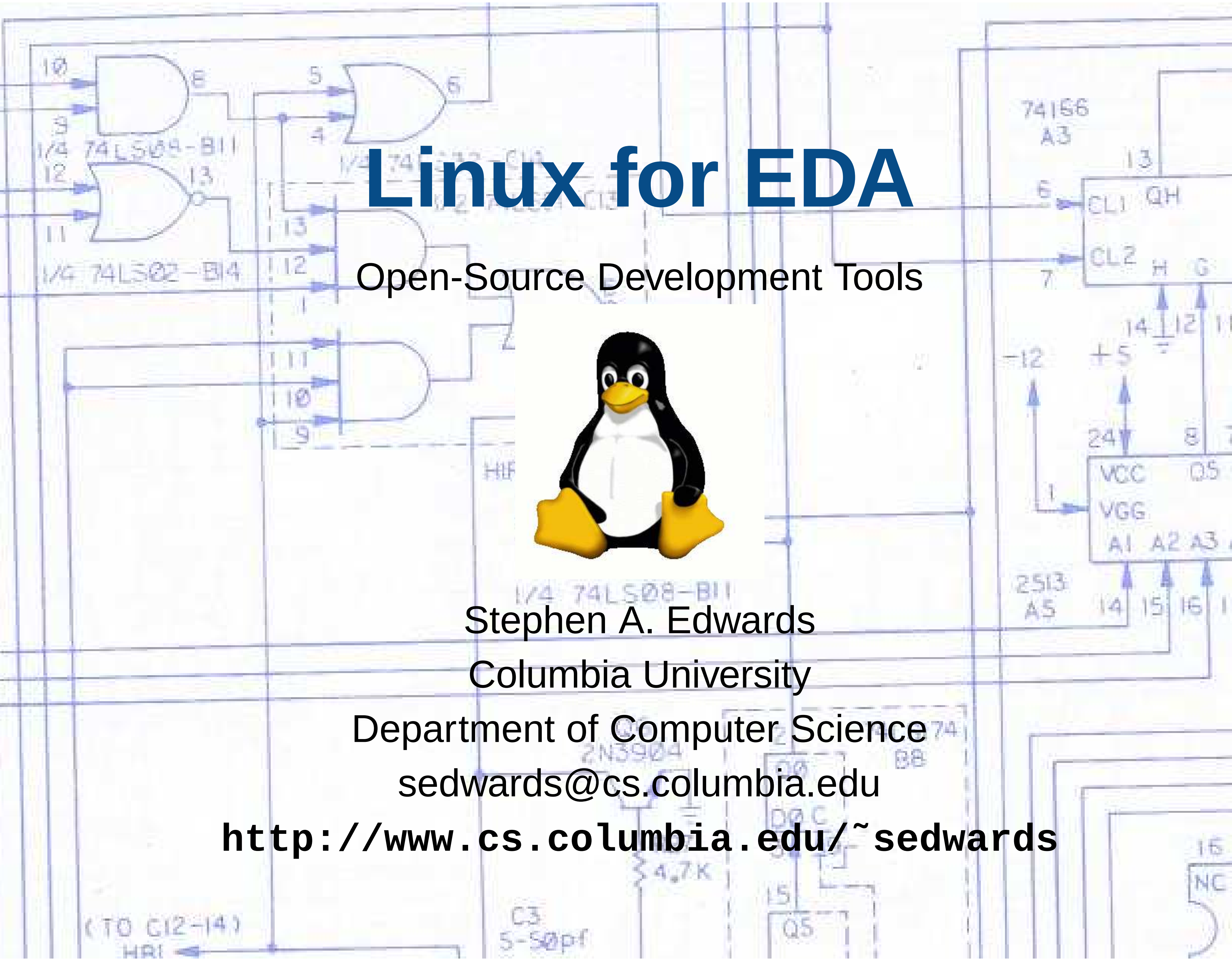


Linux for EDA

Open-Source Development Tools

Stephen A. Edwards
Columbia University
Department of Computer Science
sedwards@cs.columbia.edu
http://www.cs.columbia.edu/˜sedwards

Espresso

An example project: Berkeley's *Espresso* two-level minimizer.

