CVS bez tajemnic

Autor: Jennifer Vesperman
Tłumaczenie: Marek Pałczyński (rozdz. 5 – 11, dod. A – C), Rafał Sionek (przedmowa, rozdz. 1 – 4)
ISBN: 83-7361-710-8
Tytuł oryginału: Essential CVS
Format: B5, stron: 360

Książka „CVS bez tajemnic” to podręcznik opisujący system CVS, zarówno z punktu widzenia użytkownika, jak i administratora. Przedstawia metody zarządzania kodem źródłowym projektów oraz tworzenia repozytoriów i administrowania nimi. Zawiera zestawienie poleceń CVS oraz opisy aplikacji klienckich, umożliwiających połączenie z systemem CVS z poziomu różnych środowisk programistycznych i systemów operacyjnych.

- Instalacja systemu CVS
- Projekty i repozytoria
- Tworzenie znaczników i odgałęzień
- Wykorzystanie systemu CVS w pracy zespołu programistycznego
- Zarządzanie repozytoriami i projektami
- Zdalne repozytoria
- Polecenia systemu CVS
- Aplikacje klienckie i integracja ze środowiskami programistycznymi

„CVS bez tajemnic” to doskonałe źródło wiedzy dla programistów i administratorów.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Przedmowa **9**

Część I Wprowadzenie 13

1. Czym jest CVS **15**
Czym jest system kontroli wersji 15
Obszary zastosowań CVS 17

2. CVS — przewodnik dla początkujących **21**
Instalowanie CVS 21
Tworzenie pierwszego repozytorium 27
Importowanie projektów 29
Uzyskiwanie dostępu do repozytoriów zdalnych 32
Pobieranie plików 33
Wysyłanie poprawek 34
Aktualizowanie magazynów lokalnych 35
Dodawanie plików 37
Usuwanie plików 38
Porady na zakończenie 39

Część II Użytkowanie systemu CVS 41

3. Podstawowe zastosowania CVS **43**
Informacje ogólne 43
Magazyny lokalne i repozytoria 45
Wprowadzanie zmian do repozytorium 50
Sprawdzanie statusu plików 53
Uaktualnianie plików znajdujących się w magazynie lokalnym 55
Dodawanie plików do repozytorium 62
Usuwanie plików z repozytorium 64
Przenoszenie plików i katalogów 68
Zwalnianie magazynu lokalnego 69
Słowa kluczowe 70
Pliki binarne i warstwy otaczające 74
Określanie domyślnych opcji polecenia 76

4. Tworzenie znaczników i odgałęzień 77
Oznaczanie 77
Lepkość 86
Rozgałęzianie 88
Strategie rozgłęziania 98

5. Praca wielu użytkowników 105
Jednoczesna praca nad projektem 106
Monitorowanie pliku 106
Rezerwowanie plików 112
Porównywanie różnych wersji pliku 115
Wyświetlanie najnowszych zmian 118
Wyświetlanie historii pliku 119

Część III Administracja 123

6. Zarządzanie repozytorium 125
Tworzenie repozytorium 125
Usuwanie repozytorium 126
Zabezpieczanie projektów 127
Struktura repozytorium 130
Pliki katalogu CVSROOT 133
Zmienne środowiskowe serwera 144
Sporządzanie kopii zapasowej repozytorium 145
Edycja repozytorium 151
Struktura magazynu lokalnego 160
Zmienne środowiskowe systemu klienckiego 163
Kody statusowe 165

7. Zarządzanie projektem 167
Tworzenie projektu 167
Dystrybucja plików 174
Uruchamianie skryptów 177
Współpraca CVS z innymi programami 187
Narzędzia CVS 190
Strategie i uwagi praktyczne 193

8. Zdalne repozytoria 197
Wyznaczanie ścieżek do repozytorium 198
Metoda dostępu local 199
Metody dostępu ext i server 200
Metoda dostępu fork 203

Metoda dostępu gserver 204
Metoda dostępu kserver 207
Metoda dostępu pserver 208
Wykorzystanie usługi inetd w metodach gserver, kserver i pserver 213

9. Rozwiązywanie problemów 215
Ogólne techniki rozwiązywania problemów 215
Problemy z połączeniami 217
Problemy związane z nazwami plików 221
Problemy ze znakami końca wiersza 222
Problemy z prawami dostępu 222
Pliki blokad 223

Część IV Pomoc techniczna 225

10. Polecenia CVS 227
Opcje instrukcji wiersza poleceń 227
Polecenia CVS 230

11. Pozostałe elementy CVS 273
Pliki administracyjne 274
Pliki katalogu CVSROOT 276
Zmienne plików katalogu CVSROOT 287
Daty 289
Zmienne środowiskowe 293
Słowa kluczowe i tryby zastępowania słów kluczowych 295
Wzorce dopasowania 298
Metody dostępu do repozytorium 300

Dodatki 303

A Oprogramowanie klienckie różnych systemów operacyjnych 305

B Narzędzia administratora 323

C Najważniejsze polecenia CVS w skrócie 333

Skorowidz 339

ROZDZIAŁ 4.

Tworzenie znaczników i odgałęzień

Z pewnością jedną z najbardziej pomocnych, jak również niedocenianych funkcji CVS są znaczniki (ang. *tag*). Funkcja nadawania znaczników pozwala nadać rewizji etykietę umożliwiającą jej późniejsze pobranie. Ponadto funkcja ta pozwala na rozwidlenie prac nad projektem w taki sposób, iż możliwa staje się równoczesna praca nad dwoma lub więcej wersjami projektu. Odgałęziona wersja projektu nosi nazwę *gałęzi* (ang. *branch*), a jego część główna — *pnia* (ang. *trunk*).

W rozdziale tym obszernie wyjaśniono takie pojęcia jak oznaczanie i odgałęzianie projektów. Podano w nim także przyczyny, dla których wykonuje się operacje oznaczania i tworzenia odgałęzień projektu, oraz opisano strategie i wskazówki pomocne w efektywnym zarządzaniu odgałęzieniami w projekcie.

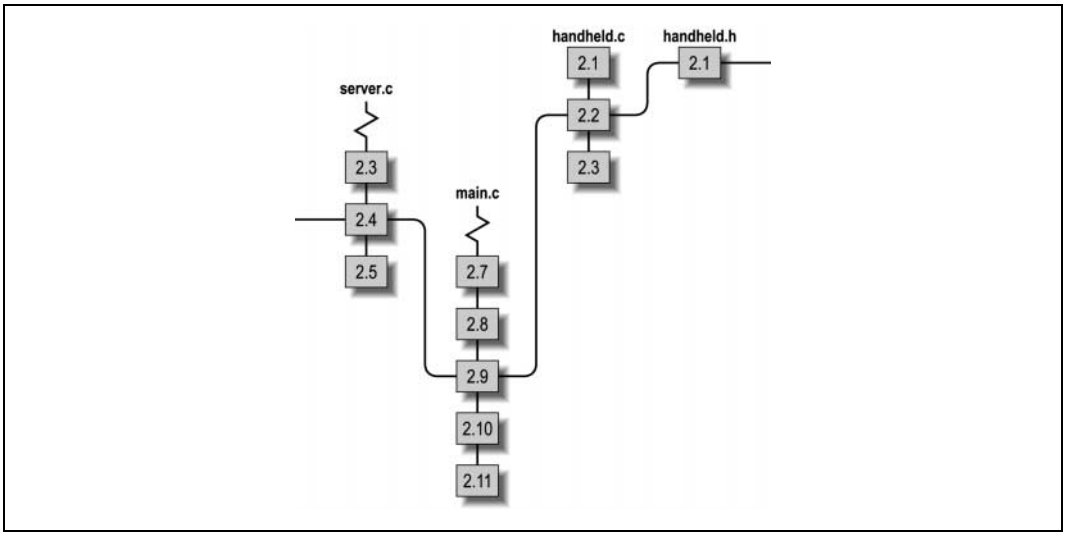
Oznaczanie

CVS umożliwia pobranie dowolnej, obecnej w repozytorium rewizji pliku. Jeżeli pobieranie różnych rewizji ma sens, dlaczego nie miałoby mieć sensu pobieranie zestawu zgodnych ze sobą rewizji innych, tworzących kompletną, w pełni funkcjonalną wersję projektu? Np. wszystkich rewizji plików, które składały się na wersję 1.0 projektu, lub stały się pierwszym wydaniem książki.

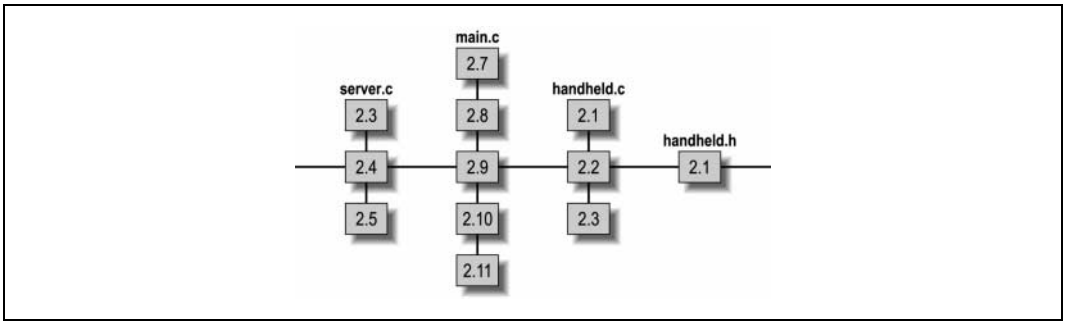
Znakowanie jest sposobem nadawania znaczników zestawowi rewizji plików tworzącemu jedną zwartą grupę. W przypadku oznaczania nie można do tego celu używać numerów rewizji, gdyż np. rewizja 2.3 jednego pliku, mogła tworzyć wspólną grupę z rewizją 2.17 innego pliku. Rysunek 4.1 przedstawia grupę rewizji plików wraz z linią określającą, które numery rewizji należą do tej samej grupy.

System CVS umożliwia utworzenie pojedynczego znacznika, wskazującego na rewizje połączone za pomocą wymienionej wcześniej linii. Aby mieć wgląd na wszystkie należące do tego znacznika rewizje, system CVS może „pociągnąć" za linię w celu zlokalizowania oznaczonych rewizji. Rysunek 4.2 przedstawia tę samą grupę plików z prostą linią wskazującą pliki należące do tego znacznika.

Znacznika można użyć do wskazania określonej wersji pojedynczego pliku lub określonego zestawu rewizji grupy plików i określenia nazwy dla tego zestawu. Od tej pory znacznik pozwala na łatwe odzyskanie tej jednej rewizji lub całego zestawu. Ponadto zapamiętanie łańcucha tekstowego jest o wiele łatwiejszym zadaniem niż zapamiętanie numeru wersji. Używanie znaczników przez programistów do oznaczania wersji według ich własnych kryteriów jest znacznie częstsze niż korzystanie z numerowania rewizji stosowanego przez system CVS.



