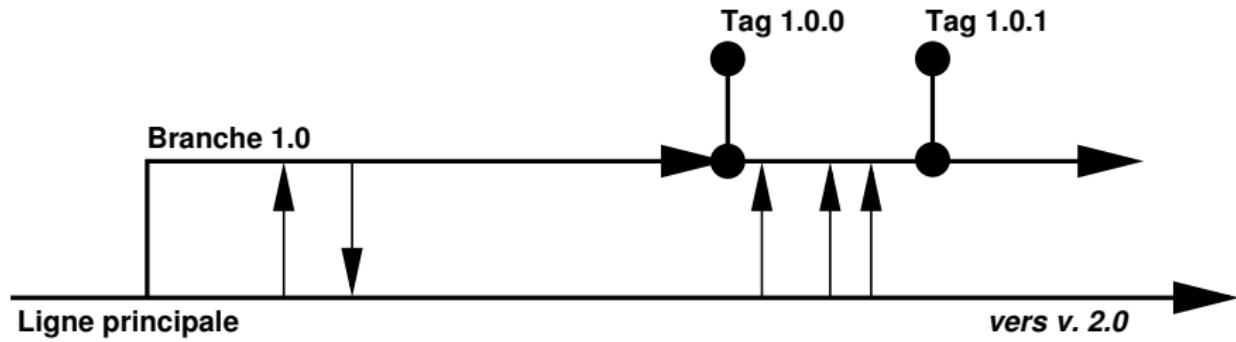
```
% svn copy svn://almighty/hello/trunk  
%           svn://almighty/hello/tags/release-1-0\  
%           -m "Première livraison du produit"  
  
Committed revision 22.
```

- ▶ 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



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

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

- ▶ Authentification avec conf/svnserve.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 svnserve simultanément



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

tortoiseSVN. Interface SVN/Windows Explorer.

(<http://tortoisessvn.tigris.org/>)

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

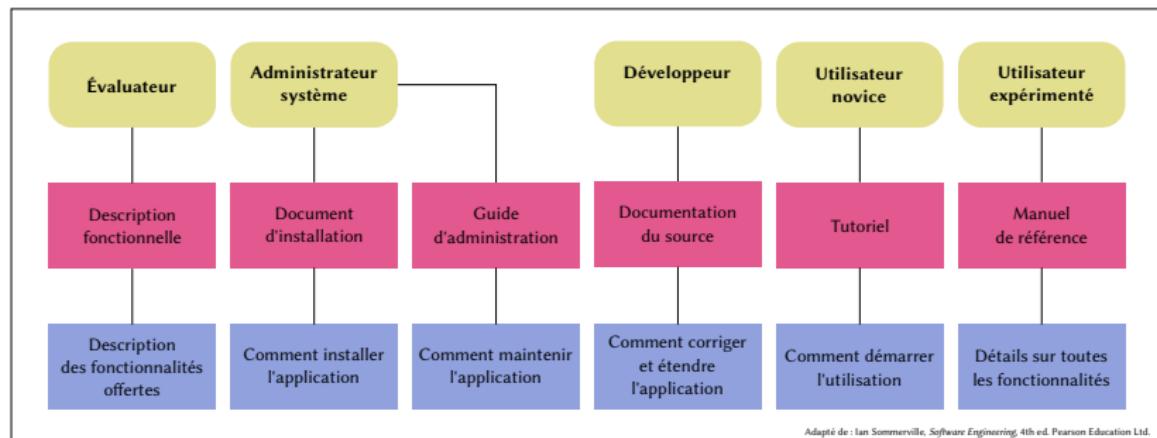
webSVN. Accès au dépôt SVN par le web

(<http://www.websvn.info/>)

Documentation



Différentes documentations



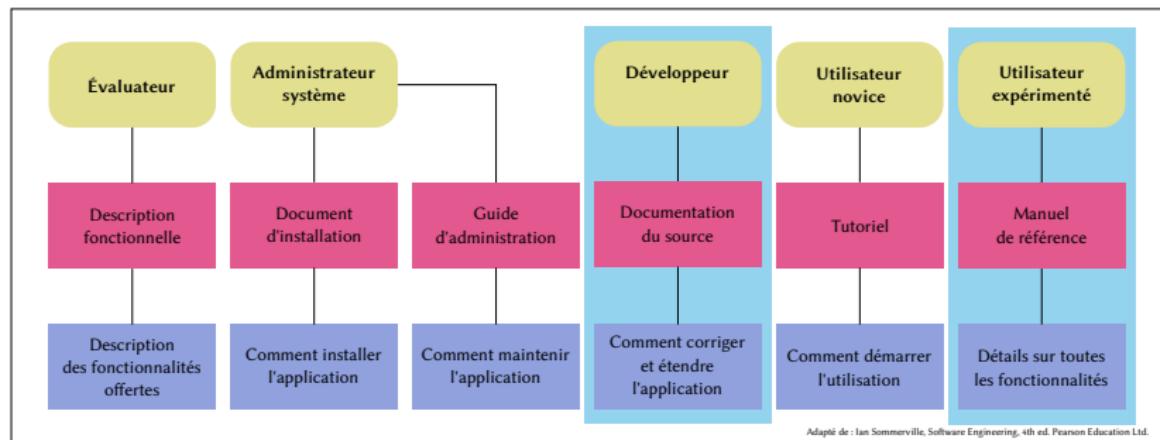
Documentation du code source

*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





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;
// Appartent 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

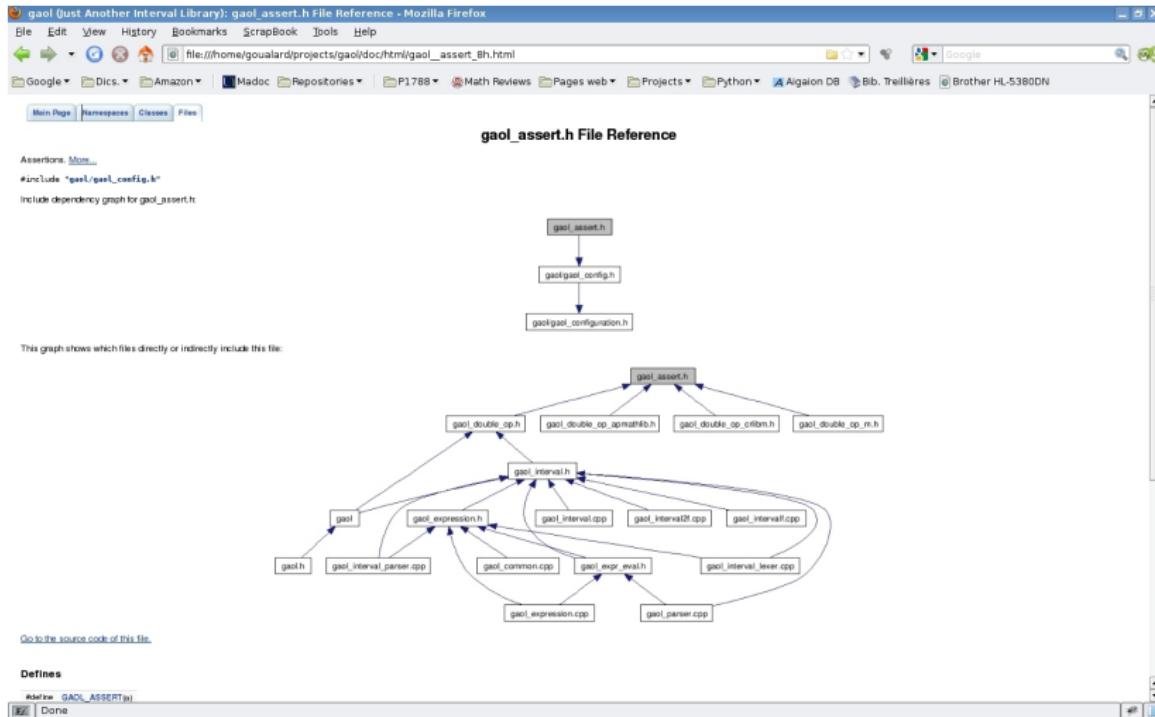
- ▶ **Simplicité** du code (*do it right [then do it fast]*)



- ▶ 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





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

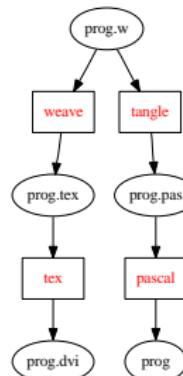
Donald Knuth, 1981 : un programme doit pouvoir être de la « littérature »

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.

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

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

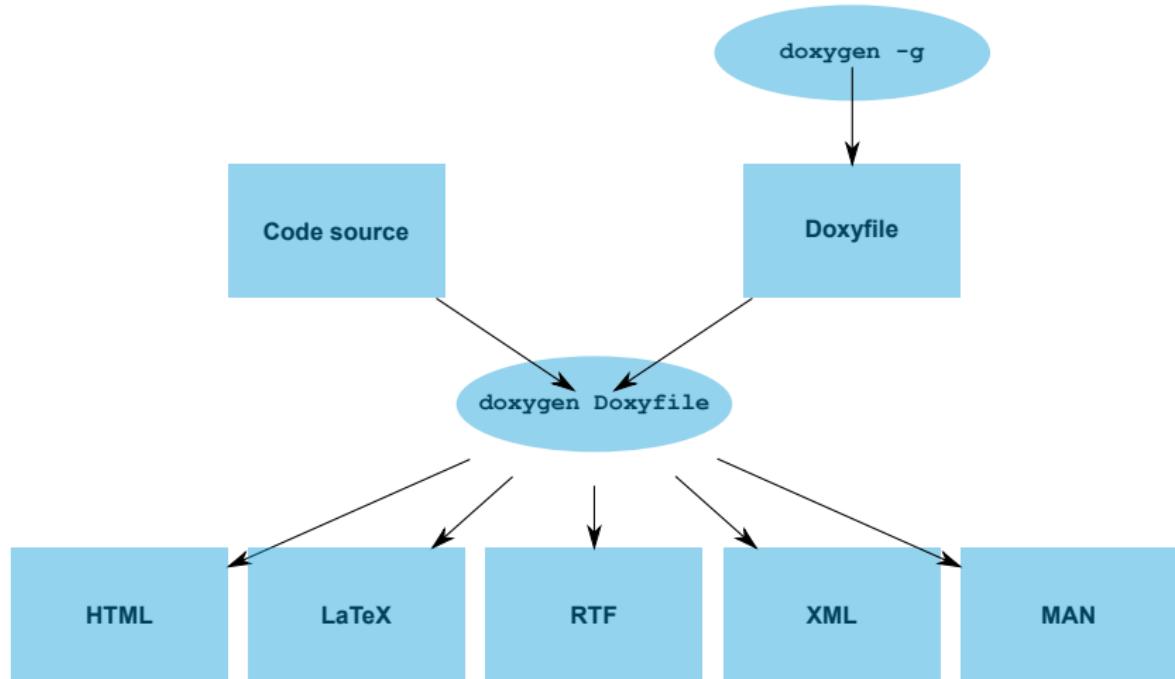
*/
struct __GAOL_PUBLIC__ interval_format {
    enum format_t {
        bounds,
        center,
        hexa,
        agreeing
    };
};

