`git`: Control de versiones estilo Linus

Horst H. von Brand

Departamento de Informática
Universidad Técnica Federico Santa María

22 de octubre 2005

Una conferencia C.A.C.L.E.

Visión general

Definición de *SCM*

- *Software Configuration Management* (SCM) es el conjunto de actividades relacionadas con la administración de versiones de software
- Muchos productos se encuentran en versiones diferentes en diversos sitios. Desarrolladores tienen versiones en prueba.
- Es una actividad vital en el desarrollo de todo proyecto.

Modelos de *SCM*

Hay dos grandes modelos:

Centralizado: Hay un repositorio central, todos los cambios se registran allí. Los desarrolladores sacan una versión del repositorio, la modifican, y luego envian sus cambios de vuelta

Distribuido: Cada desarrollador tiene un repositorio propio. Maneja sus propias modificaciones, intercambia parches con los demás

Un sistema distribuido puede manejarse en forma centralizada, en la práctica no se usan *realmente* todos contra todos. Los sistemas distribuidos son cómodos para uso personal.

El modelo de desarrollo de Linux

- Hay muchos desarrolladores independientes
- Los desarrolladores se organizan en una jeraquía informal: Linus a la cabeza, lugartenientes de Linus, responsables por grandes subsistemas, ...
- Hay múltiples ramas de desarrollo: Versiones experimentales para arquitecturas específicas, drivers individuales, colecciones de parches en prueba, ...
- El tráfico de parches es gigantesco

Linux es marca registrada de Linus Torvalds

Requisitos de SCM para Linux

- Sistema distribuido (son muchas líneas de desarrollo)
- Gran rendimiento (ingresar cientos de parches en pocos minutos)

¡Advertencia!

- Lo que se discutirá aca son sistemas en desarrollo activo, aún hay *rápidos* cambios
- Los distintos paquetes no necesariamente van al mismo ritmo. . .
- Hay paquetes binarios extraoficiales RPM y Debian
- Ya están en/para algunas distribuciones. . .

GIT – The stupid content tracker

"git" can mean anything, depending on your mood:

- Random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant
- Stupid. Contemptible and despicable. Simple. Take your pick from the dictionary of slang.
- "Global information tracker": You're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "Goddamn idiotic truckload of sh*t": When it breaks

Bases de `git`

Un repositorio es una colección de objetos indexados por contenido (*blobs*, *trees*, *commits*)

Blob: Contenido de archivos

Tree: Directorio de objetos

Commit: Punto en la historia del proyecto, hace referencia a un *tree*, y contiene un comentario

Tag: Una marca, referencia un *commit*; opcionalmente está firmada digitalmente

Bases de `git`

- Los *commit* están organizados en un árbol a través de enlaces al padre y a los hijos
- Una rama de desarrollo es simplemente una referencia a un *commit*
- Se pueden intercambiar modificaciones entre repositorios, las historias *no* necesariamente coinciden

Plomería y porcelana

Linus describe a `git` como infraestructura en bruto ("plomería"), a la que se conectan artefactos más estéticos que ofrecen una interfaz amigable.

Plomería

`git` es la plomería sobre la cual se puede construir un sistema de control de versiones que funciona.

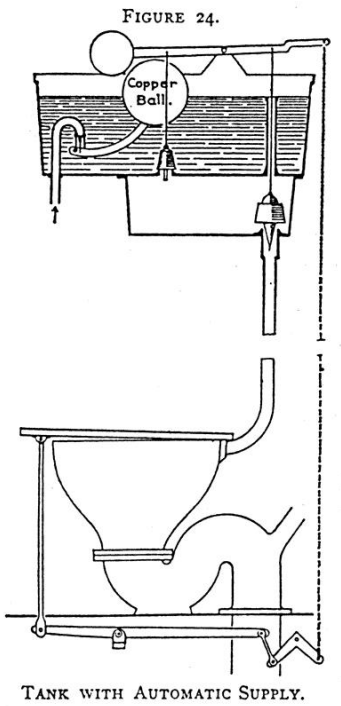

FIGURE 24.

TANK WITH AUTOMATIC SUPPLY.

Porcelana

Sobre `git` se construyen sistemas más sencillos de usar, de variadas funcionalidades.

Algunas porcelanas

`cogito`: Un conjunto de scripts que simplifican el uso de `git`

`StGIT`: Maneja colecciones de parches

`qgit`: Un ambiente gráfico para muchas de las operaciones de `git` y `StGIT`

Comandos de `cogito`

Hay un total de 36. Aceptan `--help` para ayuda detallada:

- `cg init`: Crear un repositorio
- `cg clone`: Copiar un repositorio
- `cg add`, `cg rm`: Agregar, eliminar archivos
- `cg restore`: Restaurar al estado del repositorio
- `cg commit`: Registrar los cambios hechos
- `cg export`: Exportar a un directorio o tarball
- `cg merge`: Integrar cambios remotos
- `cg update`: Traer cambios e integrarlos
- `cg tag`: Crear un tag
- `cg diff`: Diferencias entre versiones
- `cg mkpatch`: Crear serie de parches

Comandos de `git`

Hay un total de 142. Los más importantes son:

- `git init-db`: Crea un repositorio
- `git update-index [--add|--remove]`: Actualiza el índice actual, agregando/quitando archivos
- `git rename`: Cambia de nombre un archivo/directorio
- `git commit`: Registra cambios en el índice
- `git checkout`: Va a otra rama (con `-b` la crea)
- `git repack`: Compacta el repositorio
- `git fsck-objects`: Revisa consistencia
- `git prune`: Elimina inalcanzables

Comandos porcelanosos de `git`

- `git format-patch`: Crea parches
- `git send-email`: Envia parches creados por el anterior con `--mbox`
- `git applymbox`: Aplica de una vez la colección de parches en un *mbox*
- `git whatchanged`: Muestra cambios
- `gitk`: En realidad, no es parte de `git`, sino una interfaz gráfica para ver la historia

Referencias para git

git es el sistema de manejo de contenido base. Incluye herramientas para migrar desde otros sistemas y un extenso tutorial.
cogito es un conjunto de scripts para simplificar el uso de git
StGIT maneja colecciones de parches
qgit es una interfaz gráfica para manejar git y StGIT
El tutorial para cogito/git de Martin Langhoff es un buen punto de partida. El tutorial sobre git de Jeff Garzik da la perspectiva de un desarrollador activo.

Referencias para otros sistemas

El ensayo de David A. Wheeler resume muchos sistemas de control de versiones de código abierto. También discute temas relacionados a la seguridad de estos sistemas.

El standard histórico en código abierto es CVS (usado por el proyecto GNU y SourceForge, entre muchos otros); un sucesor que repara sus peores problemas es Subversion (hay proyectos importantes considerando migrar de CVS a SVN, o ya lo están haciendo); ambos centralizados.

Referencias para otros sistemas

Entre los sistemas distribuidos se cuentan GNU `arch` y sus derivados (`tla` es Arch 1, `revc` es Arch 2; Bazaar viene en versiones 1 (Bazaar, `baz`) y 2 (Bazaar-NG, `bzr`)).
El sistema `monotone` fue considerado seriamente para Linux, pero descartado por rendimiento.
Mercurial (`hg`) es un sistema distribuido diseñado para uso simple y eficiencia.
Otros sistemas incluyen Codeville (basado en un novedoso algoritmo de integración) y Darcs.