Principles of Software Construction: Objects, Design, and Concurrency

Version Control

Jonathan Aldrich  Charlie Garrod

Based on slides by Christian Kästner
Administrivia

• Extra office hours for HW6 and exam
  – See online schedule
• Sample final exam – coming soon
• Review session Wednesday 12/16
  – 2-4pm in DH 1112
• Final exam Thursday 12/17
  – 8:30-11:30 in MM 103 & MM A14
Learning Goals

• Understand the benefits and limitations of version control (history, parallel development, branching, etc)

• Ability to cooperate on a code base using branching and merging; to resolve merge conflicts

• Distinguish classes of version control systems (local, central, distributed)
HISTORY
History

- Record relevant steps in the history of the project
- Supports partial undo
- Label cohesive development steps
  - Allows understanding changes
  - Explains rationale
  - Traceability to other development artifacts (bug trackers, CVEs, requirements, etc)
- Record who performed changes
- Implies a backup
- Early local systems: SCCS and RCS
- -> Version control systems useful even when used entirely locally
Eclipse's Local History
Revision History

![Revision History](image)

<table>
<thead>
<tr>
<th>Revision</th>
<th>Tags</th>
<th>Revision Time</th>
<th>Author</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>*1.64</td>
<td>v20100831-0800, r361_v20100...</td>
<td>11.09.08 13:59</td>
<td>dmc</td>
<td>Removed trailing whitespace and organized the imports</td>
</tr>
<tr>
<td>1.63</td>
<td>r342_v20091014, r342_v20090716, ...</td>
<td>20.01.08 10:31</td>
<td>bbenu</td>
<td>[Performance][Breadcrumb] Issue editor leak after closing editor when in...</td>
</tr>
<tr>
<td>1.62</td>
<td>v2008122-0800a, v2008122-0800</td>
<td>21.01.08 13:04</td>
<td>bbenu</td>
<td>[Breadcrumb] Activating editor should set focus into breadcrumb if it was...</td>
</tr>
<tr>
<td>1.61</td>
<td>v2008115-0800, v20080108-0800, ...</td>
<td>21.06.07 14:44</td>
<td>dmcger</td>
<td>Fixed bug 182.108; [api] Add action constants for IDEActionFactory and ActionFactory...</td>
</tr>
<tr>
<td>1.60</td>
<td>R3_3_2, r332_20081018, r332_200806...</td>
<td>12.04.07 12:33</td>
<td>dmcger</td>
<td>Fixed bug 178744; [misc] Editor opened upon an out of sync resource should allo...</td>
</tr>
<tr>
<td>1.59</td>
<td>v20070410-0800, v20070403-0800, ...</td>
<td>28.03.06 18:53</td>
<td>nkeller</td>
<td>[Breadcrumb] Updated copyright.</td>
</tr>
<tr>
<td>1.58</td>
<td>v20060327-3600, v20060327-0010, ...</td>
<td>10.02.06 12:27</td>
<td>nkeller</td>
<td>[Breadcrumb] Renamed BasicEditorActionContributor to BasicEditorActionContribution</td>
</tr>
<tr>
<td>1.57</td>
<td>v20060207-0800, v20060131-0800, ...</td>
<td>18.08.05 15:44</td>
<td>nkeller</td>
<td>removed unnecessary $NON-NLS-&quot;$ tags</td>
</tr>
<tr>
<td>1.56</td>
<td>R3_1_2, r312_v20060104-0800, r31...</td>
<td>17.06.05 17:51</td>
<td>dmcger</td>
<td>Updated copyright date to 2005</td>
</tr>
<tr>
<td>1.55</td>
<td>v20050617-preCopyRightPass, v200...</td>
<td>19.04.05 14:47</td>
<td>dmcger</td>
<td>Removed trailing whitespace.</td>
</tr>
<tr>
<td>1.54</td>
<td>v20050419-0800</td>
<td>19.04.05 11:02</td>
<td>dmcger</td>
<td>Converted to Eclipse NLS.</td>
</tr>
<tr>
<td>1.53</td>
<td>v20050412-0800, v20050405-0800, ...</td>
<td>25.02.05 16:34</td>
<td>dbae...</td>
<td>Converted from CPL to EPL</td>
</tr>
<tr>
<td>1.52</td>
<td>R3_0_2, r302_v20050311-0800, v20...</td>
<td>23.06.04 21:33</td>
<td>dmcger</td>
<td>Copyright fix - automated changes with releng tool tor ...</td>
</tr>
<tr>
<td>1.51</td>
<td>v20040523-1600, pre-copyright v3</td>
<td>03.05.04 17:10</td>
<td>dmcger</td>
<td>Released from Christof Martin, some modifications...</td>
</tr>
</tbody>
</table>
Pass, buy special tile GUI impl [ci skip]

Checkpoint

finish boom

GUI Finished, Test Cases Not Finished

new environment

Showing 1,862 changed files with 0 additions and 0 deletions.