/*
 \brief test for evenness
 \warning d should not be \f$ \pm \infty \f$
 */
bool feven(double d)
{
    // FIXME: check whether 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^AT_EX, 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);
```



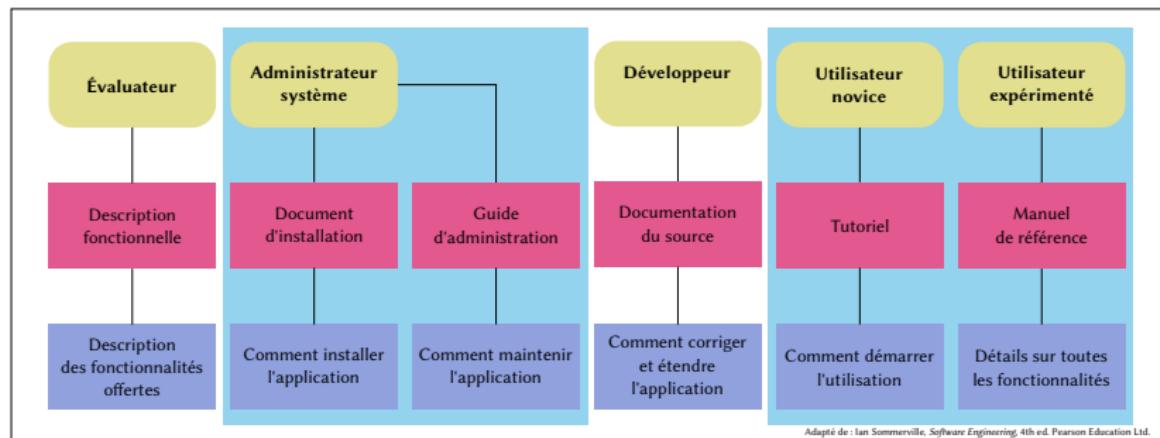
Limites des outils du type « Doxygen »

- ▶ 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 : \LaTeX
 - ▶ Avant-propos : pourquoi \LaTeX ?
 - ▶ Liens et références utiles
 - ▶ Historique de \TeX et \LaTeX
 - ▶ Possibilités de \LaTeX : une visite guidée
 - ▶ Bases de \LaTeX
 - ▶ Bibliographie et Bib \TeX
 - ▶ Makeindex
 - ▶ Pour aller plus loin avec \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 (*vs.* 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**



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^AT_EX ? (2)

- ▶ Outils de composition (troff, lout, T_EX) 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^AT_EX :
 - ▶ Sur-couche de T_EX simplifiant la rédaction
 - ▶ Outil très largement répandu



Liens et références utiles

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_E.* 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/?wakka=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 \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



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

Point important : compatibilité ascendante garantie

Les possibilités de \LaTeX

Visite guidée

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

8164062
3032971693
3832795028841971693
383238462643
P
36383238462643
36383238462643

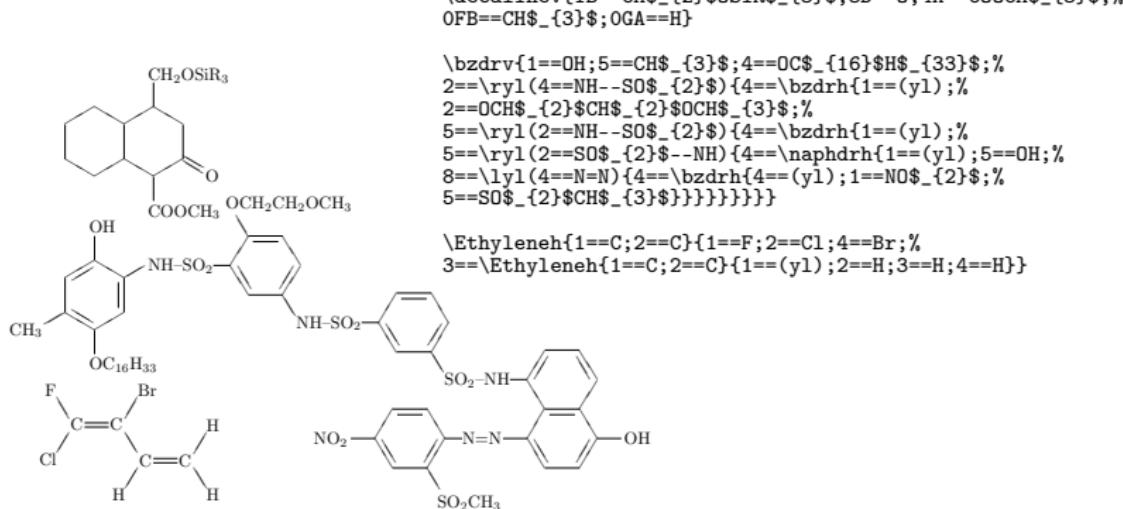


Écriture de partitions de musiques :

```
\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 1\sk\ql n\enotes
\barre
\Notes\ibu0f0\qh0{dgc}\lqp i\enotes
\notes\tbu0\qh0g|\ibbl1j3\qb1j\tbl1\qb1k\enotes
\tempo\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\finextrait
\end{music}
```

Piano

- ▶ Lilypond





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^AT_EX :

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

```
\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}

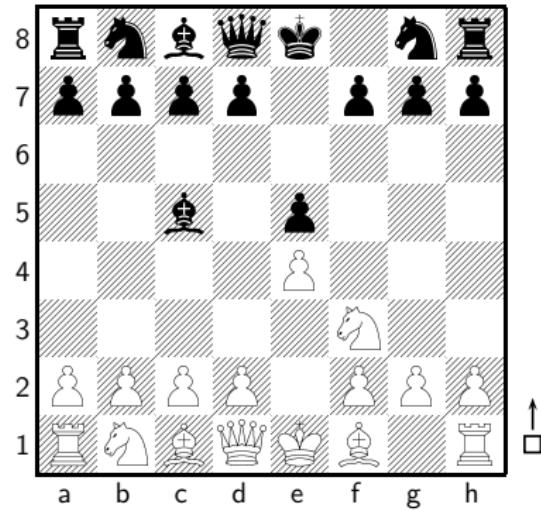
\begin{document}
\begin{sagesilent}
var('x')
f(x)=sin(log(x))+cos(x)^2
\end{sagesilent}
La dérivée de $f(x)\colon sage{f}$$ est $\sage{diff(f(x),x)}$.
\end{document}
```

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



Et bien plus encore...

1 e4 e5 2 Nf3 Bc5

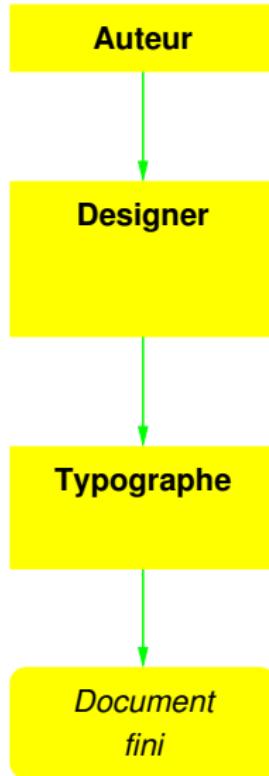


```
\newgame  
\mainline{1. e4 e5 2. Nf3 Bc5}  
\[showboard\]
```

Bases de \LaTeX



Principe de L^AT_EX

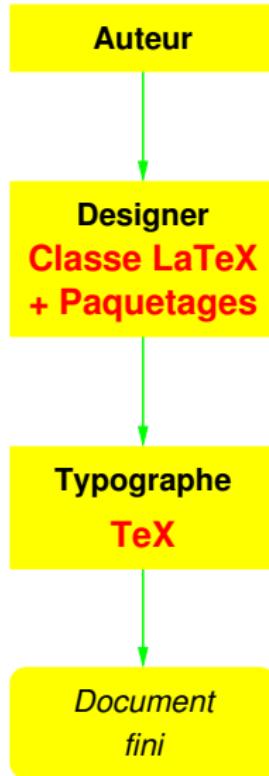


L^AT_EX :

- ▶ 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^AT_EX



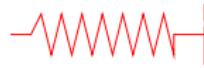
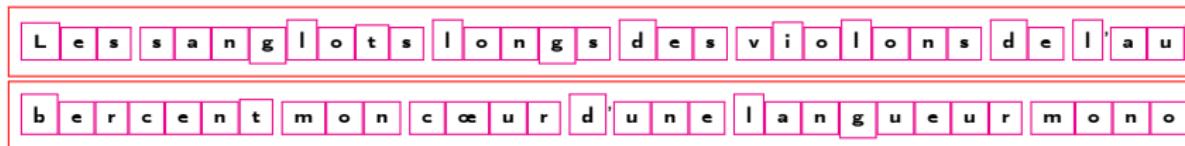
L^AT_EX :

- ▶ 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^AT_EX (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^AT_EX (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\^otel, NO\"EL, na\"{\i}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^LA_TE_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

\TeX 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}
```



- ▶ 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^AT_EX

```
\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}
```



- ▶ 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



Texte *verbatim*

- ▶ Affichage du texte tel quel sans prise en compte des caractères actifs dans L^AT_EX
- ▶ 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 & calé à droite \\
    \cline{2-3}
    Youpi & aglaglagla & 45.5 & yam\\
    \hline
    \multicolumn{3}{c|}{Sur 3 colonnes} & Bla \\
    \hline
\end{tabular}
```

34	pomme	tralala pouet	calé à droite
Youpi	aglaglagla	45.5	yam
Sur 3 colonnes			Bla

► `\multicolumn{#cols agglomérées}{position}{contenu}`



► 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



Éléments flottants (3)

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} % Dans le préambule
```

```
\begin{loi}[Loi de Murphy]\label{loi:murphy}
```

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

```
\end{loi}
```

Loi 1 (Loi de Murphy)

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



Symboles mathématiques (1)

α	<code>\alpha</code>	θ	<code>\theta</code>	\circ	<code>\circ</code>	τ	<code>\tau</code>
β	<code>\beta</code>	ϑ	<code>\vartheta</code>	π	<code>\pi</code>	υ	<code>\upsilon</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	ϕ	<code>\phi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	φ	<code>\varphi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	χ	<code>\chi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ψ	<code>\psi</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>	ω	<code>\omega</code>
η	<code>\eta</code>	ξ	<code>\xi</code>				
Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<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>	\uplus	<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^AT_EX Symbol List](#)



Symboles mathématiques (3)

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

* 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 caractères accentués :

```
\usepackage[encodage caractères]{inputenc} % entrée
```

(e.g., latin1, utf8, ...)

```
\usepackage[encodage fontes]{fontenc} % sortie
```

(e.g., OT1, T1, TS3, ...)



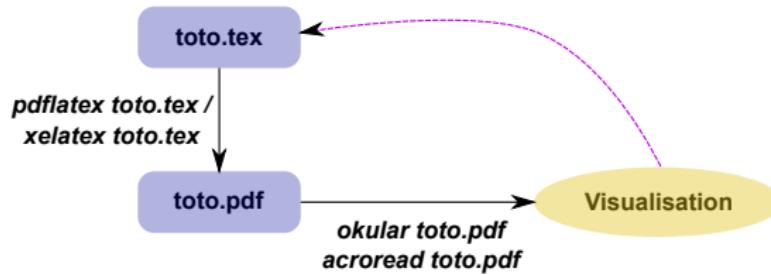
```
\includegraphics [width=5cm,height=64pt]{mon_dessin}
```

- ▶ Utilisation du paquetage `graphicx`
- ▶ Possibilité de rotation (option `angle`), mise à l'échelle, ...
- ▶ Format du fichier : dépend du moteur \LaTeX
 - ▶ « `.pdf` », « `.png` », ..., avec $\text{PDF}\text{\LaTeX}$, $\text{Xe}\text{\LaTeX}$
 - ▶ « `.ps` », « `.eps` », avec \LaTeX



Compilation (via PDF)

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





```
! 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==(yl);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_EX



Ajout d'une bibliographie

- ▶ Utilisation d'un style bibliographique

```
\bibliographystyle{plain}
```

- ▶ Inclusion d'un fichier BibT_EX contenant la bibliographie

```
\bibliography{ma_biblio}
```

- ▶ Citation d'une entrée bibliographique par sa clé

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

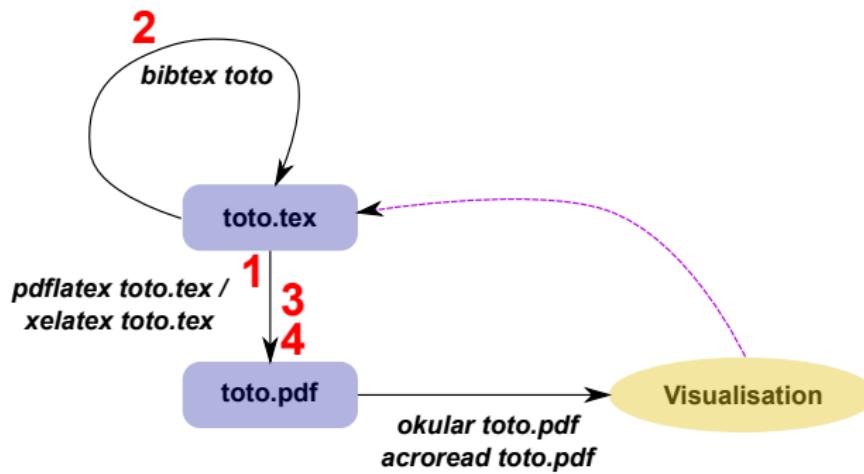


► ma_biblio.bib :

```
@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 BibTEX :



Makeindex



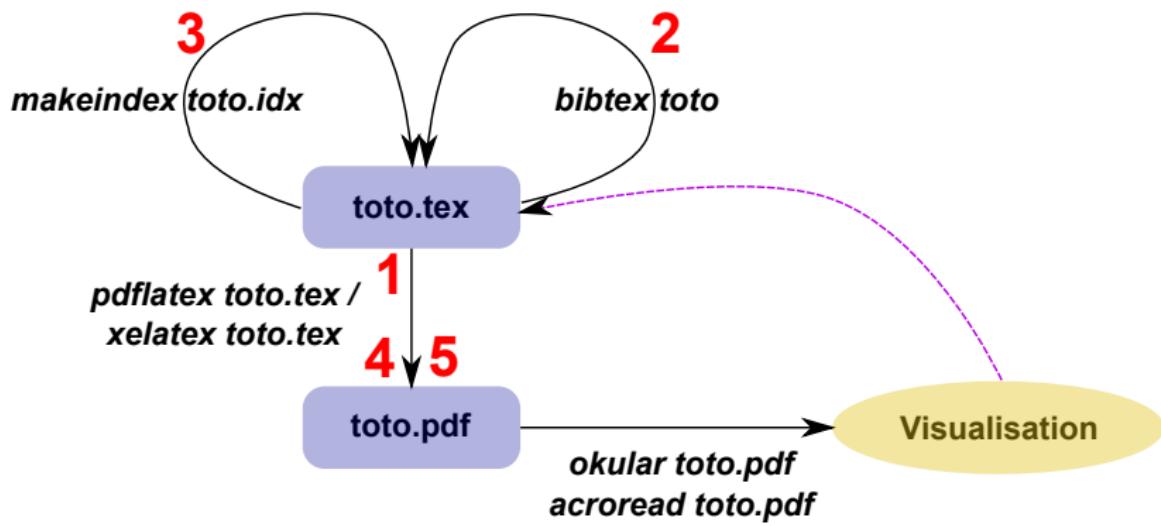
Création d'un index

- ▶ 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 :