Rysunek 4.1. Rewizje plików należących do tej samej grupy



Rysunek 4.2. Oznaczanie rewizji plików

.Nazwy znaczników muszą rozpoczynać się od znaku będącego literą i mogą składać się ze znaków alfanumerycznych, łączników i znaków podkreślenia. Nazwa każdego znacznika musi być również niepowtarzalna. Pliki znaczników oraz polecenia `cvs tag` i `cvs rtag` zostały opisane w punktach „Oznaczanie przy pomocy magazynu lokalnego" i „Oznaczanie za pomocą daty lub rewizji" niniejszego rozdziału.

Najlepiej używać zrozumiałych i prostych nazw znaczników. Powinny one wywoływać określone, związane z daną rewizją skojarzenia, podpowiadające np. czemu dane pliku należą do jednej grupy. Przykładowymi, dobrymi nazwami znaczników są: *release-1-3-beta*, *release-2-13-patch-5*, *testing-1-5-alpha* i *release-2-5-stable*.

W trakcie trwania projektu należy ustalić jednolitą konwencję nazewniczą znaczników i zachęcić osoby pracujące nad projektem do jej stosowania. Nazwy nadawane znacznikom powinny odzwierciedlać z kolei konwencję nazewniczą używaną do określania kolejnych wersji projektu, nieobcą wszakże uczestnikom projektu. Na odpowiednio nazwanych znacznikach uczestnicy projektu mogą polegać w dużo większym stopniu, przy wyborze odpowiedniej rewizji projektu, niż na wewnętrznej numeracji CVS. Wewnętrzna numeracja przeznaczona jest dla celów narzędzi wewnętrznych składających się na system CVS i może być tylko i wyłącznie utrudnieniem dla uczestników projektu, gdyby mieli oni polegać na niej przy określaniu etapów projektu.

W punkcie „Plik taginfo" w rozdziale 7. zaprezentowano skrypt, który może być użyty do automatycznego wymuszania standardów nazewniczych.

W systemie CVS istnieją jednakże dwie zastrzeżone nazwy znaczników. CVS używa nazw *BASE* i *HEAD* do nazywania odpowiednio: nazwy ostatnio zsynchronizowanej z repozytorium rewizji i nazwy znacznika ostatniej rewizji znajdującej się w repozytorium.

Rozważmy hipotetyczną sytuację, w której wszyscy uczestnicy projektu synchronizują rewizję do wersji 1.23, z tym, że dowolny uczestnik projektu dokonuje dwóch synchronizacji swoich zmian w repozytorium. Wobec tego numery rewizji pozostałych członków zespołu, którzy nie synchronizowali swoich magazynów lokalnych z repozytorium, wynoszą nadal 1.23, podczas gdy obecnym numerem wersji *HEAD* jest już 1.25, ponieważ taka była nazwa rewizji *BASE* użytkownika, który synchronizował wcześniej swój magazyn lokalny.

Oznaczanie przy pomocy magazynu lokalnego

W celu oznakowania wszystkich plików w magazynie lokalnym wraz z należącymi do niego podkatalogami, należy wykorzystać polecenie `cvs tag`. Domyślnie, oznaczana jest rewizja *BASE*. Określone pliki mogą zostać oznaczone po podaniu ich nazw jako argumentu polecenia `cvs tag`.

Składnia polecenia `cvs tag` prezentuje się nastepująco:

```
cvs [opcja_cvs] tag [opcja_polecenia] nazwa_znacznika [nazwy_plików]
```

Polecenie `cvs tag` określa pliki i rewizje do oznaczenia ich na podstawie zawartości magazynu lokalnego, lecz faktycznie oznacza je na podstawie numeru ostatnio synchronizowanej z repozytorium rewizji. Zatem, jeżeli w magazynie lokalnym pojawiłyby się nowe zmiany od czasu ostatniej synchronizacji przechowywanych w nim plików z repozytorium, zmiany te nie zostałyby odzwierciedlone w oznaczonych rewizjach.

Opcja `-c` polecenia `cvs tag` pozwala sprawdzić, czy magazyn lokalny nie został zmodyfikowany, a dokonane w nim zmiany jak dotychczas niezatwierdzone, zanim dokonano oznakowania plików. W przypadku gdy polecenie `cvs tag -c` napotka na niewprowadzone do repozytorium zmiany, zatrzyma swoje działanie, nie dokonując oznakowania żadnych plików. Aby jednak oznakować rewizję znajdującą się w repozytorium, bez wprowadzenia wszystkich zmian z magazynów lokalnych, należy podczas wydawania polecenia `cvs tag` ominąć opcję `-c`. Chcąc oznakować rewizję znajdującą się w magazynie lokalnym, należy wprowadzić zmiany do repozytorium i uruchomić ponownie polecenie `cvs tag`.

W celu zidentyfikowania rewizji do oznakowania można posłużyć się datami bądź innymi, istniejącymi już znacznikami lub też numerami rewizji. Aby zidentyfikować wersję za pomocą nazwy rewizji lub istniejącego znacznika, należy użyć polecenia `cvs tag -r` *`rewizja`* lub `cvs tag -r` *`nazwa_znacznika`*. Aby zidentyfikować wersję za pomocą daty, trzeba posłużyć się poleceniem `cvs tag -D` *`data`*. Dokonując identyfikacji za pomocą daty, CVS oznaczy pierwszą rewizję, której data jest mniejsza od daty określonej za pomocą parametru `-D`. Więcej informacji na temat dat można znaleźć w rozdziale 11.

Wraz z opcjami `-D` oraz `-r` można również stosować opcję `-f`. Opcja ta nakazuje systemowi CVS, aby używał rewizji *HEAD*, w przypadku gdy żadna inna rewizja nie spełnia wymagań określonych parametrami `-r` lub `-D`.

Domyślnie, polecenie `cvs tag` działa rekursywnie w stosunku do wszystkich podkatalogów magazynu lokalnego. Za taki sposób wykonywania polecenia `cvs tag` odpowiada opcja `-R`, którą można również używać, choć jak już wspomniano wcześniej, nie ma takiej potrzeby.

Przykład 4.1 prezentuje oznakowanie plików znajdujących się w magazynie lokalnym znacznikiem o nazwie *pre_alpha_0-1* za pomocą polecenia `cvs tag`.

Przykład 4.1. Użycie polecenia cvs tag

```
bash-2.05a$ cvs tag pre_alpha_0-1
cvs server: Tagging .
T Changelog
T INSTALL
T Makefile
T README
T TODO
cvs server: Tagging doc
cvs server: Tagging doc/design
T doc/design/AcceptanceTest.doc
T doc/design/Analysis.rtf
T doc/design/Requirements.doc
T doc/design/Specification.rtf
cvs server: Tagging doc/plan
T doc/plan/Schedule.rtf
cvs server: Tagging lib
cvs server: Tagging man
cvs server: Tagging src
T src/config.h
T src/main.c
```

Oznaczanie za pomocą daty lub rewizji

Polecenie `cvs rtag` pozwala na oznakowanie plików bez odwoływania się do konkretnego magazynu lokalnego. Polecenie `rtag` w celu zidentyfikowania rewizji plików do oznakowania polega na parametrach podawanych wraz z poleceniem, toteż podanie opcji `-r` lub `-D` wraz z nazwą co najmniej jednego pliku, katalogu lub modułu staje się obligatoryjne. Zagadnienie modułów zostało szerzej ujęte w rozdziale 7. Podając jako argument opcji nazwy kilku katalogów, plików lub modułów, należy rozdzielić je spacjami.

Składnia polecenia `cvs rtag` jest następująca:

```
cvs [opcja_cvs] rtag opcja_polecenia nazwa_znacznika nazwy_plików
```

Przykład 4.2 prezentuje nadanie znacznika *pre_alpha_0-2* wszystkim plikom znajdującym się w podkatalogach, jak również w samym katalogu *wizzard* za pomocą polecenia `cvs rtag`. Opcja `-r HEAD` oznacza, że znacznik *pre_alpha_0-2* zostanie przyporządkowany rewizjom `HEAD` wszystkich plików.

Przykład 4.2 Użycie polecenia cvs rtag

```
bash-2.05a$ cvs -d cvs:/var/lib/cvs rtag -r HEAD pre_alpha_0-2 wizzard
cvs rtag: Tagging wizzard
cvs rtag: Tagging wizzard/doc
cvs rtag: Tagging wizzard/doc/design
cvs rtag: Tagging wizzard/doc/plan
cvs rtag: Tagging wizzard/lib
cvs rtag: Tagging wizzard/man
cvs rtag: Tagging wizzard/src
```

Jeżeli w chwili wydania polecenia `cvs rtag` znajdujemy się w magazynie lokalnym, system CVS jako repozytorium, w którym należy oznakować pliki, domyślnie przyjmie repozytorium powiązane z tym konkretnym magazynem lokalnym. Znajdując się więc w magazynie lokalnym powiązanym z innym repozytorium niż to, w którym ma dojść do oznakowania plików, należy opuścić je lub użyć opcji `-d` *`ścieżka_do_repozytorium`* , tak jak zostało to pokazane w przykładzie 4.2.

Jeśli bieżącym katalogiem roboczym nie jest katalog magazynu lokalnego, można określić je za pomocą zmiennej środowiskowej *CVSROOT* (w przypadku komputera będącego jedynie klientem CVS) lub opcji `-d` *`ścieżka_do_repozytorium`* (w przypadku komputera będącego równocześnie serwerem, jak i klientem CVS).

Chcąc oznakować każdą najnowszą rewizję wszystkich plików znajdujących się w repozytorium, należy użyć opcji `-r HEAD`. Warto przy tym mieć świadomość tego, że operacje wykonywane przez CVS nie są niepodzielne i w przypadku, gdy podczas oznaczania plików za pomocą opcji `-r HEAD`, któryś z użytkowników będzie równocześnie wprowadzał swoje zmiany do repozytorium, może zdarzyć się sytuacja, w której jeden katalog zostanie oznakowany przed wprowadzeniem zmian przez użytkownika, podczas gdy inny zostanie oznakowany już po dokonaniu operacji wprowadzenia zmian do repozytorium.

Używając opcji `-D`, trzeba pamiętać, że jeśli nie została określona konkretna godzina, system CVS oznakuje ostatnią rewizję, datującą się przed północą określonego za pomocą opcji `-D` dnia. Przykładowo, opcja `-D 12 Feb 2002` spowoduje oznakowanie plików tak, jak wyglądały one o północy 12 lutego 2002 roku czasu lokalnego. Formaty daty zostały omówione w rozdziale 11.

Większość opcji polecenia `cvs tag` wykazuje podobne właściwości jak w przypadku polecenia `cvs rtag`. Opcje `-l` i `-R` odpowiadają za rekursywność działania opcji, a opcje `-r`, `-D` i `-f` służą określaniu rewizji, podobnie jak w poleceniu `cvs tag`. Opcja `-c` polecenia `cvs tag` nie ma zastosowania w przypadku polecenia `cvs rtag`.