18k LOC total in 59 .c files and 17 .h files.

Written in C in pre-ANSI days.

Ported extensively. Supports ANSI C and K&R C on VAX, SunOS 3 & 4, Ultrix, Sequent, HPUX, and Apollo.

Autoconf

A modern approach to cross-platform portability.

How do you compile a program on multiple platforms?

- Multiple code bases
- Single code base sprinkled with platform-checking **`#ifdef`**s
- Single code base with platform-checking **`#ifdef`**s confined to a few files: (**`#include "platform.h"`**)
- Single code base with feature-specific **`#ifdef`**s computed by a script that tests each feature

Autoconf

Basic configure.ac:

```
AC_INIT(espresso.c)
AM_INIT_AUTOMAKE(espresso, 2.3)
AC_PROG_CC
AC_LANG(C)

AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

Autoconf

```
$ autoconf
$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
...
checking dependency style of gcc... gcc3
configure: creating ./config.status
config.status: creating Makefile
config.status: executing depfiles commands
$ ls Makefile
Makefile
$
```

Espresso’s port.h (fragment)

```
#ifdef __STDC__
#include <stdlib.h>
#else
#ifdef hpux
extern int abort();
extern void free(), exit(), perror();
#else
extern VOID_HACK abort(), free(), exit(), perror();
#endif /* hpux */
extern char *getenv(), *malloc(), *realloc(), *calloc();
#ifdef aiws
extern int sprintf();
#else
extern char *sprintf();
#endif
extern int system();
extern double atof();
extern int sscanf();
#endif /* __STDC__ */
```

Checking for features in Autoconf

```
AC_HEADER_STDC
AC_CHECK_FUNCS(abort free exit qsort)

#if STDC_HEADERS
#  include <stdlib.h>
#else
#  if !HAVE_ABORT
extern void abort();
#  endif
#  if !HAVE_FREE
extern void free(void *);
#  endif
#  if !HAVE_EXIT
extern void exit(int);
#  endif
#  if !HAVE_QSORT
extern qsort();
#  endif
#endif
```

Automake

Makefiles for large projects tend to be fussy.

Often, common patterns for building libraries, executables, distributions, clean-up, etc.

Many people use ad-hoc templates or includes.

Automake a way to address many of these problems.

sources.redhat.com/autobook/

Automake

Knows about building executables, libraries, and distributions, installation, generating dependencies, creating tags, running tests, and recursive make.

Makefile.am:

```
bin_PROGRAMS = espresso

espresso_SOURCES = black_white.c exact.c \
mm_int.h sparse.c expand.c sparse.h \
canonical.c gasp.c opo.c sparse_int.h \
... copyright.h