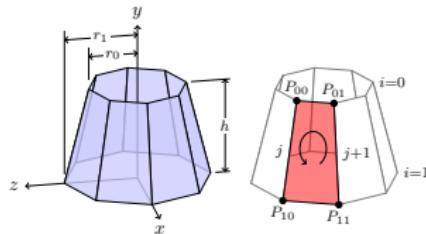
Aller plus loin avec L^AT_EX



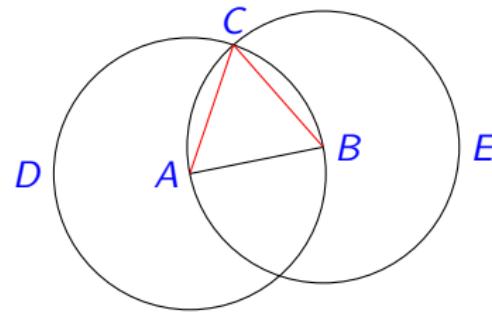
- ▶ Seminar
- ▶ Powerdot (Prosper)
- ▶ Beamer



- ▶ 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 de macros avec ou sans arguments

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

```
\newcommand{\valeurAbsolue}[1]{\ensuremath{|#1|}}
\newcommand{\itv}[2]{\ensuremath{[#1\mathrel{..} #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 L^AT_EX

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.

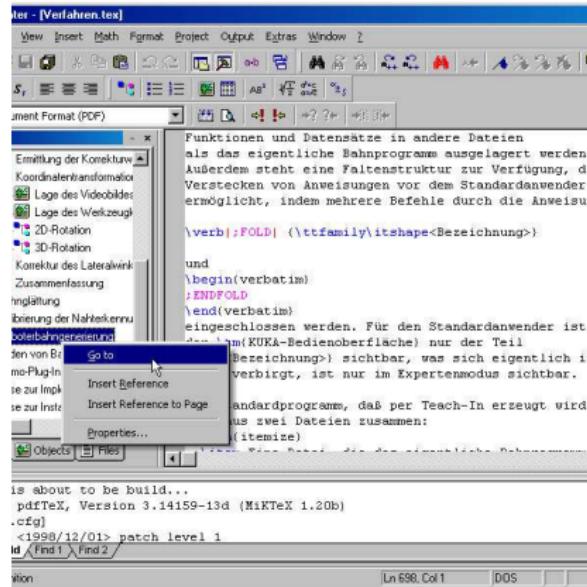
$e^{i\pi} + 1 = 0$. Uglier equations such as the integral of $1/x$ can be written as

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

/example.lyx (Changed)

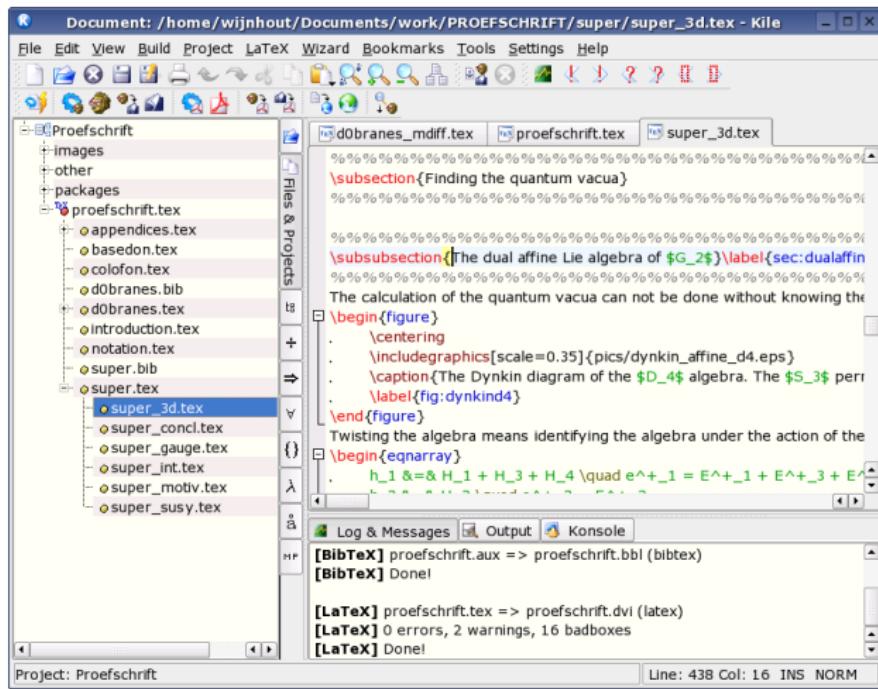
URL : <http://www.texniccenter.org/>

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_AT_EX 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 \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 1filll
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
@node First Chapter, Concept Index, Top, Top
@comment node-name, next, previous, up
@chapter First Chapter
@cindex Sample index entry
This is the contents of the first chapter.
@cindex 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?



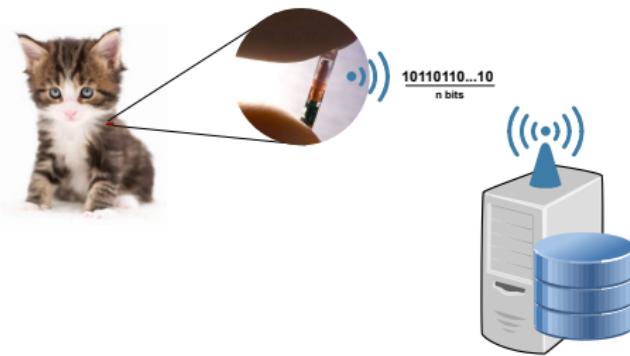
<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>



Is Your Cat Infected with a Computer Virus?



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

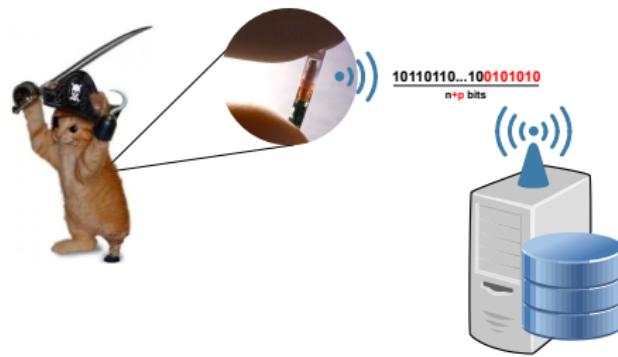




Is Your Cat Infected with a Computer Virus?



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

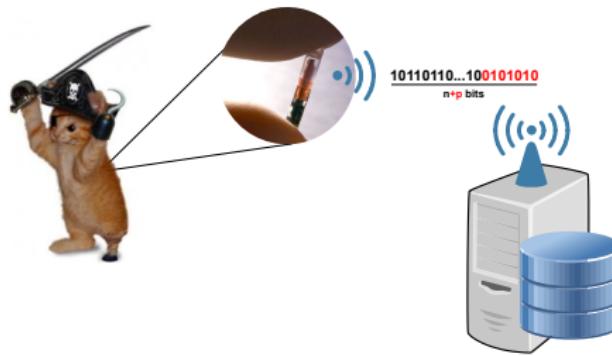




Is Your Cat Infected with a Computer Virus?



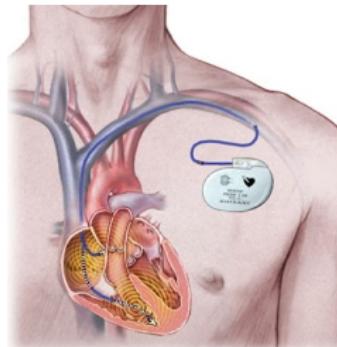
<http://zef.me/wp-content/uploads/2009/09/funny-cat.jpg>



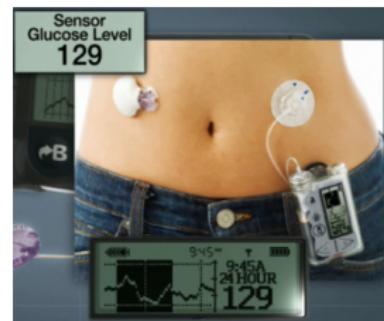
Is Your Cat Infected with a Computer Virus? M. R. Rieback, B. Crispo, A. S. Tanenbaum. PERCOM '06 Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, 2006.



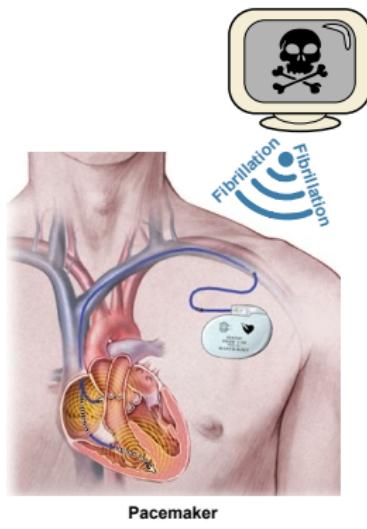
Le crime parfait



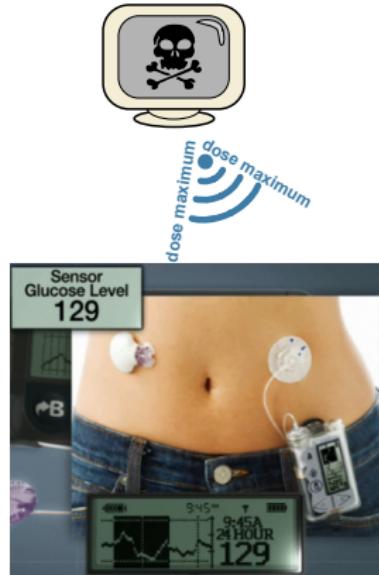
Pacemaker



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.
- ▶ *Rain Fall (aka A Clean Kill in Tokyo)*, Barry Eisler, 2002.

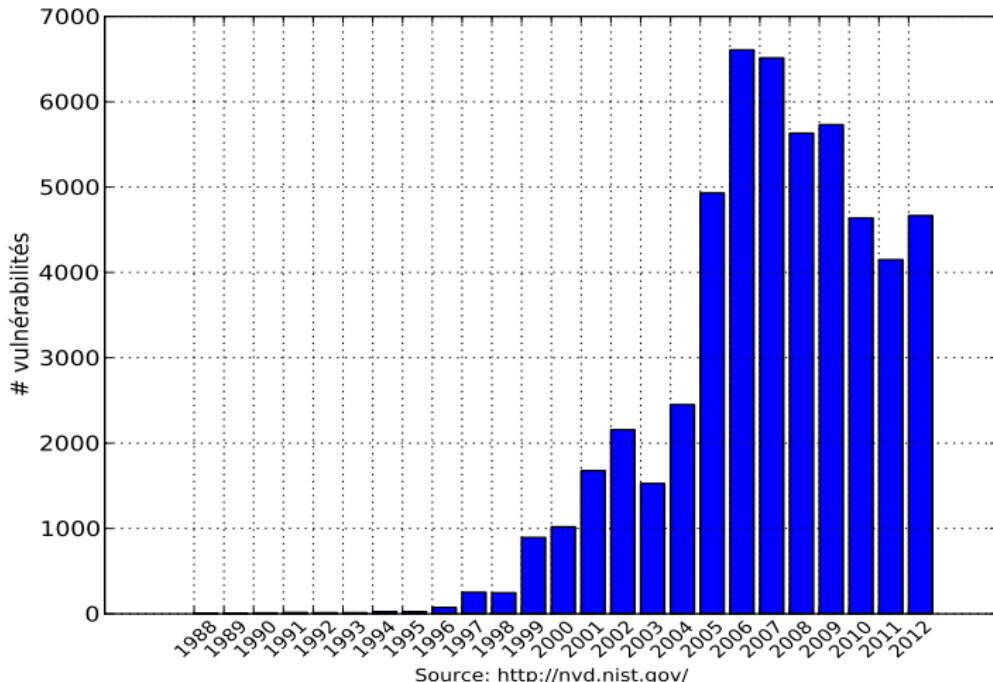


Les programmes informatiques contrôlent tous les aspects critiques de la vie :

- ▶ 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

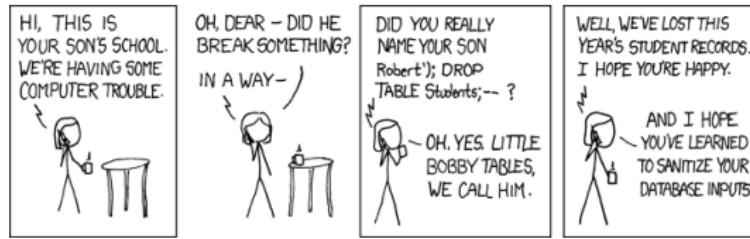
If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.

— 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;
    }

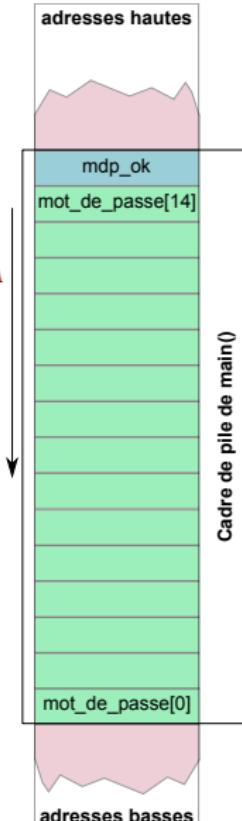
    // [...]

    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



Pile d'appels

Problème : Buffer overflow



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];

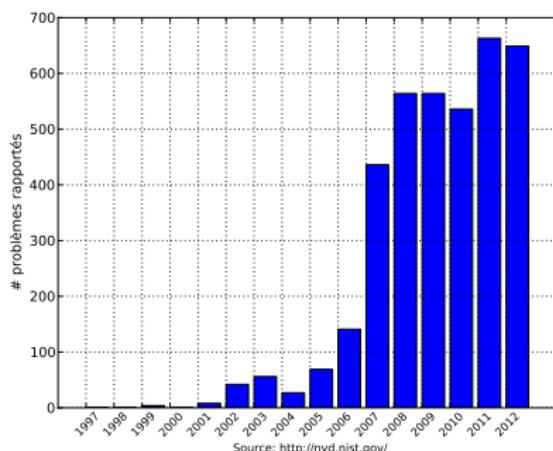
    if (szbig < 100) {
        strcpy(tinybuf, bigbuf);
        pbOV(bigbuf);
    }
    // [...]
}
```

