



ACM & MUG Tech Talk

Software Development Tools on Unix
October 15, 2002

<http://acm.cs.virginia.edu/presentations.php3>



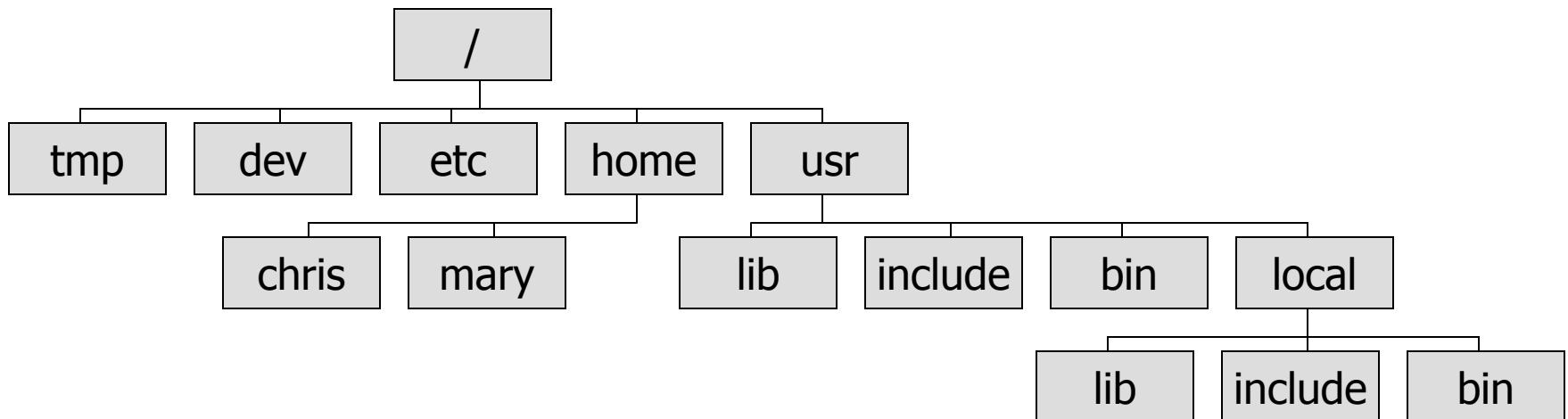
Overview

- Intro to Unix, getting around
- Text editors
- Gcc
- Debugging
- Profiling
- Revision management
- Searching files
- Useful programming and misc tools



Intro to Unix: Files

- Filesystem a single tree
- Filenames case sensitive
- Physical devices can be mounted anywhere





Intro to Unix: Essential Cmds

- `cd` - change directory - *cd*
- `mkdir` - make a directory - *md*
- `cp` - copy a file - *copy*
- `ls` - list files - *dir*
- `rm` - remove a file - *del*
- `mv` - move a file - *move & ren*
- `grep` - expression searching
- `top` - cpu and memory usage
- `who/w` - who else is logged in
- `man` - read documentation



Intro to Unix: More

- A central idea in Unix:
 - Make small tools that do one job well and that can be connected
- ACM Intro to Unix presentation:
<http://acm.cs.virginia.edu/archives/events/workshop/unix/>
- Online resources
 - Lots, just search
- Books
 - Again, lots
- CHUUG: <http://www.chuug.org/>



Text Editors

- Crucial tools for using Unix
- Two main editors (easy flamewar topic):
 - emacs
 - vi
- Great features in both:
 - Syntax highlighting
 - Brace matching
 - Sophisticated text manipulation/movement
 - Scriptable
 - ...



Text Editors: vi

- If it's unix, vi is installed
 - vim, <http://www.vim.org>
- Simple and small
- Modal
 - Command: move, perform commands
 - Insert
 - Others (replace, selection, more)
- Built in help, “: help”



Text Editors: emacs

- Kitchen-sink powerful
- Configurable, extensible, complicated
- emacs and xemacs
- emacsclient
- Tutorial: C-h t



Text Editors: Pure IDEs

- KDevelop: KDE
- Anjuta: GTK/GNOME

Text Editors: IDEs

KDevelop

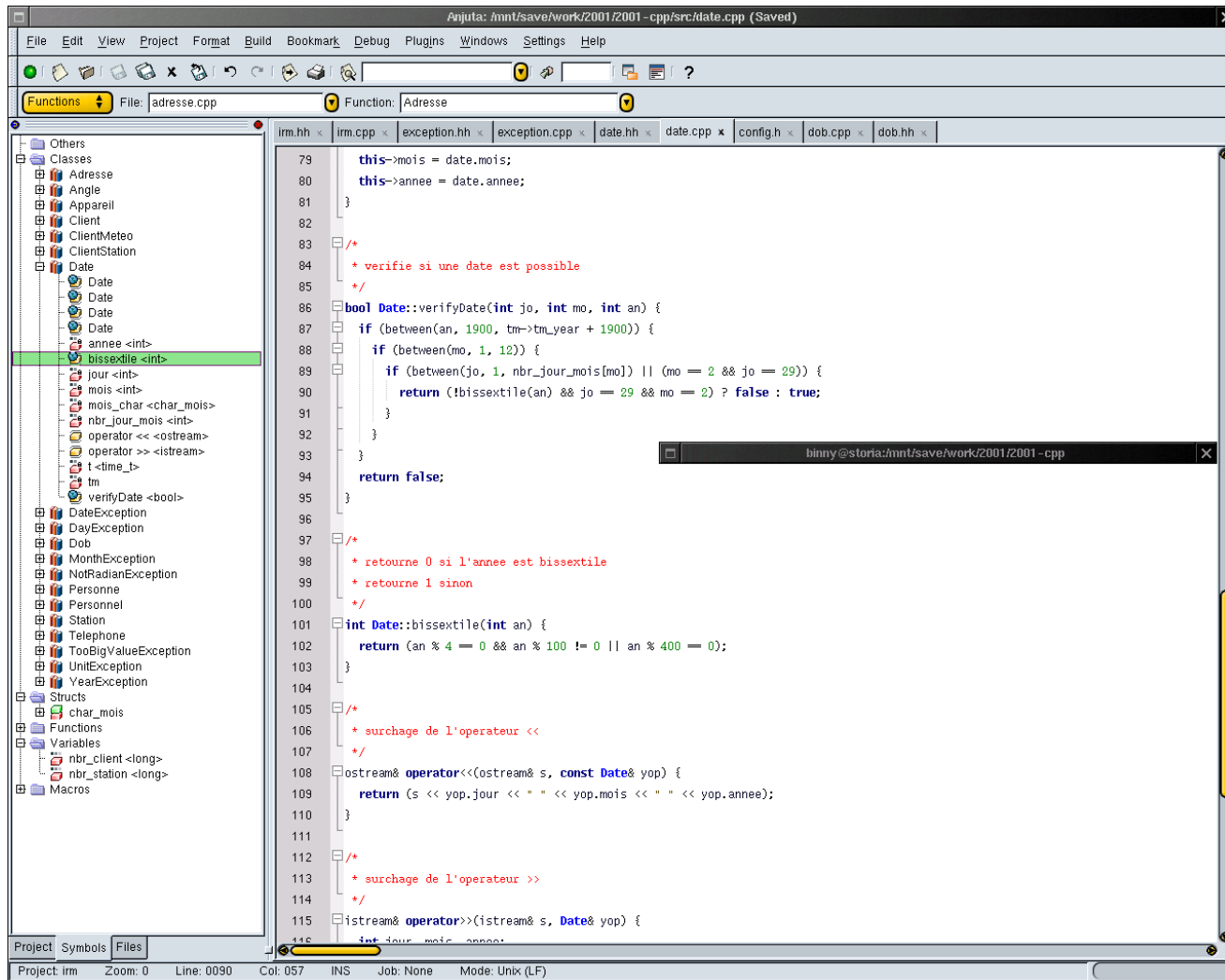
The screenshot displays the KDevelop 3.0 IDE interface. The main window shows the file `file:/home/caleb/gt/src/ginaglobal.cpp` with the following code:

```
58 . . . if(rFilter.contains("active") && !p.current()->IsActive()) continue;
59 . . . if(rFilter.contains("fuelreading") &&!p.current()->IsFuelReadingParticipant())
60 . . . if(rFilter.contains("nospecial")) {
61 . . . . . if(p.current()->Name()=="fuel" || p.current()->Name()=="fuel_rate" ||
62 . . . . . p.current()->Name()=="bcfc" || p.current()->Name()=="flame_g") continu
63 . . . }
64 . . . names += p.current()->Name();
65 . . . }
66 . . . return names;
67 }
68
69 QStringList GinaGlobal::MeasurementUnits(QStringList rNames)
70 {
71     QStringList temp_list;
72     for(QStringList::Iterator it = rNames.begin(); it != rNames.end(); ++it) {
73         QString temp_name = *it;
74         if(mvMeasurementMap.find(temp_name)) temp_list << mvMeasurementMap[temp_name]-
75         else temp_list << "";
76     }
77     return temp_list;
78 }
79
80 double GinaGlobal::QueryMeasurementValue(QString rName)
81 {
82     if(!mvMeasurementMap.find(rName)) return -1;
83     return mvMeasurementMap[rName]->FinalValue();
84 }
85
86
87 DoubleList GinaGlobal::MeasurementValues(QStringList rNames, QString rFilter)
88 {
89     DoubleList temp_list;
90     for(QStringList::Iterator it = rNames.begin(); it != rNames.end(); ++it) {
91         QString temp_name = *it;
92         if(mvMeasurementMap.find(temp_name) && rFilter.contains("final"))
93             temp_list << mvMeasurementMap[temp_name]->FinalValue();
94         else if(mvMeasurementMap.find(temp_name) && rFilter.contains("nospecial"))
```

