Computers
Fundamentals of Programming
Grau en Ciència i Enginyeria de Dades

Facultat d’Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)
2021-2022 Q2
Creative Commons License

This work is under a Creative Commons Attribution 4.0 Unported License

The details of this license are publicly available at https://creativecommons.org/licenses/by-nc-nd/4.0
Table of Contents

• Understanding your application
• Control Version Management
• Compilation, debugging, and code management toolchains
• Testing is really worthy
• Automated Software Deployment
Understanding your application

• While developing/maintaining code, it is extremely important!!
• Have in mind:

  • Goal of the application
  • Algorithms
  • Data structures
  • Which services it uses from the system

  • Structure of the source code
    • Directory structure
    • Header files
    • Source files

  • Structure of the binary files
    • When compiled
Understanding your application

- Example... 🍺 Geany git

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Theme improvements (#1382)</td>
</tr>
<tr>
<td>doc</td>
<td>manual: added documentation about replacement of 'untitled.ext' with ...</td>
</tr>
<tr>
<td>icons</td>
<td>icon: regenerate png/ico files based on the svg</td>
</tr>
<tr>
<td>m4</td>
<td>Update Scintilla to version 3.7.5 (#1503)</td>
</tr>
<tr>
<td>plugins</td>
<td>filebrowser: Don’t change directory on project save</td>
</tr>
<tr>
<td>po</td>
<td>Small update of German translation</td>
</tr>
<tr>
<td>scintilla</td>
<td>Update Scintilla to version 3.7.5 (#1503)</td>
</tr>
<tr>
<td>scripts</td>
<td>Update Scintilla to version 3.7.5 (#1503)</td>
</tr>
<tr>
<td>src</td>
<td>Merge pull request #1748 from kugel-/msgwin-api</td>
</tr>
<tr>
<td>tests</td>
<td>bash may not found in the system (#1574)</td>
</tr>
</tbody>
</table>
## Understanding your application

- **autotools**
- **src**
- **include**
- **libs**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makefile.am</td>
<td>Merge pull request #1095 from eht16/issue1076_win32_build_working_dir...</td>
</tr>
<tr>
<td>about.c</td>
<td>Remove a space (#1790)</td>
</tr>
<tr>
<td>about.h</td>
<td>Normalize use of header guards and extern &quot;C&quot; guards</td>
</tr>
<tr>
<td>app.h</td>
<td>Add utils_get_real_path() and use it</td>
</tr>
<tr>
<td>build.c</td>
<td>Work around a <code>-Wformat-overflow</code> warning</td>
</tr>
<tr>
<td>build.h</td>
<td>doxygen: various doxygen-related fixes in preparation for gtkdoc gene...</td>
</tr>
<tr>
<td>callbacks.c</td>
<td>Show status message on attempt to execute empty context action.</td>
</tr>
<tr>
<td>callbacks.h</td>
<td>Allow to set a keybinding for File-&gt;Properties</td>
</tr>
<tr>
<td>dialogs.c</td>
<td>Fix canceling keybinding overriding by discarding the dialog</td>
</tr>
<tr>
<td>dialogs.h</td>
<td>Protect private definitions by the GEANY_PRIVATE macro in headers</td>
</tr>
<tr>
<td>document.c</td>
<td>Add missing space in string. Fixes #1789</td>
</tr>
<tr>
<td>document.h</td>
<td>Added option to auto reload files changed on disk (#1246)</td>
</tr>
<tr>
<td>documentprivate.h</td>
<td>Add support for Keyed Data Lists for documents</td>
</tr>
<tr>
<td>editor.c</td>
<td>Remove some unused variables</td>
</tr>
</tbody>
</table>
Library support according to language and OS

**python:** libpython3.6m.so
libpthread.so
libdl.so
libm.so, libmvec.so
libc.so

**xclock:** libX11.so
libXext.so
libm.so
libc.so
...

**cpmd.x.omp:** libmpi.so
libgfortran.so
libpthread.so, libgomp.so
libm.so, libmvec.so
libc.so
...

**perl:** libdl.so
libcrypt.so
libpthread.so
libm.so
libc.so
Code Version Management

• Version Control tools
  • A set of tools that help to keep track of changes in code using a hierarchy of internal structures and files that help to manage different concurrent versions of code
    • Centralized Version Control System (e.g. cvs, svn): there is a single copy of the repository (i.e. all code versions)
    • Distributed Version Control System (e.g. git, Mercurial): there are multiple copies of the repository
  • There are web-accessible repositories where people/companies manage code
    • E.g.: Github, Gitlab

https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs
Basic concepts (examples based on Git)

- **Repository**
  - The tracking of all changes done in docs of a Project
    - “.git” folder
  - **Clone** operation
    - Copy an existing repository into a new local repository