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

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

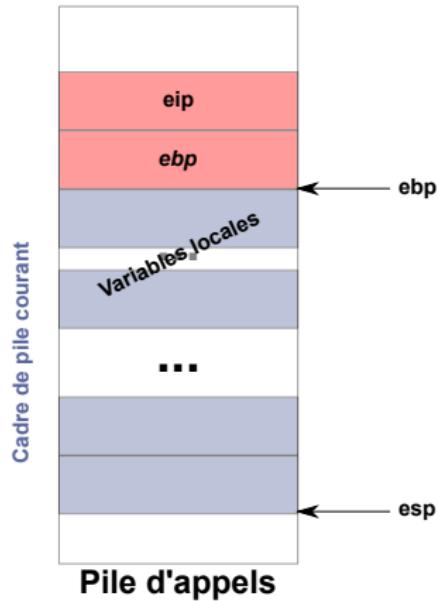


- ▶ Trou de sécurité courant
- ▶ Ecriture en mémoire au-delà de l'espace spécifiquement alloué
- ▶ Risques :
 - ▶ Crash du programme
 - ▶ Exécution inattendue de code
 - ▶ Prise de contrôle du programme





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+0x1c],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+0x1c],0x1
0x0804843b <+30>:   8b 44 24 1c                 mov   eax,DWORD PTR [esp+0x1c]
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+0x1c],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+0x1c],0x1
0x0804843b <+30>:   b8 44 24 1c          mov   eax,DWORD PTR [esp+0x1c]
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

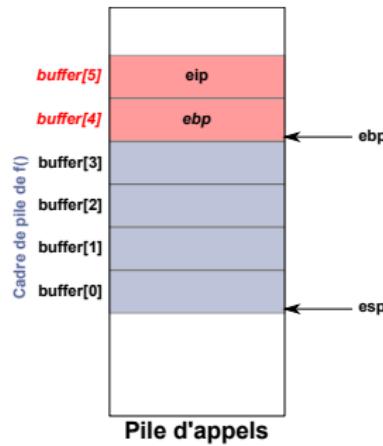
int main(void)
{
    int x = 0;

    f();
    x = 1;

    printf("%d\n", x);

    return 0;
}
```

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, ...)



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



Chaîne de caractères fournie par l'utilisateur comportant des « commandes » :

- ▶ Chaîne de format (e.g., pour `printf()`)
- ▶ Injection SQL
- ▶ Injection de commandes

« Entrée utilisateur » à prendre au sens large :

- ▶ Frappe au clavier
- ▶ Lecture de fichier (*Buffer overflow* dans Acrobat Reader [7](#), [8](#) et [9](#))



Exemple : chaîne de format

```
#include <stdio.h>
void display(int gui, char *str)
{
    if (gui) {
        // [Affichage graphique]
    } else {
        printf(str);
    }
}

int main(int argc, char *argv[])
{
    // [code]
    display(0, argv[1]);
    // [code]
}
```

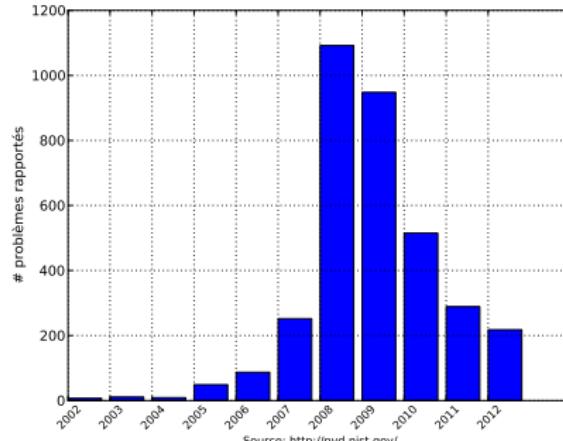
- ▶ % format_string toto
 - ⇒ toto
- ▶ % format_string "%p"; format_string "%p"
 - ⇒ 0x7fff8f60be3 0x7ffffadafe3f
 - ⇒ Info : Address Space Layout Randomization (**ASLR**)
- ▶ % format_string "%x %x"
 - ⇒ ca176e3c ca174ad0
 - ⇒ Info : valeurs se trouvant sur la pile



```
// [Création d'une commande SQL en C++]

commande = string("SELECT * FROM utilisateurs WHERE name = '"') +
    nomUtilisateur + string("';");

▶ Utilisation correcte avec nomUtilisateur valant « Dupont » :
    SELECT * FROM utilisateurs WHERE name = 'Dupont';
▶ Utilisation malveillante avec nomUtilisateur valant « ' ; DROP TABLE
    utilisateurs -- » :
    SELECT * FROM utilisateurs WHERE name = ''; DROP TABLE utilisateurs --';
```





Injection de commande

Code d'une application de gestion d'imprimante (IRIX, 1994, d'après Howard, LeBlanc & Viega) :

```
char buf[1024];
// [Demande du nom de l'imprimante]
// [...]
snprintf("buf", "lpr -P %s", nom_imprimante, sizeof(buf)-1);
system(buf);
```

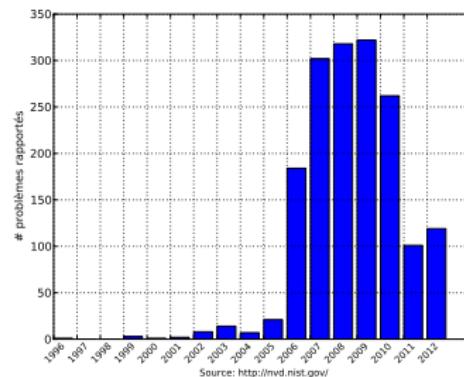


Injection de commande

Code d'une application de gestion d'imprimante (IRIX, 1994, d'après Howard, LeBlanc & Viega) :

```
char buf[1024];
// [Demande du nom de l'imprimante]
// [...]
snprintf("buf, "lpr -P %s", nom_imprimante,sizeof(buf)-1);
system(buf);
```

- ▶ Nom de l'imprimante : « toto; xterm& »
- ▶ Obtention d'un terminal *administrateur*



- ▶ Septembre 2014 : ShellShock



Integer overflow

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

int main(void) {
    int *mat;
    unsigned int matSZ, ligne, colonne;
    int val;

    printf("Taille de la matrice ? ");
    scanf("%d", &matSZ);
    mat = (int*)malloc((matSZ*matSZ)*
                        sizeof(int));
```

```
printf("Position de l'élément ? ");
scanf("%d %d", &ligne, &colonne);
printf("Valeur de l'élément ? ");
scanf("%d", &val);
if (ligne >= matSZ || colonne >= matSZ) {
    printf("Position invalide\n");
} else {
    mat[ligne*matSZ+colonne] = val;
}
```



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

int main(void) {
    int *mat;
    unsigned int matSZ, ligne, colonne;
    int val;

    printf("Taille de la matrice ? ");
    scanf("%d", &matSZ);
    mat = (int*)malloc((matSZ*matSZ)*
                        sizeof(int));
}

printf("Position de l'élément ? ");
scanf("%d %d", &ligne, &colonne);
printf("Valeur de l'élément ? ");
scanf("%d", &val);
if (ligne >= matSZ || colonne >= matSZ) {
    printf("Position invalide\n");
} else {
    mat[ligne*matSZ+colonne] = val;
}
```

- ▶ Avec `sizeof(int)==4` : si `matSZ==32768`, allocation de 0 octet dans le tas
- ▶ Écrasement possible d'autres variables dans le tas, même avec une saisie contrôlée



- ▶ *Buffer overflow au niveau de la JVM*
- ▶ *Buffer overflow possible avec JNI*
- ▶ Sensible au problème de l'*integer overflow*
- ▶ Sensible aux injections SQL
- ▶ Injection de code possible (`Class.forName()` et `Runtime.exec()`)



```
class DOS {  
    public static void main(String[] args) {  
        double d = Double.parseDouble("2.2250738585072012e-308");  
        System.out.println(d);  
    }  
}
```

- ▶ Bouclage infini : déni de service
- ▶ Même problème en PHP
- ▶ CVE-2010-4476



Danger 2 : protection des données

- ▶ Stockage de mots de passe

```
const char pass[] = "123456";
// [...]
if (!strcmp(mot_de_passe, pass)) {
    // [...]
}
```

- ▶ Machines à voter Diebold, 2007 : mot de passe stocké en clair (« diebold »)
- ▶ Cisco Wireless LAN Solution Engine (WLSE), 2004 :
login/password administrateur codé « en dur » dans le code du logiciel d'administration de réseaux

- ▶ Fuite de données

- ▶ Page par défaut d'une erreur 404

- ▶ Encryptage faible des données

- ▶ Content Scramble System (CSS)
 - ▶ Advanced Access Content System (AACS)

- ▶ *Race conditions*



Problèmes :

- ▶ Stockage d'un mot de passe en clair dans l'application ou dans un fichier mal protégé
- ▶ Autoriser un mot de passe faible (`pwgen`)
- ▶ Définir un mot de passe par défaut



Race conditions

```
FIC := $(shell mktemp)
$(shell echo "double pow(double,double);\
    int main(void){return pow(0.0,0.0)==1.0;}" > $(FIC).c)
$(shell gcc -o $(FIC) $(FIC).c -lm 1>&2 2> /dev/null)
correct_pow := $(shell $(FIC); echo $$?)
```