Pobieranie oznakowanych plików

W celu wyświetlenia znaczników przyporządkowanych danemu plikowi, należy wykonać polecenie `cvs status -v` w katalogu, w którym znajduje się rzeczony plik. Polecenie to wyświetli również informacje na temat aktualnej rewizji znajdującej się w magazynie lokalnym, aktualnej rewizji w repozytorium oraz informację na temat „lepkich" znaczników aktualnego magazynu lokalnego. Nazwy znaczników zostaną umieszczone w dolnej części wyświetlonego raportu. Warto zwrócić uwagę na znaczniki, których nazwy są opatrzone nie tylko numerami rewizji, lecz także słowem „branch" — są to znaczniki będące równocześnie znacznikami bazowymi danego odgałęzienia. Więcej informacji na ich temat można znaleźć w podrozdziale „Rozgałęzianie" w dalszej części tego rozdziału. Przykład 4.3 demonstruje sposób użycia polecenia `cvs status` w celu wyświetlenia znaczników pliku *main.c*.

Przykład 4.3 Znaczniki pliku

```
bash-2.05a$ cvs status -v src/main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c                Status: Up-to-date

   Working revision:    1.9
   Repository revision: 1.9     /var/lib/cvs/wizzard/src/main.c,v
   Sticky Tag:          (none)
```

```
Sticky Date:        (none)
Sticky Options:     (none)

Existing Tags:
  pre_alpha_0-2               (revision: 1.9)
  pre_alpha_0-1               (revision: 1.9)
```

Pobranie oznakowanego pliku lub całego ich zestawu jest możliwe dzięki zastosowaniu parametru `-r` *`nazwa_znacznika`* jako opcji polecenia `cvs checkout` lub `cvs update`. Polecenia `checkout` należy użyć w celu utworzenia nowego magazynu lokalnego, a polecenia `update` przy założeniu, że pliki mają być pobrane do aktualnego magazynu lokalnego. W drugim przypadku wszystkie znajdujące się w nim pliki zostaną nadpisane plikami oznakowanej rewizji, jednakże zmiany, które zostały dokonane od czasu ostatniej synchronizacji z repozytorium, zostaną scalone z nowymi plikami. Przykład 4.4 prezentuje efekt działania polecenia `cvs checkout` w przypadku oznakowanego magazynu lokalnego.

Przykład 4.4. Pobieranie plików do oznakowanego magazynu lokalnego za pomocą polecenia cvs checkout

```
bash-2.05a$ cvs -d cvs:/var/lib/cvs checkout -r pre_alpha_0-2 wizzard
cvs server: Updating wizzard
U wizzard/Changelog
U wizzard/INSTALL
U wizzard/Makefile
U wizzard/README
U wizzard/TODO
cvs server: Updating wizzard/doc
cvs server: Updating wizzard/doc/design
U wizzard/doc/design/AcceptanceTest.doc
U wizzard/doc/design/Analysis.rtf
U wizzard/doc/design/Requirements.doc
U wizzard/doc/design/Specification.rtf
cvs server: Updating wizzard/doc/plan
U wizzard/doc/plan/Schedule.rtf
cvs server: Updating wizzard/lib
cvs server: Updating wizzard/man
cvs server: Updating wizzard/src
U wizzard/src/config.h
U wizzard/src/main.c
```

Wykonując operacje pobrania plików lub uaktualnienia magazynu lokalnego za pomocą daty lub znacznika nie będącego znacznikiem odgałęzienia (temat odgałęzień opisany został w dalszej części tego rozdziału), znacznik lub data określająca pliki w magazynie lokalnym będą „lepkie". Pliki znajdujące się w magazynie lokalnym, które zostały pobrane poprzez określenie ich daty lub nazwy znacznika nie będącego znacznikiem odgałęzienia są tylko statyczną reprezentacją stanu projektu w danym czasie. Wprowadzenie zmian do takiego pliku nie jest możliwe. Termin *lepkość* odnosi się tylko do plików znajdujących się w magazynie lokalnym, nie dotyczy zaś repozytorium. Dokładniejsze informacje na ten temat można znaleźć w podrozdziale „Lepkość".

Usuwanie oraz przenoszenie znaczników

Nadawanie znaczników ma zazwyczaj charakter stały, jednak czasami pojawia się potrzeba usunięcia znacznika, zmiany jego nazwy lub przeniesienia go. Operacje takie należy przeprowadzać ze szczególną ostrożnością, gdyż mogą spowodować usunięcie wcześniejszych informacji i równocześnie uniemożliwienie ich odzyskania.

Przy próbie usunięcia specjalnego znacznika — nazywanego znacznikiem odgałęzienia (patrz podrozdział „Rozgałęzianie") — system CVS wyświetli informację o błędzie i nie usunie znacznika, ani nie przeniesie go. Aby wymusić usunięcie lub przeniesienie takiego znacznika, należy posłużyć się opcją `-B`.

Należy unikać przenoszenia, jak również zastępowania nazwy znacznika odgałęzienia bez potrzeby, a decydując się na to, należy zadbać, aby kopia zapasowa repozytorium odzwierciedlała wszystkie ostatnie zmiany, gdyż istnieje duże zagrożenie utraty danych.

Usuwanie znacznika

Zwykle nie ma potrzeby usuwania poprawnie umieszczonego w pliku znacznika. Sytuacja taka może jednak zaistnieć w przypadku błędnego nadania znacznika i wtedy, być może, trzeba będzie usunąć dany znacznik i spróbować nadać go ponownie.

Aby usunąć znacznik, należy posłużyć się opcją `-d`:

```
cvs tag -d nazwa_znacznika [nazwa_pliku]
```

lub

```
cvs rtag -d nazwa_znacznika nazwa_pliku
```

Podczas wykonywania polecenia `rtag` poza magazynem lokalnym podanie ścieżki do repozytorium staje się obligatoryjne. Po wydaniu polecenia `rtag` wewnątrz magazynu lokalnego CVS przeszuka podkatalog *CVS* w celu znalezienia odpowiedniego repozytorium.

Polecenie `tag` musi być wykonywane wewnątrz magazynu lokalnego. Domyślnie odnosi się ono do aktualnego magazynu lokalnego i znajdujących się w nim podkatalogów. System CVS przeszuka podkatalog *CVS*, by znaleźć odpowiednie repozytorium.

Przykład 4.5 pokazuje sposób użycia polecenia `cvs tag` z zamiarem usunięcia znacznika. Polecenie zostało wydane z najwyższego poziomu hierarchii katalogów magazynu lokalnego danego projektu.

Przykład 4.5. Usuwanie znacznika

```
bash-2.05a$ cvs tag -d pre_alpha_0-2
cvs server: Untagging .
cvs server: Untagging doc
cvs server: Untagging doc/design
cvs server: Untagging doc/plan
cvs server: Untagging src
```

Przenoszenie znacznika

Najczęstszą przyczyną przenoszenia znacznika jest poprawienie błędu w oznakowaniu. W niektórych zespołach projektowych używa się również przenośnego znacznika w celu oznakowania najnowszej wersji projektu, gotowej do wydania, lub najnowszej wersji poprawek. Znacznik taki przenoszony jest za każdym razem, kiedy nowa wersja projektu zostaje ukończona.

Aby przenieść znacznik z jednej rewizji na inną rewizję tego samego pliku lub ich zestawu, należy posłużyć się poleceniem `cvs tag` lub `cvs rtag` wraz z opcją `-F`. Za pomocą opcji `-r` określa się rewizję, do której ma zostać przeniesiony znacznik, podczas gdy opcja `-F` służy do określenia nazwy znacznika do przeniesienia. Przykład 4.6 ilustruje operację przeniesienia znacznika

wykonaną z poziomu magazynu lokalnego przy użyciu polecenia `cvs rtag`. Raport dotyczący statusu pliku wyświetlił się przed i po operacji przeniesienia. Używając do przeniesienia znacznika polecenia `rtag`, zamiast `tag`, należy określić pełną ścieżkę z katalogu głównego repozytorium do pliku *main.c*, włączając w to również nazwę projektu. Przy wykonywaniu polecenia `cvs status` określenie pełnej ścieżki nie jest konieczne ze względu na to, że polecenie to zostaje wydane z poziomu magazynu lokalnego.

Przykład 4.6. Przenoszenie znacznika

```
bash-2.05a$ cvs status -v src/main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c               Status: Up-to-date

   Working revision:    1.9
   Repository revision: 1.9     /var/lib/cvs/wizzard/src/main.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)

   Existing Tags:
     pre_alpha_0-1                (revision: 1.9)
bash-2.05a$ cvs rtag -r 1.8 -F pre_alpha_0-1 wizzard/src/main.c
bash-2.05a$ cvs status -v src/main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c               Status: Up-to-date

   Working revision:     1.9
   Repository revision:  1.9     /var/lib/cvs/wizzard/src/main.c,v
   Sticky Tag:           (none)
   Sticky Date:          (none)
   Sticky Options:       (none)

   Existing Tags:
     pre_alpha_0-1       (revision: 1.8)
```

Usuwanie i przenoszenie znaczników w plikach znajdujących się w katalogu Attic

Pliki usunięte z głównego pnia projektu poleceniem `cvs remove` lub też pliki, które nigdy się w nim nie znalazły, przechowywane są w podkatalogu *Attic*, znajdującym się w katalogu repozytorium. Pliki takie mogą być skojarzone ze znacznikami, które wymagają usunięcia, przeniesienia lub zmiany nazwy. Operacji tej nie da się wykonać na plikach znajdujących się w katalogu *Attic* za pomocą polecenia `cvs tag`. Jednak polecenie `cvs rtag` ma opcję `-a`, która potrafi wymusić wykonanie polecenia `cvs rtag -d -F` w stosunku do usuniętych wcześniej plików (np. takich, które są umieszczone w katalogu *Attic*) znajdujących się w odpowiednim module lub katalogu .

Po wydaniu polecenia `cvs tag` lub `cvs rtag` wraz z opcją `-r` *`nazwa_rewizji`*, system CVS sprawdza katalog *Attic* pod kątem obecności określonej rewizji w plikach tam się znajdujących, co tym samym oznacza, że w takim wypadku opcja `-a` nie jest wymagana.

Zmienianie nazwy znacznika

Jeżeli któryś z członków zespołu projektowego dodał do systemu znacznik nie odpowiadający standardom nazewniczym przyjętym w danym projekcie, można przeprowadzić operację zmiany nazwy takiego znacznika.

Co prawda, CVS nie dysponuje specjalnym poleceniem do zmiany nazwy znacznika, jednakże za pomocą polecenia `cvs tag` lub `cvs rtag` wraz z opcją `-r` nadanie nowego znacznika błędnie oznaczonej rewizji jest bardzo proste. Po pomyślnie zakończonej operacji można usunąć stary znacznik. Takie podejście nie jest jednak odpowiednie w przypadku rozgałęzień.

Przykład 4.7. prezentuje operację zmiany nazwy znacznika. Celem operacji jest zmiana nazwy znacznika *pre_alpha_0-1* na *pre_beta_0-1*. W pierwszej kolejności plikom oznaczonym jako *pre_alpha_0-1* nadaje się znacznik *pre_beta_0-1*, a następnie usuwa się stary znacznik poleceniem `cvs tag -d`. Efektem takiej operacji jest zmiana nazwy znacznika *pre_alpha_0-1* na *pre_beta_0-1*.

