Το Mercurial SCM*

Γιώργος Κεραμίδας

30 Νοέμβρη 2007

Περίληψη

Το *Mercurial* είναι ένα κατανεμημένο σύστημα για διαχείριση πηγαίου κώδικα (και άλλων αρχείων). Αυτό το άρθρο παρουσιάζει τις βασικές αρχές της λειτουργίας του Mercurial[12], και περιγράφει συνοπτικά τον τρόπο χρήσης του, τα πλεονεκτήματα που έχει σε σχέση με άλλα συστήματα, και τα χαρακτηριστικά που το ξεχωρίζουν.

Εισαγωγή

Σαν προγραμματιστές ή μέλη μιας ομάδας που δουλεύει σε ένα κοινό «project» έχουμε πολλές φορές την ανάγκη να συνεργαστούμε στην ανάπτυξη ενός κοινού σετ από αρχεία πηγαίου κώδικα, συνοδευτικά scripts και Makefiles, κοκ.

Σε άλλες καταστάσεις, είναι αρκετό να κρατάμε ένα ασφαλές, «backup» αντίγραφο από τον πηγαίο κώδικα. Υπάρχουν περιπτώσεις που κάτι τέτοιο είναι αρκετό, όπως π.χ. όταν θέλουμε να κρατήσουμε ένα αντίγραφο από τις φωτογραφίες μιας εκδρομής σε CD-ROM ή DVD-ROM. Η διαχείριση του πηγαίου κώδικα έχει λίγο διαφορετική φύση όμως.

Συνήθως, ακόμα κι αν στο project εργάζεται ένα μόνο άτομο, υπάρχουν διάφορες εκδόσεις από τον ίδιο πηγαίο κώδικα: η έκδοση της Δευτέρας πρωί, η έκδοση με το GUI menu, η έκδοση με το έξτρα configuration file, και πάει λέγοντας. Οι περισσότεροι από εμάς έχουν χρησιμοποιήσει, τουλάχιστον μια φορά στη ζωή μας ο καθένας, τον απλούστερο τρόπο διαχείρισης των διαφορετικών εκδόσεων: ένα αριθμημένο σετ από καταλόγους (*dir1*, *dir2*, . . . , *dir85*, . . .).

Αυτός ο τρόπος διαχείρισης του κώδικα φαίνεται αρχικά εύκολος, αλλά μετά από λίγο τείνει να γίνει μπελάς και δύσκολος στη συντήρηση. Ας πούμε, είναι πολύ δύσκολο να απαντήσει κανείς σε ερωτήσεις όπως:

- Ποιά ήταν η πιο πρόσφατη έκδοση του κώδικα τη Δευτέρα, στις 09:00 το πρωί;
- Ποιά εβδομάδα ξεκίνησε το πρόγραμμα να πετάει core dumps όταν δέχεται κενή τιμή στο πεδίο X;
- Πόσες διαφορετικές αλλαγές έγιναν μεταξύ της έκδοσης *2.0-alpha5* και της έκδοσης *2.0-alpha6*, και τι άλλαξε με την καθεμία από αυτές;

Τα SCM συστήματα φτιάχτηκαν για να καλύψουν ακριβώς αυτές τις ανάγκες, καθώς και διάφορες άλλες οργανωτικές ανάγκες οι οποίες προκύπτουν όταν στο ίδιο project πρέπει να συνεργαστούν παραπάνω από ένα άτομα. Ορισμένα από τα SCM συστήματα είναι τόσο επιτυχημένα στο ρόλο αυτό, που χρησιμοποιούνται εδώ και αρκετά χρόνια από ομάδες εκατοντάδων ή και χιλιάδων ατόμων· ακόμη και από ομάδες που είναι διασκορπισμένες σχεδόν σε κάθε γωνιά της Γης.

Αυτό το άρθρο περιγράφει αρχικά ποιές είναι οι ανάγκες ενός developer που είναι μέλος μιας τέτοιας ομάδας. Ύστερα αναφέρεται συνοπτικά στον τρόπο με τον οποίο δουλεύουν τα «παραδοσιακά» SCM συστήματα (όπως το CVS ή το RCS/SCCS). Ο κεντρικός άξονας του άρθρου, όμως, είναι ο τρόπος με τον οποίο λειτουργεί ένα μοντέρνο SCM σύστημα: το Mercurial. Με βάση τον τρόπο λειτουργίας του Mercurial, θα αναφερθούμε στα νέα χαρακτηριστικά που εισάγει ένας τύπος συστήματος που λέγεται «*κατανεμημένο SCM*» (distributed SCM), και στους λόγους για τους οποίους μια ομάδα είναι προτιμότερο να χρησιμοποιήσει ένα τέτοιο σύστημα.

* Το άρθρο γράφτηκε για την εκδήλωση «Source Code Management Tools», του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιά.

1 Οι ανάγκες ενός developer

Η πικρή αλήθεια είναι ότι οι περισσότεροι developers *σιχαίνονται* τα συστήματα SCM. Πιστεύουν, τουλάχιστον στην αρχή, ότι τα συστήματα αυτά είναι «εμπόδιο» στη δουλειά τους, ότι τους καθυστερούν αδικιολόγητα ενώ έχουν πολλή δουλειά να κάνουν, ότι όλα τα SCM συστήματα είναι ενοχλητικά περίπλοκα στη χρήση, ή ότι βασικά μπορούν να κάνουν τη δουλειά τους πολύ καλύτερα στέλνοντας ο ένας στον άλλο «patches» ή αρχεία zip.

Για παράδειγμα, ο ίδιος ο Linus Torvalds, για πολύ καιρό, δεν ήθελε να ακούσει με τίποτα για το CVS. Προτιμούσε να του στέλνουν email με patches οι εκατοντάδες contributors του πυρήνα, τα οποία αυτός ή κάποιος άλλος kernel hacker έκανε review, merge και τέλος import στον επίσημο πυρήνα του Linux!

Είναι πάρα πολλοί οι προγραμματιστές οι οποίοι έχουν την ίδια άποψη με τον Linus για το CVS και αντίστοιχα SCM συστήματα.

Γενικά, αν λάβουμε υπόψη μας τα παράπονα όλων αυτών των προγραμματιστών, θα δούμε ότι σχετίζονται με μία ή περισσότερες από τις εξής απαιτήσεις ενός προγραμματιστή:

- Έχω *δουλειά* να κάνω, δε μπορώ να χάνω χρόνο με αυτό το πράγμα (ταχύτητα)
- Θέλω κάτι *απλό*, που να δουλεύει *σωστά* (ευκολία και σταθερότητα)
- Τα SCM εργαλεία μου πρέπει:
    1. να είναι εύκολα στη χρήση
    2. να με βοηθούν στη συνεργασία με άλλους
    3. να είναι γρήγορα

Αυτές ακριβώς είναι και οι βασικές ανάγκες ενός προγραμματιστή από ένα σύστημα SCM. Ένα σύστημα που δεν καλύπτει μία ή περισσότερες από αυτές τις ανάγκες, ενδέχεται να προκαλέσει σθεναρή αντίσταση όταν προσπαθήσουμε να το εισάγουμε σε μια ομάδα που δε χρησιμοποιεί ήδη κάποιο σύστημα SCM. Ακόμη και οι ομάδες που ήδη χρησιμοποιούν κάποιο σύστημα SCM, δεν έχουν κίνητρο να «αλλάξουν» σε κάποιο άλλο σύστημα, αν το νέο σύστημα κάνει τη ζωή τους ανυπόφορη, αφήνοντας ανικανοποίητες κάποιες από αυτές τις ανάγκες ή χειροτερεύοντας την καθημερινή εμπειρία των χρηστών του νέου SCM σε κάποιον από αυτούς τους τομείς.

Όπως κάθε άλλο «προϊόν», τα συστήματα SCM πρέπει να «πείσουν τους πελάτες τους (τους προγραμματιστές, στη συγκεκριμένη περίπτωση) ότι τα έχουν ανάγκη», αλλιώς, στην πλειοψηφία των περιπτώσεων, δε θα τα αποδεχτούν και δε θα τα ενσωματώσουν στον καθημερινό τρόπο εργασίας τους.

2 Κεντρικοποιημένα SCM συστήματα

Τα *κεντρικοποιημένα* συστήματα SCM ήταν τα πρώτα που φτιάχτηκαν για να καλύψουν δύο ειδών ανάγκες. Τις ανάγκες των *release engineers*, και τις ανάγκες των προγραμματιστών.

Οι release engineers συνήθως ενδιαφέρονται για το ιστορικό ενός project και τη δυνατότητα να αναπαράγουν πιστά οποιαδήποτε έκδοση του πηγαίου κώδικα.

Οι προγραμματιστές, από την άλλη, θέλουν εργαλεία τα οποία να είναι εύκολα στη χρήση και γρήγορα.

Το πρώτο source code management σύστημα που φτιάχτηκε ήταν το *SCCS*[15]. Το SCCS γράφτηκε από τον Marc J. Rochkind, σε έναν υπολογιστή IBM System/370, που έτρεχε OS/MVT. Αργότερα μεταφέρθηκε σε ένα PDP-11, το οποίο έτρεχε UNIX. Το SCCS ήταν ουσιαστικά το μοναδικό SCM, μέχρι την εμφάνιση του *Revision Control System*[14, 13, 3] (*RCS*), το 1980. Τόσο το SCCS όσο και το RCS μπορούν να χρησιμοποιηθούν από πολλά άτομα, αλλά πρέπει όλοι οι χρήστες να έχουν πρόσβαση στο ίδιο μηχάνημα.

Δεν υπάρχει πρόβλεψη για client-server χρήση σε αυτά τα πρωταρχικά συστήματα SCM, αλλά η επιρροή τους ήταν πολύ σημαντική στον τρόπο με τον οποίο αναπτύχθηκαν μελλοντικά συστήματα SCM. Η Sun Microsystems, χρησιμοποίησε το SCCS ως βάση για το *TeamWare*[9] (ένα από τα πρώτα συστήματα SCM που υποστήριζαν το κατανεμημένο μοντέλο που σήμερα χρησιμοποιεί και το Mercurial). Ο Larry McVoy, ο οποίος σχεδίασε το TeamWare, έφτιαξε ύστερα και το *BitKeeper*[1].

Το RCS ήταν η βάση πάνω στην οποία αναπτύχθηκε αργότερα το *PRCS*[5] και το *CVS*[2, 10]. Το PRCS χρησιμοποιεί ένα διαφορετικό μηχανισμό storage από το RCS, τον *Xdelta* (τον ίδιο μηχανισμό που χρησιμοποιεί και το δημοφιλές εργαλείο *rsync*[6]). Το CVS χρησιμοποιεί την ίδια μορφή αρχείων με το RCS, αλλά επεκτείνει το μοντέλο του RCS έτσι ώστε να υποστηρίζει client-server λειτουργία, tunnelling μέσα από διάφορα πρωτόκολα (RSH, SSH) και διάφορους μηχανισμούς πιστοποίησης ταυτότητας (authentication). Η μορφή αποθήκευσης του RCS χρησιμοποιείται, με κάποιες proprietary επεκτάσεις, και σε εμπορικά συστήματα SCM, όπως το *Perforce*[7], το οποίο επεκτείνει το client-server μοντέλο του CVS, και προσθέτει σε αυτό διάφορα χρήσιμα χαρακτηριστικά, όπως *cherry-picking* των αλλαγών, atomic changesets, καλύτερη υποστήριξη για merge-tracking μεταξύ branches, κλπ.

Σήμερα υπάρχουν δεκάδες συστήματα που δουλεύουν με κεντρικοποιημένο τρόπο. Στη Wikipedia αναφέρονται *40* διαφορετικά συστήματα. Η συντριπτική πλειοψηφία από αυτά είναι εμπορικά συστήματα, τα οποία υποστηρίζονται από κάποια εταιρεία. Αυτό δείχνει ότι το πεδίο του source code management έχει ενδιαφέρον και είναι τομέας στον οποίο η ανάπτυξη και εξέλιξη συνεχίζεται και σήμερα.

Η τάση για εξέλιξη στα συστήματα SCM έχει συνήθως διαφορετικό χαρακτήρα στα συστήματα SCM που είναι free software και στα συστήματα SCM που είναι εμπορικά προϊόντα. Τα μεν ελεύθερα SCM τείνουν να είναι *γραμμένα από προγραμματιστές για προγραμματιστές*, ενώ τα δε εμπορικά τείνουν να *είναι σχεδιασμένα με έμφαση στο management μέρος της διαδικασίας ανάπτυξης λογισμικού*. Χαρακτηριστικά είναι δύο quotes, από τον Dick Grune, που έφτιαξε το CVS, και από το web site της εταιρείας Perforce, που έφτιαξε το ομώνυμο SCM:

> I created CVS to be able to cooperate with my students Erik Baalbergen and Maarten Waage on the ACK (Amsterdam Compiler Kit) C compiler. The three of us had vastly different schedules (one student was a steady 9-5 worker, the other was irregular, and I could work on the project only in the evenings). Their project ran from July 1984 to August 1985. CVS was initially called *cmt*, for the obvious reason that it allowed us to commit versions independently.
> Dick Grune, *Dick Grune's web site*[10]

> All software development requires software configuration management (SCM) tools to manage the changes you make developing your software. Without SCM, releases go out the door with uncertain contents. Bug fixes get lost. And engineers see their development efforts wasted in a blundered process.
> Perforce company, *The Ten Minute Pitch*[8]

Το μοντέλο εργασίας των κεντρικοποιημένων SCM, ανεξάρτητα από το αν η ανάπτυξή τους κατευθύνεται από προγραμματιστές ή από release engineering ανάγκες, χρησιμοποιούν σχεδόν πάντα το ίδιο μοντέλο εργασίας: το μοντέλο copy-modify-merge.

2.1 Το μοντέλο εργασίας των κεντρικοποιημένων SCM

Τα πρώτα SCM συστήματα φτιάχτηκαν με βάση τον τρόπο που ήταν ήδη οργανωμένες οι ομάδες των προγραμματιστών εκείνο τον καιρό. Η αρχική υπόθεση για τον σχεδιασμό τους ήταν ότι όλοι οι προγραμματιστές έχουν πρόσβαση στο ίδιο σύστημα ή κάποιου είδους πρόσβαση σε ένα, κεντρικό *repository* πηγαίου κώδικα.

Για παράδειγμα, το RCS, το οποίο δεν έχει καν υποστήριξη για λειτουργία client-server, αποθηκεύει τα *metadata* του ιστορικού ενός αρχείου σε έναν υποκατάλογο με όνομα `RCS`. Κι αυτό είναι όλο. Δεν υπάρχει κάποιος *server*, αλλά όποιος θέλει να κάνει αλλαγές σε ένα versioned αρχείο, πρέπει να συνδεθεί κάπως στο ίδιο το μηχάνημα στο οποίο είναι αποθηκευμένα τα versioned αρχεία. Ο τρόπος με τον οποίο ένας προγραμματιστής μπορεί να κάνει αλλαγές χωρίς να υπάρχουν προβλήματα συγχρονισμού με άλλους προγραμματιστές, οι οποίοι εργάζονται με τα ίδια αρχεία, βασίζεται στο «κλείδωμα» των αρχείων που είναι υπό επεξεργασία κάθε στιγμή.

Αυτός ο τρόπος συγχρονισμού μεταξύ των μελών μιάς ομάδας είναι σχετικά απλός, και δεν έχει μεγάλο κόστος συγχρονισμού όταν η ομάδα αποτελείται από λίγα άτομα. Όσο αυξάνει το μέγεθος μιας ομάδας όμως,

η πιθανότητα να θέλουν περισσότερα από ένα άτομα να κάνουν αλλαγές στο ίδιο αρχείο αυξάνει εκθετικά, και με τον ίδιο ρυθμό αυξάνει και η καθυστέρηση την οποία πρέπει να υποστεί κάθε προγραμματιστής, μέχρι να βρει ένα κατάλληλο «χρονικό παράθυρο» και να κλειδώσει τα αρχεία τα οποία τον ενδιαφέρουν. Το RCS δεν είναι *scalable* όσο η ομάδα μεγαλώνει, και σύντομα οι προγραμματιστές θα αρχίσουν να θεωρούν ότι τους καθυστερεί κι ότι είναι εμπόδιο στην *πραγματική* τους δουλειά. Το αποτέλεσμα είναι συνήθως είτε να σταματήσουν εντελώς να το χρησιμοποιούν, είτε να κάνουν μεγάλες αλλαγές σε όλο και πιο αραιά μεταξύ τους χρονικά διαστήματα, προσπαθώντας να μειώσουν το χρόνο που «χάνουν λόγω του SCM συστήματος», αλλά με μεγάλο κόστος: πλέον οι αλλαγές δεν είναι πάντα self-contained, αυτόνομες και εύκολο να ξεχωρίσουν η μία από την άλλη, γιατί γίνονται σε «batches» με όλο και μεγαλύτερο μέγεθος.

Η λύση των κεντρικοποιημένων SCM που φτιάχτηκαν ύστερα από το RCS είναι το μοντέλο *copy-modify-merge*.

2.2 Το μοντέλο copy-modify-merge

TBD.

3 Το Mercurial

TBD.

3.1 Τι είναι ένα Mercurial repository

TBD.

3.2 Τι περιέχει ένα Mercurial repository

TBD.

3.3 Το μοντέλο clone-commit-merge

TBD.

4 Ευκολία χρήσης

TBD.

4.1 Δημιουργία ενός repository

TBD.

4.2 Τυπική χρήση ενός repository

TBD.

5 Ασφάλεια χρήσης

TBD.

5.1 Commit και μετά merge

TBD.

6 Συνεργασία μέσα σε μια ομάδα

TBD.

