

# **“Introduzione alla programmazione in ambiente GNU/Linux (con particolare riferimento al linguaggio C)”**

Marco 'Embyte' Balduzzi <[embyte@spine-group.org](mailto:embyte@spine-group.org)>

**MOCA METROLOGRAFIX  
CAMP 2004**

# glibc: la libreria GNU C (1)

- glibc: cos'è?

Libreria che fornisce funzioni standard quali I/O, gestione della memoria, manipolazione delle stringhe link nel rispetto degli standard.

- standard

\* ISO C: standard adottato dall'American National Standards Institute (ANSI) come “American National Standard X3.159-1989” (ANSI C) e poi dall'International Standardization Organization (ISO) come ISO/IEC 9899:1990

## glibc: la libreria GNU C (2)

- \* POSIX: acronimo di “Portable Operating System Interface for Computer Environments” (ISO/IEC 9945), conosciuto anche come ANSI/IEEE Std 1003.

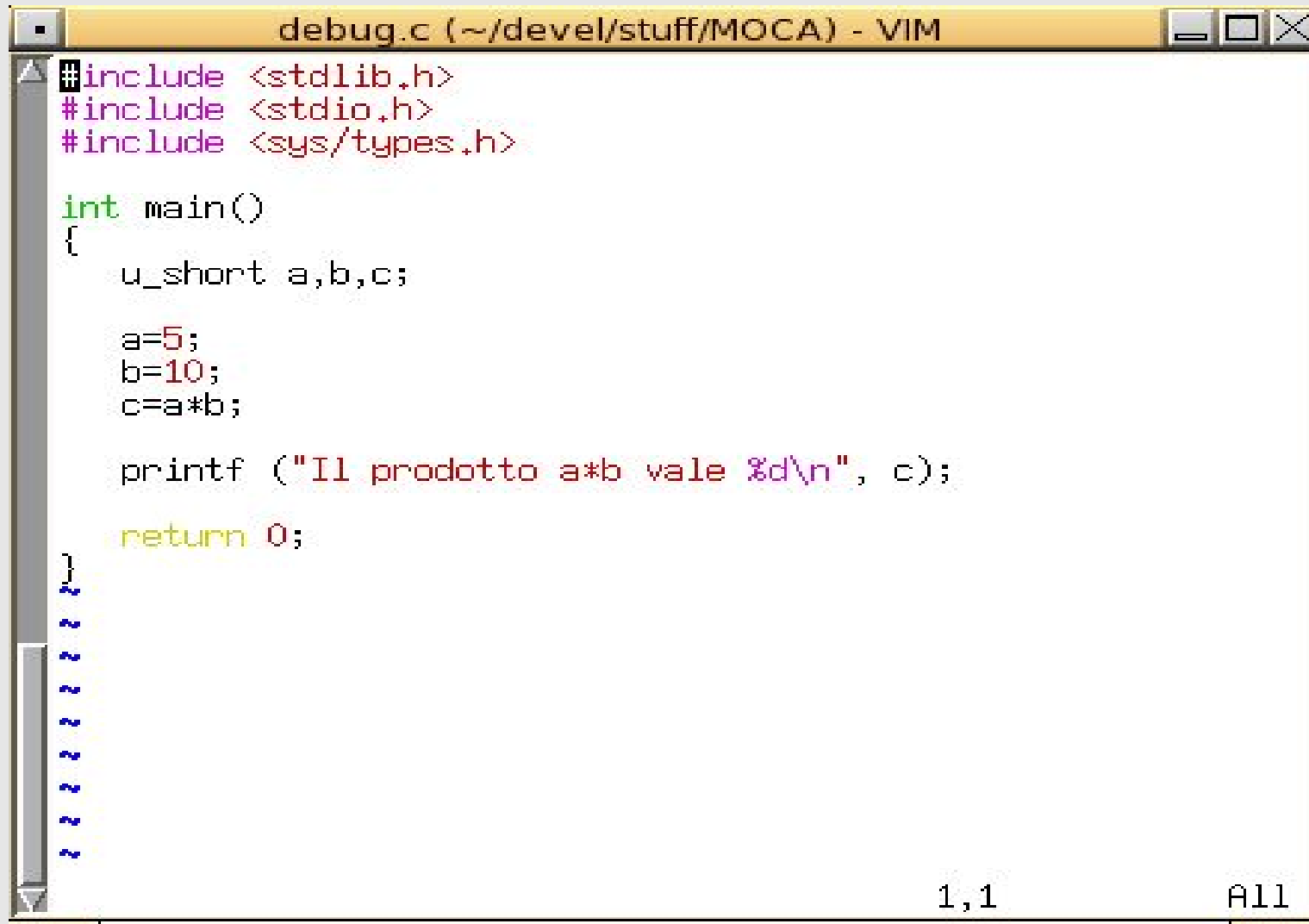
E' lo standard usato dai sistemi unix e unix-like.