man_MANS = espresso.1 espresso.5
EXTRA_DIST = $(man_MANS)
```

Generates a 517-line Makefile with over 60 rules.

Cygwin

`http://cygwin.com/`

A port of virtually all Gnu libraries and tools to the Windows environment.

What to run if you’re forced to run Windows.

gcc, emacs, glibc, make, cvs, bash, etc.

Even an X server: xfree86.cygwin.com

Eclipse

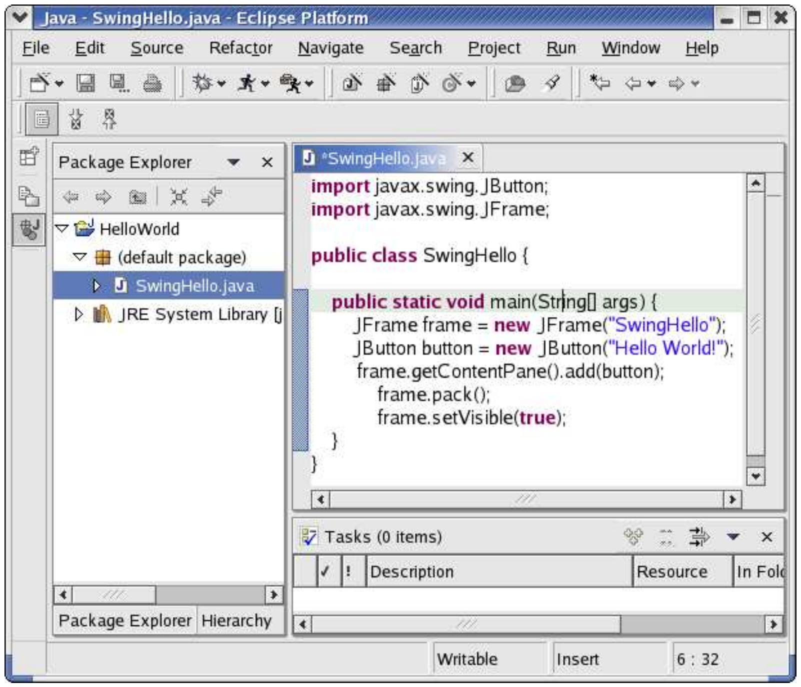
`http://eclipse.org/`

Platform for building integrated development environments.

Written in Java by IBM et al.

Extensible through plug-ins.

Currently supports Java development best.



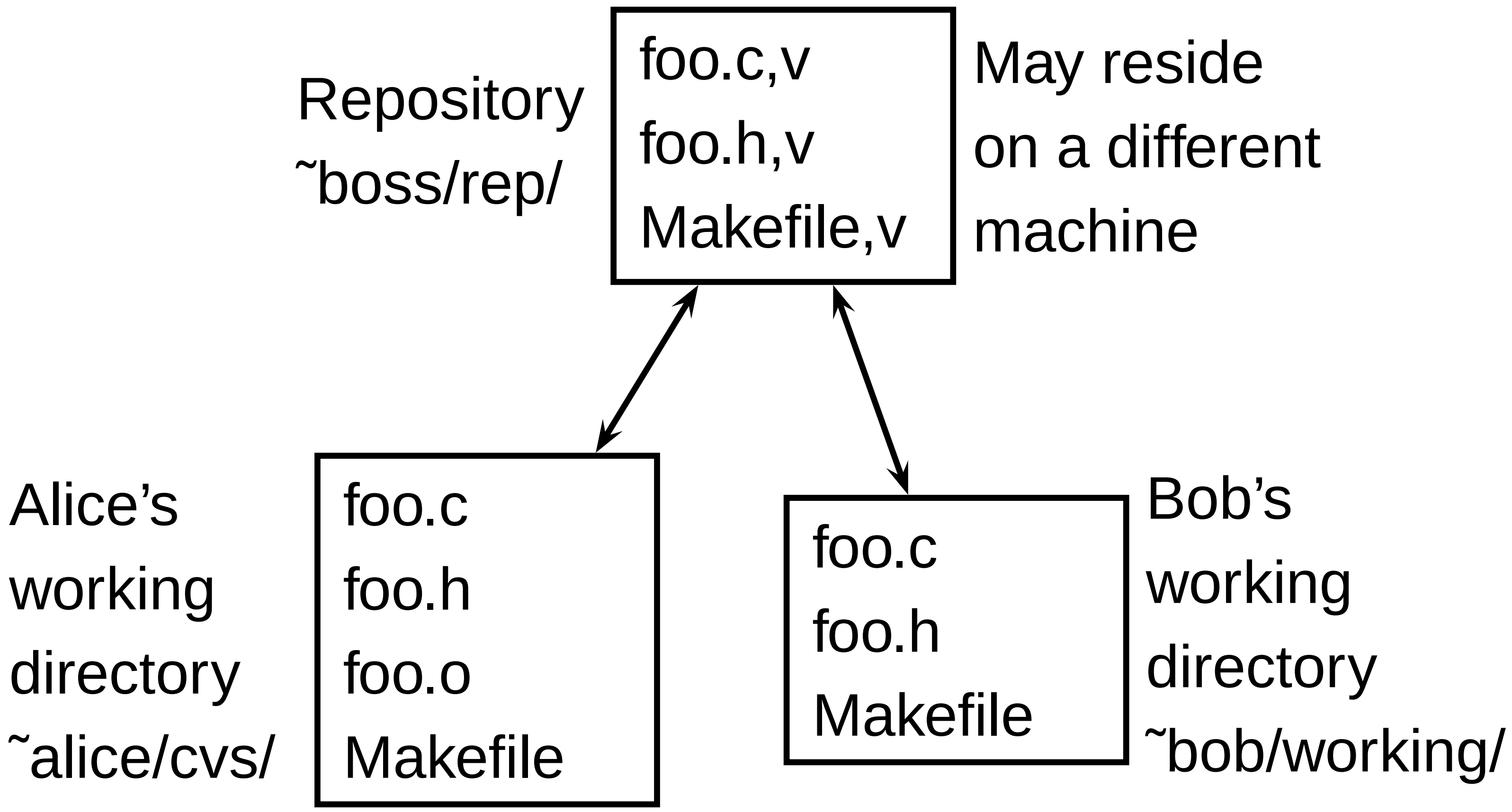
```
import javax.swing.JButton;
import javax.swing.JFrame;