Sorry, we could not display the entire diff because too many files (1,862) changed.
Recording Rationale and Traceability

Index: kernel/git/torvalds/linux.git

commit: 29fa68256e3c97e11972c4b031f746b1b5e95b72b (patch)
author: Andy Luomiski <luto@amascapital.net> 2014-12-06 03:03:26 (GMT)
committer: Paolo Bonzini <pbonzini@redhat.com> 2014-12-10 11:49:39 (GMT)
parent: e08e434b16776ecb19a67b2e8c38c3b25885 (diff)

x86, kvm: Clear paravirt_enabled on KVM guests for espfx32's benefit

paravirt_enabled has the following effects:

- Disables the FOOF bug workaround warning. There is no FOOF bug workaround any more because Linux's standard IDT handling already works around the FOOF bug, but the warning still exists. This is only cosmetic, and, in any event, there is no such thing as KVM on a CPU with the FOOF bug.

- Disables 32-bit APIC BIOS detection. On a KVM paravirt system, there should be no APIC BIOS anyway.

- Disables boot. I think that the boot code should check the CPUID hypercall hit directly if it matters.

- paravirt_enabled disables espfx32. espfx32 should *not* be disabled under KVM paravirt.

The last point is the purpose of this patch. It fixes a leak of the high 16 bits of the kernel stack address on 32-bit KVM paravirt guests. Fixes CVE-2014-014.

---

Co: Stable@vgtc.kernel.org
Suggested-by: Konrad Haaseneck Wilt <konrad.wilt@oracle.com>
Signed-off-by: Andy Luomiski <luto@amascapital.net>
Signed-off-by: Paolo Bonzini <pbonzini@redhat.com>

Diffstat
---
-wr14-arch/x86/kvm/kvm.c  9
Versioning entire projects
Versioning and Tags

- Version numbers for files individually vs. per project
- Version numbers vs. hashes
- Tags to name specific states
While files to manage

• All code and noncode files
  – Java code
  – Build scripts
  – Documentation
• Exclude generated files (.class, ...)
• Most version control systems have a mechanism to exclude files (e.g., .gitignore)
COLLABORATION
Collaborating on Files

• How to exchange files
  – Send changes by email
  – Manual synchronization at project meeting
  – All files on shared network directory

• Permission models
  – Each file has an owner; only person allowed to change it
  – Everybody may change all files (collective ownership)
Concurrent Modifications

- Allowing concurrent modifications is challenging
- Conflicts (accidental overwriting) may occur
- Common strategies
  - Locking to change
  - Detecting conflicts (optimistic model)
Change Conflicts
Locking Files

Practical problems of locking model?
Locking Problems

• How to lock?
  – Central system vs announcement on mailing list
  – Forgetting to unlock common

• Unnecessary sequentializing
  – Cannot work on different concepts in same file

• False sense of security
  – Changing dependant files can cause conflicts not prevented by locking
Merging (1/2)
Merging (2/2)

Harry compares the latest version to his own

A new merged version is created

The merged version is published

Now both users have each others’ changes
Example

```java
import java.util.LinkedList;
public class Stack<T> implements Cloneable {
    private LinkedList<T> items = new LinkedList<T>();
    public void push(T item) {
        items.addFirst(item);
    }
    public T pop() {
        if (items.size() > 0) return items.removeFirst();
        else return null;
    }
}
```
import java.util.LinkedList;
public class Stack<T> implements Cloneable {
    private LinkedList<T> items = new LinkedList<T>();
    public void push(T item) {
        items.addFirst(item);
    }
    public int size() {
        return items.size();
    }
    public T pop() {
        if(items.size() > 0) return items.removeFirst();
        else return null;
    }
}

import java.util.LinkedList;
public class Stack<T> implements Cloneable {
    private LinkedList<T> items = new LinkedList<T>();
    public void push(T item) {
        items.addFirst(item);
    }
    public T top() {
        return items.getFirst();
    }
    public T pop() {
        if(items.size() > 0) return items.removeFirst();
        else return null;
    }
}
Example

import java.util.LinkedList;
public class Stack<T> implements Cloneable {
    private LinkedList<T> items = new LinkedList<T>();
    public void push(T item) {
        items.addFirst(item);
    }
    public int size() {
        return items.size();
    }
    public T pop() {
        if(items.size() > 0) return items.removeFirst();
        else return null;
    }
}

import java.util.LinkedList;
public class Stack<T> implements Cloneable {
    private LinkedList<T> items = new LinkedList<T>();
    public void push(T item) {
        items.addFirst(item);
    }
    public int size() {
        return items.size();
    }
    public T pop() {
        if(items.size() > 0) return items.removeFirst();
        else return null;
    }
}