Ridefinisce e aggiunge nuove caratteristiche e funzioni rispetto all'ANSI C: filesystem, processi, terminali.

- \* Berkeley Unix: estensione di BSD 4.2, 4.3, 4.4 e SunOS  
Aggiunge alcune caratteristiche: link simbolici, socket BSD, select(), segnali BSD.
- \* System V Interface Description" (SVID): estensione dello standard POSIX da parte della AT&T

# gli editor: VIM (Vi Improved) (1)

E' consigliabile utilizzare editor che supportano l'highlighting del codice



The image shows a screenshot of a VIM editor window. The title bar reads "debug.c (~/.devel/stuff/MOCA) - VIM". The editor displays a C program with syntax highlighting. The code is as follows:

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

int main()
{
    u_short a,b,c;

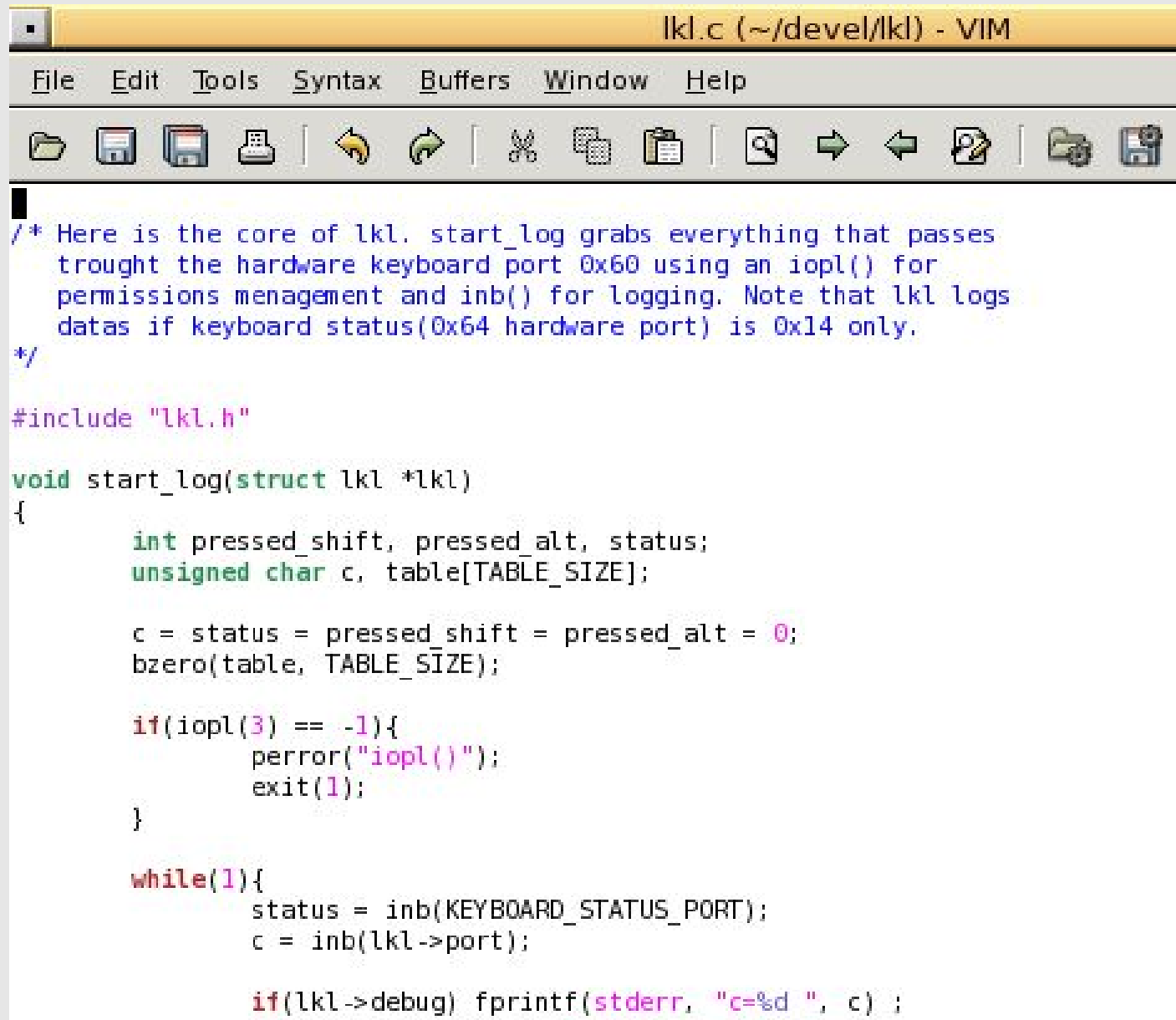
    a=5;
    b=10;
    c=a*b;