Przykład 4.7. Zmiana nazwy znacznika

```
bash-2.05a$ cvs tag -r pre_alpha_0-1 pre_beta_0-1
cvs server: Tagging .
T Changelog
T INSTALL
T Makefile
.
.
.
cvs server: Tagging src
T src/config.h
T src/main.c
bash-2.05a$ cvs tag -d pre_alpha_0-1
cvs server: Untagging .
D Changelog
D INSTALL
D Makefile
.
.
.
cvs server: Untagging src
D src/config.h
D src/main.c
```

Usunięte pliki

Trzeba wiedzieć, że jeśli usunie się plik z oznakowywanego właśnie projektu, to ów plik nie zostanie oznakowany. Nie ma to większego znaczenia w przypadku, gdy plik ten nie będzie ponownie dodany do projektu.

Jeżeli jednak plik został usunięty z projektu, a następnie do niego przywrócony, to trzeba mieć świadomość, że nie istnieje prosty sposób na sprawdzenie, czy plik ten został oznakowany, czy nie. Dzieje się tak, gdyż znacznik mógł zostać nadany pomiędzy usunięciem pliku, a jego przywróceniem, lub też z tego powodu, że znacznik może być starszy niż plik. By zdobyć więcej informacji na ten temat, można posiłkować się datami plików lub wykonać polecenie `cvs rdiff -s -r` *`nazwa_znacznika nazwa_projektu`*. Opcja `-s` polecenia `rdiff` generuje raport podsumowujący, w którym wyświetlą się wszystkie pliki bądź to zmienione, bądź dodane lub też usunięte.

Aby oznakować zarówno usunięte, jak i nieusunięte pliki wraz z poleceniam `cvs tag` lub `cvs rtag`, należy używać opcji `-r`, na przykład `-r HEAD`.

Przy oznakowywaniu rewizji *HEAD*, być może będzie trzeba zapewnić ochronę plikom znajdującym się w repozytorium przed zmianami innych użytkowników, pomiędzy momentem, w którym zostaje podjęta decyzja, że pliki są gotowe do oznakowania, a ich faktycznym oznakowaniem. Rady dotyczące tego, jakie podjąć w takim wypadku działania, zawarte zostały w punkcie „Blokowanie repozytorium” znajdującym się w rozdziale 6.

Strategie oznakowywania

Nadawanie znaczników ułatwia późniejsze pobieranie plików należących dodanego etapu projektu. Znaczniki należy nadawać zawsze wtedy, gdy projekt osiąga kolejne, znaczące etapy realizacji. Jako absolutne minimum należy przyjąć oznaczanie każdego rozgałęzienia projektu i każdego kolejnego, ukończonego wydania projektu.

Dobrze jest obmyślić własną lokalną strategię oznakowywania. Poniższa lista ważnych punktów projektu, które należałoby wziąć pod uwagę podczas opracowywania strategii, jest silnie zorientowana na tworzenie programów:

- po zakończeniu prac nad każdą ważniejszą funkcją,
- w momencie osiągnięcia każdej znaczącej fazy projektu,
- tuż przed odrzuceniem istniejącej już funkcji,
- tuż przed rozpoczęciem testowania,
- przed wprowadzeniem zmian mogących zepsuć uruchamiający się już kod,
- tuż przed rozszczepieniem gałęzi,
- tuż po scaleniu odgałęzień.

Należy używać zrozumiałych znaczników o wcześniej ustalonym formacie, zawierających w swojej nazwie wszystkie potrzebne informacje. Poniższy przykład jest jednym z możliwych do zastosowania, lecz równocześnie niezwykle uszczegółowionym formatem nazywania znaczników.

```
wersja-[alpha-|beta-][test-|final-|patch-][patch#-][pub|priv]
```

W przypadku, gdy zaistnieje konieczność pobrania starszej wersji kodu w celu przetestowania lub opracowania poprawki, należałoby mieć pod ręką łatwy sposób identyfikacji tej wersji, o którą nam chodzi. Powyższy przykład formatu nazwy znacznika zapewnia informację na temat numeru rewizji oraz mówi o tym, czy oznakowana rewizja jest wersją testową, czy też finalną, określa stadium testów, a także powiadamia o tym, czy dane wydanie jest wersją ogólnodostępną, czy wewnętrzną.

Należy mieć na uwadze fakt, że zaprezentowany powyżej przykład jest tylko jednym z możliwych do zastosowania formatów. Bazując na potrzebach własnego zespołu projektowego najlepiej jest jednak opracować swój własny format nazywania znaczników. Większość zespołów będzie preferowała format krótszy od tego, jaki został tutaj zaprezentowany.

Lepkość

Pojęcie *lepkości* (ang. *stickiness*) jest jedną z ważniejszych cech systemu CVS, szczególnie gdy mowa o nadawaniu znaczników i rozgałęzianiu. Lepkość jest zasadniczo cechą wewnętrzną systemu CVS, ogranicza ona jednak zakres zmian, jakie można wprowadzić do magazynu lokalnego.

Kiedy któryś z plików znajdujących się w repozytorium osiąga stan, w którym nie powinien być on już poddawany żadnym modyfikacjom, co nie jest domyślnym typem zachowania dla plików znajdujących się w magazynie lokalnym, stan taki nazywa się stanem *lepkim* (ang. *sticky*).

Lepkość pliku może być spowodowana przynależnością pliku do pewnej określonej rewizji lub odgałęzienia projektu, lub też może on posiadać pewne opcje słów kluczowych.

Dany plik można włączyć do określonej rewizji poprzez pobranie go z repozytorium za pomocą określającej go nazwy znacznika, numeru rewizji lub daty. W przypadku, gdy plik zostanie pobrany za pomocą daty, mówi się, że ma on *lepką datę*. Jeżeli zostanie pobrany za pomocą znacznika, mówi się o *lepkim znaczniku*, a przypadku pobrania za pomocą numeru rewizji, analogicznie plik taki określa się jako plik posiadający *lepką rewizję*.

Podobnie jak w powyższym przykładzie, o plikach znajdujących się w magazynie lokalnym, należących równocześnie do odgałęzienia, mówi się, że posiadają one lepkie odgałęzienie, a w przypadku pliku o określonych dla danego magazynu lokalnego słowach kluczowych, mówi się jako o pliku mającym lepkie słowa kluczowe.

Znacznik lepkości determinuje zachowanie wszystkich uruchamianych poleceń mających wpływa na tenże plik. Plik, z lepką data, rewizją lub ze znacznikiem nie należącym do jakiegokolwiek odgałęzienia nie zostanie zaktualizowany do nowszej wersji; plik taki nie może zostać również zsynchronizowany z repozytorium. Z kolei plik określony lepkim odgałęzieniem zawsze zostanie zaktualizowany lub zsynchronizowany z określonym odgałęzieniem.

Stan lepkości plików może być monitorowany za pomocą polecenia `cvs status`. Przykładowo, aby zobaczyć raport dla pliku *index.htm*, należy wydać polecenie `cvs status index.htm`. Przykład 4.8. prezentuje raport na temat statusu pliku mającego lepkie odgałęzienie.

Przykład 4.8. Wyświetlenie statusu lepkiego pliku

```
bash-2.05a$ cvs status main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c               Status: Up-to-date

   Working revision:     1.9
   Repository revision:  1.9     /var/lib/cvs/wizzard/src/main.c,v
   Sticky Tag:           beta_0-1_branch (branch: 1.9.2)
   Sticky Date:          (none)
   Sticky Options:       (none)
```

Katalogi magazynów lokalnych również mogą być lepkie, a stan taki implikuje przekazywanie go również każdemu plikowi dodanemu do takiego katalogu. Nazwy katalogów oznaczonych jako lepkie znajdują się w pliku *Tag* umiejscowionym w podkatalogu *CVS* takiego lepkiego katalogu. Przykład 4.9. prezentuje plik *Tag*.

Przykład 4.9. Zawartość pliku Tag

```
bash-2.05a$ less CVS/Tag
Tbeta_0-1_branch
```

Stan lepkości nadaje się lub zmienia za pomocą polecenia `cvs checkout` lub `cvs update` wraz z opcjami `-k`, `-D` lub `-r`. Stan taki można usunąć poprzez uruchomienie polecenia `cvs update -A`, które to polecenie spowoduje pobranie z pnia repozytorium rewizji *HEAD*, nadanie ustawień domyślnych odpowiednim plikom oraz usunięcie wszystkich lepkich znaczników z katalogów.

Aby usunąć oznaczenie lepkości z danego katalogu, należy posłużyć się poleceniem `cvs update -A`, określając dany katalog jako cel działania tegoż polecenia. Polecenie `cvs update -A` *`nazwa_pliku`* nie spowoduje usunięcia znacznika lepkości katalogu, jak również nie będzie ono odnosiło się do innych plików znajdujących się w katalogu.

Rozgałęzianie

We wcześniejszej wersji tego rozdziału określono rozgałęzienie jako rozwidloną linię projektową, z wyróżnieniem linii nazywanej *rozgałęzieniem* lub *odgałęzieniem* (ang. *branch*) oraz linią główną projektu nazywaną *pniem* (ang. *trunk*). W skład zarówno odgałęzienia, jak i pnia wchodzą te same dane do momentu, w którym zostały one rozdzielone — moment ten nazywany jest *podstawą* (ang. *base*) odgałęzienia. Od tego momentu dane na temat zmian dokonanych w odgałęzieniu, jak i pniu, przechowywane są oddzielnie. Kolejnym wersjom odgałęzienia nadawane są numery rewizji bazujące na numerze rewizji podstawy odgałęzienia.

Odgałęzić można nie tylko pojedynczy plik, ale także ich grupę lub nawet cały projekt, podobnie jak przypadku oznakowywania, gdzie można nadawać znaczniki pojedynczym plikom, jak również całej ich grupie. Najlepiej jednak rozgałęziać cały projekt. Doświadczenie wskazuje na to, że dokonując odgałęzienia jednego pliku, koniec końców trzeba będzie odgałęzić również i inne pliki wchodzące w skład tego samego projektu z tych samych powodów, z jakich odgałęziono pierwszy z plików. Dużo łatwiej jest więc śledzić w trakcie projektu pojedynczą gałąź niż dużą ilość niezwiązanych ze sobą gałęzi, stworzonych dla pojedynczych plików.

Znacznik określa rewizję każdego oznakowanego pliku, a oznakowany nim magazyn lokalny nie może być zmodyfikowany. W odróżnieniu od takiego zachowania, za pomocą mechanizmu rozgałęziania tworzone są zdatne do edytowania rewizje, które mogą być niezależnie synchronizowane z repozytorium, pobierane, aktualizowane i oznakowywane. Pod wieloma względami odgałęzienie można traktować jako zupełnie niezależną linię rozwojową projektu.

Plik można oznakować dowolną ilością znaczników, przy czym żadna z nazw znaczników nie może się powtarzać. CVS pozwala również na oznakowywanie odgałęzionych rewizji za pomocą różnych znaczników gałęzi. Bazuje on przy tym na niepowtarzalności nazw znaczników i odgałęzień, przyjmując nazwę odgałęzienia jako nazwę znacznika.

Odgałęzienie jest aktywną linią rozwojową projektu, w związku z czym znaczniki są wykorzystywane do oznaczania określonych wersji projektu w istotnych dla niego momentach. Pomimo faktu, iż zarówno odgałęzienia, jak i znaczniki tworzy się za pomocą polecenia `cvs tag`, ich zadanie jest zgoła różne. Odgałęzienie reprezentuje wiele różnych rewizji, podczas gdy znacznik — tylko jedną.