System cannot decide order
3-way merge

• File changed in two ways
  – Overlapping changes -> conflicts
  – Merge combines non-conflicting changes from both
• Merging not always automatic
  – diff tool to show changes
  – Manual resolution of conflicts during merge (potentially requires additional communication)
• Automatic merge potentially dangerous
  -> syntactic notion of conflicts
• Merging of binary files difficult
• In practice: most merges are conflict free
BRANCHING
Branching

• Parallel copies of the source tree
• Can be changed independently, versioned separately, and merged later (or left separate)
• Often used for exploratory changes or to isolate development activities
• Many usage patterns, common:
  – Main branch for maintenance OR main development
  – New branches for experimental features; merge when successful
  – New branches for nontrivial maintenance work
  – Branches for maintenance of old versions
Release management with branches
Variants and Revisions

- **Revision** replaces prior revision (temporal)
- **Variant** coexists with other variants
- **Version** describes both
- **Release**: Published and named version

<table>
<thead>
<tr>
<th></th>
<th>V1.0</th>
<th>V1.1</th>
<th>V2.0</th>
<th>V3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base system (Windows)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Linux variant</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Server variant</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Extension for A</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Extension for B</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Semantic Versioning for Releases

• Given a version number MAJOR.MINOR.PATCH, increment the:
  – MAJOR version when you make incompatible API changes,
  – MINOR version when you add functionality in a backwards-compatible manner, and
  – PATCH version when you make backwards-compatible bug fixes.

• Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

http://semver.org/
Variants and Revisions

[Staples&Hill, APSEC’04]
Managing variants

- Branching for variants does not scale well
- Requires special planning or tooling

- Many solutions
  - Configuration files
  - Preprocessors
  - Build systems
  - DSLs
  - Software product lines
  - ...

```c
/* common parts */
...
/* dependent on operating system */
#if (OS == Unix)
...
#else
    ...
#endif
...
CENTRALIZED VERSION CONTROL
(E.G., SVN)
Classes of version control systems

• Systems supporting merging and/or locking
• Local version control
• Central version control
  – Versions stored on central master server
  – Clients synchronize with server (update, commit)
  – CVS (1990), SVN (2004), Perforce, Visual SourceSafe
• Distributed version control
  – Many repositories; synchronization among repositories (push, pull)
  – Git (2005), Mercurial, Bitkeeper, ClearCase
Centralized Version Control

Client (version 5, branch M)

Client (version 5, branch M)

Client (revision 4, branch B)

Server (all versions)

checkout/ update/ commit

access control
Typical work cycle

- Once: Create local workspace
  - `svn checkout`

- Update workspace:
  - `svn update`

- Perform changes in workspace:
  - `svn add`
  - `svn delete`
  - `svn copy`
  - `svn move`

- Show workspace changes:
  - `svn status`
  - `svn diff`

- Revert changes in workspace:
  - `svn revert`

- Update and merge conflicts:
  - `svn update`
  - `svn resolved`

- Push workspace changes to server:
  - `svn commit`
## CVS vs. SVN

**CVS**
- Improvement over RCS in tracking entire directories
- Revision number per file
- Text files (binary files possible)

**SVN**
- Revision numbers for project
- Atomic commits (commiting multiple files at once)
- Tracking files and directories
- Support renaming
- Tracking of Metadata
DISTRIBUTED VERSION CONTROL (E.G., GIT)
Git

- Distributed version control
- No central server necessary (but possible)
- Local copies of repositories (containing all history)
  - Locally SVN like functionality: checkout, update, commit, branch, diff
- Nonlinear development: each local copy can evolve independently
- Synchronization among repositories (push/fetch/pull)
- Fast local operations (branch, commit, diff, ...)

15-214
Overview

Diagram showing the relationships and processes between GH, M2, M3, M1, and other components. The diagram includes arrows indicating flows and actions such as "clone, push, pull," "checkout / update," and "commit."
Distributed Versions

• Versions not globally coordinated/sorted
• Unique IDs through hashes, relationships tracked in successor graph
  – e.g., 52a0ff44aba8599f43a5d821c421af316cb7305
  – Possible to merge select changes (cherry picking)
  – Possible to rewrite the history as long as not shared remotely (git rebase etc)
• Cloning creates copy of repository (including all versions)
  – Tracks latest state when cloned, relevant for updating and merging
  – Normal checkout and commit operations locally
  – Commits don't change original repository
• Fetch and pull get missing versions from remote repository (one or more)
• Push operations sends local changes to remote repository (one or more), given access rights
Example workflow

Kernel developer
- checkout / update
- commit
- edit
- pull & merge

New developer
- checkout
- commit
- edit

Clone / pull
- push
- clone
Pull Requests

Kernel developer

Pull request: Github feature to ask developer to pull a specific change (alternative to sending email); integration with Travis CI
Forks

Kernel developer

clone / pull
push

pull & merge

commit
edit

checkout / update

clone
checkout
commit
edit

Fork: Github feature to clone repository on Github (own copy with full rights)
Forks

Kernel developer
checkout / update

Caution:
Please to not fork 214 repositories.

214 Collaboration Policy: "Here are some examples of behavior that are inappropriate: Making your work publicly available in a way that other students (current or future) can access your solutions, even if others’ access is accidental or incidental to your goals."

Fork: Github feature to clone repository on Github (own copy with full rights)
Repositories in mustache.js

GitHub Network Graph Viewer v4.0.0
Git History