The interface includes a menu bar (File, Edit, View, Project, Build, Go, Bookmarks, Settings, Help, Tools), a toolbar, and a status bar at the bottom showing "Line: 70 Col: 1". On the right side, there is an Automake Manager showing a project structure for `gt (Program in bin)` with files `ginaglobal.h`, `ginaglobal.cpp`, `main.h`, and `main.cpp`. A Documentation Browser is also visible on the far right.

Text Editors: IDEs

Anjuta

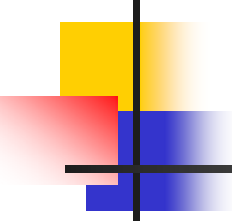




gcc

the Gnu Compiler Collection

- Frontends for:
 - C: gcc
 - C++: g++
 - More (ada, java, objective-c, fortran, ...)
- Backends for:
 - x86, ia-64, ppc, m68k, alpha, hppa, mips, sparc, mmix, pdp-11, vax, ...



gcc/g++ : Usage

- `g++ foo.cpp`
 - Outputs `./a.out` binary
- `g++ -g -Wall -pedantic foo.cpp bar.cpp baz.o -o foo`
 - `-g`: produce debugging information
 - `-Wall -pedantic`: many types of warnings
 - `-o foo`: output to file "foo"
- Linking libraries:
 - eg math library: `-lm`
 - The library foo is named `libfoo.a` under unix



gcc/g++: Usage

Optimizations

- None/-O0: Compile afap, debugging as expected
- -O1: some reduction of code size & exec time
- -O2: most opts w/o space-speed tradeoffs
- -O3: more
- cpu-specific optimization flags as well
 - eg scheduling, assembly style, native types (get 128bit doubles on x86!), aligning, ...

gcc/g++ : Usage

Misc

- Run only preprocessor: -E
 - -C: don't discard comments
 - -MM: dependency info (useful for makefiles)
- Macros:
 - Define: `-Dmacro` and `-Dmacro=defn`
 - See defined after preprocessing: `-E -dM`
- -fall-virtual: treat all member fns as virtual
 - Except constructor and new/delete
- Cross-compile