    printf ("Il prodotto a*b vale %d\n", c);

    return 0;
}
```

The status bar at the bottom right shows "1,1" and "All".

# gli editor: Xvim (vim -g) (2)



```
lkl.c (~/devel/lkl) - VIM
File Edit Tools Syntax Buffers Window Help
[*] Here is the core of lkl. start_log grabs everything that passes
trought the hardware keyboard port 0x60 using an iopl() for
permissions management and inb() for logging. Note that lkl logs
datas if keyboard status(0x64 hardware port) is 0x14 only.
*/
#include "lkl.h"

void start_log(struct lkl *lkl)
{
    int pressed_shift, pressed_alt, status;
    unsigned char c, table[TABLE_SIZE];

    c = status = pressed_shift = pressed_alt = 0;
    bzero(table, TABLE_SIZE);

    if(iopl(3) == -1){
        perror("iopl()");
        exit(1);
    }

    while(1){
        status = inb(KEYBOARD_STATUS_PORT);
        c = inb(lkl->port);

        if(lkl->debug) fprintf(stderr, "c=%d ", c) ;
    }
}
```

# gli editor: JED (3)

```
F10 key ==> File Edit Mode Search Buffers Windows System Help
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>

int main()
{
    u_short a,b,c;

    a=5;
    b=10;
    c=a*b;

    printf ("Il prodotto a*b vale %d\n", c);

    return 0;
}

-----+ (Jed 0.99.16) Emacs: debug.c (C) All 11:06am -----
```

CppSupportPart m(const CodeModelItem\* item, bool shortDescription)

```

codeModel()->removeFile( codeModel()->fileByName(fileName) );
}

bool CppSupportPart::isValidSource( const QString& fileName ) const
{
    QFileInfo fileInfo( fileName );
    QString path = URLUtil::canonicalPath( fileInfo.absFilePath() );

    return project()->isProjectFile( path )
        && (isSource( path ) || isHeader( path ))
        && !QFile::exists(fileInfo.dirPath(true) + "/.kdev_ignore");
}