```
FIC := $(shell mktemp)
$(shell echo "double pow(double,double);\
    int main(void){return pow(0.0,0.0)==1.0;}" > $(FIC).c)
$(shell gcc -o $(FIC) $(FIC).c -lm 1>&2 2> /dev/null)
correct_pow := $(shell $(FIC); echo $$?)
```

- ▶ Exécution du script avec `nice` pour ralentir l'exécution
- ▶ Implémentation 4.3BSD-compliant de `mktemp` : nom créé à partir du PID et d'une lettre de l'alphabet → facile à connaître à l'avance
- ▶ *Race condition* entre le test d'existence du fichier `$(FIC)` et sa création
- ▶ Possibilité de création du fichier par une tierce personne après le test d'existence



Danger 3 : mauvaise gestion des erreurs

- ▶ Utilisation de `assert()` (disponible seulement en mode débogage)
- ▶ Vol Ariane 5 501 (mauvaise gestion d'un *integer overflow*)
(coût : 370 millions de dollars)



Danger 4 : mauvaise initialisation des données

Commande tar sous SUN Solaris 2.0 (1993) :

Chaque archive tar contenait une partie de /etc/passwd

- ▶ tar chargeait /etc/passwd sur le tas puis libérait la mémoire
- ▶ tar allouait un tampon pour lire les fichiers blocs par blocs sans initialiser la mémoire
- ▶ La partie non utilisée du tampon contenait des fragments des valeurs précédentes

Solution : initialiser le tampon utilisé au départ



- ▶ Utilisations : simulations, cryptographie, jeux, ...
- ▶ Génération pseudo-aléatoire (implémentation POSIX.1-2001) :

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

static unsigned long next = 1;

void srand(unsigned seed)
{
    next = seed;
}

int rand(void)
{
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536)%32768;
}

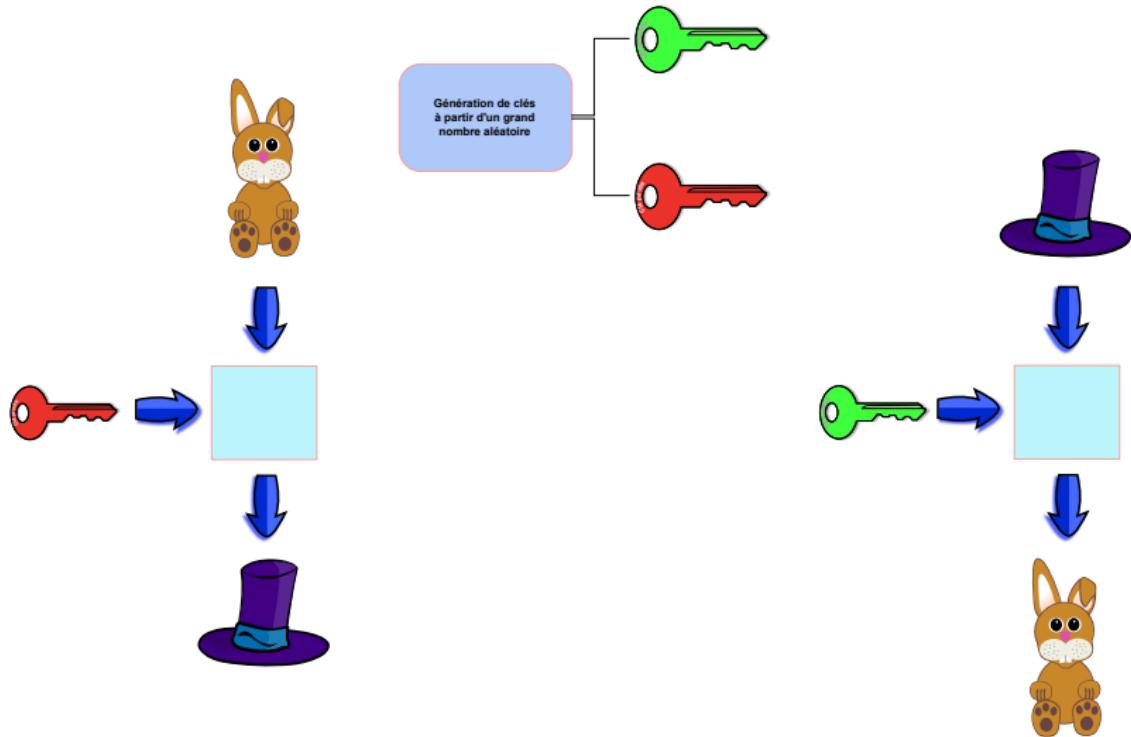
int main(void)
{
    srand(4);
    for (int i = 0; i < 10; ++i) {
        printf("%d ",rand());
    }
}
```

Sortie : 1817 24166 10491 3711 15407
4496 11138 3221 28482 23883

- ▶ Génération aléatoire :
 - ▶ Bruit d'un circuit électronique (`/dev/random`)
 - ▶ Phénomènes quantiques
 - ▶ ...



Cryptographie à clé asymétrique





Kerberos authentication protocol 4 :

- ▶ Création de clé secrète « au hasard »
- ▶ Générateur de nombres pseudo-aléatoires de mauvaise qualité
 - ➡ Clé secrète facile à retrouver



Kerberos authentication protocol 4 :

- ▶ Création de clé secrète « au hasard »
 - ▶ Générateur de nombres pseudo-aléatoires de mauvaise qualité
 - ➡ Clé secrète facile à retrouver
 - ▶ Bon générateur aléatoire : création complexe (voir **TAOCP**, D. Knuth)
 - ▶ Cryptographie : affaire de spécialistes
- ➡ Ne pas improviser son propre code



L'exploitation d'une vulnérabilité dépend de :

- ▶ Système d'exploitation (32 bits/64 bits)
- ▶ Langage utilisé
- ▶ Compilateur utilisé (et options de compilation utilisées)



Do's and Don'ts (1)

- ▶ Nettoyer les entrées de l'utilisateur
- ▶ Tester les bornes (*bound checking*)
- ▶ Définir des valeurs par défaut sûres
- ▶ Se méfier des références aux fichiers (`mktemp`, ...)
- ▶ Vérifier l'ordre de recherche de données ou de programmes (« . » dans PATH)
- ▶ Faire attention au stockage de données sensibles (nécessité ?, cryptage professionnel)
- ▶ Utiliser du code validé et connu plutôt que de réinventer la roue (moins de code écrit ➔ moins de risque d'erreur)
- ▶ Faire auditer le code par une tierce personne
- ▶ Éliminer complètement tout code obsolète
- ▶ Toujours tester la complétion d'une fonction (`malloc()`, ...)



Do's and Don'ts (2)

- ▶ Créer une base de tests de régression pour toutes les vulnérabilités trouvées dans l'application au fil du temps
- ▶ Appliquer les correctifs de sécurité
- ▶ Ne pas utiliser de nom de fichier relatif
- ▶ Ne pas invoquer un programme non validé dans un programme sûr
- ▶ Éviter le recours au *setuid*
- ▶ Attendre le pire de l'utilisateur
- ▶ Se méfier des nombres « aléatoires »
- ▶ Ne pas stocker de données sensibles en clair dans le programme



- ▶ Règle cardinale : ne pas faire confiance à l'utilisateur
- ▶ À qui faire confiance ?
 - ▶ Bibliothèques logicielles connues (vulnérabilité possible)
 - ▶ Compilateur compromis
 - Reflections on Trusting Trust*, K. Thompson. Comm. ACM 27(8), 1984
- ▶ Matériel (firmware compromis, design malveillant—**cas Huawei** en 2012, **badUSB** en 2014)



- ▶ *24 deadly sins of software security : programming flaws and how to fix them.* M. Howard, D. LeBlanc, J. Viega. McGraw Hill, 2010
- ▶ *Hacking : The art of Exploitation.* J. Erickson. No Starch Press, 2008
- ▶ *CERT C Programming Language Secure Coding Standard.* 2007.
- ▶ *Writing Secure Code.* M. Howard, D. LeBlanc. Microsoft Press. 2002

Tests

Beware of bugs in the above code; I have only proved it correct, not tried it. — Donald Knuth



*If debugging is the process of removing bugs,
then programming must be the process of putting them in.*

— Edsger W. Dijkstra

- ▶ *Mariner 1, 1962* (18,5 millions de dollars) : destruction d'une fusée ($\overline{\dot{R}_n} \rightsquigarrow \dot{R}_n$)
- ▶ *Mars Climate Orbiter, 1998* (125 millions de dollars) : sonde spatiale détruite (système métrique / système impérial)
- ▶ *Ariane 5, 1996* (370 millions de dollars) : overflow lors d'un *cast*
- ▶ *Instituto Oncologico Nacional, Panama, 2000* (8 morts, 20 blessés) : incohérence manuel/logiciel de pilotage d'un appareil de radiothérapie et non vérification des entrées ⇒ sur-irradiations
- ▶ *Patriot vs. Scud, 1991* (28 morts, 100 blessés) : erreur de calcul dans la trajectoire des missiles
- ▶ ...

Coût annuel estimé des *bugs* aux USA : ≈ 60 milliards de dollars



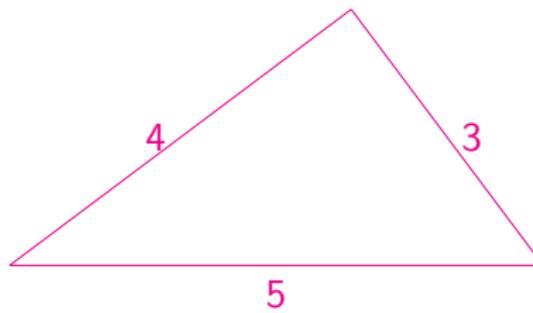
Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

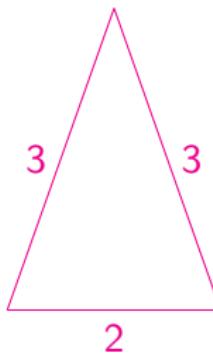


Test pour un triangle scalène



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

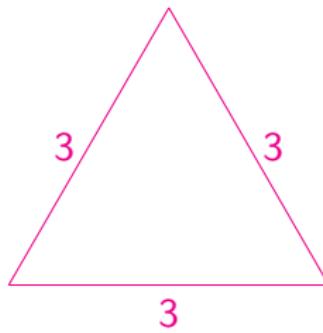


Test pour un triangle isocèle



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

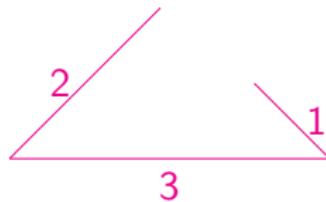


Test pour un triangle équilatéral



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

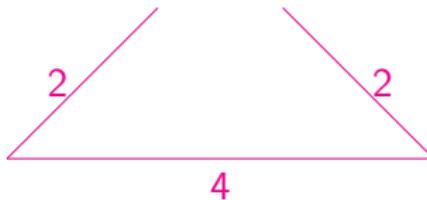


Triangle non scalène



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.



Triangle non isocèle



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?

Tests avec trois côté nuls (0,0,0) ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?

Tests avec trois côté nuls (0,0,0) ?

Tests avec un côté de longueur négative (2,-6,3) ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?

Tests avec trois côté nuls (0,0,0) ?

Tests avec un côté de longueur négative (2,-6,3) ?

Test avec des nombres non entiers ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?

Tests avec trois côté nuls (0,0,0) ?

Tests avec un côté de longueur négative (2,-6,3) ?

Test avec des nombres non entiers ?

Permutations de tous les cas précédents ?



Exemple de motivation

- ▶ Écrire un programme lisant trois longueurs de côtés (nombres entiers) et indiquant si le triangle associé est *scalène*, *isocèle*, ou *équilatéral*;
- ▶ Proposer un jeu de tests validant le programme.

Tests avec un côté nul (2,4,0) ?

Tests avec deux côté nuls (0,4,0) ?

Tests avec trois côté nuls (0,0,0) ?

Tests avec un côté de longueur négative (2,-6,3) ?

Test avec des nombres non entiers ?

Permutations de tous les cas précédents ?

Définir un bon ensemble de test : difficile



À quoi servent les tests ?

- ▶ Montrer qu'un programme ne contient pas d'erreur ?



À quoi servent les tests ?

- ▶ Montrer qu'un programme ne contient pas d'erreur ?
- ▶ Montrer qu'un programme fonctionne correctement ?



À quoi servent les tests ?

- ▶ Montrer qu'un programme ne contient pas d'erreur ?
- ▶ Montrer qu'un programme fonctionne correctement ?
- ▶ Montrer qu'un programme fait ce qu'il est censé faire ?



À quoi servent les tests ?

- ▶ Montrer qu'un programme ne contient pas d'erreur ?
- ▶ Montrer qu'un programme fonctionne correctement ?
- ▶ Montrer qu'un programme fait ce qu'il est censé faire ?

Montrer qu'un programme contient des erreurs !

Tester : processus destructeur et non constructeur
Expérience positive = trouver des *bugs* !

Program testing can be used to show the presence of bugs, but never to show their absence!
— Edsger W. Dijkstra

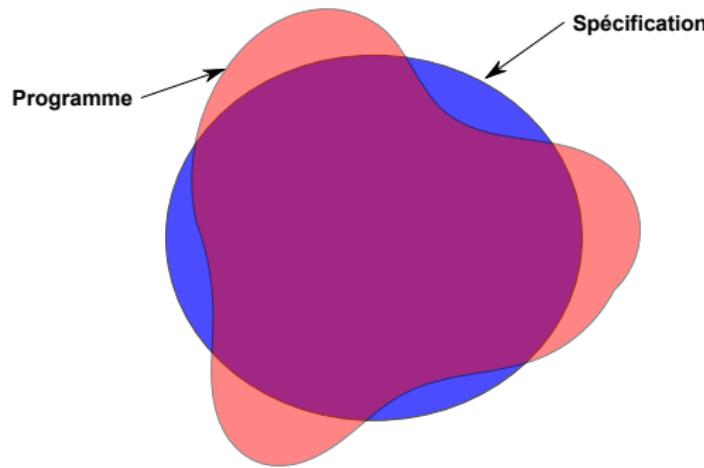


Correctness is a relative notion

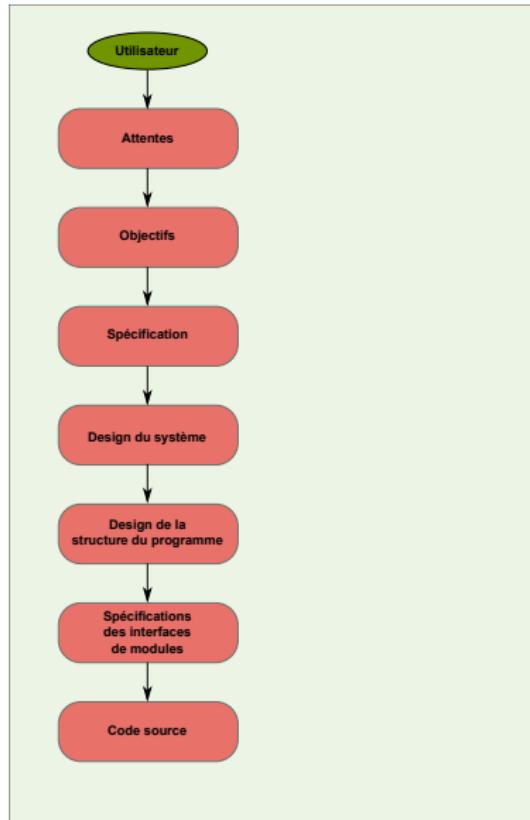
— B. Meyer

That's not a bug, it's a feature.