W przypadku niektórych poleceń, takich jak np. `cvs diff`, system CVS wymaga, aby odgałęzienie wskazywało tylko na jedną rewizję. W takich przypadkach system CVS wskazuje na najnowszą rewizję znajdującą się w danym odgałęzieniu.

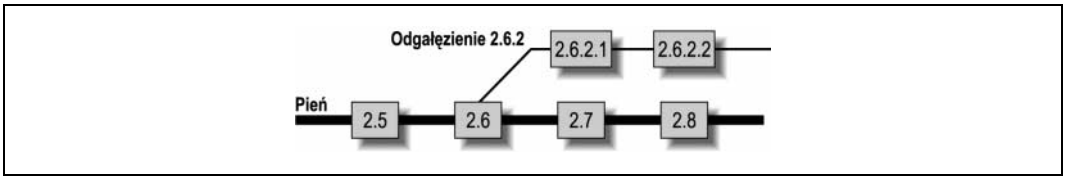
Odgałęzienia wykorzystywane są często do tego, aby umożliwić testerom przeprowadzenie testów aplikacji na wersji *release candidate*, gdy przygotowywana jest już ogólnodostępna wersja wydania programu, podczas gdy niezależnie od przeprowadzanych testów może już być opracowywana nowa linia rozwojowa produktu. Odgałęzienia mogą być również wykorzystywane do opracowywania w oddzieleniu od głównej linii rozwojowej eksperymentalnych zmian w projekcie, np. do kompletnego przepisania kodu biblioteki.

W idealnym przypadku dobrze byłoby mieć świadomość potrzeby stworzenia odgałęzienia jeszcze przed rozpoczęciem modyfikacji plików projektu znajdujących się w magazynie

lokalnym. Rozgałęzienie projektu z poziomu magazynu lokalnego zaprezentowano w punkcie „Tworzenie odgałęzienia" znajdującego się w dalszej części tego rozdziału.

Zdarzają się sytuacje, gdy świadomość eksperymentalnego charakteru dokonanych zmian lub potrzeba przeprojektowania całej sekcji projektu przychodzi dopiero po zmodyfikowaniu plików. W takich sytuacjach warto utworzyć odgałęzienie projektu umożliwiające śledzenie dokonanych zmian, a które nie będzie wpływało na resztę projektu. Sposób wykonania takiego odgałęzienia został opisany w punkcie „Wsteczne tworzenie odgałęzień" w dalszej części tego rozdziału. Wsteczne tworzenie odgałęzień może być nieco trudniejsze do wykonania standardowego rozgałęzienia, najlepiej więc planować tego typu działania z wyprzedzeniem.

Na rysunku 4.3 zaprezentowano operacje rozgałęzienia projektu. Rewizja 2.6 jest rewizją bazową odgałęzienia, a odgałęzienie jako takie jest reprezentowane jako rewizja 2.6.2. Odgałęzienie posiada własną, niezależną od głównego pnia projektu linię rozwojową, numerowaną począwszy od rewizji 2.6.2.1. Numeracja rewizji głównego pnia jest kontynuowana jako 2.7.



Rysunek 4.3. Odgałęzienie

Odgałęzienie bazuje na pniu głównym. Jednak zarówno pień, jak i odgałęzienie przechowywane są w tym samym pliku znajdującym się w repozytorium, co wiąże się z tym, że polecenia operujące na tym pliku na poziomie repozytorium mogą wpływać na pień, a także na odgałęzienie. Przykładowo, ten sam znacznik nie może być użyty w tym samym pliku do oznakowania pnia i odgałęzienia. Oprócz tego, wynik działania polecenia `cvs status` i `cvs log` wyświetlają całościową informacje na temat pliku, włączając w to informacje zarówno na temat pnia jak i odgałęzienia.

Zmiany dokonane w pniu mogą zostać włączone do odgałęzienia, ale także zmiany poczynione w odgałęzieniu mogą zostać włączone do głównego pnia projektu. Dalszy rozwój odgałęzienia może zostać albo porzucony, albo kontynuowany w zależności od przeznaczenia scalenia zmian.

Obszary zastosowań odgałęzień

Odgałęzienia mają wiele zastosowań. Zwykle są one wykorzystywane do tworzenia oddzielnych linii rozwojowych projektu, podczas gdy pień służy opracowywaniu głównej linii projektowej produktu.

W projektach programistycznych i w zadaniach dotyczących zarządzania treścią, odgałęzienia są często wykorzystywane do prac eksperymentalnych, opracowywania wersji kandydackich produktu, refaktoryzacji kodu lub treści, a także do wprowadzania poprawek.

W przypadku zarządzania konfiguracją za pomocą CVS, pień główny może służyć jako konfiguracja domyślna, a odgałęzienia jako poszczególne warianty konfiguracyjne, np. jedna gałąź odpowiada za konfiguracje usług sieciowych, inna za konfigurację usług związanych z dostarczaniem poczty itd.

Poniższa lista prezentuje niektóre z typowych zastosowań dla różnych typów odgałęzień. Typy odgałęzień opisano w podrozdziale „Strategie rozgałęziania" w dalszej części tego rozdziału.

Warianty zastosowań jednego schematu, takie jak przechowywanie konfiguracji serwerów
: W przypadku zastosowania odgałęzień jako repozytorium wariantów konfiguracyjnych, najlepiej używać długich odgałęzień lub też okazjonalnie — odgałęzień zagnieżdżonych. Dodatkowe zmiany najlepiej wprowadzać w pniu, a następnie scalać je z odgałęzieniem, jeśli jest to wymagane do poprawnego funkcjonowania odgałęzienia.

Zarządzanie poprawkami
: W celu zarządzania poprawkami błędów należy używać długich rozgałęzień, a zmiany scalać z pniem głównym projektu.

Prace eksperymentalne, takie jak nowe fragmenty kodu lub np. projekt nowego wyglądu strony
: W takich wypadkach należy używać krótkich rozgałęzień, a zmiany scalać z pniem głównym.

Zasadnicze zmiany projektu, takie jak kompletne przepisanie kodu
: W zależności od rozpiętości zmian można stosować długie odgałęzienia scalane do pnia, długie odgałęzienia scalane w obu kierunkach, jak i odgałęzienia krótkie.

Testowe wersje kandydackie
: Należy stosować długie odgałęzienia scalane do pnia głównego. Można również pokusić się o konwersję tego typu odgałęzień na odgałęzienia zarządzania poprawkami po wypuszczeniu pełnej wersji produktu.

Tworzenie odgałęzienia

Odgałęzienie tworzy się za pomocą polecenia `cvs tag` lub `cvs rtag` wraz z opcją `-b`. Opcja ta może występować w połączeniu z dowolnymi innymi opcjami tych poleceń, odpowiedzialnymi za tworzenie znaczników. W celu identyfikacji wersji mającej zostać rozgałęzionej można użyć zarówno daty, istniejącego już znacznika lub numeru rewizji.

Za pomocą polecenia `cvs tag` można również utworzyć odgałęzienie na podstawie ostatnio zsynchronizowanej rewizji magazynu lokalnego. Utworzenie odgałęzienia w ten sposób przypomina sposób oznaczania plików, jaki opisano w punkcie „Oznaczanie przy pomocy magazynu lokalnego".

Przykład 4.10. ilustruje sposób utworzenia odgałęzienia za pomocą polecenia `cvs tag` z wykorzystaniem nazwy znacznika do identyfikacji rewizji do rozgałęzienia. Wydając polecenie `cvs update` można upewnić się, czy wszystkie pliki z rewizji *pre_beta_0.1* znajdują się w magazynie lokalnym. Wyświetlony wynik działania polecenia `cvs update` pozwala również na upewnienie się, że żaden z plików nie został zmieniony.

Przykład 4.10. Tworzenie odgałęzienia

```
bash-2.05a$ cvs update -d -r pre_beta_0-1
.
.
.
bash-2.05a$ cvs tag -r pre_beta_0-1 -b pre_beta_0-1_branch
cvs server: Tagging .
T Changelog
T INSTALL
T Makefile
```

```
T README
T TODO
cvs server: Tagging doc
cvs server: Tagging doc/design
T doc/design/AcceptanceTest.doc
T doc/design/Analysis.rtf
T doc/design/Requirements.doc
T doc/design/Specification.rtf
cvs server: Tagging doc/plan
T doc/plan/Schedule.rtf
cvs server: Tagging src
T src/config.h
T src/main.c
```

Odgałęzienie zostaje utworzone w repozytorium, a nie w magazynie lokalnym. Aby rozpocząć edycję plików wchodzących w skład odgałęzienia, należy najpierw pobrać pliki do magazynu lokalnego, realizując polecenie `cvs checkout`, lub zaktualizować magazyn za pomocą polecenia `cvs update`, tak aby odzwierciedlał on odgałęzienie.

Dobrą praktyką jest z kolei oznaczanie głównego pnia projektu tuż przed wykonaniem operacji rozgałęzienia, z tego względu, że ułatwi to przyszłe scalenie zmian. Aby mieć pewność, że pliki oznaczone jako te, które mają być podstawą odgałęzienia w istocie stają się plikami, na bazie których jest ono wykonywane, należy utworzyć odgałęzienie za pomocą polecenia `cvs rtag -r` *`znacznik_podstawy`* `-b` *`znacznik_odgałęzienia nazwa_projektu`*. Polecenie to wykorzystuje znacznik podstawy do określenia rewizji będącej podstawą do tworzenia następnych odgałęzień.

Przykład 4.11 przedstawia sposób, w jaki należy utworzyć znacznik podstawy, a następnie wykonać operację rozgałęzienia. Polecenie `cvs status` zostało wykorzystane do pokazania stanu znacznika jednego z plików.