public class SwingHello {

   public static void main(String[] args) {
      JFrame frame = new JFrame("SwingHello");
      JButton button = new JButton("Hello World!");
      frame.getContentPane().add(button);
         frame.pack();
         frame.setVisible(true);
   }
}
```

CVS: Concurrent Versioning System

Model:



CVS: Remote Access

Repository need not reside on local machine. Can use **`ssh`** for remote authentication and communication.

```
$ export CVS_RSH=ssh
$ cvs -d sedwards@arthur:/mnt/repository \
  checkout foo
Password:
$ ls -F
foo/
$
```

I use CVS to keep files synchronized among my various home and work desktops and notebooks.

CVS Features

Files are not locked when checked out (c.f. RCS).

Simultaneous modifications possible; text files merged when changes committed.

Simple merging (adding a function, modifying two different places in the file) usually works; CVS warns on failure.

Has all the usual confusing multiple development sequences, global version marking, etc.

More features than RCS (e.g., remote update, merging).

Less fancy, transparent than (commercial) ClearCase. CVS has better multi-site support and you have a fighting chance of understanding it.

GCC: “Gnu Compiler Collection”

`http://gcc.gnu.org/`

Frontends	**Backends**	
C	Alpha	PDP-11
C++	ARM	RS6000
Objective C	AVR	SuperH
Fortran 77	HPPA	SPARC
Java	x86	VAX
Ada	i960	Xtensa
Pascal	ia64	
Cobol	68k	
Modula-2	68hc11	
Modula-3	MIPS	
VHDL	PowerPC	

C Front end standards

ANSI C (1989)/ISO C (1990):

```
gcc -ansi
gcc -std=c89
gcc -std=iso9899:1990
```

ISO/IEC 9899:1999: “C99”

```
gcc -std=c99
gcc -std=iso9899:1999
```

Mostly supported.

The C99 Standard

```
int main(int argc, char argv[])
{
  int a;
  a = 1;
  int b; /* Declarations mixed with statements */
  _Bool bb; // New built-in type
  long long c; // At least 64 bits
  char myargv[argc]; // Variable-length auto array
  for (int i = 0 ; i < 10 ; i++) ; // local declaration
  struct { int x, y; } p = { .x = 1, .y = 2 };
  int *restrict p1, *restrict p2; // p1 and p2 assumed !=
}

inline int min(int x, int y) { return x < y ? x : y; }
```

C++

With G++ 3.0, most C++ features finally work:

- The standard template library: Sets, Maps, Vectors,...
- Standard header files, e.g., **`#include <vector>`**
- Namespaces
- RTTI, e.g., **`dynamic_cast<Foo*>(p)`**

GCC and Java

`gcj` compiles Java programs to (big) executables.

Implements JDK 1.2 (Sun up to JDK 1.4)

libgcj largely compatible, but missing, e.g., **`java.awt`**.

```
class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
$ gcj --main=Hello -o hello Hello.java
$ ./hello
Hello World!
$ file hello
hello: ELF 32-bit LSB executable, Intel 80386
```

The Intel C++ Compiler 7.1 for Linux

`http://www.intel.com/software/products/global/eval.htm`

Not technically open-source, but available.

Free, unsupported, non-commercial version plus a commercial version.

Offers extra performance. Claims as much as 30% over gcc 3.2

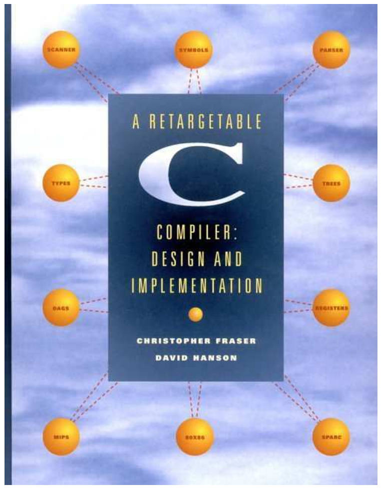
Interesting feature: profile-driven optimization

Lcc

www.cs.princeton.edu/software/lcc

Described in David R. Hanson and Christopher W. Fraser, *A Retargetable C Compiler: Design and Implementation* (Addison-Wesley, 1995)

Non-optimizing, but very fast compilation.



Performance comparison

Running *Espresso* on all the “hard” examples on a Pentium 4 1.7 GHz

lcc	gcc			Icc		
	-g	**-O**	**-O7**	**-g**	**-O3**	**-prof_use**
41s	43s	23s	24s	43s	18s	17s

Electric Fence

Performs purify-like checking: array bounds checking and accessing unallocated/freed memory.

Modified malloc library that puts an empty page after or before each block. Free actually deallocates the page. Illegal accesses cause a segmentation fault.

The “ef” command invokes its argument with the Electric Fence library. Can also link executables against it.

```
$ ef ./espresso < pdc > result
$
```

On this example, Espresso runs without any access violations.

Electric Fence

```
#include <stdlib.h>
int main() {
  int *p = malloc(sizeof(int) * 10);
  p[0] = 0; /* OK */
  p[9] = 1; /* OK */
  p[10] = 2; /* ILLEGAL */
  return 0;
}
```

```
$ cc -g -o access access.c -lefence
$ gdb ./access
(gdb) run
Program received signal SIGSEGV, Segmentation fault.
0x08048477 in main () at access.c:9
9         p[10] = 2; /* ILLEGAL */
(gdb)
```

Splint

A C lint checker from the University of Virginia. Looks for security vulnerabilities and coding mistakes. Based on static analysis. Can be run without annotations, but works better with them.

`http://splint.org/`

Finds plenty of problems with everything.

Splint on Espresso

```
$ splint verify.c
.
.
.
verify.c:93:3: Index of possibly null pointer permute: permute
A possibly null pointer is dereferenced. Value is either the result of a
function which may return null (in which case, code should check it is not
null), or a global, parameter or structure fi eld declared with the null
qualifi er. (Use -nullderef to inhibit warning)
verify.c:88:15: Storage permute may become null
.
.
.
```

```
88  permute = ALLOC(int, PLA2->F->sf_size);
89  for(i = 0; i < PLA2->F->sf_size; i++) {
90      labi = PLA2->label[i];
91      for(j = 0; j < PLA1->F->sf_size; j++) {
92          if (strcmp(labi, PLA1->label[j]) == 0) {
93              permute[npermute++] = j;
```

gprof: Runtime Profiling