— Attribué à William Gates, III



- ▶ Notion d'erreur liée à une *spécification*



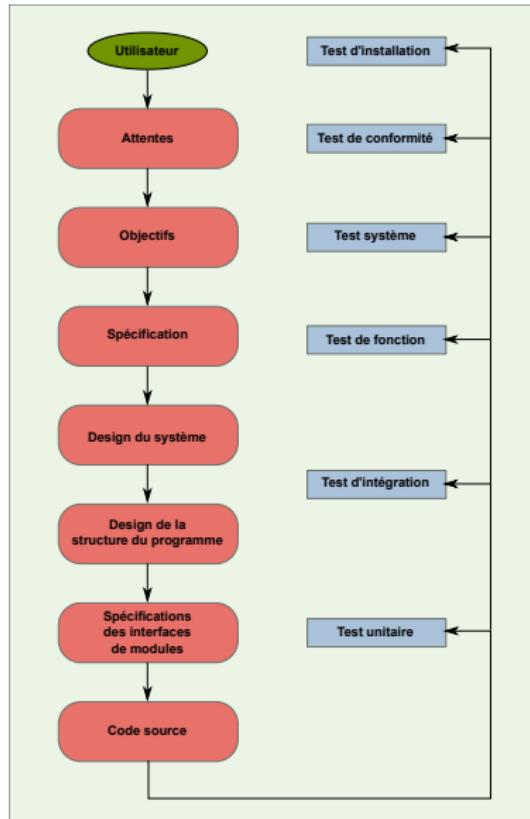
Spécification des interfaces :

Préconditions. Ce que l'utilisateur doit garantir en entrée

Post-conditions. Ce que le code garantit en sortie

Erreurs/bugs possibles à des niveaux différents :

- ▶ Erreur dans la spécification du système par rapport aux attentes
- ▶ Erreur dans la structure du programme implémentant les spécifications
- ▶ Erreur dans un module individuel



Spécification des interfaces :

Préconditions. Ce que l'utilisateur doit garantir en entrée

Post-conditions. Ce que le code garantit en sortie

Erreurs/bugs possibles à des niveaux différents :

- ▶ Erreur dans la spécification du système par rapport aux attentes
- ▶ Erreur dans la structure du programme implémentant les spécifications
- ▶ Erreur dans un module individuel



But des différents tests

Test unitaire. Montrer une différence entre ce que fait ou ne fait pas un module et la spécification de son interface

Test de fonction. Prouver qu'un programme ne répond pas strictement à sa spécification

Test système. Prouver que le produit n'est pas cohérent avec les objectifs originaux

- ▶ Comparaison avec la documentation utilisateur (fonctionnalités réellement implémentées ?)
- ▶ Test de charge : le système supporte t'il les charges maximum prévues ?
- ▶ Utilisabilité du système
- ▶ Performances, sécurité du code
- ▶ Test de la documentation (complète, fidèle, cohérente, lisible)

Test de conformité. Comparaison par l'utilisateur du système et de ses attentes



- ▶ Tests « black box » : tests réalisés sans connaissance de l'implémentation (si possible par une tierce personne). Basé sur les spécifications
- ▶ Tests « white box » : tests réalisés avec connaissance de l'implémentation. Basé sur le code source.



« Black box »

```
int find_min(int T[], int sz) { ... }
```

« White box »

```
int find_min(int T[], int sz) {
    int m = T[0];
    for (int i = 1; i < sz; ++i) {
        if (T[i] < m) {
            m = T[i];
        }
    }
    return m;
}
```

- ▶ **Préconditions** : T contient au moins un élément, sz est égal au nombre d'éléments dans T
- ▶ **Post-condition** : la valeur rentrée est la première occurrence du plus petit élément de T



- ▶ Spécifications peu claires
 - ▶ Précondition : T non nul
 - ▶ Postcondition : Retourne le minimum de T
- ▶ Spécifications imprécises
 - ▶ Précondition trop forte : T non nul et contient un élément
 - ▶ Postcondition trop faible : Le résultat est inférieur ou égal à tous les éléments de T
- ▶ Spécifications incorrectes
 - ▶ Précondition trop faible : T non nul (et contient au moins un élément)
 - ▶ Postcondition trop forte : Le résultat est inférieur ou égal à tous les éléments de T et est strictement positif



1. Définir à l'avance le résultat attendu des tests
Évite le problème de « voir ce que l'on attend »
2. Ne pas tester son propre programme
Passage difficile du constructeur au destructeur
3. Vérifier attentivement le résultat de chaque test
4. Tester les cas attendus valides ET les cas inattendus invalides
5. Tester si le programme fait ce que l'on attend ET s'il ne fait pas ce que l'on n'attend pas
6. Ne pas utiliser de tests « jetables »
Garder précieusement chaque test pour la régression
7. Partir du principe que l'on *doit* et que l'on va trouver des bugs
8. Plus on trouve de bugs dans une section, plus il est probable qu'il y en a encore



- ▶ Plus un bug est trouvé tard, plus il coûte cher
 - ➡ Tester dès le départ (tests unitaires)
- ▶ Automatiser les tests pour qu'ils soient faciles à relancer (régression)
 - ▶ CppUnit
- ▶ En cas d'échec lors d'un test :
 - ▶ Erreur dans le module
 - ▶ Erreur dans le test ou dans le résultat attendu



1. Relecture du code

- ▶ Comité de 3/4 personnes (leader, secrétaire, implémenteur, testeur confirmé)
- ▶ Lecture du code à voix haute avec explication de sa logique par l'implémenteur
- ▶ Comparaison du code avec une checklist d'erreurs communes
- ▶ Focus sur la recherche de bug, pas leur correction
- ▶ Examen des qualités du code, pas des qualités de l'implémenteur

2. Création de jeux de tests en *black boxing*, puis ajout de tests en *white boxing*

- ▶ Test de toutes les entrées impossible ou prohibitif
- ▶ → Trouver le plus petit ensemble de tests assurant la détection du plus grand nombre d'erreurs possibles



- ▶ On se base sur les spécifications pour créer les tests (pas de connaissance du code)

```
// Préconditions: size(T)==sz, sz >= 1
// Postconditions: find_min(T) == min(T)
int find_min(int T[], int sz) {
    int m = T[0];
    for (int i = 1; i < sz; ++i) {
        if (T[i] < m) {
            m = T[i];
        }
    }
    return m;
}
```

Choix des tests :

- ▶ Création de classes d'équivalences (inutile *a priori* de tester `find_min([3],1)` et `find_min([5],1)`)
 - ▶ Choisir un représentant dans chaque classe d'équivalence
 - ▶ Choisir des tests aux limites entre les classes
 - ▶ Tests de comportement sur les cas anormaux
-
- | | |
|---------------------------------------|---------------------------------------|
| ▶ <code>find_min([3],1)==3</code> | ▶ <code>find_min([7,5,3],3)==3</code> |
| ▶ <code>find_min([-6,5],2)==-6</code> | ▶ <code>find_min([],0) ?</code> |
| ▶ <code>find_min([-2,5],2)==-2</code> | ▶ <code>find_min([5,6],1) ?</code> |
| ▶ <code>find_min([3,5,7],3)==3</code> | ▶ ... |



- ▶ Connaissance du code → recherche de jeux de tests couvrant tout le code
- ▶ Couverture exhaustive du code par des tests (classes d'équivalences)
- ▶ Vérification de la couverture (`gcov/lcov`)

```
// Préconditions: size(T)==sz, sz >= 1
// Postconditions: find_min(T) == min(T)
int find_min(int T[], int sz) {
    int m = T[0];
    for (int i = 1; i < sz; ++i) {
        if (T[i] < m) {
            m = T[i];
        }
    }
    return m;
}
```

- ▶ Test si entrée dans le « for » et si pas d'entrée (tableau à 1 case ou à plusieurs cases)
- ▶ Test si $T[0]$ est le minimum et si ce n'est pas le cas
- ▶ Test d'*integer overflow* sur sz ?



Checklist dépendant du langage utilisé

Exemples :

- ▶ Une variable est-elle déclarée et non initialisée ?
- ▶ Un pointeur fait-il référence à de la mémoire allouée ?
- ▶ Un calcul utilise t'il des variables de types différents ?
- ▶ ...

Voir [Code Complete, 2nd Ed. Checklist](#)



Rapport de couverture

Utilisation de gcov/lcov :

- ▶ Compilation des tests avec options « `-g -O0 --coverage` »
- ▶ Utilisation de lcov pour le rapport en HTML

```
% g++ -c -g -O0 --coverage find_min.cpp
% g++ -g -O0 --coverage -o test_find_min test_find_min.cpp find_min.o -lcppunit
% lcov --capture --directory . --output-file coverage.info
% genhtml coverage.info --output-directory html
```

The screenshot shows a web-based LCOV code coverage report. At the top, there's a navigation bar with links for File, Edit, View, Go, Bookmarks, Settings, Window, and Help. Below that is a toolbar with icons for back, forward, search, and refresh. The main content area has a title "LCOV - code coverage report". It displays the current view as "top level - home/goulaud/enseignement/2012-2013/Master/X7II030_outils-developpement/slides/find_min.cpp (source / functions)". Below this, it shows the test configuration: "Test: coverage.info" and "Date: 2012-11-20". A summary table at the top right shows coverage statistics: Lines: 6 / 6 (100.0 %) and Functions: 1 / 1 (100.0 %). The main part of the page shows the source code of the `find_min.cpp` file with line numbers and coverage information. Lines 1 through 13 are shown, with line 13 highlighted in yellow. The code implements a function to find the minimum value in an array. At the bottom, a footer notes "Generated by: LCOV version 1.10".

```
Line data  Source code
1 : #include <find_min.h>
2 : // Préconditions: size(T)==sz, sz >=1
3 : // Postconditions: find_min(T) == min(T)
4 : int find_min(int T[], int sz) {
5 :     int n = T[0];
6 :     for (int i = 1; i < sz; ++i) {
7 :         if (T[i] < n) {
8 :             n = T[i];
9 :         }
10:    }
11:    return n;
12: }
13:
```



Quand arrêter les tests ?

- ▶ Couverture exhaustive du code dans tous les cas de figure en pratique prohibitif
- ▶ Quand arrêter d'ajouter des jeux de tests ?
 - ▶ Seuil sur le nombre de bugs trouvés en un temps donné
 - ▶ Seuil sur le nombre total de bugs trouvés

La certitude absolue de correction n'existe pas

► Bibliothèque pour faciliter les tests unitaires en C++

```
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/ui/text/TestRunner.h>
#include <cppunit/extensions/HelperMacros.h>
#include "find_min.h"

class test_find_min :
public CppUnit::TestFixture {
CPPUNIT_SUITE(test_find_min);
CPPUNIT_TEST(test_singleton);
// [...]
CPPUNIT_TEST(test_regulier);
CPPUNIT_SUITE_END();

public:
void setUp() {}
void tearDown() {}

void test_singleton() {
    int Ta[] = { 2 };
    CPPUNIT_ASSERT(find_min(Ta,1)==2);
    int Tb[] = { -6 };
    CPPUNIT_ASSERT(find_min(Tb,1)==-6);
}
```

```
void test_regulier() {
    int T[] = {3, 4, 6, -7, 2 };
    CPPUNIT_ASSERT(find_min(T,5)==-7);
};

CPPUNIT_SUITE_REGISTRATION(test_find_min)

int main(int argc, char **argv)
{
    CppUnit::TextUi::TestRunner runner;
    CppUnit::TestFactoryRegistry &registry =
        CppUnit::TestFactoryRegistry::
            getRegistry();
    runner.addTest(registry.makeTest());
    bool success = runner.run();
    return !success;
}
```



- ▶ **Test** : démonstration de l'existence d'un problème
 - `cppunit`, `gcov`, ...
- ▶ Tests à faire faire par un tiers et non par l'implémenteur
- ▶ **Débogage** : recherche de la source du problème
 - `gdb`, `Valgrind`, affichages, ...
- ▶ Débogage à faire faire par l'implémenteur



- ▶ *The Economic Impacts of Inadequate Infrastructure for Software Testing.* Planning Report 02-3, National Institute of Standards & Technology, 2002
- ▶ *The Art of Software Testing.* Glenford J. Myers, Tom Badgett, Todd M. Thomas, and Corey Sandler. Second edition, John Wiley & Sons, 2004.

Débogage

If you want more effective programmers, you will discover that they should not waste their time debugging, they should not introduce the bugs to start with. — Edsger W. Dijkstra

When debugging, novices insert corrective code ; experts remove defective code.

— Richard Pattis



Origine de « bug »

92

9/9

0800 Antam started
1000 " stopped - antam ✓
13° UC (033) MP - MC $\frac{1.2700}{1.3047645} \times 10^6 = 9.0378467985$ correct
(033) PRO. 2 2.13047645
correct 2.13047645

Relays 6-2 in 033 failed special speed test
in Relay 11.000 test.

Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.

1545



Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.
1600 Antam not started.
1700 closed down.

Relay
2145
Relay 3378

Harvard Mark II, 1947



Origine de « bug »



Harvard Mark II, 1947

- **M. Quinion** : vocabulaire d'ingénieur et de télégraphiste au XIX^e siècle



Origine de « bug »



Harvard Mark II, 1947

- M. Quinion : vocabulaire d'ingénieur et de télégraphiste au XIX^e siècle
- Bug : euphémisme pour « ERREUR »



Coût :

- ▶ 50–75 % du coût de développement
- ▶ 50 % du temps de développement

Difficultés :

- ▶ Symptôme et bug géographiquement et/ou temporellement éloignés
- ▶ Symptôme difficile à reproduire
- ▶ Symptôme non lié à un bug du programme (dernier recours)
(Rayons cosmiques ?)



1. L'utilisateur informe le fabricant d'un problème
2. Le fabricant affecte la résolution du problème à un développeur
3. Le développeur reproduit le problème
4. Le développeur isole les circonstances du problème
5. Le développeur isole et corrige le défaut
6. Le fabricant fournit le correctif aux utilisateurs



The Jargon File :

Bohr bug. Bug reproductible de façon garantie ;

Heisenbug. Bug disparaissant ou induisant un comportement différent lorsque l'on observe le programme pour l'éradiquer ;

Mandelbug. Bug dont les causes sont multiples et complexes, induisant un comportement semblant non déterministe ;

Schroedelinbug. Bug ne se manifestant qu'à partir de sa découverte fortuite dans le code source.



Bohr bug

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

int minimum(int T[], unsigned int sz)
{ // Précondition: sz > 0
    assert(sz != 0);
    if (sz == 1) {
        return T[0];
    } else {
        int minimum = T[0];
        for (int i=1; i < sz-1; ++i) {
            if (T[i] < minimum) {
                minimum = T[i];
            }
        }
        return minimum;
    }
}

int main(void)
{
    printf("%d\n", minimum({2,5,6,4}, 4));
    // sortie: 2 (OK)
    printf("%d\n", minimum({3,5,6,1}, 4));
    // sortie: 3 (Pas OK)
    // Hypothèse: retourne l'élément en position 0
    printf("%d\n", minimum({5,3,6,1}, 4));
    // sortie: 3 (hypothèse réfutée)
    return 0;
}
```



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

int minimum(int T[], unsigned int sz)
{ // Précondition: sz > 0
    assert(sz != 0);
    if (sz == 1) {
        return T[0];
    } else {
        int minimum = T[0];
        for (int i=1; i < sz-1; ++i) {
            if (T[i] < minimum) {
                minimum = T[i];
            }
        }
        return minimum;
    }
}

int main(void)
{
    printf("%d\n", minimum({2,5,6,4}, 4));
    // sortie: 2 (OK)
    printf("%d\n", minimum({3,5,6,1}, 4));
    // sortie: 3 (Pas OK)
    // Hypothèse: retourne l'élément en position 0
    printf("%d\n", minimum({5,3,6,1}, 4));
    // sortie: 3 (hypothèse réfutée)
    return 0;
}
```

- ▶ Dernier élément du tableau jamais pris en compte



```
#include <iostream>
#include <fenv.h>

using namespace std;

void rnd_dnup(double a, double b, double& lo, double& hi)
{
    fesetround(FE_DOWNWARD);
    lo = a/b;
    fesetround(FE_UPWARD);
    hi = a/b;
}

int main(void)
{
    double lo, hi;

    rnd_dnup(1.0,10.0,lo,hi);
    cout << boolalpha << (lo==hi) << endl;
    return 0;
}
```

- ▶ Essai du programme : « true » (**Problème**)
- ▶ Débogage du programme : « false » (**OK**)



```
#include <iostream>
#include <fenv.h>

using namespace std;

void rnd_dnup(double a, double b, double& lo, double& hi)
{
    fesetround(FE_DOWNWARD);
    lo = a/b;
    fesetround(FE_UPWARD);
    hi = a/b;
}

int main(void)
{
    double lo, hi;

    rnd_dnup(1.0,10.0,lo,hi);
    cout << boolalpha << (lo==hi) << endl;
    return 0;
}
```

- ▶ Essai du programme : « true » (**Problème**)
- ▶ Débogage du programme : « false » (**OK**)
- ▶ Compilation classique : optimisation du calcul de « a/b »
- ▶ Compilation pour débogage : pas d'optimisation du code



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

unsigned long int fun(unsigned long int *a)
{
    assert(sizeof(unsigned long int) == sizeof(unsigned long int*));
    // [...]
    return (unsigned long int)a;
}

int main(void)
{
    unsigned long int i = 45;
    // [...]
    i = fun(&i);
    // [...]
    printf("Valeur ? ");
    scanf("%ld",i);
    // [...]
    printf("%ld\n",i);

    return 0;
}
```



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

unsigned long int fun(unsigned long int *a)
{
    assert(sizeof(unsigned long int) == sizeof(unsigned long int*));
    // [...]
    return (unsigned long int)a;
}

int main(void)
{
    unsigned long int i = 45;
    // [...]
    i = fun(&i);
    // [...]
    printf("Valeur ? ");
    scanf("%ld",i);
    // [...]
    printf("%ld\n",i);

    return 0;
}
```

- ▶ Programme fonctionnant parfaitement jusqu'au jour où quelqu'un définira une autre variable pour récupérer le résultat de `fun()`.



Qui crée les bugs ?

- ▶ Le programmeur de routines (tests unitaires)
 - ▶ Erreur de logique dans l'algorithme
 - ▶ Erreur de transcription de l'algorithme
 - ▶ Copier/coller
 - ▶ Faute de syntaxe (e.g., « if (x = 3) »)
- ▶ Personne en particulier (tests d'intégration)
 - ▶ Erreur découlant de l'assemblage de codes d'origines différentes (e.g., métrique/impérial)
- ▶ Erreurs extérieures
 - ▶ Sur-optimisation du compilateur
 - ▶ Erreur du matériel (exemple : DIV)
 - ▶ Phénomènes cosmiques (particules α , ...)



- ▶ *Brute force*
 - ▶ *Dumps* mémoire, trace, *log* de tout
 - ▶ **Trop de données**
- ▶ **Retour arrière**
 - ▶ Partir du point où apparaît le symptôme et remonter pas-à-pas
 - ▶ Difficile à faire pour un gros programme
- ▶ **Déduction/élimination**
 - ▶ Formuler des hypothèses, écrire les tests pour les valider/invalider



- ▶ Étudier les données disponibles (Quel test passe/échoue)
- ▶ Former une nouvelle hypothèses cohérente avec *toutes* les données
- ▶ Définir un nouveau test pour *réfuter* l'hypothèse

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

int minimum(int T[], unsigned int sz)
{ // Précondition: sz > 0
    assert(sz != 0);
    if (sz == 1) {
        return T[0];
    } else {
        int minimum = T[0];
        for (int i=1; i < sz-1; ++i) {
            if (T[i] < minimum) {
                minimum = T[i];
            }
        }
    }
    return minimum;
}

int main(void)
{
    printf("%d\n", minimum({2,5,6,4}, 4));
    // sortie: 2 (OK)
    printf("%d\n", minimum({3,5,6,1}, 4));
    // sortie: 3 (Pas OK)
    // Hypothèse: retourne l'élément en position 0
    printf("%d\n", minimum({5,3,6,1}, 4));
    // sortie: 3 (hypothèse réfutée)
    return 0;
}
```



Règles applicables pas seulement aux logiciels :

1. Vérifier les données (format attendu, ...);
2. Comprendre le système (quelle partie fait quoi, comment, pourquoi);
3. Reproduire le problème de la façon la plus simple possible;
4. Accumuler les observations (sans sauter immédiatement aux conclusions);
5. Réduire l'espace de recherche;
6. Modifier un seul élément à la fois;
7. Garder la trace des actions;
8. Tester d'abord les hypothèses les plus simples (+ rasoir d'Occam);
9. Prendre un avis extérieur;
10. Vérifier l'effet du correctif (un bug ne disparaît pas spontanément).



Vérification des données

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int v1, v2, sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file >> v1 >> v2) {
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
    file.close();
}
```



```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int v1, v2, sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file >> v1 >> v2) {
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
    file.close();
}
```

- ▶ Erreur avec le fichier `values.dat` :

```
34.0 12.0
23.0 -6.0
```



```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int v1, v2, sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file >> v1 >> v2) {
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
    file.close();
}
```

- ▶ Erreur avec le fichier `values.dat` :

```
34.0 12.0
23.0 -6.0
```

- ▶ Entiers attendus, flottants fournis



Comprendre le système (1/2)

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (!file.eof()) {
        int v1, v2;
        file >> v1 >> v2;
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
}
```

- ▶ Entrée :

```
34 12
23 -6
```

- ▶ Sortie : 80 0



Comprendre le système (1/2)

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file.good()) {
        int v1, v2;
        file >> v1 >> v2;
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
}
```

- ▶ Entrée :

```
34 12
23 -6
```

- ▶ Sortie : 80 0



Comprendre le système (1/2)

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int sum1 = 0, sum2 = 0;
    file.open("values.dat");
    while (file.good()) {
        int v1, v2;
        file >> v1 >> v2;
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
}
```

- ▶ Entrée :

```
34 12
23 -6
```

- ▶ Sortie : 80 0
- ▶ Suppression du dernier retour-chariot du fichier d'entrée : OK



Comprendre le système (1/2)

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int v1, v2, sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file >> v1 >> v2) {
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
}
```

- ▶ Entrée :

```
34 12
23 -6
```

- ▶ Sortie : 80 0

- ▶ Suppression du dernier retour-chariot du fichier d'entrée : OK



Comprendre le système (1/2)

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ifstream file;
    int v1, v2, sum1 = 0, sum2 = 0;

    file.open("values.dat");
    while (file >> v1 >> v2) {
        sum1 += v1;
        sum2 += v2;
    }
    cout << sum1 << " " << sum2 << endl;
}
```

- ▶ Entrée :

```
34 12
23 -6
```

- ▶ Sortie : 80 0
- ▶ Suppression du dernier retour-chariot du fichier d'entrée : OK
- ▶ Shotgun debugging



Comprendre le système (2/2)

- ▶ Connaître le langage et les bibliothèques utilisées
- ▶ Connaître les flux de données (direction et contenu des informations)
- ▶ Connaître l'environnement (*endianness*, contraintes d'alignement, ...)
- ▶ Connaître les outils (influence des *flags* du compilateur, ...)

Dans le doute, s'informer



Reproduire le problème (1/2)

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

using namespace std;

union entier_reel {
    unsigned int i[2];
    double d;
};

int main(void)
{
    // Pi arrondi vers le bas
    entier_reel pi_dn = { {1413754136, 1074340347} };

    // [...]
    double v = cos(er.d);
    // [...]
}
```

- ▶ Bug signalé par l'utilisateur : mauvaise valeur de π
- ▶ Bug non reproductible par le testeur



Reproduire le problème (1/2)

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

using namespace std;

union entier_reel {
    unsigned int i[2];
    double d;
};

int main(void)
{
    // Pi arrondi vers le bas
    entier_reel pi_dn = { {1413754136, 1074340347} };

    // [...]
    double v = cos(er.d);
    // [...]
}
```

- ▶ Bug signalé par l'utilisateur : mauvaise valeur de π
- ▶ Bug non reproductible par le testeur
- ▶ Machine *little/big endian* ?
- ▶ Système 32/64 bits ?



Reproduire le problème (2/2)

- ▶ Refaire la manipulation censée créer le problème
- ▶ Prendre connaissance de la chaîne *complète* des évènements conduisant au bug
- ▶ *Stimuler* pour obtenir le problème (envoi massif de données, ...)
- ▶ Ne pas *simuler* l'environnement d'échec
- ▶ Bug intermittent : faire varier tous les paramètres un par un pour déterminer les causes réelles
- ▶ Garder trace de tous les tests déjà faits
- ▶ Garder l'esprit ouvert



Accumuler les observations (1/2)

```
#include <stdio.h>
#include <alloca.h>
#include <stdint.h>

double f(double *mat, uint32_t sz)
{
    double res = 0;
    double *tmpmat =
        (double*)alloca(sz*sizeof(double));
    // [...]
    return res;
}

double g(double *mat, double *vec,
         uint32_t szmat, uint32_t szvec)
{
    double out = 0;
    for (uint32_t i = 0; i < szvec; ++i) {
        out += f(mat,szmat);
    }
}
int main(void)
{
    // [...]
    cout << g(M,V,1000,100000) << endl;
}
```

- ▶ *Stack overflow chez le client*
- ▶ Initialement, pas d'erreur chez le testeur



Accumuler les observations (1/2)