Przykład 4.11. Oznakowywanie rewizji przed wykonaniem operacji rozgałęzienia

```
bash-2.05a$ cvs tag beta_0-1_branch_root
cvs server: Tagging .
T Changelog
T INSTALL
.
.
.
cvs server: Tagging src
T src/config.h
T src/main.c
bash-2.05a$ cvs rtag -r beta_0-1_branch_root -b beta_0-1_branch wizzard
cvs rtag: Tagging wizzard
cvs rtag: Tagging wizzard/doc
cvs rtag: Tagging wizzard/doc/design
cvs rtag: Tagging wizzard/doc/plan
cvs rtag: Tagging wizzard/lib
cvs rtag: Tagging wizzard/man
cvs rtag: Tagging wizzard/src
bash-2.05a$ cvs status -v src/main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c               Status: Up-to-date

   Working revision:    1.9
   Repository revision: 1.9     /var/lib/cvs/wizzard/src/main.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

```
Existing Tags:
  beta_0-1_branch               (branch: 1.9.2)
  beta_0-1_branch_root          (revision: 1.9)
  pre_beta_0-1                  (revision: 1.8)
```

Wsteczne tworzenie odgałęzień

W sytuacji, gdy po dokonaniu pewnej ilości zmian, ale jeszcze przed ich zatwierdzeniem w repozytorium, powstaje potrzeba wykonania odgałęzienia, należy spróbować utworzyć odgałęzienie na podstawie wersji przed zmianami. Jeśli zmiany nie zostały jeszcze zatwierdzone, odgałęzienie można wykonać wstecznie, postępując zgodnie z poniższymi wskazówkami:

1. Nie należy wprowadzać zmian do repozytorium, aż do momentu wykonania odgałęzienia. Informacja ta jest ważna z tego względu, że wykorzystywana jest funkcja polecenia `cvs tag` pozwalająca na utworzenie odgałęzienia przed ostatnimi zmianami.

   Ze względu na brak możliwości wprowadzenia zmian do repozytorium przed rozpoczęciem procedury, najlepiej jest utworzyć kopię zapasową zmienionych plików, wydając systemowe polecenie `copy`. Kopia powinna obejmować cały magazyn lokalny i powinna być przechowywana aż do momentu wprowadzenia zmian do repozytorium.

2. Odgałęzienie należy utworzyć za pomocą polecenia `cvs tag -b` *`nazwa_odgałęzienia`*. Polecenie `cvs tag` oznakowuje ostatnią rewizję, która została zsynchronizowana z repozytorium, właśnie tę, która ma zostać poddana operacji rozgałęziania. Polecenie to nie powoduje modyfikacji magazynu lokalnego.

3. Za pomocą polecenia `cvs update -r` *`nazwa_odgałęzienia`* należy wczytać pliki danego odgałęzienia do magazynu lokalnego. Spowoduje to próbę scalenia plików odgałęzienia z plikami znajdującymi się w magazynie lokalnym, lecz ponieważ pliki z magazynu lokalnego bazują na plikach odgałęzienia, operacja scalenia nie wywoła zmiany jakichkolwiek plików. CVS oznaczy pliki znajdujące się w magazynie lokalnym jako pliki będące częścią odgałęzienia, nadając im lepkie znaczniki.

4. Wydając polecenie `cvs commit`, należy zapisać zmiany dokonane w plikach do repozytorium jako kolejną rewizję odgałęzienia.

5. Sprawdzić poprawność plików, a następnie usunąć pliki tymczasowej kopii zapasowej

Powyższy proces bazuje na założeniu, że polecenie `cvs tag` oznakowuje repozytorium takim, jakie było ono w chwili ostatniego wprowadzenia zmian z magazynu lokalnego. Odgałęzienie jest zatem tworzone dla tego momentu, a ewentualna próba aktualizacji magazynu lokalnego z odgałęzieniem powoduje, że system CVS stara się scalić pliki podstawy, na bazie których został utworzony magazyn lokalny z plikami w nim się znajdującymi, w wyniku czego magazyn lokalny pozostaje niezmieniony.

Przykład 4.12 przedstawia operację wstecznego utworzenia odgałęzienia.

Przykład 4.12. Wsteczne utworzenie odgałęzienia

```
ash-2.05a$ cvs tag -b test_0-1_branch
cvs server: Tagging .
T config.h
T main.c
bash-2.05a$ cvs update -r test_0-1_branch
cvs server: Updating .
M config.h
bash-2.05a$ cvs commit
```

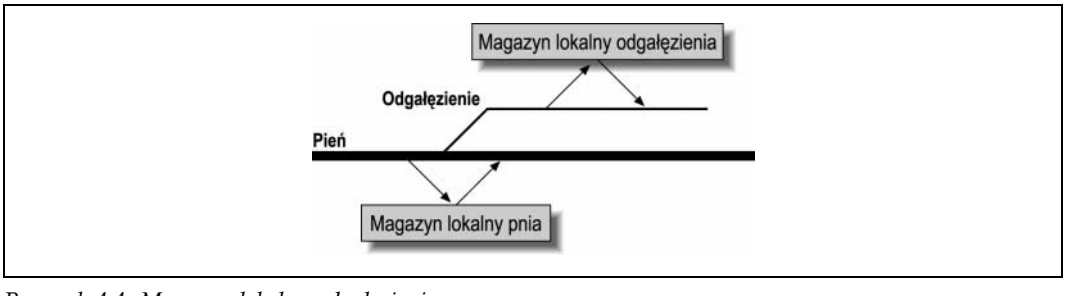
Po wprowadzeniu zmian do repozytorium można również wykonać operację wstecznego utworzenia rozgałęzienia za pomocą określonej daty, tak jak to pokazano na rysunku 4.2, trzeba jednak zastąpić opcję `-b nazwa_znacznika`, opcją `-D data`.

W przypadku częstego kopiowania i wklejania godzin podawanych przez polecenie `cvs log`, należy dodatkowo określić strefę czasową UTC za pomocą opcji `-D data`.

Tworzenie magazynu lokalnego odgałęzienia

Aby zmodyfikować pliki odgałęzienia, należy pobrać pliki z magazynu lokalnego, będącego protoplastą odgałęzienia, które ma zostać zmodyfikowane. Polecenie `cvs commit` wydane z poziomu magazynu lokalnego odgałęzienia spowoduje wprowadzenie zmian z magazynu lokalnego odgałęzienia do repozytorium, a polecenie `cvs update` pobranie plików z repozytorium do magazynu lokalnego.

Magazyn lokalny odgałęzienia tworzy się za pomocą polecenia `cvs checkout` lub `cvs update` wraz z opcją `-r nazwa_znacznika_odgalezienia`. Rysunek 4.4 prezentuje operację pobrania plików do magazynu lokalnego odgałęzienia



Rysunek 4.4. Magazyn lokalny odgałęzienia

CVS oznacza pliki znajdujące się w magazynie lokalnym odgałęzienia lepkim znacznikiem, aby tym samym podkreślić ich przynależność do odgałęzienia. Przykład 4.13 prezentuje sposób utworzenia magazynu lokalnego odgałęzienia i raport dotyczący statusu plików, z których jeden oznaczony jest lepkim znacznikiem odgałęzienia.

Przykład 4.13. Utworzenie magazynu lokalnego odgałęzienia

```
bash-2.05a$ cvs -d cvs:/var/lib/cvs checkout -r beta_0-1_branch wizzard
cvs server: Updating wizzard
U wizzard/Changelog
U wizzard/INSTALL
.
.
.
cvs server: Updating wizzard/src
U wizzard/src/config.h
U wizzard/src/main.c
bash-2.05a$ cd wizzard/src/
bash-2.05a$ cvs status main.c
=  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =
File: main.c               Status: Up-to-date
```

```
Working revision:    1.9
Repository revision: 1.9     /var/lib/cvs/wizzard/src/main.c,v
Sticky Tag:          beta_0-1_branch (branch: 1.9.2)
Sticky Date:         (none)
Sticky Options:      (none)
```

Istnieje możliwość pobrania poszczególnych plików odgałęzienia do głównego magazynu lokalnego, jednak taka operacja, skutkująca mieszaniną plików należących do odgałęzienia i pnia, nie jest zalecana. Pobranie pojedynczych plików nie należących do tego samego odgałęzienia lub pnia, co aktualny magazyn lokalny, jest możliwe z poziomu katalogu nie będącego magazynem lokalnym.

Polecenie `cvs update -A` umożliwia uaktualnienie plików znajdujących się w magazynie lokalnym za pomocą plików z pnia projektu. Polecenie to usuwa znaczniki lepkości i pobiera najnowsze wersje plików należące do pnia. Wszystkie zmiany z magazynu lokalnego wykonane od momentu, gdy zostały pobrane plik,i na których ów magazyn bazuje (rewizji *BASE*), są scalane z nowo pobieranymi plikami.

Magazynu lokalnego odgałęzienia można używać jak normalnego magazynu. Czynności wykonywane na poszczególnych rewizjach plików będą jednak bardziej dotyczyły samego odgałęzienia, a nie pnia projektu. Z kolei czynności dotyczące kopii pliku znajdującego się w repozytorium będą dotyczyły wszystkich jego instancji. Przykładowo, polecenie `cvs status` wydane z poziomu magazynu lokalnego wygeneruje raport przedstawiający status lokalnych kopii plików, numery rewizji pnia dla rewizji roboczych i tych znajdujących się już w repozytorium oraz aktualny znacznik odgałęzienia i rewizji oznaczony jako lepki.

Dodawanie i usuwanie plików

Po każdym wykonaniu polecenia `cvs add` lub `cvs remove` na plikach przechowywanych w magazynie lokalnym odgałęzienia, operacja dodania lub usunięcia plików ogranicza się do plików należących do odgałęzienia i nie obejmuje swoim działaniem pnia. W przykładzie 4.14 pokazano odpowiedź systemu CVS po dodaniu pliku do odgałęzienia.

Przykład 4.14. Dodawanie pliku do odgałęzienia

```
bash-2.05a$ cvs add handheld.c
cvs server: scheduling file `handheld.c' for addition on branch `beta_0-1_branch'
cvs server: use 'cvs commit' to add this file permanently
```

Scalanie odgałęzień

Proces scalenia odgałęzienia z pniem powoduje uwzględnienie zmian dokonanych w trakcie istnienia odgałęzienia w najnowszej wersji kodu pnia. To, czy działanie takie jest pożądane, zależy od powodu, dla którego zostało ono utworzone. Jednym z takich powodów może być włączenie poprawek do kodu głównego, innym — chęć włączenia zawartości eksperymentalnego odgałęzienia do kodu głównego. Można również scalić zmiany z kodu głównego do odgałęzienia lub połączyć razem zawartość dwóch odgałęzień.

Oznakowanie odgałęzienia w momencie scalenia go z pewnością należy do dobrej praktyki. Znaczniki takie zachowują się niczym markery wskazujące na to, kiedy miało miejsce każde scalenie.

Po scaleniu dwóch odgałęzień lub odgałęzienia i pnia, zwykle w efekcie otrzymuje się jeden element zmodyfikowany i jeden, który pozostał bez zmian. Po scaleniu zmian z odgałęzienia do pnia głównego, a następnie po wprowadzeniu do repozytorium zmian z magazynu lokalnego, następna rewizja kodu głównego będzie uwzględniała wszystkie te zmiany. Aby móc jednak pracować na kopii kodu głównego pozbawionej scalonych zmian, należy rozważyć scalenie kodu głównego z odgałęzieniem lub utworzenie nowego odgałęzienia zawierającego zarówno dane wchodzące w skład kodu głównego, jak i dane z odgałęzienia.

Kończąc pracę z odgałęzieniami logicznym poleca się usunięcie jej lub zamknięcie jej w jakikolwiek inny sposób. CVS jednak nie oczekuje na to, iż niewykorzystywane odgałęzienia będą usuwane. Zamiast tego przechowuje on je jako część zapisu projektu. Nie istnieje również żadne polecenie oznakowujące dane odgałęzienie jako nieużywane. Jako znacznika końcowego odgałęzienia należy użyć komunikatu dziennika.

Za pomocą polecenia `cvs commit -f` można wymusić na CVS zsynchronizowanie z repozytorium plików, w których nie zostały dokonane żadne zmiany, po to aby móc wprowadzić do dziennika komunikat na temat wykonanego scalenia tych plików.

Scalanie zmian z odgałęzienia do kodu głównego

W celu scalenia zmian z odgałęzienia do kodu głównego, należy najpierw pobrać pliki do aktualnego magazynu lokalnego pnia, a następnie uruchomić polecenie `cvs update -j` *`znacznik_bazowy_odgałęzienia`* `-j` *`nazwa_odgałęzienia`*. Następnie konieczne jest rozwiązanie wszystkich zgłoszonyh przez operację scalenia konfliktów i wprowadzenie zmian do repozytorium. Jeżeli wprowadzane zmiany mają złożony i kompleksowy charakter, rozwiązaniem konfliktów powinien zająć się jeden z deweloperów lub osoba prowadząca projekt, odpowiedzialna za zarządzanie odgałęzieniami i kodem głównym.

W przypadku, gdy odgałęzienie zostało już wcześniej scalone z kodem głównym i zostało ono również na tym etapie oznakowane, polecenie `cvs update -j` *`znacznik_ostataniego_scalenia`* `-j` *`nazwa_odgałęzienia`* spowoduje scalenie zmian dokonanych od czasu ostatniej operacji.

Przykład 4.15 przedstawia proces scalenia odgałęzienia z kodem głównym. W tym wypadku CVS odmówił usunięcia zmienionego pliku z kodu głównego, lecz usunął taki plik w odgałęzieniu. Przykład ten demonstruje również operacje dodania pliku *handheld.c* utworzonego wcześniej w odgałęzieniu.

Przykład 4.15. Scalanie zmian z odgałęzienia do kodu głównego