```
$ gcc -o espresso -pg *.c
$ espresso < pdc
$ gprof espresso
  % cumulative self
time   seconds seconds  calls name
34.74   0.74   0.74   153981 massive_count
 6.57   0.88   0.14     2926 elim_lowering
 6.10   1.01   0.13    11082 cofactor
 5.63   1.13   0.12     2514 setup_BB_CC
 5.16   1.24   0.11   204420 scofactor
 4.23   1.33   0.09  2598408 full_row
 4.23   1.42   0.09  1675360 malloc
 3.76   1.50   0.08   698471 set_or
 3.76   1.58   0.08   569514 sm_insert
 2.82   1.64   0.06   133195 taut_special_cases
 2.82   1.70   0.06     2889 essen_parts
 1.88   1.74   0.04  1675360 free
```

rpm: Redhat Package Manager

Database tracks package file ownership for convenient uninstalls & upgrades.

```
$ rpm -i automake-1.6.3-5.rpm
$ rpm -qi automake
Name        : automake  Relocations: (not relocateable)
Version     : 1.6.3     Vendor: Red Hat, Inc.
Release     : 5         Build Date: Mon 27 Jan 2003
...
$ rpm -ql automake
/usr/bin/aclocal
/usr/bin/aclocal-1.6
/usr/bin/automake
/usr/bin/automake-1.6
/usr/share/aclocal
/usr/share/aclocal-1.6
/usr/share/aclocal-1.6/amversion.m4
/usr/share/aclocal-1.6/as.m4
...
```

rpm: Writing a .spec file (1)

The .spec file describes unpackaging, compiling, installing, and cleaning up. Works well with autoconf.

```
Summary: A two-level logic minimizer
Name: espresso
Version: 2.3
Release: 1
License: BSD
Group: Applications/Engineering
URL: http://www.cs.columbia.edu/~sedwards
Source0: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-buildroot
Packager: Stephen A. Edwards

%description
Espresso minimizes a two-level logic function ...

%prep
%setup -q
```

rpm: Writing a .spec file (2)

```
%build
%configure
make

%install
rm -rf $RPM_BUILD_ROOT
%makeinstall

%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root,-)
%doc
%{_bindir}/*

/usr/share/man/man1/espresso.1.gz
/usr/share/man/man5/espresso.5.gz
```

rpm: Making a package (1)

Simple once you have written the .spec file.

```
$ cat ~/.rpmmacros
%_topdir                        /home/sedwards/redhat
%_tmppath                       /var/tmp/rpm
$ ls -F ~/redhat
BUILD/  RPMS/  SOURCES/  SPECS/  SRPMS/
$ cp espresso-2.3-1.spec ~/redhat/SPECS
$ make dist
...
$ cp espresso-2.3.tar.gz ~/redhat/SOURCES
```

rpm: Making a package (2)

```
$ cd ~/redhat/SPECS
$ rpmbuild -ba espresso-2.3-1.spec
...configure
...make
...make install
$ ls ../RPMS/i386/
espresso-2.3-1-i386.rpm
$ rpm -qlp ../RPMS/i386/espresso-2.3-1.i386.rpm
/usr/bin/espresso
/usr/share/man/man1/espresso.1.gz
/usr/share/man/man5/espresso.5.gz
$
```

Lex, Yacc, and ANTLR

- Lex and Yacc have been around forever for writing scanners and parsers. Both generate C.

  Lex takes regular expressions; Yacc context-free grammars.
- Flex and Bison: newer replacements for lex and yacc.
- ANTLR (antlr.org) is a comparatively new compiler generation tool. Generates C++, Java, or C#.

  Integrated scanner, parser, and tree-walker language.

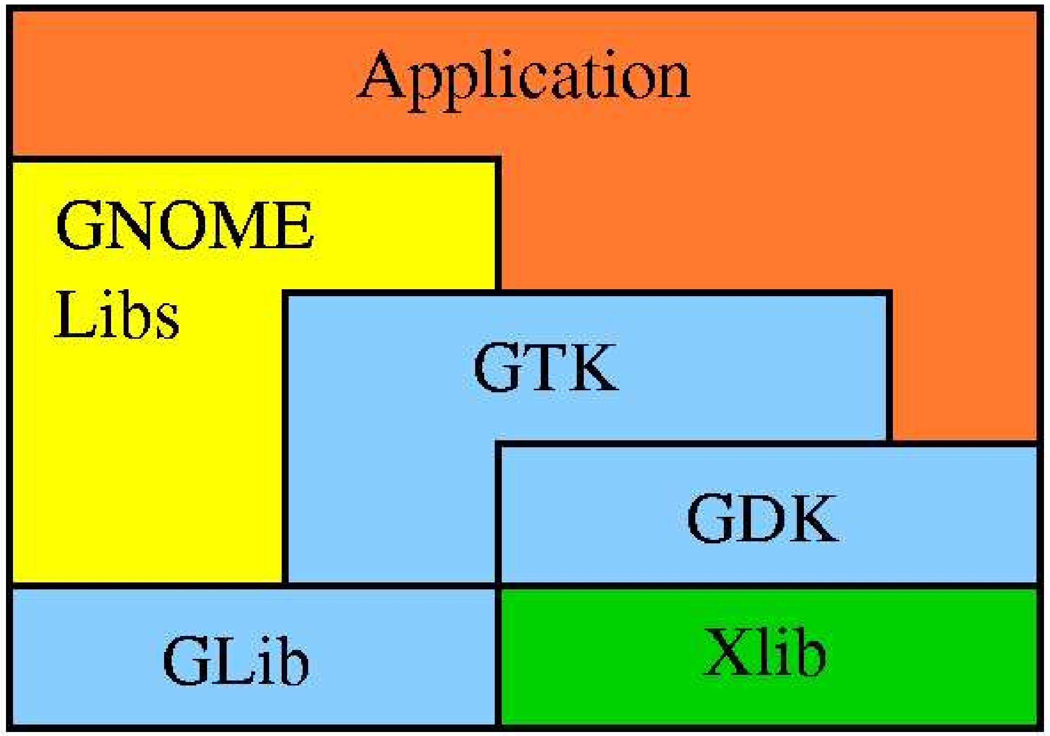
  Generates top-down parsers. Certain grammars easier to parse, others harder.

  Easier error recovery. Uses exceptions.

GUIs: Gnome

C-based toolkit designed for X.

developer.gnome.org/doc/tutorials



Gnome Hello World

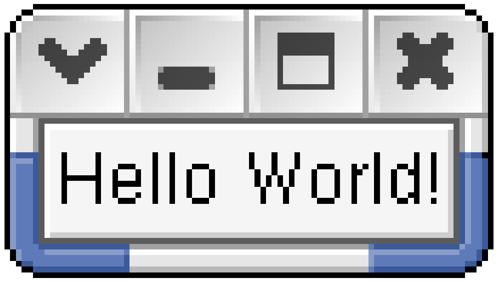
```c
#include <gnome.h>

int main(int argc, char *argv[])
{
  GtkWidget *app;
  GtkWidget *button;
  GtkWidget *hbox;

  gnome_init("gnome-hello", "0.1", argc, argv);
  app = gnome_app_new("gnome-hello", "GNOME Hello World");
  hbox = gtk_hbox_new(FALSE,5);
  gnome_app_set_contents(GNOME_APP (app), hbox);
  button = gtk_button_new_with_label("Hello World!");
  gtk_box_pack_start(GTK_BOX(hbox), button, FALSE, FALSE, 0);
  gtk_widget_show_all(app);
  gtk_main();
  return 0;
}
```

Gnome Hello World

```
$ gcc -o gnome-hello gnome-hello.c \
   ‘gnome-config --cflags --libs gnome gnomeui‘
$ ./gnome-hello
```


Hello World!

GUIs: Qt

Developed by Trolltech, a Norwegian company with an unusual business model. Offers identical commercial and open-source versions of their system.

Basically, if you want to sell your product, you pay them, otherwise it’s free.

Remarkable: C++ WIMP environment that supports all major platforms: X, Windows, Macintosh.

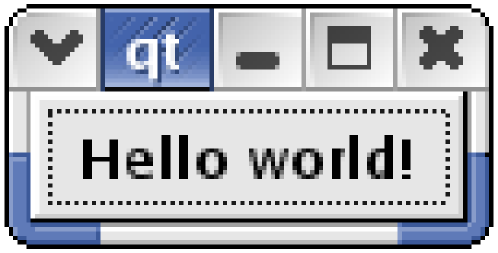
Qt Hello World

```
#include <qapplication.h>
#include <qpushbutton.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    QPushButton hello( "Hello world!", 0 );
    hello.resize( 100, 30 );
    a.setMainWidget( &hello );
    hello.show();
    return a.exec();
}
```

Qt Hello World

```
$ g++ -o qt-hello qt-hello.cpp \
  -I/usr/lib/qt-3.1/include \
  -L/usr/lib/qt-3.1/lib -lqt -lfreetype
$ ./qt-hello
```



KDE Hello World

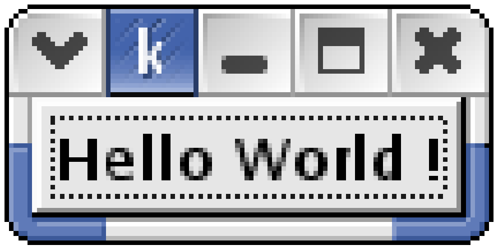
C++-based toolkit. Built on Qt for X.

```
#include <kapp.h>
#include <klocale.h>
#include <qpushbutton.h>

int main(int argc, char **argv)
{
  KApplication a( argc, argv , "p2");
  QPushButton *hello =
     new QPushButton( i18n("Hello World !"), 0 );
  hello->setAutoResize( TRUE );
  QObject::connect( hello, SIGNAL(clicked()),
                    &a, SLOT(quit()) );
  a.setMainWidget( hello );
  hello->show();
  return a.exec();
}
```

KDE Hello

```
$ g++ -o kde-hello kde-hello.cpp \
  -I/usr/include/kde -I/usr/lib/qt-3.1/include
  -lkdeui -lkdecore -lfreetype
$ ./kde-hello
```



Swing Hello World

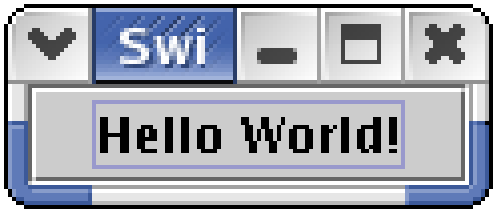
The now-standard GUI for Java.

```
import javax.swing.*;

public class SwingHello {
    public static void main(String[] args) {
        JFrame frame = new JFrame("SwingHello")
        JButton button = new JButton("Hello Wor
        frame.getContentPane().add(button);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Swing Hello World

```
$ javac SwingHello.java
$ java SwingHello
```



Hello World!

GUIs compared

Toolkit	Language	Platforms
Gnome	C	X11
Qt	C++	X11, Windows, MacOS, Qtopia
KDE	C++	X11
Swing	Java	X11, Windows, MacOS

Summary

- Build tools: Autoconf, Automake
- Development environments: Cygwin, Eclipse
- Version control: CVS
- Compilers: gcc, icc, lcc
- Debugging aids: Electric Fence, splint
- Profiliers: gprof
- Packagers: rpm
- Code generators: lex, yacc, ANTLR
- GUIs: Gnome, Qt, KDE, Swing