- **Basic workflow of git like environment**
  - Working Directory → Staging Area → Git Repository
  - **Add** operation
    - Add a change from the working directory to the staging area
  - **Commit** operation
    - Capture a change that has been previously promoted to the staging area
      - Additional related details (e.g. timestamp)
  - **Push** operation
    - Updates remote repository: upload commits
  - **Pull** operation
    - Updates local repository: download commits
Basic concepts (examples based on Git)

• Branch
  • It is a working line (that is, a similar to a path) to develop in a Project
    • Modifications in a branch don’t impact on other branches
  • At least there is one branch in a repo: the **master**
    • It is the main Branch
  • Management of branches
    • Example: merge development branch to master branch

• You can switch either from branch to branch and from commit to commit
Ex. Code Management
Compilation and code management toolchains

• Software Version Formats
  • (X.Y.Z)
    • X: Major changes, usually incompatible with previous versions;
    • Y: Minor changes, new functionalities added in a backwards-compatible manner;
    • Z: Revision/Patch (bug fixing)
  • Odd-even System
    • Odd numbers for development and even numbers for stable releases

• Build Process Tools
  • A set of tools for software developers to create/distribute automatically buildable source code and make software packages portable
  • Autotools (by GNU) make it easier to support portability, (build configuration) based on common build conventions (e.g. well known paths), and automate dependency tracking to create the makefile
    • autotools = autoconf + automake + libtool + ...
  • Cmake is the next generation of autotools
Example of compile and install process

(t)ernal point of view)

#> Download source code
...
#> ./configure

http://freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools/
Example of automated build process

(terminal point of view)

#> Download source code
...  
#> ./configure
#> make all
#> make install

http://freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools/
Debugging

• A Debugger is a tool that can execute a process under controlled conditions to help programmers to find bugs
  • It can pause/resume the execution and analyse critical values, such as variables, parameters, memory addresses, among other capabilities
  • But it has some limitations and requirements, such as difficulties to analyse parallel executions, some debuggers depend on the language, and complexity to know how to properly use it

• A program...
  • without debug information can also be debugged, but it is difficult
  • with debug information, the debugger shows information that relates high level source code to the low level source code of the execution to ease the debug process

• An interpreted code is easier to be debugged to find a bug since there is no low level code
  • But this simplicity can hide low level issues

• Examples: **GDB – The GNU Debugger**, PDB (The Python Debugger), Visual Studio
Table of Contents

• Understanding your application
• Control Version Management
• Compilation, debugging, and code management toolchains
• Testing is really worthy
• Automated Software Deployment
Types of Testing

• From isolated to integrated and from faster to slower approaches
  • **Unit Test**: a single unit of code
  • **Integration Test**: two or more units communicated with each other
  • **Functional Test**: a feature
  • **User Acceptance Test**: a feature, but from the user point of view
  • **Smoke Test**: the system is operational
  • **Regression Test**: other functionalities previously implemented
  • **Usability Test**: related to User eXperience (UX)
  • **Exploratory Test**: the tester itself manually discovers and learns the different checks to the system

• Test automation: implement automated testing tasks
Development Methodologies

• Waterfall
  • Work broken down into sequential phases
    • Requirements->Design->Development->Testing->Deployment->Final Outcome
  • When time/cost are limited, and product scope/requirements are clear from the beggining

• Agile
  • Cross-functional teams discovering and building through iterative process
    • Scrum approach: Broken down into sprints (a.k.a. iterations)
  • When the details are not clear from the beggining, deliver micro-outcomes and requirements are discovered/adapted through the different iterations

• Hybrid Model
Automated Software Deployment

• Everytime there are new code developments, the mainline code has to be updated in a secure way. That is, the code needs to be tested
  • There are software projects that need to automate this task

• There are different procedures depending on the target
  • **Continuous Integration**: automates built and test of new code. The main goal is to integrate it to the mainline code. Thus, it has to be tested before and after the integration
  • **Continuous Delivery**: automates a new software release. The main goal is to be sure it can be delivered (e.g. go to production). Thus, it is key the user acceptance tests
  • **Continuous Deployment**: is a step up, since it deploys the results of Continuous Delivery into the deployment environment (e.g. production environment). It is assumed all automated tests are passed.

• There are tools that integrates multiple tools to perform these tasks
  • E.g. Jenkins
Bibliography

• Git
  • https://git-scm.com/doc

• GDB – The GNU Debugger
  • https://www.gnu.org/software/gdb/
    • Manual (Web & PDF)

• Introduction to software testing
  • https://en.wikipedia.org/wiki/Software_testing

• Continuous management of software releases
  • Atlassian Wiki
    • https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment