

Index

A

Access, anonymous, password-authenticating server, 102–104
Accessibility of source code, 10
Accessorizing business model, 4
add command, 59, 60, 128–129, 261–262, 291
admin command, 140–141, 182, 262–266
ALL keyword, 121
—**allow-root** option, 257
annotate command, 145–150, 266–267
Annotation, 148–150
Announcements, 212
Anonymous access, password-authenticating server, 102–104
-A option
 admin command, 263
 checkout command, 267
 update command, 71, 79–80, 201, 288
-a option, 257
 admin command, 263
 edit command, 271
 history command, 274
 modules file, 298–299
 rtag command, 286
 watch command, 290
Apache Group, 218–219
Apache WWW server, 12, 13
Archiving projects, 223
Artistic License, 3
Atomic transaction
 lack of ability, 178–179
 locking repositories for atomicity, 249–251
@ signs, RCS format, 110–112
\$**Author**\$ keyword, 292
autoconf command, 194, 196–197, 237
Autoconfiguration, 88

B

Bar, Moshe, 4
Base file, 174, 302
Baserev file, 174, 302
Baserev.tmp file, 174, 302
“Bazaar-style” development, 7
Behlendorf, Brian, 13
Berkeley Software Distribution License. *See* BSD License.
Berliner, Brian, 9
Binary files, 60, 182
Blandy, Jim, 232
-b option, 257
 admin command, 263
 history command, 274
 import command, 276–277
 log command, 280
 rtag command, 286
 tag command, 287
Branches, 74–86, 152–168
 annotating, 148–150
 creating without working copies, 84–85
 keyword expansion, 163–164
 merging in and out of trunk, 160–162
 merging into trunk, 74–80
 merging repeatedly into trunk, 153–160
 replacing with new branch, 162–164
 splitting code for development and stable releases, 191–192
 vendor branches, 164–168
 working copies on multiple branches, 180
Branching code, 213–214, 220–222
Branch numbers, 79
Brand licensing business model, 4
BSD License, 3
Bug(s). *See* Problems.
Bug report mailing list, 96
BUGS file, 93

334 Building CVS from source

Building CVS from source, 88–90
Business models, open source, 4

C

Canonical sites, downloading CVS, 88
“The Cathedral and the Bazaar” (Raymond), 2, 6, 10, 204, 205
“Cathedral-style” development, 6
Cederqvist (manual), 25, 95–96
Cederqvist, Per, 25, 95, 239, 240
Cervisia, 244
Change(s)
 announcing in NEWS file, 200
 communication about, 58
 control over who can commit changes to CVSROOT, 114
 examining and reverting, 54–58
 merging from branch to trunk, 80–82
 process of making, 35
 reverting. *See* Reverting changes.
ChangeLogs, `cvs2cl.pl` program to create from CVS logs, 247–249
Checking out
 bad error message, 181
 definition, 23
 working copies, 33–36
Checkin.prog file, 174, 302
checkout command, 75, 267–268, 291
checkoutlist file, 122–124, 295
Civility
 forking code, 221–222
 rejecting code, 214
Code design, 229–233
 design document, 229–230
 dividing code into files and directories, 230–231
 dividing code into modules, 231–233
Code freezes, 190
Command(s), 255–291. *See also specific commands.*
 failing remotely but not locally, 184–185
 general patterns, 256
 list, 261–291
 options, 26, 257–261. *See also specific options.*
 short forms, 46
Command options, 28
Commit(s)
 definition, 23
 lack of atomic transaction ability, 178–179
 stored as diffs, 20
 wrong log message, 183
commit command, 43–51, 52, 128, 136, 201, 268–269
Commit emails, 120, 139–141

commitinfo file, 117–120, 295–296
Communication, language issues, 215
Communications features, 125–142
 getting rid of working copies, 141–142
 log messages and commit emails, 139–141
 watches, 125–139
Compatibility,
diff command options, 270–271
Compilation
 compiling from source code, 194–196
 instructions, 89–90
Computer Supported Collaborative Work. *See* CSCW.
\$COMSPEC environment variable, 304
Concurrent Versions System. *See* CVS.
config file, 114–115, 296
Configuration, 196–197
configure command, 90
Conflict(s), 19
 definition, 23
 detecting and resolving, 48–51
Conflict markers, 19, 50
Conflict of interests between developer needs and user needs, 188
Consistency of interfaces, 235–236
contrib/ directory, 252
-c option, 37–39
 checkout command, 267
 history command, 274
 rdiff command, 283
 tag command, 287
Copying .cvspass files, 183
Copy-modify-merge model, 18–21
 commits stored as diffs, 20
 terminology, 20–21
Courtesy
 forking code, 221–222
 rejecting code, 214
Cox, Alan, 13
Credit, giving to developers, 13
CSCW, 10
CVS
 beginnings, 9–10
 functions, 10–12
 scope of use, 11–12
cvschroot, 246
\$CVS_CLIENT_LOG environment variable, 304
\$CVS_CLIENT_PORT environment variable, 304
cvs2cl.pl command, 247–249
cvsco, 247
cvsdiscard, 246–247
CVS distributions, 92–97

- informational files, 92–93
- information sources, 96–97
- Macintosh, 92
- subdirectories, 94–96
- Windows, 91–92
- cv**do, 245
- \$CVSEEDITOR** environment variable, 304
- \$CVSIGNORE** environment variable, 304
- cvsignore file, 296–297
- .cvsignore file, 301
- \$CVS_IGNORE_REMOTE_ROOT** environment variable, 305
- cvslck command, 249–251
- .cvspass file, 302
 - copying, 183
 - inability to find, 180
- \$CVS_PASSFILE** environment variable, 305
- cvspurge, 246
- .cvsrc file, 301
- \$CVS_RCMD_PORT** environment variable, 305
- \$CVSREAD** environment variable, 305
- cvsrmdm, 246
- CVSROOT directory, 113–124, 294–301
 - checkoutlist file, 122–124, 295
 - commitinfo file, 117–120, 295–296
 - config file, 114–115, 296
 - control over who can commit changes, 114
 - cvsignore file, 296–297
 - cvswrappers file, 122, 297
 - editinfo file, 122, 297
 - history file, 297
 - loginfo file, 117–120, 298
 - modules file, 115–117, 298–299
 - notify file, 122, 299
 - passwd file, 100, 101–104, 300
 - rcsinfo file, 120–121
 - taginfo file, 121–122, 300
 - users file, 300
 - val-tags file, 301
 - verifysg file, 120–121, 301
- \$CVSROOT** environment variable, 28–29, 98, 142, 305
- \$CVS_RSH** environment variable, 305
- \$CVS_SERVER** environment variable, 305
- \$CVS_SERVER_SLEEP** environment variable, 305
- cvsu, 244–245
- \$CVSUMASK** environment variable, 305
- CVSUp command, 252
- CVS username, 128
- cvstools programs, 243–247
 - Cervisia, 244
 - cvschroot, 246

- cvscsco, 247
- cvstdiscard, 246–247
- cvstdo, 245
- cvstdump, 246
- cvstrmdm, 246
- cvstu, 244–245
- CVSWeb, 252
- \$CVSWRAPPERS** environment variable, 306
- cvswrappers file, 122, 297
- .cvswrappers file, 302
- Cyclic Software Web site, 97

D

- Data structures, documenting, 236–237
- Date(s)
 - formats, 67–68, 256
 - retrieving previous revisions by, 63–68
- \$Date\$** keyword, 293
- DEFAULT** keyword, 121
- Design. *See* Software design.
- Design documents, 229–230
- DEVEL-CVS file, 93
- Developers
 - conflict of interests between developer needs and user needs, 188
 - expertise, 13–14
 - giving credit, 13
 - language issues in communicating, 215
- Development charters, 219
- Diff(s), commits stored as, 20
- diff** command, 8, 37–40, 54–55, 60, 71, 80–81, 269–271
 - compatibility options, 270–271
 - keyword substitution, 291
 - viewing changes one file at a time, 41–43
- Directories. *See also* Subdirectories.
 - adding, 59
 - code design, 230–231
 - removing, 61–62
 - renaming, 62–63
- Distributions. *See* CVS distributions.
- Documentation, 209, 210
 - data structures, 236–237
 - design, 229–230
- D option
 - annotate** command, 266
 - checkout** command, 267
 - diff** command, 269
 - export** command, 181, 272
 - history** command, 274, 276
 - rdiff** command, 283

336 -d option

- rtag** command, 286
 - tag** command, 287
 - update** command, 64–66, 288
- d** option, 26, 257–258
 - checkout** command, 76, 267
 - export** command, 272
 - import** command, 277
 - log** command, 280
 - modules file, 299
 - release** command, 142, 284–285
 - rtag** command, 286
 - tag** command, 287
 - update** command, 173, 288
- Downloading CVS from canonical sites, 88
- Dreilinger, Sean, 97

E

- Economics
 - cost of proprietary versus free software design, 227–228
 - free software, 1–4, 13–14
- edit** command, 126, 131, 136, 137, 271–272
- editinfo file, 122, 297
- \$EDITOR** environment variable, 306
- editors** command, 134, 272
- Elib, 240
- Emacs/CVS interfaces, 240. *See also* pcl-cvs interface.
- Emails
 - commit**, 120, 139–141
 - turning on notification, 127–128
- Endres, Tim, 27
- Entries.Backup file, 172, 303
- Entries file, 35, 50, 303
- Entries.Log file, 172, 303
- Entries.Static file, 173, 303
- Environments
 - design for software portability, 237–238
 - repository access, 28–31
 - supported, 27
- Environment variables, 304–306
- e** option, 258
 - admin** command, 263
 - history** command, 274
 - modules file, 299
- Error handling in pcl-cvs, 243
- Evaluating code, 215–217
- Evolutionary design, 233–234
- Expanding keywords, 60, 150–151
- Expertise of developers, 13–14
- export** command, 198–199, 272–273
 - D** option, 181
- ext** command, 29–30

F

- Factionalism, 14–15
- Failure of projects, 203–204
- FAQ file, 93
- Feature freezes, 190, 191
- Fenner, Bill, 252
- Files. *See also specific file names.*
 - adding, 59
 - binary, 60
 - code design, 230–231
 - lock, removing, 176–177
 - removing, 61, 112–113
 - renaming, 62
 - run control, 301–302
 - working copy, 302–304
- Flags.
 - See also specific options.*
 - adding automatically, 63
- Fogel, Karl, 206–207
- F** option
 - commit** command, 268
 - rtag** command, 286
 - tag** command, 287
- f** option, 120, 258
 - annotate** command, 266
 - checkout** command, 267
 - commit** command, 268
 - export** command, 272
 - history** command, 274
 - rdiff** command, 283
 - remove** command, 285
 - rtag** command, 286
 - tag** command, 287
 - update** command, 288
- Forking code, 213–214, 219, 220–222
 - civility toward original maintainer, 221–222
 - naming forked versions, 220–221
 - trademarks, 221
- Formats for dates, 67–68, 256
- FreeBSD, 6, 12
- Free software, 1–4
 - design, proprietary software design versus, 226–228
 - design principles, 234–238
 - reasons for writing, 12–15
- Free Software Foundation, 5
- Freezing, 190–191



G

Global options, 28, 257–261. *See also specific options.*
 GNOME free desktop environment, 12
 GNU, 5–6
 GNU Emacs/XEmacs divergence, 222
 GNU General Public License. *See* GPL.
 GNU/RCS, 25
 GNU site, releases available, 89
 GPL, 3, 209, 210
 Grune, Dick, 9
gserver command, 29, 30, 273

H

HACKING file, 93
 Hard freezes, 190–191
 Hardware producers, support for free software, 14
\$Header\$ keyword, 293
 —**help** option, 258
 —**help-options** option, 258
 —**help-synonyms** option, 258–259
history command, 142–145, 273–276
 History file, 123, 297
\$HOMEDRIVE environment variable, 306
\$HOME environment variable, 306
 Home pages, project information, 210
\$HOMEPATH environment variable, 306
 -**h** option, **log** command, 281

I

\$Id\$ keyword, 293
import command, 31–32, 276–279
 info-cvs mailing list, 96
init command, 279
 Installing pcl-cvs, 240–242
 Interfaces, consistency, 235–236
 Interim releases, 89
 Internet resources, CVS distributions, 96
 Invariants in software design, 228–229
 Invoking CVS, 27–28
 -**I** option
 admin command, 263
 import command, 277–279
 update command, 288
 -**i** option
 admin command, 263
 modules file, 299
 ISO 8601 format, 67

J

Jalindi Igloo command, 251
 Java client, 27
 -**j** option
 checkout command, 267
 update command, 58, 288–289

K

-**kb** option, 182
 add command, 60
 keyword expansion, 292
 update command, 164
 Keyword expansion, 60, 150–151, 291–294
 branches and, 163–164
 controlling, 291–292
 list of keywords, 292–294
 Kingdon, Jim, 9
 -**kk** option
 keyword expansion, 292
 update command, 164
 -**kkvl** option, keyword expansion, 292
 -**kkv** option
 admin command, 182
 keyword expansion, 292
 -**ko** option
 add command, 60
 keyword expansion, 292
 -**k** option
 admin command, 141, 262, 263
 checkout command, 267–268
 cvswrappers file, 297
 diff command, 269
 export command, 272
 import command, 279
 keyword substitution, 291–292
 update command, 289
kserver command, 29, 30, 279

L

Language, communicating with developers, 215
 Licenses, 1, 3–4, 209–210
 Line-end conversion, problems, 182
 Linux, 6, 13
 Linux kernel, 191
LockDir parameter, config file, 115
\$Locker\$ keyword, 293
 Lock files, removing, 176–177
 Lock-modify-unlock development model, 18

338 log command

- log command, 51–53, 279–282
 - cvs2cl.pl program to create GNU-style ChangeLogs from CVS logs, 247–249
- login command, 282
- loginfo file, 117–120, 298
- \$Log\$ keyword, 151–152, 293
- Log messages
 - browsing, 51–54
 - changing after committing, 140–141
 - committing files with wrong log message, 183
 - definition, 21
- logout command, 282
- LokDir parameter, config file, 296
- l option, 259
 - admin command, 263
 - annotate command, 266
 - checkout command, 268
 - commit command, 268
 - diff command, 269
 - edit command, 272
 - editors command, 272
 - export command, 272
 - history command, 274
 - log command, 281
 - rdiff command, 283
 - remove command, 285
 - rtag command, 286
 - status command, 287
 - tag command, 287
 - unedit command, 288
 - watch command, 290
 - watchers command, 291
- L option, admin command, 263
- Loss leader business model, 4

M

- Macintosh, 27
 - getting CVS, 92
 - limitations of CVS version, 92
- Maintainers, 217–220
 - changing, 222–223
 - committees, 218–220
 - individual, 217–218
 - rejecting proposed code, 213, 214
- make command, 194–196
- merge command, 74–75
- Merging
 - alternative to, 162–164
 - branches into trunk, 74–80
 - branches into trunk repeatedly, 153–160
 - changes from branch to trunk, 80–82
 - multiple merges, 82–84
 - in and out of trunk, 160–162
- Microsoft Visual Source Safe. *See* VSS.
- Mirroring, CVSup program, 252
- Modules, code design, 231–233
- modules file, 115–117, 298–299
- Molli, Pascal, 97
- Monnier, Stefan, 239
- m option, 32, 120
 - admin command, 140–141, 262, 263
 - commit command, 269
 - cvs wrappers file, 297
 - history command, 274
 - import command, 279
- Mosix (software), 4
- Mosix, Inc., 4
- Moving
 - directories, 63
 - files without losing revision history, 183
- Mozilla, 208–209
- Mozilla Public License. *See* MozPL.
- MozPL, 4