```
#include <stdio.h>
#include <alloca.h>
#include <stdint.h>

double f(double *mat, uint32_t sz)
{
    double res = 0;
    double *tmpmat =
        (double*)alloca(sz*sizeof(double));
    // [...]
    return res;
}

double g(double *mat, double *vec,
         uint32_t szmat, uint32_t szvec)
{
    double out = 0;
    for (uint32_t i = 0; i < szvec; ++i) {
        out += f(mat,szmat);
    }
}
int main(void)
{
    // [...]
    cout << g(M,V,1000,100000) << endl;
}
```

- ▶ *Stack overflow* chez le client
- ▶ Initialement, pas d'erreur chez le testeur
- ▶ Utilisation du même compilateur que l'utilisateur ↳ *Stack overflow*
- ▶ Hypothèse du testeur : la pile allouée au processus est trop petite ↳ changer la taille (`ulimit`) ↳ **OK**



Accumuler les observations (1/2)

```
#include <stdio.h>
#include <alloca.h>
#include <stdint.h>

double f(double *mat, uint32_t sz)
{
    double res = 0;
    double *tmpmat =
        (double*)alloca(sz*sizeof(double));
    // [...]
    return res;
}

double g(double *mat, double *vec,
         uint32_t szmat, uint32_t szvec)
{
    double out = 0;
    for (uint32_t i = 0; i < szvec; ++i) {
        out += f(mat,szmat);
    }
}
int main(void)
{
    // [...]
    cout << g(M,V,1000,100000) << endl;
}
```

- ▶ *Stack overflow* chez le client
- ▶ Initialement, pas d'erreur chez le testeur
- ▶ Utilisation du même compilateur que l'utilisateur ↳ *Stack overflow*
- ▶ Hypothèse du testeur : la pile allouée au processus est trop petite ↳ changer la taille (`ulimit`) ↳ **OK**
- ▶ Hypothèse pas cohérente avec les observations (limite imposée par le système, pas par le compilateur)



Accumuler les observations (1/2)

```
#include <stdio.h>
#include <alloca.h>
#include <stdint.h>

double f(double *mat, uint32_t sz)
{
    double res = 0;
    double *tmpmat =
        (double*)alloca(sz*sizeof(double));
    // [...]
    return res;
}

double g(double *mat, double *vec,
         uint32_t szmat, uint32_t szvec)
{
    double out = 0;
    for (uint32_t i = 0; i < szvec; ++i) {
        out += f(mat,szmat);
    }
}
int main(void)
{
    // [...]
    cout << g(M,V,1000,100000) << endl;
}
```

- ▶ *Stack overflow* chez le client
- ▶ Initialement, pas d'erreur chez le testeur
- ▶ Utilisation du même compilateur que l'utilisateur ↳ *Stack overflow*
- ▶ Hypothèse du testeur : la pile allouée au processus est trop petite ↳ changer la taille (`ulimit`) ↳ **OK**
- ▶ Hypothèse pas cohérente avec les observations (limite imposée par le système, pas par le compilateur)
- ▶ Source du problème : *inlining* de `f()` dans `g()`



Accumuler les observations (2/2)

- ▶ Observer attentivement les conséquences de l'erreur avant d'imaginer la cause
- ▶ Déterminer tous les détails de l'erreur visible (réduction des hypothèses)
- ▶ Instrumenter le code pour extraire de l'information
- ▶ L'imagination doit guider la recherche d'un bug, pas déterminer le coupable



- ▶ Rechercher les sources d'erreurs classiques ([Code Complete, 2nd Ed. Checklist](#))
- ▶ Éliminer les endroits où ne se trouve pas le bug (et non ceux où « il ne peut pas se trouver »)
- ▶ Attention aux effets de bord et aux instabilités (distance entre le bug et le symptôme)



Modifier un seul élément à la fois

- ▶ Isoler les éléments clés intervenant dans la survenue du symptôme
- ▶ Chaque test doit valider un seul élément à la fois
- ▶ Garder en tête ce qui a changé depuis la fois où le symptôme n'apparaissait pas



- ▶ Noter par écrit chaque test fait, dans quel ordre et le résultat
- ▶ Noter les détails de l'environnement



- ▶ Problème sur une fonction `badfun()` : lancement d'une exception indépendamment de la valeur du paramètre d'entrée
- ▶ Testeur : test avec le fichier `test.cpp` : pas d'exception

```
#include <iostream>
#include <exception>
using namespace std;

extern int badfun(int);

int main(void)
{
    try {
        // [...]
        int x = badfun(45);
    } catch (std::exception& e) {
        cout << e.what() << endl;
    }
}
```



Tester les hypothèses simples

- ▶ Problème sur une fonction `badfun()` : lancement d'une exception indépendamment de la valeur du paramètre d'entrée
- ▶ Testeur : test avec le fichier `test.cpp` : pas d'exception

```
#include <iostream>
#include <exception>
using namespace std;

extern int badfun(int);

int main(void)
{
    try {
        // [...]
        int x = badfun(45);
    } catch (std::exception& e) {
        cout << e.what() << endl;
    }
}
```

- ▶ « `test` » : commande interne de *bash*



Tester les hypothèses simples

- ▶ Problème sur une fonction `badfun()` : lancement d'une exception indépendamment de la valeur du paramètre d'entrée
- ▶ Testeur : test avec le fichier `test.cpp` : pas d'exception

```
#include <iostream>
#include <exception>
using namespace std;

extern int badfun(int);

int main(void)
{
    try {
        // [...]
        int x = badfun(45);
    } catch (std::exception& e) {
        cout << e.what() << endl;
    }
}
```

- ▶ « `test` » : commande interne de *bash*
- ▶ Vérifier tous les éléments à partir du début
- ▶ Vérifier toutes les hypothèses de travail



Prendre un avis extérieur

Experience is the name we give to our mistakes.
— Oscar Wilde

- ▶ Observations par un esprit neuf
- ▶ Recours à un expert dans un domaine précis
- ▶ Écoute de personnes expérimentées
- ▶ Donner les symptômes pas les théories les expliquant



Vérifier l'effet du correctif



Wubba Wubba Dance (Super Grover et l'ordinateur)



- ▶ Instrumentation du code
- ▶ Compilateur (options de gcc « `-pedantic` », « `-Wall` », ...)
- ▶ Gestionnaires de bugs
- ▶ Outils de débogage
 - ▶ `gdb` (débogueur en ligne de commande)
 - ▶ `ddd` (interface graphique pour `gdb`)
 - ▶ `valgrind` (cadre logiciel pour l'instrumentation : `memcheck`, `cachegrind`, ...)



Instrumentation du code (1/2)

But : faciliter le débogage futur

- ▶ Utilisation d'assertions (`assert()`)

```
#include <assert.h>
// [...]
void trier_tableau(int T[], unsigned int sz)
{
    assert(sz != 0);
    // [...]
}
```

- ▶ Macro `assert()` dépendant de la macro `NDEBUG` (au moment de l'inclusion de `assert.h`)

Attention : l'instrumentation peut faire apparaître/disparaître un bug



Instrumentation du code (2/2)

- ▶ Compilation conditionnelle de code pour l'affichage (*logging*) vers un canal séparé de `stdout`

```
#include <stdio.h>
// [...]
#ifndef DEBUG
#define DBGLOG(fmt, ...)
    do {
        fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, \
                __LINE__, __func__, __VA_ARGS__);
    } while (0)
#endif

int main(void)
{
    int i = 4;
    // [...]
    DBGLOG("i vaut %d\n", i);
}
```

- ▶ Affichage dépendant de la macro DEBUG



Gestionnaire de bugs

- ▶ GNATS
- ▶ Bugzilla
- ▶ Trac

Summary: contains all of the words/strings

Search

Product:	Component:	Version:	Target:
abakus adept aki akire	*base.kde.org 802.1x AFT AT Envelope	0.0.1 0.0.2 0.0.4 n.1	0.8.0 (KDE 4.3.0) 0.8.1 (KDE 4.3.1) n.9.0 (KDE 4.4.0)

A Comment: contains the string

The URL: contains all of the words/strings

Keywords: contains all of the keywords

Status:	Resolution:	Severity:	Priority:	Hardware:	OS:
UNCONFIRMED NEW ASSIGNED REOPENED RESOLVED WONTFIXINFO	FIXED INVALID WONTFIX LATER REMIND FUTUREDATE	critical grave major crash normal minor wishlist	VHI HI NOR LO VLO	Unlisted Binaries Compiled Sources Caldera RPMs Connectiva RPMs Debian stable Debian testing	Linux FreeBSD NetBSD OpenBSD AIX HP-UX

Email Addresses, Bug Numbers, and Votes

Any of:

the bug assignee
 the reporter
 a CC list member
 a commenter

contains

Any of:

the bug assignee
 the reporter
 a CC list member
 a commenter

contains

Bug Changes

Only bugs changed between: and Now (YYYY-MM-DD or relative dates)

where one or more of the following changed:

[Bug creation]
Alias
Assignee

< >



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

int main(void)
{
    int e;

    printf("Valeur ? ");
    scanf("%d",e);

    printf("Valeur choisie: %d",e);

}

% gcc -o bad bad.c
% ./bad
Valeur ? 4
Segmentation fault
% gcc -g -o bad bad.c
% gdb -q bad
Reading symbols from /home/goualard/enseignement/2012-2013/Master/X7II030_outils-developpement/slides/bad
(gdb) b 9
Breakpoint 1 at 0x7ffff7abde30: file vfscanf.c, line 9.
(gdb) run
Starting program: /home/goualard/enseignement/2012-2013/Master/X7II030_outils-developpement/slides/bad

Breakpoint 1, main () at bad.c:9
9         scanf("%d",e);
(gdb) step
(gdb) next
```



ddd : DDD: /home/goualard/enseignement/2012-2013/Master/X71030_outils-developpement/slides/bohrbug.c

File Edit View Program Commands Status Source Data Help

(1) main()

// Minimum function

```
int minimum(int T[], unsigned int sz)
{
    assert(sz != 0);
    if (sz == 1)
        return T[0];
    else
    {
        int minimum = T[0];
        for (int i=1; i < sz-1; ++i)
            if (T[i] < minimum)
                minimum = T[i];
    }
    return minimum;
}

int main(void)
{
    printf("2d\n", minimum({int[]{2,5,6,4}, 4}));
    printf("2d\n", minimum({int[]{3,5,6,1}, 4}));

    Dump of assembler code for function minimum:
    0x00000000004005c: <0>    push   %rbp
    0x00000000004005d: <1>    mov    %rbp,%rbp
    0x00000000004005f: <2>    sub    $0x20,%rbp
    0x0000000000400574: <3>    mov    %rdi,-0x18(%rbp)

(gdb) run
2
3

Program exited normally.
(gdb) clear bohrbug.c:14
(gdb) break bohrbug.c:11
Breakpoint 2 at 0x4005a8: file bohrbug.c, line 11.
(gdb) run

Breakpoint 2, minimum (T=0x2fffffffdbf0, sz=4) at bohrbug.c:11
(gdb) graph minor minimum
(gdb) graph display 1
(gdb) graph display *1 dependent on 2
(gdb) graph undisplay 2
(gdb) graph undisplay 3
```

Registers

Register	Value	Type
rax	0x7fffffffdbf0	1f
rbx	0x0	0
rcx	0x400630	4195384
rdx	0x7fffffffda08	1f
rsi	0x0	0
rdi	0x7fffffffdbf0	1f
rbp	0x7fffffffdb80	0x
rsp	0x7fffffffdbc0	0x
r8	0x7ffff7dd5600	1f
r9	0x7ffff7def10	1f
r10	0x7ffff7fd4750	1f
r11	0x7ffff7fa2e00	1f

Integer registers All registers

Run Interrupt Step Step Next Nexti Until Finish Cont Kill Up Down Undo Redo Edit Make



Scénario

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

using namespace std;

struct MyType {
    double v1;
    char c;
    double v2;
};

int main(void)
{
    MyType V[10];
    char *ptr = &(V[0].c);

    // Remplissage de V
    // [...]
    for (int i=0; i < 10; ++i) {
        *ptr = 'a';
        ptr += 16;
    }
    // [...]
    for (int i=0; i<10; ++i) {
        cout << V[i].c << " ";
    }
    cout << endl;
}
```

Sortie : a a a p a %



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

using namespace std;

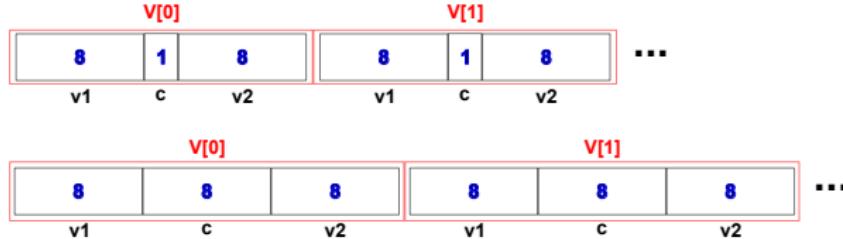
struct MyType {
    double v1;
    char c;
    double v2;
};

int main(void)
{
    MyType V[10];
    char *ptr = &(V[0].c);

        // Remplissage de V
        // [...]
        for (int i=0; i < 10; ++i) {
            *ptr = 'a';
            ptr += 16;
        }
        // [...]
        for (int i=0; i<10; ++i) {
            cout << V[i].c << " ";
        }
        cout << endl;
}
```

Sortie : a a a p a %

► Padding



Fin du cours