7 Η ταχύτητα του Mercurial

TBD.

7.1 Στατιστικά με το FreeBSD src/ tree

TBD.

7.2 Στατιστικά με το FreeBSD doc/ tree

TBD.

8 Αποδοτικότητα με το Mercurial

TBD.

8.1 Ελάχιστος χαμένος χρόνος

TBD.

8.2 Πολλαπλά feature-based clones

TBD.

9 Χρήσιμα features

TBD.

9.1 Επεκτασιμότητα

Το Mercurial δεν είναι μόνο γρήγορο και εύκολο στη χρήση. Είναι και επεκτάσιμο, με τη χρήση *extensions*[11] γραμμένων σε Python.

TBD.

9.1.1 Hooks

TBD.

9.1.2 Python extensions

TBD.

9.2 Patch queues

΄Ενα από τα πιο δημοφιλή extensions του Mercurial είναι το *MQ extension*[4].

TBD.

9.3 Merge extensions

TBD.

9.4 Υποστήριξη από το Emacs

TBD.

9.5 Υποστήριξη από το Trac

TBD.

10 Μελλοντικές εξελίξεις

TBD.

10.1 Windows GUI

TBD.

10.2 Tortoise Hg

TBD.

10.3 Καλύτερη υποστήριξη από το Trac

TBD.

10.4 Υποστήριξη από δημοφιλή IDEs

TBD.

Ευχαριστίες

Το άρθρο αυτό θα είχε σίγουρα πολύ περισσότερα λάθη και ελλείψεις χωρίς την πολύτιμη βοήθεια του Παναγιώτη Χατζή, που διάβασε κάποιες αρχικές εκδόσεις και μου έστειλε πολύτιμα σχόλια, παρατηρήσεις και διορθώσεις. Το άρθρο διάβασαν επίσης και ο Μανώλης Κιαγιάς, του οποίου η εμπειρία από την μέχρι τώρα χρήση του Mercurial με βοήθησε πολύ στο να γράψω ό,τι περιμένει κανείς να δει από ένα εισαγωγικό άρθρο στο Mercurial.

΄Εχω προσπαθήσει να διορθώσω όποια ορθογραφικά ή συντακτικά λάθη βρήκαμε, αλλά όσα έχουν παραμείνει είναι δικό μου φταίξιμο. Μπορείτε πάντα να με ενημερώσετε για όποιες ελλείψεις ή λάθη βρείτε, ή απλά να μου στείλετε σχόλια για μέρη στα οποία θα θέλατε περισσότερες λεπτομέρειες, στην διεύθυνση ηλεκτρονικής αλληλογραφίας *keramida@FreeBSD.org*.

Βιβλιογραφία

[1] BitMover Inc. BitKeeper. Available on: `http://www.bitkeeper.com/`.

[2] Dick Grune et al. Concurrent Versions System. Available on: `http://www.nongnu.org/cvs/`.

[3] Free Software Foundation. Revision Control System. Available on: `http://www.gnu.org/software/rcs/rcs.html`.

[4] Chris Mason. The Mercurial Queues extension. Available on: `http://www.selenic.com/mercurial/wiki/index.cgi/MqExtension`.

[5] Joshua MacDonald Paul N. Hilfinger, Luigi Semenzato. PRCS: the Project Revision Control System. Available on: http://prcs.sourceforge.net/.

[6] rsync Team. rsync. Available on: http://samba.anu.edu.au/rsync/.

[7] Perforce Software. Perforce. Available on: http://www.perforce.com/.

[8] Perforce Software. Perforce: The Ten Minute Pitch. Available on: http://www.perforce.com/pitch/page01.html.

[9] Sun Microsystems Inc. Introduction to Sun Workshop TeamWare. Available on: http://docs.sun.com/source/806-3573/intro.html.

[10] CVS Team. CVS: Some early history. Available on: http://www.cs.vu.nl/~dick/CVS.html#History.

[11] Mercurial Team. Mercurial Extensions. Available on: http://www.selenic.com/mercurial/wiki/index.cgi/Extensions.

[12] Mercurial Team. The Mercurial Wiki. Available on: http://www.selenic.com/mercurial/.

[13] Daniel Trinkle. Official RCS homepage. Available on: http://www.cs.purdue.edu/homes/trinkle/RCS/.

[14] Wikipedia. Revision Control System. Available on: http://en.wikipedia.org/wiki/Revision_Control_System.

[15] Wikipedia. Source Code Control System. Available on: http://en.wikipedia.org/wiki/Source_Code_Control_System.