N

- \$Name\$ keyword, 293
- NetBSD, 6, 12
- Netscape Communications Corporation, 208
- Netscape Public License. *See* NPL.
- New features
 - in CVS, 168–169
 - sending, 186
- New projects, starting. *See* Starting projects.
- NEWS file, 92–93, 200
- Nodes, 115
- N option
 - admin command, 263
 - checkout command, 268
 - export command, 272
 - log command, 281
- n option, 259
 - admin command, 264–265
 - checkout command, 268
 - commit command, 269
 - export command, 273
 - history command, 274
 - rtag command, 286
 - tag command, 287
- Notify file, 122, 174, 299, 303
- Notify.tmp file, 174, 303
- NPL, 4

O

Official CVS site, downloading CVS, 88
 O'Neill, Melissa, 207
 -o option
 history command, 274
 modules file, 299
 OpenBSD, 6, 12
 Open source business models, 4
 Open source software, 2–3
 licenses, 3–4
 origins, 5–6
 types of development, 6–7

P

Packaging, 197–199, 209–212
 checklist, 210–212
 documentation, 209, 210
 licenses, 209–210
passwd command, 100
 passwd file, 100, 101–104, 124, 300
 Password-authenticating server, 99–104
 anonymous access, 102–104
patch command, 8
 Patches, sending, 186
\$PATH environment variable, 306
 pcl-cvs interface, 239–243
 error handling, 243
 installing, 240–242
 using, 242–243
 % code, 119
 Perl, 13
 Permissions, changing, 179–180
 Platforms. *See* Environments.
 -P option
 checkout command, 268
 export command, 273
 update command, 61–62, 289
 -p option
 checkout command, 268
 history command, 274
 update command, 54–56, 289
 Portability, 174
 software design, 237–238
 PostgreSQL database, 12
PreservePermissions parameter, config file, 115, 296
 Problems, 171–186
 common, solving, 175–186
 reporting, 185–186, 211
 in repository permissions, 174–175
 in working copy administrative area, 172–174

Project(s)

 listing, 184
 new, starting, 31–33
 running. *See* Running projects.
 starting. *See* Starting projects.
 success and failure, 203–204
 Project history, 142–150
 annotate command, 145–150
 history command, 142–145
 Project home pages, 210
projname argument, 32
 Proprietary software, design, free software design versus,
 226–228
pserver command, 28–29, 91, 128, 282–283
 inability to get method to work, 177–178
:pserver: command, 99

Q

-Q option, 35–36, 41, 259
 -q option, 41, 259
 admin command, 265

R

rannotate command, 169, 283
 Raymond, Eric, 2, 6, 10, 12, 204, 205, 217, 218
RCBBIN parameter, config file, 296
 RCS, 9, 24–25, 79
\$RCSfile\$ keyword, 293
 RCS format
 @ signs, 110–112
 version control files, 105–112
 rcsinfo file, 120–121
 RCS keyword strings, 60
rdiff command, 283–284
 Red Hat Package Manager. *See* RPM.
 Red Hat Software, 14
 Regular expressions, 118
 Rejecting proposed code, 213, 214
 Release(s), 187–192
 available on GNU site, 89
 avoiding
 “code cram” effect, 189–190
 development versus stable branches, 191–192
 freezing, 190–191
 interim, 89
 recording, 200–201
 starting release process, 188–189
 testing, 192–194
 usefulness, 206–209
 uses, 187–188

340 release command

- release command, 141–142, 284–285
- release tag, **rtag** command, 84
- releasetag** argument, 32
- Releasing, 199–201
- remove** command, 285
- Removing
 - directories, 61–62
 - files, 61, 112–113
 - subdirectories, 182–183
- Renaming files and directories, 62–63
- Repositories, 19
 - access, 28–31
 - comparing working copies with state of project in repository, 36–40
 - definition, 21, 34
 - enabling watches, 127–130
 - listing projects, 184
 - locking for atomicity, 249–251
 - permissions, 174–175
 - RCS, 23
 - sending changes to repository, 43–51
 - starting.—See Starting repositories.
 - structure, 104–110
 - Web interface to browsing, 252
- Repository administration, 87–124
 - administrator's role, 87
 - building CVS from source, 88–90
 - CVS distributions. See CVS distributions.
 - getting and installing CVS on Macintosh, 92
 - getting and installing CVS under Windows, 91–92
 - starting repositories. See Starting repositories.
- Repository administrative files, 294–301
 - list, 295–301
 - shared syntax, 294–295
- Repository file, 34–35, 303
- Repository revision, 48
- Reverting changes, 54–58
 - fast method, 58
 - slow method, 55–57
- Revision(s)
 - definition, 21
 - moving files without losing revision history, 183
 - previous, retrieving by tag, 68–74
 - retrieving by date, 63–68
 - retrieving by tag, 63
 - versions versus, 44
- Revision Control System. See RCS.
- \$Revision\$** keyword, 152, 293
- Revision numbers, 44–48
 - merging branches with trunk, 78–80
 - recording releases, 200–201
- rlog** command, 169, 285
- Root file, 34, 303
- R** option
 - annotate** command, 267
 - checkout** command, 268
 - commit** command, 269
 - diff** command, 269
 - edit** command, 272
 - editors** command, 272
 - export** command, 273
 - log** command, 281
 - rdiff** command, 283
 - remove** command, 285
 - rtag** command, 286
 - status** command, 287
 - tag** command, 287
 - unedit** command, 288
 - update** command, 289
 - watch** command, 290
 - watchers** command, 291
- r** option, 259
 - annotate** command, 267
 - checkout** command, 268
 - commit** command, 201, 269
 - diff** command, 71, 269–270
 - export** command, 273
 - history** command, 274, 276
 - log** command, 281
 - rdiff** command, 284
 - rtag** command, 286
 - tag** command, 287
 - update** command, 55–57, 289
- Roskin, Pavel, 243
- RPM, 211
- rsh** command, 29–30
- rtag** command, 84–85, 169, 285–286
 - release** tag, 84
- Run control files, 301–302
- Running projects, 212–223
 - archiving projects, 223
 - communicating with developers, 215
 - evaluating code, 215–217
 - forking code, 213–214, 219, 220–222
 - maintainers, 217–220, 222–223
 - rejecting proposed code, 213, 214