Debugging

- gdb and dbx
- Frontends:
 - insight (part of gdb)
 - ddd
 - gvd
 - kdbg
 - emacs



Debugging

`gdb`: the Gnu DeBugger

- Debugs C, C++, and Modula-2
- Text and graphical interfaces
- Program crashes
 - Leaves a "core" file
 - Memory dump of the program
 - Run `gdb` on binary and core, explore state of program at crash
- Common usage:
 - `gdb binary`
 - `gdb binary core`



Debugging

`gdb: Essential Commands`

- `break [file:]function` - set breakpoint
`break class::member_fn`
- `run [arglist]` - run program
- `bt` - print function stack
- `print expr` - print the result of `expr`
- `c` - continue execution
- `next, step` - next instruction/line
- `watch` - watch an `expr`
- `info locals` - local variables
- `help` - builtin help



Profiling

- “90% exec time spent in 10% of the code”
- Popular tools for profiling:
 - gprof
 - Flat profile: time spent in fns and how often called
 - Call graph: Which functions called which and how often
 - Annotated source: source w/ #times each line executed
 - gcov
 - Test coverage
 - SimpleScalar
 - Program performance analysis
 - Mirco-arch modeling
 - Hardware-software co-verification



Profiling gprof Usage

- Use cc/cpp options: `-pg -a`
- Run the binary as normal, outputs `gmon.out` and `bb.out`
- Run `gprof` to analyze



Revision Management

- diffutils:
 - Find differences among files (diff and diff3)
 - Update a set of files from generated differences (patch)
- CVS
 - Maintains a history of changes for groups of files
 - Including: who, when, what, and optionally why
 - Analogue of Visual SourceSafe
 - New alternatives: subversion and bitkeeper



Revision Management

What CVS lets you do

- Modify source code locally until it's ready to be merged with what everyone sees
 - Multiple people can modify the same file and be ok
- Develop multiple versions of software simultaneously
 - And merge these branches later on
- Recall past versions of code base
 - Based on time ago or version
- See committer's comments on their changes



Searching files: grep

- Searches lines of text for given pattern
- “grep”?
 - From ed subcommand “g/re/p”
 - “Globally search for the Regular Expression and Print the lines containing matches to it”
- Also great for parsing program output
 - `program_foo | grep regex`



Searching files: grep

Regex Examples

<u>Regex</u>	<u>Will match</u>	<u>Won't match</u>
ab	baby	borris
^ab	abby	baby
ad\$	monad	monads
[cC]hris	Chris, chris	cHris
[0-9a-zA-Z]	Any alphanumeric	?, \$
[^aeiou]s	Zrvwls	is
a.b	arbitrary	arkabane
a*b	arkabane	bardvark



Other Useful Programming Tools

- indent: format source code
- static checking
 - C: splint
 - Java: escjava
- lex and yacc (gnu: flex and bison)
 - Code generators for lexical analyzers and parsers
- nm: lists symbols from object files



Other Useful Programming Tools, *cont*

- gengetopt: generates code to parse options
- Dynamic code generation
 - lightning and vcode
- GUI layout
 - gtk: glade (C and C++ code generation)
 - qt: Kdevelop
 - Glade and KDevelop: layout by hand or from XML
 - fltk: fluid
- make
 - Automates generation of files from sources files



Other Useful Programs

- screen: multiple terminals within a terminal
- wc: Word Count
- ssh and scp: remote login and file copy
- find: find files
 - `find . -name "foo*bar" -type f -exec wc -l {$1} \;`
- Printing: a2ps and enscript
 - Syntax highlighting, multipage, toc



Other Useful Programs, *cont*

- tar and gzip/bzip2: collect and compress files
 - Analogue of winzip
- less/more: look through files
- wget: retrieve files via http and ftp
- xmms: winamp-like mp3 player



ACM & MUG Tech Talk

Software Development Tools on Unix
October 15, 2002

<http://acm.cs.virginia.edu/presentations.php3>