```
2009-12-08 575c2d8 bchesneau@gmail.com (bchesneau@gmail.com) Merge remote branch 'thomo/master' into mthomo
* 2009-12-07 a315955 Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Merge branch 'master' of git://github.com/couchapp/couchapp
  into master of git://github.com/couchapp/couchapp
* 2009-12-01 cbd2bc8 Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Bug: Trailing carriage returns are not stripped.
* 2009-11-28 c649863 Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Merge branch 'master' of git://github.com/couchapp/couchapp
  into master of git://github.com/couchapp/couchapp
* 2009-11-27 1a18fba Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Fix wrong couchapp path
* 2009-11-27 1944c8b Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Merge branch 'master' of git://github.com/couchapp/couchapp
  into master of git://github.com/couchapp/couchapp
* 2009-11-27 ae97905 Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Fix UTF-8 issue on WinXP
  Merge branch 'master' of git://github.com/couchapp/couchapp
  into master of git://github.com/couchapp/couchapp
* 2009-11-19 00f4d99 Thomas.Mohaupt@gmail.com (Thomas.Mohaupt@gmail.com) Fix WinXP problem: Attachment name is not UTF-8 encoded
* 2009-11-17 6a52137 mo@.sql.int (mo@.sql.int) Fix: USERPROFILE handling
* 2009-11-08 34628c8 bchesneau@gmail.com (bchesneau@gmail.com) fix hooks and compress hook
* 2009-11-08 8ebdec2 bchesneau@gmail.com (bchesneau@gmail.com) fix compress options search path
* 2009-11-07 3c89e38 jasondavies@apache.org (jasondavies@apache.org) Generate attachment signatures before processing macros.
* 2009-11-07 54c3e83 jasondavies@apache.org (jasondavies@apache.org) Merge branch 'master' of git://github.com/couchapp/couchapp
* 2009-11-06 9b97679 bchesneau@gmail.com (bchesneau@gmail.com) fix__iter_
* 2009-11-06 4d4c6c7 bchesneau@gmail.com (bchesneau@gmail.com) fix typo. Thanks to markh.
* 2009-11-04 b37239e bchesneau@gmail.com (bchesneau@gmail.com) patch from @mg, thanks!
* 2009-11-02 2323238 bchesneau@gmail.com (bchesneau@gmail.com) bump version number to 0.5.1.
* 2009-11-01 d8c9789 bchesneau@gmail.com (bchesneau@gmail.com) add update template to generators
* 2009-11-01 0964b63 bchesneau@gmail.com (bchesneau@gmail.com) couchapp standalone for macosx via py2app
* 2009-11-28 4def8a6 bchesneau@gmail.com (bchesneau@gmail.com) make sure octent type is defined
  2009-11-28 1805363 benoitc@.none (benoitc@.none) new fixes while testing
  2009-11-28 051e778 bchesneau@gmail.com (bchesneau@gmail.com) rethink hook system, final version. Now like extensions each hooktype is a list of
  2009-11-28 c47718f bchesneau@gmail.com (bchesneau@gmail.com) rethink extensions. So now extensions are a list of key=value pair. where key is
  2009-11-27 53a2840 bchesneau@gmail.com (bchesneau@gmail.com) fix compress extension & load extensions like we load hooks
  2009-11-27 343862b bchesneau@gmail.com (bchesneau@gmail.com) fix template paths with windows
  2009-11-27 bfe21bc bchesneau@gmail.com (bchesneau@gmail.com) backport import_module from python 2.7
  2009-11-27 33804c benoitc@.none (benoitc@.none) more windows fix
  2009-11-27 a049b41 bchesneau@gmail.com (bchesneau@gmail.com) more fixes
```
Git and Central Repositories

© Scott Chacon “Pro Git”
Social Coding

Awareness/News Feeds
Git Internals

© Scott Chacon “Pro Git”
Git Internals

© Scott Chacon “Pro Git”
Git Internals

© Scott Chacon “Pro Git”
Git Internals

© Scott Chacon "Pro Git"
Summary

- Version control has many advantages
  - History, traceability, versioning
  - Collaborative and parallel development
- Locking vs. merging and merge conflicts
- Collaboration with branches
- From local to central to distributed version control