S

- SCCS, 25–26
- SCCS Identification Number. See SID.
- Sell it, free it business model, 4
- Sending

- changes to repository, 43–51
 - new features, 186
- server** command, 286
- Service enabler business model, 4
- Shared variables, repository administrative files, 295
- SID, 25–26
- Snapshots, 68
 - retrieving by tag name, 68–74
- Soft freezes, 190
- Software design, 225–238
 - avoiding limits on size of data, 235
 - caution regarding, 238
 - code. *See* Code design.
 - comprehensibility, 225, 234
 - documenting data structures, 236–237
 - evolution-centered, 233–234
 - importance, 225–226
 - interface consistency, 235–236
 - invariants, 228–229
 - portability, 237–238
 - for proprietary versus free software, 226–228
- Software franchising business model, 4
- Software licenses, 1, 3–4, 209–210
- s** option, 259–260
 - admin** command, 265
 - checkout** command, 268
 - log** command, 281
 - modules file, 299
 - rdiff** command, 284
- Source code
 - accessibility, 10
 - building CVS from, 88–90
 - compiling from, 194–196
- \$Source\$** keyword, 294
- ssh** command, 29, 30
- Stallman, Richard, 5, 14
- Starting projects, 31–33, 204–212
 - announcing projects, 212
 - packaging, 209–212
 - usefulness of releases, 206–209
- Starting repositories, 97–124, 113–124
 - @ signs in RCS files, 110–112
 - password-authenticating server, 99–104
 - removing files, 112–113
 - repository structure, 104–110
- \$State\$** keyword, 294
- status** command, 46–48, 79, 286–287
- Sticky tags, 181
- Subdirectories, 34
 - code design, 231
 - CVS distributions, 94–96
 - removing, 182–183

- Success of projects, 203–204
- Support sellers business model, 4
- SystemAuth** parameter, config file, 114, 296

T

- Tag(s)
 - creating without working copies, 84–85
 - listing, 183–184
 - names, 110
 - recording releases, 200–201
 - retrieving previous revisions by, 63, 68–74
 - sticky, 181
- tag** command, 76, 77, 287
- Tag file, 173–174, 304
- taginfo file, 121–122, 300
- tcommit** command, 136
- Technical judgment, 215–217
- tedit** command, 136
- \$TEMP** environment variable, 306
- Template file, 174, 304
- Temporary watchers, 132
- Testing releases, 192–194
 - automated, 193–194
 - recruiting and retraining testers, 193
- Third-party tools, 239–253. *See also specific tools.*
 - definition, 239
 - writing, 252–253
- 386BSD, 6
- Tichy, Walter, 9
- \$TMPDIR** environment variable, 306
- \$TMP** environment variable, 306
- TopLevelAdmin** parameter, config file, 114, 296
- T** option, 260–261
 - history** command, 274
- t** option, 261
 - admin** command, 265
 - history** command, 274
 - log** command, 281
 - modules file, 299
 - rdiff** command, 284
- Torvalds, Linus, 6, 13, 218
- Trademarks, forking code, 221
- Tromey, Tom, 243
- Troubleshooting. *See* Problems.
- Trunk, 74
 - merging branches. *See* Branches.
 - merging branches into trunk, 74–79
 - merging changes from branch to trunk, 80–81
- tunedit** command, 136

U

- unedit** command, 126, 132–133, 136, 287–288
- Unix, 27
 - diff** and **patch** commands, 8
- u** option
 - admin** command, 265
 - history** command, 274
 - modules file, 299
 - rdiff** command, 284
- U** option, **admin** command, 265
- Update(s), definition, 21
- update** command, 40–41, 48, 49, 61–62, 288–290
 - A** option, 72, 79–80, 201
 - D** option, 64–66
 - d** option, 173
 - fast method of reverting, 58
 - kb** option, 164
 - keyword substitution, 291
 - kk** option, 164
 - multiple merges, 81, 82–84
 - naming specific files, 41
 - slow method of reverting, 55–57
- Update.prog file, 174, 304
- Up-to-date check, files failing, 177
- Usenet newsgroups, CVS information, 97
- USERNAME**, 100
- users file, 300
- User variables, repository administrative files, 295