QString CppSupportPart::formatModelItem( const CodeModelItem *item, bool shortDesc
{
    if (item->isFunction() || item->isFunctionDefinition() )
    {
        const FunctionModel *model = static_cast<const FunctionModel*>(item);
        QString function;
        QString args;
        ArgumentList argumentList = model->argumentList();
        for (ArgumentList::const_iterator it = argumentList.begin(); it != argumen
        {
            args.isEmpty() ? args += "" : args += ", ";
            args += formatModelItem(*it).data();
        }
        if( !shortDescription )
            function += (model->isVirtual() ? QString("virtual ") : QString("")) )

        function += model->name() + "(" + args + ")" + (model->isConstant() ? QString
            (model->isAbstract() ? QString(" = 0") : QString("")) );
    }
}

```

languages.cpp

- KParts
- KTextEditor
- TagUtils
- AddAttributeDialog
- AddAttributeDialog
- AddMethodDialog
- AddMethodDialog
- BackgroundParser
- CCConfigWidget
- CCConfigWidget
- ClassGeneratorConfig
- ClassGeneratorConfig
- CodeInformation
- ComputeRecover
- ConfigureProblemReporter
- CppBaseClass
- CppCodeCompletion
- CppCodeCompletion
- CppCodeCompletion
- CppDriver
- CppFunction
- CppNewClassDialog
- CppNewClassDialog
- CppSupportFactory

languages

- ada
  - app\_t...
  - ada...
- doc
- file\_tem...
- bash
  - app\_t...
  - bas...
- doc
- cpp

problemrepo...  
kdevdriver.cpp  
doxydoc.cpp  
createpcsd...  
createpcsd...  
cppsupportp...  
cppsupportf...  
cppsupport\_...  
cppnewclass...  
cppnewclass...  
cppcodecom...  
cppcodecom...

```

NO_STL -DQT_NO_COMPAT -DQT_NO_TRANSLATION -o libkdevcppsupport.la -rpath /opt/kde3.2/lib/kde3 -L/usr/X11R6/lib -L/develop/kde/qt-copy/lib -L/opt/kde3.2/lib -avoid-version -module -no-undefined
-Wl,--no-undefined -Wl,--allow-shlib-undefined -R /opt/kde3.2/lib -R /develop/kde/qt-copy/lib -R /usr/X11R6/lib cppsupportpart.lo cconfigwidget.lo kdevdriver.lo cppcodecompletion.lo problemreporter.lo b
ackgroundparser.lo ast_utils.lo store_walker.lo KDevCppSupportInterface.lo cppsupportfactory.lo tag_creator.lo codeinformationrepository.lo doxydoc.lo cppcodecompletionconfig.lo cppnewclassdlg.lo classgen
eratorconfig.lo subclassingdlg.lo addattributedialog.lo addmethoddialog.lo cppsupport_utils.lo createpcsdialog.lo KDevCppSupportInterface_skel.lo cconfigwidgetbase.lo configproblemreporter.lo cppnewclassd
lgbase.lo classgeneratorconfigbase.lo subclassingdlgbase.lo addattributedialogbase.lo addmethoddialogbase.lo createpcsdialogbase.lo ../../lib/libkdevelop.la ../../lib/catalog/libkdevcatalog.la ../../lib/cpppar
ser/libkdevcppparser.la

```

File	Rev
printing	D
scintilla	D
scripts	D
src	D
tagmanager	D
widgets	D
ABOUT-NLS	1.5
AUTHORS	1.6
COPYING	
ChangeLog	1.436
FUTURE	1.2
HACKING	1.22
INSTALL	1.1.1.1
Makefile	
Makefile.am	1.35

```

66     "Dave Huseby <huseby@shockfusion.com>",
67     "Sebastien Cote <cots01@ge1.usherb.ca>",
68     "Stephen Knight <steven.knight@unh.edu>",
69     NULL
70 };
71
72 static const char *documentors[] = {
73     "Naba Kumar <naba@gnome.org>",
74     "Andy Piper <andy.piper@freeuk.com>",
75     "Biswapesh Chattopadhyay <biswapesh_chatterjee@tcsca1.co.in>",
76     NULL
77 };
78
79 GtkWidget *
80 about_box_new ()
81 {
82     GtkWidget *dialog;
83     GdkPixbuf *pix;
84
85     pix = anjuta_res_get_pixbuf (ANJUTA_PIXMAP_LOGO);
86     dialog1 = gnome_about_new ("Anjuta", VERSION,
87                               _("Copyright (c) Naba Kumar"),
88                               _("Integrated Development Environment"),
89                               authors, documentors, NULL, pix);

```

```

make about.o
gcc -DHAVE_CONFIG_H -I. -I. -I.. -DORBIT2=1 -I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include -I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include -I/usr/include/atk-1.0
about.c: In function `about_box_new':
about.c:86: `dialog1' undeclared (first use in this function)
about.c:86: (Each undeclared identifier is reported only once
about.c:86: for each function it appears in.)
make: *** [about.o] Error 1

```



# gcc: il compilatore GNU (1)

- preprocessing -> compilation -> assembly -> linking
- linguaggi supportati
  - \* C (gcc)
  - \* C++ (g++)
  - \* Fortran (gcc-g77)
  - \* Ada 95 (gcc-gnat)
  - \* Java (gcc-java)
  - \* etc...

Principalmente utilizzato come compilatore C/C++

# gcc: il compilatore GNU (2)

- comandi di base:

- \* compilazione

- \$ gcc -Wall -Idir sorgente.c -o binario

- \* compilazione ANSI

- \$ gcc -Wall -ansi -pedantic sorgente.c -o binario

- \* compilazione per debugging

- \$ gcc -Wall -g sorgente.c -o binario

## gcc: il compilatore GNU (3)

- \* compilazione ottimizzata

```
$ gcc -Wall -O3 -march=athlon sorgente.c -o binario
```

- \* compilazione statica:

```
$ gcc -Wall -static sorgente.c -o binario
```

- \* solo compilazione (no linking):

```
$ gcc -Wall -c sorgente.c (produce sorgente.o)
```

- \* specificare una ulteriore libreria al linker:

```
$ gcc -Wall -Ldir sorgente.c -o binario -l libreria
```

## gcc: gestione delle librerie (4)

un esempio:

```
$ gcc -Wall -I/usr/include/libnet-1.1 -L/usr/lib/net net.c -o net -lnet
```

- ulteriore libreria libnet.a
- ulteriore percorso non standard delle librerie /usr/lib/net (libnet.a si trova lì dentro..)
- ulteriore percorso non standard degli include /usr/include/libnet-1.1

# usare pkg-config (1)

- cosa è pkg-config?

Programma utilizzato per recuperare informazioni sulle librerie installate nel sistema.

- come funziona?

Le informazioni sono contenute in speciali file con estensione .pc presenti in prefix/lib/pkgconfig (PKG\_CONFIG\_PATH specifica nuove fonti).

Questi file fan parte dell'installazione della libreria.

- come si usa?

```
$ gcc -Wall sorgente.c -o binario `pkg-config --cflags --libs libreria`
```

## usare pkg-config (2)

- come son fatti i file .pc?

```
embyte@darkstar:/usr/lib/pkgconfig> less gtk+-2.0.pc
prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include
target=x11
```

```
gtk_binary_version=2.4.0
gtk_host=i686-pc-linux-gnu
```

```
Name: GTK+
Description: GIMP Tool Kit (${target} target)
Version: 2.4.3
Requires: gdk-${target}-2.0 atk
Libs: -L${libdir} -lgtk-${target}-2.0
Cflags: -I${includedir}/gtk-2.0
```

## usare pkg-config: esempio (3)

- elenco delle librerie presenti nel sistema:

```
$ pkg-config --list-all
gdk-pixbuf-xlib-2.0      GdkPixbuf Xlib - GdkPixbuf rendering for Xlib
gimpthumb-2.0          GIMP Thumb - GIMP Thumbnail Library
gdk-x11-2.0             GDK - GIMP Drawing Kit (x11 target)
gmodule-2.0            GModule - Dynamic module loader for GLib
gtk+-x11-2.0           GTK+ - GIMP Tool Kit (x11 target)
gdk-pixbuf-2.0         GdkPixbuf - Image loading and scaling
gtk+-2.0               GTK+ - GIMP Tool Kit (x11 target)
pangoft2               Pango FT2 - Freetype 2.0 font support for Pango
```

- flags di compilazione (libreria GTK+2):

```
$ pkg-config --cflags gtk+-2.0
-I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include -I/usr/include/atk-1.0
-I/usr/include/pango-1.0 -I/usr/X11R6/include -I/usr/include/freetype2 -I/
usr/include/glib-2.0 -I/usr/lib/glib-2.0/include
```

- librerie (GTK+2):

```
$ pkg-config --libs gtk+-2.0
-Wl,--export-dynamic -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -lgdk_pixbuf-2.0
-lm -lpangoxft-1.0 -lpangox-1.0 -lpango-1.0 -lgobject-2.0 -lgmodule-2.0
-ldl -lglib-2.0
```

# make

- utility per automatizzare le procedure di compilazione e ricompilazione
- make esegue target specificati in un file di “scripting” chiamato [Makefile](#)
- Makefile: struttura predefinita
  - \* possibilità di definire variabili
  - \* indentazione a tabulazione
  - \* dipendenza tra i target
- integrazione con configure (vedi dopo)



# un esempio di Makefile

```
CC          = gcc
CFLAGS      = -Wall -O2
LIBS        = -lnet
prefix      = /usr/local

OBJS        = main.o funct.o

all:        test install

test:       $(OBJS)
            $(CC) $(CFLAGS) $(LDFLAGS) -o test $(OBJS) $(LIBS)
            @echo "Done! Type make install from root"

.c.o:
            $(CC) $(CFLAGS) -c $< -o $@

install:
            cp -f test ${prefix}/bin

clean:
            rm -fr *.o test
```

# configure e autoconf

- [configure](#): che file è?

Script di shell che automatizza la configurazione delle opzioni e dei parametri di compilazione di un programma.

[Makefile.in](#) -> Makefile

- quando è utile?

Qualora si voglia rendere portabile il proprio software o si ha la necessità di modularizzarlo (possibilità di abilitare/disabilitare determinate features).

- a cosa serve autoconf?

Utility per generare il configure da un file template

([configure.ac](#) o [configure.in](#))

# procedura di configurazione e compilazione

1. scrittura dei template Makefile.in e configure.ac (sviluppatore)

2. generazione dell'include di configurazione (sviluppatore)

configure.ac, config.h.in -> \$ autoheader -> config.h

3. generazione dello script di configurazione (sviluppatore)

configure.ac -> \$ autoconf -> configure

4. generazione dello script di compilazione (utente)

Makefile.in -> \$ ./configure -> Makefile, config.h (ok)

-> config.log (errore)

5. compilazione e installazione (utente)

\$ make (è implicito il target 'all')

## un esempio di configure.ac (1)

```
AC_INIT(main.c)
AC_CONFIG_HEADER(config.h)                # header config.

AC_PREFIX_DEFAULT(/usr/local)             # prefix
if test "$prefix" = "NONE"; then
    prefix="/usr/local"
fi

AC_PROG_CC                                # compilatore
AC_FUNC_MALLOC                             # malloc() e simili
AC_FUNC_VPRINTF                            # vprintf() e simili
AC_HEADER_STDC                             # header comuni
AC_OUTPUT(Makefile)                        # scrive il Makefile
```

## un esempio di configure.ac (2)

```
filechk="yes" # test libpcap
AC_CHECK_FILE(/usr/lib/libpcap.a,, filechk="no")
if test "$filechk" = "no"; then
  AC_CHECK_FILE(/usr/local/lib/libpcap.a, filechk="yes" ;
    LDFLAGS="-L/usr/local/lib"; CPPFLAGS="-I/usr/local/include")
fi

AC_CANONICAL_TARGET # OS
case "$target" in
*linux*)
  AC_MSG_NOTICE([Found Linux, happy day!])
  ;;
esac # etc...
```

# pkg-config: integrazione con autoconf (1)

- macro

VARIABLE\_CFLAGS  
VARIABLE\_LIBS



```
PKG_CHECK_MODULES(VARIABLE,  
MODULELIST[,ACTION-IF-FOUND[,ACTION-IF-NOT-FOUND]])
```

- esempio

```
AC_SUBST(GTKOBJJS)
```

```
AH_TEMPLATE(HAVE_GTK, gtk+ 2.x support)
```

[configure.ac](#)

```
PKG_CHECK_MODULES(GTK, gtk+-2.0 >= 2.0.0 pango >= 1.0 atk >= 1.0, [  
    AC_DEFINE(HAVE_GTK)  
    GTKOBJJS="interface.o callbacks.o gfuncts.o support.o gtk.o",  
    [AC_MSG_WARN([Librerie GTK+ non trovate!])] )
```

## pkg-config: integrazione con autoconf (2)

GTK\_LIBS = @GTK\_LIBS@

GTK\_CFLAGS = @GTK\_CFLAGS@

OBJS = main.o console.o common.o @GTKOBJ@

Makefile.in

.c.o:

\$(CC) \$(CFLAGS) \$(CPPFLAGS) \$(GTK\_CFLAGS) -c \$< -o \$@

gspooof: \$(OBJS)

\$(CC) \$(CFLAGS) \$(LDFLAGS) \$(OBJS) -o gspooof -lnet \$(GTK\_LIBS)

# **gdb: il debugger GNU (1)**

- ricordarsi di compilare con il flag -g
- eseguire con `$ gdb programma`
- esempio sessione

```
$ gcc -Wall -g debug.c -o debug
```

```
$ gdb ./debug
```

```
GNU gdb 6.1.1
```

```
(gdb) break main
```

```
Breakpoint 1 at 0x8048380: file debug.c, line 9.
```

```
(gdb) run
```

```
Starting program: /home/embyte/devel/stuff/MOCA/debug
```

```
Breakpoint 1, main () at debug.c:9
```

```
9      a=5;
```



## **gdb: il debugger GNU (2)**

```
(gdb) step
10      *b=10;
(gdb) step
11      c=a>(*b);
(gdb) step
13      printf ("Il prodotto a*b vale %d\n", c);
(gdb) print c
$1 = 50
(gdb) print b
$2 = (u_short *) 0x4014860c
(gdb) print *b
$3 = 10
(gdb) next
Il prodotto a*b vale 50
15      return 0;
(gdb) help comando
```

## altri comandi utili

- diff: trova le differenze tra due file
- patch: applica delle differenze ad un file originale
  
- ldd: stampa le librerie dinamiche dipendenti di un binario
- ldconfig: configura i link ed effettua il caching delle librerie dinamiche contenute nei percorsi di default e specificati in ld.so.conf
- objdump: stampa informazioni dettagliate sui file oggetto
- strace: traccia le chiamate di sistema
- ltrace: traccia le chiamate alle librerie
- ld: il linker
  
- cvs: client CVS

# riferimenti

- VIM, <http://www.vim.org> 
- JED, <http://www.jedsoft.org/jed/>
- GCC, <http://gcc.gnu.org/>
- GDB, <http://www.gnu.org/software/gdb/gdb.html>
- KDevelop, <http://www.kdevelop.org/>
- Anjuta, <http://anjuta.sourceforge.net/>

La presentazione sarà disponibile sul sito dello S.P.I.N.E. Research Group

<http://www.spine-group.org>



Associazione culturale Olografix, <http://www.olografix.org/>