V

- val-tags file, 301
- Variables
 - environment, 304–306
 - shared, repository administrative files, 295
 - user, repository administrative files, 295
- VC, 240
- Vendor branches, 164–168
- vendortag** argument, 32
- verifymsg file, 120–121, 301
- Version(s), revisions versus, 44
- Version Control.
 - See* VC.
- Version control files, RCS format, 105–112
- Version numbers, 211
 - stability, 192
- \$VISUAL** environment variable, 306
- Visual Source Safe. *See* VSS.
- v** option, 261
 - status** command, 287
- V** option, **admin** command, 266

VSS, 21–22

W

- Wall, Larry, 8, 13
 - Watch(es), 125–139
 - controlling what actions are watched, 133–134
 - enabling in repository, 127–130
 - ending editing sessions, 132–133
 - finding out who is watching what, 134–136
 - inability to turn off, 182
 - reminding people to use, 136–139
 - using in development, 130–132
 - watch** command, 126, 138, 290
 - subcommands, 290
 - watchers** command, 134–135, 290–291
 - Web interface, to CVS repositories, 252
 - Web sites, CVS information, 97
 - Widget frosting business model, 4
 - Windows, 27
 - getting CVS, 91–92
 - limitations of CVS version, 92
 - w** option, 261
 - history** command, 274
 - import** command, 279
 - log** command, 281–282
 - W** option, **update** command, 290
 - Workarounds, 11
 - Working copies, 18
 - checking out, 33–36
 - comparing with state of project in repository, 35–40
 - definition, 21
 - files, 302–304
 - getting rid of, 141–142
 - on multiple branches, 180
 - Working revision, 48
 - World-writable files, 123
 - writing tools, 252–253
- ## X
- XEmacs text editor, 12
 - x** option, 261
 - admin** command, 266
 - history** command, 275, 276
 - X** option, **history** command, 275
- ## Z
- Zawinski, Jamie, 208
 - Zeller, Henner, 252
 - z** option, 261
 - history** command, 275, 276

CVS Commands

Note: This card contains the most important CVS commands, options, and keywords. For a complete list, see Chapter 11.

Basic Structure of CVS Commands

`cv`s [*global_options*] *command* [*command_options*]
[*command_args*]

Global Options

Option	Purpose
<code>--allow-root=<i>rootdir</i></code>	Specify legal CVSROOT directory (server only) (not in CVS 1.9 and older).
<code>-a</code>	Authenticate all communication (client only) (not in CVS 1.9 and older).
<code>-b</code>	Specify RCS location (CVS 1.9 and older).
<code>-d <i>rootdir</i></code>	Specify the CVSROOT.
<code>-e <i>editor</i></code>	Edit messages with editor.
<code>-f</code>	Do not read the <code>~/cvsrc</code> file.
<code>-H [<i>command</i>] or --help [<i>command</i>]</code>	Print a help message. If a command is indicated, the help will be context-sensitive.
<code>-l</code>	Do not log in to CVSROOT/history file.
<code>-n</code>	Do not change any files.
<code>-Q</code>	Operate in quiet mode.
<code>-q</code>	Be somewhat quiet.
<code>-r</code>	Make new working files read-only.
<code>-s <i>variable=value</i></code>	Set a user variable.
<code>-T <i>tempdir</i></code>	Put temporary files in <i>tempdir</i> .
<code>-t</code>	Trace CVS execution.
<code>-v or --version</code>	Display version and copyright information for CVS.
<code>-w</code>	Make new working files read-write.
<code>-x</code>	Encrypt all communication (client only).
<code>-z <i>gzip-level</i></code>	Set the compression level (client only).

Keyword Expansion Modes

Mode	Expansion
<code>-kb</code>	<i>no expansion; file is binary</i>
<code>-kk</code>	<code>\$Id\$</code>
<code>-kkv</code>	<code>\$Id: file1,v 1.1 2000/12/09 03:21:13 moshe</code>
<code>Exp \$</code>	
<code>-kkvl</code>	<code>\$Id: file1,v 1.1 2000/12/09 03:21:13 moshe</code>
<code>Exp karl \$</code>	
<code>-ko</code>	<i>no expansion</i>
<code>-kv</code>	<code>file1,v 1.1 2000/12/09 03:21:13 moshe Exp</code>

Keywords

Keyword	Expanded Form Example
<code>\$Author\$</code>	<code>\$Author: moshe \$</code>
<code>\$Date\$</code>	<code>\$Date: 2000/12/09 03:21:13 \$</code>
<code>\$Header\$</code>	<code>\$Header: /home/files/file1,v 1.1 1993/12/09 \ 03:21:13 moshe Exp karl \$</code>
<code>\$Id\$</code>	<code>\$Id: file1,v 1.1 1993/12/09 03:21:13 moshe \ Exp karl \$</code>
<code>\$Locker\$</code>	<code>\$Locker: karl \$</code>
<code>\$Name\$</code>	<code>\$Name: snapshot_1_14 \$</code>
<code>\$RCSfile\$</code>	<code>\$RCSfile: file1,v \$</code>
<code>\$Revision\$</code>	<code>\$Revision: 1.1 \$</code>
<code>\$Source\$</code>	<code>\$Source: /home/files/file1,v \$</code>
<code>\$State\$</code>	<code>\$State: Exp \$</code>
<code>\$Log\$</code>	<code>\$Log: file1,v \$</code>

Commands, Command Options, and Command Arguments

add

Syntax: **add** [*options*] [*files*]

Used for: Adding a new file or directory.

Option/Argument	Purpose
<code>-kkflag</code>	Set keyword expansion.
<code>-m <i>msg</i></code>	Set file description.

admin

Syntax: **admin** [*options*] [*files*]

Used for: Administration of history files in the repository.

Option/Argument	Purpose
<code>-b[<i>rev</i>]</code>	Set default branch.
<code>-ksubst</code>	Set keyword substitution.
<code>-l[<i>rev</i>]</code>	Lock revision <i>rev</i> , or latest revision.
<code>-mrev:<i>msg</i></code>	Replace the log message of revision <i>rev</i> with <i>msg</i> .
<code>-orange</code>	Delete revisions from the repository.
<code>-q</code>	Run quietly; do not print diagnostics.
<code>-ssstate[<i>rev</i>]</code>	Set the state.
<code>-t</code>	Set file description from standard input.
<code>-tfile</code>	Set file description from <i>file</i> .
<code>-t-string</code>	Set file description to <i>string</i> .
<code>-u[<i>rev</i>]</code>	Unlock revision <i>rev</i> , or latest revision.



annotate

Syntax: **annotate** [*options*] [*files*]

Used for: Showing the last revision where each line was modified.

Option/Argument	Purpose
-D date	Annotate the most recent revision no later than <i>date</i> .
-f	Use head revision if tag/date not found.
-l	Local; run only in current working directory.
-R	Operate recursively (default).
-r rev	Annotate revision <i>rev</i> .

checkout

Syntax: **checkout** [*options*] [*project(s)*]

Used for: Checking out a module from the repository into a working copy.

Option/Argument	Purpose
-A	Reset any sticky tags/date/options.
-c	Output the module database.
-D date	Check out revisions as of <i>date</i> (is sticky).
-d dir	Check out into <i>dir</i> .
-f	Use head revision if tag/date not found.
-j rev	Merge in changes.
-k kflag	Use <i>kflag</i> keyword expansion.
-l	Local; run only in current working directory.
-N	Don't "shorten" module paths if -d specified.
-n	Do not run module program (if any).
-P	Prune empty directories.
-p	Check out files to standard output (avoids stickiness).
-R	Operate recursively (default).
-r tag	Check out revision <i>tag</i> (is sticky).
-s	Like -c , but include module status.

commit

Syntax: **commit** [*options*] [*files*]

Used for: Checking changes into the repository.

Option/Argument	Purpose
-F file	Read log message from <i>file</i> .
-f	Force the file to be committed; disables recursion.
-l	Local; run only in current working directory.
-m msg	Use <i>msg</i> as log message.
-n	Do not run module program (if any).
-R	Operate recursively (default).
-r rev	Commit to <i>rev</i> .

diff

Syntax: **diff** [*options*] [*files*]

Used for: Showing differences between revisions. In addition to the options shown here, accepts a wide variety of options to control output style—for example, **-c** for context diffs.

Option/Argument	Purpose
-D date1	Diff revision for <i>date1</i> against working file.
-D date2	Diff <i>rev1/date1</i> against <i>date2</i> .
-l	Local; run only in current working directory.
-R	Operate recursively (default).
-r rev1	Diff revision for <i>rev1</i> against working file.
-r rev2	Diff <i>rev1/date1</i> against <i>rev2</i> .

edit

Syntax: **edit** [*options*] [*files*]

Used for: Getting ready to edit a watched file.

Option/Argument	Purpose
-a actions	Specify actions for temporary watch, where <i>actions</i> is edit , unedit , commit , all , or none .
-l	Local; run only in current working directory.
-R	Operate recursively (default).

editors

Syntax: **editors** [*options*] [*files*]

Used for: Seeing who is editing a watched file.

Option/Argument	Purpose
-l	Local; run only in current working directory.
-R	Operate recursively (default).

export

Syntax: **export** [*options*] [*project(s)*]

Used for: Exporting files from CVS.

Option/Argument	Purpose
-D date	Check out revisions as of <i>date</i> .
-d dir	Check out into <i>dir</i> .
-f	Use head revision if tag/date not found.
-k kflag	Use <i>kflag</i> keyword expansion.
-l	Local; run only in current working directory.
-N	Don't shorten module paths if -d specified.
-n	Do not run module program (if any).
-P	Prune empty directories.
-R	Operate recursively (default).
-r tag	Check out revision <i>tag</i> (is sticky).

history

Syntax: **history** [*options*] [*files*]

Used for: Showing repository access history.

Option/Argument	Purpose
-a	Apply to all users (default is self).
-b str	Back to record with <i>str</i> in module/file/repos field.
-c	Report on committed (modified) files.
-D date	Check out revisions as of <i>date</i> .
-e	Report on all record types.
-f file	Report the most recent event concerning a file.
-l	Last modified (committed or modified report).
-m module	Report on <i>module</i> (repeatable).
-n module	Check out revision in <i>module</i> .
-o	Report on checked out modules.
-p repository	Show data for a particular directory in the repository.
-r rev	Check out revision since revision <i>rev</i> .
-T	Produce report on all tags.
-t tag	Check out revision since <i>tag</i> record was placed in history file (by anyone).
-u user	Check out revisions for user <i>user</i> (repeatable).
-w	Working directory must match.
-x types	Report on <i>types</i> .
-z zone	Output for time zone <i>zone</i> .

import

Syntax: **import** [*options*] *repository vendor_tag release_tag(s)*

Used for: Importing new sources into the repository, either creating a new project or a new vendor revision on a vendor branch of an existing project.

Option/Argument	Purpose
-b branch	Import to vendor branch <i>branch</i> .
-d	Use the file's modification time as the time of import.
-l ign	Indicate files to ignore (! to reset).
-k kflag	Set default keyword substitution mode.
-m msg	Use <i>msg</i> for log message.
-W spec	Indicate which wrappers.

init

Syntax: **init**

Used for: Creating a CVS repository if it doesn't exist.

log

Syntax: **log** [*options*] [*files*]

Used for: Printing out history information for files.

Option/Argument	Purpose
-b	Only list revisions on the default branch.
-ddates	Specify dates (<i>d1</i> < <i>d2</i> for range, <i>d</i> for latest before).
-h	Only print header.
-l	Local; run only in current working directory.
-N	Do not list tags.
-R	Only print name of RCS file.
-r revs	Only list revisions <i>revs</i> .
-sstates	Only list revisions with specified states.
-t	Only print header and descriptive text.
-wlogins	Only list revisions checked in by specified logins.

login

Syntax: **login**

Used for: Prompting for password for authenticating server.

logout

Syntax: **logout**

Used for: Removing stored password for authenticating server.

rdiff

Syntax: **rdiff** [*options*] *projects*

Used for: Showing differences between releases.

Option/Argument	Purpose
-c	Use context diff output format (default).
-D date	Select revisions based on <i>date</i> .
-f	Use head revision if tag/date not found.
-l	Local; run only in current working directory.
-R	Operate recursively (default).
-r rev	Select revisions based on <i>rev</i> .
-s	Show differences in "short patch form"—one line per file.
-t	Show differences in the top two diffs—last change made to the file.
-u	Use unidiff output format.

release

Syntax: **release** [*options*] *directory*

Used for: Indicating that a directory is no longer in use.

Option/Argument	Purpose
-d	Delete the given directory.



remove

Syntax: **remove** [*options*] [*files*]

Used for: Removing an entry from the repository.

Option/Argument	Purpose
-f	Delete the file before removing it.
-l	Local; run only in current working directory.
-R	Operate recursively (default).

rtag

Syntax: **rtag** [*options*] *tag project(s)*

Used for: Adding a symbolic tag to a module.

Option/Argument	Purpose
-a	Clear tag from removed files that would not otherwise be tagged.
-b	Create a branch named <i>tag</i> .
-D <i>date</i>	Tag revisions as of <i>date</i> .
-d	Delete the given tag.
-F	Move tag if it already exists.
-f	Force a head revision match if tag/date not found.
-l	Local; run only in current working directory.
-n	No execution of tag program.
-R	Operate recursively (default).
-r <i>tag</i>	Tag existing tag <i>tag</i> .

status

Syntax: **status** [*options*] [*files*]

Used for: Displaying status information in a working directory.

Option/Argument	Purpose
-l	Local; run only in current working directory.
-R	Operate recursively (default).
-v	Include tag information for file.

tag

Syntax: **tag** [*options*] *tag [files]*

Used for: Adding a symbolic tag to checked out version of files.

Option/Argument	Purpose
-b	Create a branch named <i>tag</i> .
-D <i>date</i>	Tag revisions as of <i>date</i> .
-d	Delete the given tag.
-F	Move tag if it already exists.
-f	Force a head revision match if tag/date not found.
-l	Local; run only in current working directory.
-n	No execution of tag program.
-R	Operate recursively (default).
-r <i>tag</i>	Tag existing tag <i>tag</i> .

unedit

Syntax: **unedit** [*options*] [*files*]

Used for: Undoing an edit command.

Option/Argument	Purpose
-l	Local; run only in current working directory.
-R	Operate recursively (default).

update

Syntax: **update** [*options*] [*files*]

Used for: Bringing work tree in sync with repository.

Option/Argument	Purpose
-A	Reset any sticky tags/date/options.
-D <i>date</i>	Check out revisions as of <i>date</i> (is sticky).
-d	Create directories.
-f	Use head revision if tag/date not found.
-l <i>ign</i>	Ignore these files while syncing (! to reset).
-j <i>rev</i>	Merge in changes.
-k <i>kflag</i>	Use <i>kflag</i> keyword expansion.
-l	Local; run only in current working directory.
-P	Prune empty directories.
-p	Check out files to standard output (avoids stickiness).
-R	Operate recursively (default).
-r <i>tag</i>	Check out revision <i>tag</i> (is sticky).
-W <i>spec</i>	Use these wrappers.

watch

Syntax: **watch on|off|add|remove** [*options*] [*files*]

Used for: Turning on/off, or adding/removing a watch on read-only files.

Option/Argument	Purpose
-a <i>actions</i>	Specify actions for temporary watch, where <i>actions</i> is edit , unedit , commit , all , or none .
-l	Local; run only in current working directory.
-R	Operate recursively (default).

watchers

Syntax: **watchers** [*options*] [*files*]

Used for: Seeing who is watching a file.

Option/Argument	Purpose
-l	Local; run only in current working directory.
-R	Operate recursively (default).