```
bash-2.05a$ cvs update -j beta_0-1_branch_root -j beta_0-1_branch
cvs server: Updating .
cvs server: file config.h has been modified, but has been removed in revision beta_0-
1_
branch
U handheld.c
```

Scalanie zmian z kodu głównego do odgałęzienia

Przed scaleniem zmian z kodu głównego do odgałęzienia konieczne jest pobranie plików do aktualnego magazynu lokalnego odgałęzienia, a następnie uruchomienie z poziomu tego magazynu polecenia `cvs update -j` *`znacznik_bazowy_odgałęzienia`* `-j HEAD`. Następnie należy

rozwiązać wszystkie zgłoszone przez operację scalenia konflikty i wprowadzić zmiany do repozytorium. Jeżeli wprowadzane zmiany mają złożony i kompleksowy charakter, rozwiązaniem konfliktów powinien zająć się jeden z deweloperów lub osoba prowadząca projekt, odpowiedzialna za zarządzanie odgałęzieniami i kodem głównym.

W przypadku, gdy kod główny został już wcześniej scalony z odgałęzieniem i został on również na tym etapie oznakowany, polecenie `cvs update -j znacznik_ostataniego_scalenia -j HEAD` spowoduje scalenie zmian dokonanych od czasu ostatniej operacji.

Przykład 4.16 prezentuje wynik próby scalenia kodu głównego z odgałęzieniem. Plik *config.h* został usunięty z odgałęzienia, lecz nadal jest obecny w kodzie głównym. Konflikt ten musi zostać rozwiązany, najprawdopodobniej poprzez anulowanie operacji usunięcia. Z kolei plik *handheld.c* nie istniał wcześniej w kodzie głównym, lecz ponieważ został do niego dodany podczas operacji scalenia zaprezentowanej w przykładzie 4.15, w związku z tym wygenerowany komunikat o błędzie dotyczący tego pliku może zostać zignorowany.

Przykład 4.16. Scalanie zmian z kodu głównego do odgałęzienia

```
bash-2.05a$ cvs update -j beta_0-1_branch_root -j HEAD
cvs server: Updating .
cvs server: file config.h does not exist, but is present in revision HEAD
cvs server: file handheld.c exists, but has been added in revision HEAD
U server.c
```

Scalanie zmian z odgałęzienia do odgałęzienia

Scalenia zmian z jednego odgałęzienia do drugiego można dokonać pobierając pliki do magazynu lokalnego odgałęzienia docelowego, a następnie uruchomiając polecenie `cvs update -j bazowa_nazwa_znacznika -j nazwa_drugiego_odgałęzienia`. Dzięki temu scalona zostanie tym samym zawartość jednego magazynu lokalnego z drugim, odświeżonym przed chwilą, magazynem lokalnym odgałęzienia.

W przypadku gdy dane odgałęzienia były już wcześniej scalane, a odgałęzienie źródłowe na tym etapie oznakowane, polecenie `cvs update -j znacznik_ostataniego_scalenia -j nazwa_odgałęzienia` uruchomione z poziomu magazynu lokalnego odgałęzienia docelowego spowoduje scalenie zmian dokonanych od czasu ostatniej operacji.

Słowa kluczowe a scalanie odgałęzień

Zastąpienia słów kluczowych (patrz: rozdział 3.) mogą powodować konflikty w trakcie scalania dwóch różnych rewizji plików. Najczęstszą przyczyną konfliktów jest słowo kluczowe *Revision*, ponieważ zostaje ono zastąpione aktualną rewizją pliku. Konfliktów tych można uniknąć za pomocą trybu zastępowania słów kluczowych `-kk`, dzięki któremu słowa kluczowe nie są zastępowane odpowiednimi wartościami.

Tryb `-kk` może spowodować uszkodzenie plików binarnych, jeżeli będą one zawierały łańcuch znaków, który CVS rozpozna jako nazwę lub wartość słowa kluczowego.

Scalanie plików binarnych i plików specjalnych

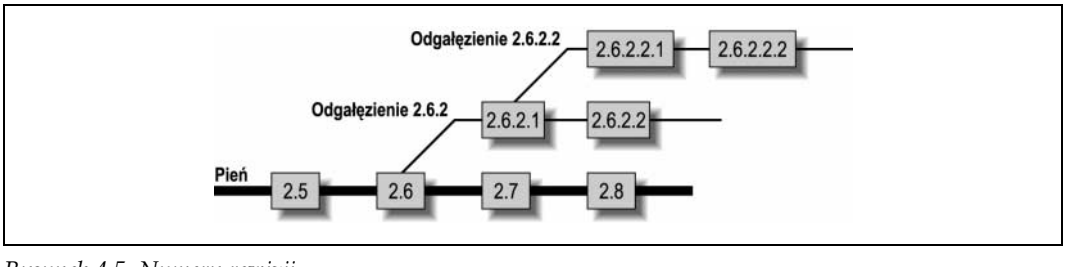
Zmiany poczynione w plikach binarnych, jak również w innych plikach, których nie da się scalić, nie mogą niestety zostać automatycznie scalone. Pliki takie mogą być scalone, pod warunkiem że istnieją dla nich odpowiednie narzędzia działające na podobnych zasadach jak narzędzia *diff* i *patch* dla plików tekstowych. W innym wypadku scalenie tego rodzaju plików trzeba będzie przeprowadzić ręcznie. Fakt ten powinien zostać wzięty pod rozwagę przy planowaniu rozgałęziania linii rozwojowej tego typu plików.

Numery rewizji odgałęzień

Numery rewizji zwykłych plików składają się z prefiksu oraz ze zwiększającego się skokowo identyfikatora, stąd też po zatwierdzeniu kolejnych zmian po rewizji 1.1 następuje rewizja 1.2, 1.3 i 1.4.

Numer rewizji plików odgałęzienia tworzony jest na podstawie numeru rewizji, z której zostało ono odgałęzione. A zatem numerami rewizji dla odgałęzienia 1.4.2 będą numery 1.4.2.x, gdzie za x należy przyjąć zmieniający się identyfikator rewizji. Warto przy tym pamiętać, że odgałęzienie nie składa się z pojedynczej rewizji, w jego skład wchodzi cała linia rewizji.

Na rysunku 4.5 pokazano rozgałęziony plik wraz z kolejnymi numerami jego rewizji.



Rysunek 4.5. Numery rewizji

System CVS nie nadaje nigdy odgałęzieniu stworzonemu przez użytkownika numeru 1.1.1, ten konkretny numer rewizji zarezerwowany jest dla specjalnej gałęzi nazywanej gałęzią dostawcy.

Numery specjalne odgałęzień

System CVS wstawia czasami na pozycji drugiej z prawej strony numeru odgałęzienia cyfrę 0. Czyni to, aby zapewnić lepszą wydajność samego systemu CVS. Skutek takiego działania może czasem odbijać się na działaniu polecenia `cvs log`, może również wpływać na polecenia `cvs admin` wykonywane z poziomu odgałęzienia, lecz poza tymi przypadkami działanie to nie jest w żaden sposób widoczne na zewnątrz. W przypadku gałęzi 1.4.2, jej numer będzie wyświetlany jako 1.4.0.2. Kolejne rewizje odgałęzienia nie będą jednak już zawierały wspomnianego zera, numerem pierwszej rewizji w gałęzi 1.4.0.2, będzie rewizja 1.4.2.1, a rewizją drugą będzie 1.4.2.2.

Ta niestabilność zachowania w stosunku do liczby zero nie dotyczy jednak samego sposobu użytkowania CVS, do gałęzi można się odwoływać bez podawania zera. Przykładowo polecenia `cvs update -r 1.4.2` i `cvs update -r 1.4.0.2` pobiorą dokładnie te same rewizje plików.

Usuwanie lub przenoszenie odgałęzień

Odgałęzienie usuwa lub przenosi się za pomocą polecenia `cvs tag` lub `cvs rtag` wraz z opcjami, odpowiednio: `-d` lub `-F`. System CVS wymaga również aby została użyta opcja `-B`, aby upewnić się, że ma być usunięta lub przeniesiona dokładnie ta gałąź, na którą wskazuje odpowiedni znacznik.

Nie zaleca się usuwania lub przenoszenia gałęzi z wyjątkiem przypadku, gdy została ona dopiero utworzona i jak do tej pory nikt na niej nie pracował. Podczas gdy większość znaczników można usunąć bądź przenieść bez wpływania na zapis historii projektu, zmiany tego typu dokonywane na gałęziach stają się częścią zapisu zmian w każdym pliku wchodzącym w skład danej gałęzi.

Przykład 4.17 ilustruje operację utworzonej chwilę wcześniej gałęzi

Przykład 4.17. Usuwanie gałęzi

```
bash-2.05a$ cvs tag -d -B pre_beta_0-1_branch
cvs server: Untagging .
D Changelog
D INSTALL
D Makefile
D README
D TODO
cvs server: Untagging doc
cvs server: Untagging doc/design
D doc/design/AcceptanceTest.doc
D doc/design/Analysis.rtf
D doc/design/Requirements.doc
D doc/design/Specification.rtf
cvs server: Untagging doc/plan
D doc/plan/Schedule.rtf
cvs server: Untagging src
D src/config.h
D src/main.c
```

Strategie rozgłęziania

Podrozdział ten przedstawia podstawy strategii tworzenia odgałęzień i dwie główne przesłanki filozofii rozgałęziania. Będzie tu można również znaleźć odpowiedzi na kilka pytań o to, kiedy i dlaczego powinno tworzyć się odgałęzienia.

Filozofia rozgałęziania

Podstawową zasadą rozgałęziania jest zasada utrzymywania głównej linii rozwojowej projektu jako kodu głównego (pnia) — wszystko inne może zostać rozgałęzione. Zasadniczy problem powstaje w momencie, gdy trzeba określić zasięg głównej linii rozwojowej. Czy główna część kodu powinna składać się tylko z kodu stabilnego czy może również zawierać kod niestabilny? Czy każde kolejne wydanie powinno być odgałęziane w celu przeprowadzenia testów? Czy nowe funkcje powinny być opracowywane w kodzie głównym, czy w odgałęzieniu?

Odpowiedzi na wszystkie powyższe pytania zależą od sposobu podejścia określanego przez dwie zasadnicze definicje „głównej linii rozwojowej". Te dwie różne definicje skutkują w dwóch różnych filozofiach rozgałęziania, które można określić jako stabilności i niestabilności

Filozofia stabilności

Filozofia stabilności opiera się na założeniach, że kodem główny programu powinien być kod niemalże gotowy do wydania. Do prac rozwojowych wykorzystywane są odgałęziania, poprawki, wersje testowe i refaktoryzacja. Odgałęzienia są również używane do opracowywania kodu eksperymentalnego.

Najbardziej ortodoksyjna odmiana tej filozofii zakłada, że nic co nie przeszło przez dział kontroli jakości nie powinno być scalone do kodu głównego. Daje to pewność, że jakakolwiek wersja *release candidate* bazująca na kodzie głównym, może być opublikowana po poniesieniu niewielkiego nakładu środków na testowanie, lub też sprzedana.

Łagodniejsze podejście pozwala na scalenie z kodem głównym wszystkiego, co zostało przetestowane, co najmniej na poziomie zespołu projektowego. Podejście takie wymaga jednak tego, aby wersja *release candidate* została odgałęziona, a następnie przed publikacją przeszła pełną procedurę testową i analizę działu kontroli jakości.

Do zalet filozofii stabilności można zaliczyć:

- możliwość utworzenia w każdej chwili nowego wydania projektu na podstawie kodu głównego, po jego uprzednim przetestowaniu;
- ze względu na powolne tempo zachodzenia zmian w kodzie możliwość przeprowadzania prac eksperymentalnych na odgałęzieniach przy niewielkim prawdopodobieństwie zniweczenia wysiłków innych osób po dokonaniu scalenia z powrotem do kodu głównego.

Największą niedogodnością takiego podejścia jest to, że scalanie zwykle jest wtedy wykonywane przez testera, a nie przez osobę rozumiejącą semantykę scalanego kodu. Również zagrożenie tym, że kod główny mógł zmienić się znacznie od momentu, w którym nastąpiło odgałęzienie do momentu scalenia, jest znaczne. Oba z tych problemów mogą zostać zniwelowane poprzez dokonywanie okresowych scaleń, przy mniej restrykcyjnych zasadach, przeprowadzane przez jednego z deweloperów.

Filozofia niestabilności

Filozofia niestabilności zakłada, że kod główny programu powinien zawsze zawierać najnowszą wersję zmian, bez względu na jego stabilność. Wersja *release candidate* powinna być przy tym odgałęziana w celu przeprowadzenia testów kontroli jakości.

Najbardziej ortodoksyjna odmiana tej filozofii zakłada, że cały proces rozwoju projektu odbywa się na poziomie kodu głównego, podczas gdy odgałęzienia używane są dla wersji *release candidate*, wersji odgałęzień poprawkowych, a także kolejnych wydań.

Łagodniejsze podejście pozwala na odgałęzianie w celu przeprowadzania prac nad kodem eksperymentalnym, refaktoryzacji i w innych specjalnych przypadkach. Ponowne scalanie odgałęzienia do pnia głównego wykonywane jest przez menedżerów odgałęzień.

Do zalet tej filozofii można zaliczyć brak konieczności przeprowadzania częstych scaleń, a co znacząco je również ułatwia, gdyż mogą być one przez to wykonywane przez osoby bardziej obeznane z kodem.

Największą niedogodnością tej filozofii, szczególnie w odniesieniu do jej najbardziej restrykcyjnej wersji, jest to, że kod główny zawiera często niestabilny kod z dużą ilością błędów, kod eksperymentalny, lub nawet czasem taki, który zupełnie nie daje się skompilować. Niedogodność ta może zostać zmniejszona poprzez częste oznaczanie plików w momentach, gdy ich kod jest poprawny,b też w momentach, gdy ktoś rozpoczyna na nim prace eksperymentalne lub refaktoryzacyjne. Łagodniejsze podejście preferuje przechowywanie kodu eksperymentalnego oraz takiego, który może spowodować największe problemy z dala od kodu głównego, co redukuje czas potrzebny do przygotowania wydania projektu do publikacji.

Sposoby rozgałęziania

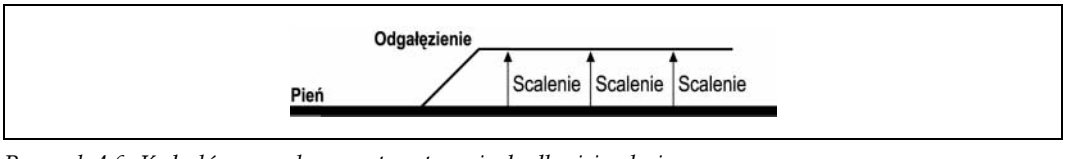
System CVS umożliwia tworzenie i wykorzystywanie odgałęzień, nie narzuca on jednak żadnej szczególnej techniki wykonywania tych zadań. W punkcie tym zostaną przedstawione niektóre z tych technik.

W niniejszym podpunkie określenie *gałąź długa* oznacza albo aktualne odgałęzienia, albo odgałęzienia scalone już kilkakrotnie wcześniej, a nie używane obecnie. *Gałąź krótka* oznacza z kolei odgałęzienie scalone raz z kodem głównym, lecz nie używane od tego czasu.

CVS umożliwia ponowną aktywację odgałęzienia w dowolnym momencie; nie można powiedzieć zatem, że dane odgałęzienie nie jest dłużej ważne. Zakończenie prac nad gałęzią polega po prostu na poinformowaniu deweloperów o tym, że gałąź ta nie będzie już dłużej przedmiotem aktywnego rozwoju.

Scalanie do długiej gałęzi

Posiadanie aktywnego odgałęzienia, do którego systematycznie scalane są zmiany kodu głównego jest użyteczne w sytuacjach, gdy zmiany z odgałęzień nie powinny wpływać na kod główny, ale są one potrzebne w odgałęzieniu. Metoda ta może być wykorzystywana do tworzenia i zarządzania kopiami lustrzanymi witryn sieciowych. Rysunek 4.6 ilustruje zasady tej techniki.

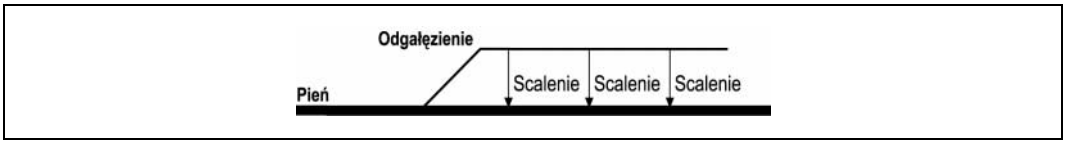


Rysunek 4.6. Kod główny scalany systematycznie do długiej gałęzi

Scalanie długiej gałęzi do kodu głównego

Posiadanie aktywnego odgałęzienia, scalanego systematycznie z kodem głównym jest użyteczne w sytuacjach, gdy zmiany będące rezultatem trwających prac nad kodem głównym nie powinny wpływać na kod odgałęzienia, przy równoczesnej potrzebie zawarcia zmian z odgałęzienia w kodzie głównym. Przykładem zastosowania może być każda sytuacja, w której należy przetestować i przygotować do opublikowania dane odgałęzienie. Projekty, w których testowanie już trwa mogą wykorzystywać tę technikę poprzez włączenie zawartości

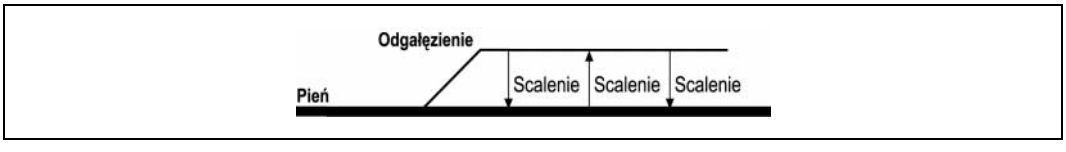
testowanego kodu do odgałęzienia, a następnie ponownego scalenia poprawek z powrotem do kodu głównego (rysunek 4.7).



Rysunek 4.7. Scalanie długiej gałęzi do kodu głównego

Scalanie długiej gałęzi zarówno do kodu głównego, jak i do innej gałęzi

Długa gałąź scalana w obydwu kierunkach zapewnia zarówno odzwierciedlenie zmian odgałęzienia w kodzie głównym, jak i sytuację odwrotną tj. uwzględnienie zmian kodu głównego w odgałęzieniu. W przypadku, gdy deweloperzy pracujący na poziomie odgałęzienia prowadzą na nim bieżące prace rozwojowe, odgałęzienie może być scalone z powrotem do kodu głównego po każdym zakończonym etapie. Wprowadzanie do odgałęzienia zmian z kodu głównego może być dokonywane zawsze wtedy, gdy są do tego przygotowani deweloperzy odgałęzienia. Technika ta może być przydatna, jeśli sekcja po sekcji przepisywany jest stabilny już kod, podczas gdy do kodu głównego dodawane są nowe funkcje programu (rysunek 4.8).

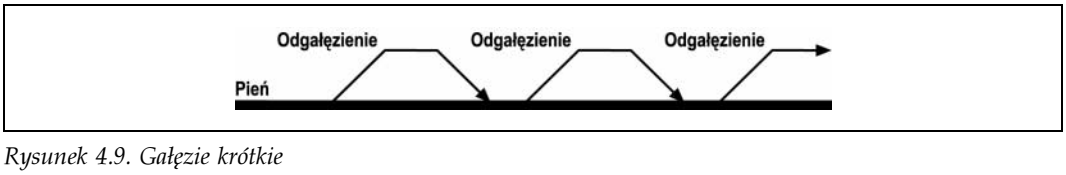


Rysunek 4.8. dwukierunkowe scalanie długiej gałęzi

Gałęzie krótkie

Przykładem krótkiej gałęzi może być pojedyncze odgałęzienie użyte w celu wykonania jednej, prostej zmiany. Gałąź taka może być idealnym wyborem w momencie, gdy powstaje potrzeba dodania jednej funkcji, napisania kodu eksperymentalnego, czy też np. przygotowania do kolejnego wydania.

Gałęzi krótkich można również użyć w celu zasymulowania długiej gałęzi scalonej do kodu głównego i z nim. Seria krótkich gałęzi pozwala uniknąć konieczności scalania zmian odgałęzienia do kodu głównego, a następnie zmian kodu głównego do odgałęzienia. Zamiast tego wystarczy scalić zmiany z odgałęzienia do kodu głównego, a następnie utworzyć nowe odgałęzienie, bazujące na kodzie głównym. Metoda ta może być użyteczna w momencie, gdy zmiany zarówno w kodzie głównym, jak i w odgałęzieniu są znaczne (rysunek 4.9).

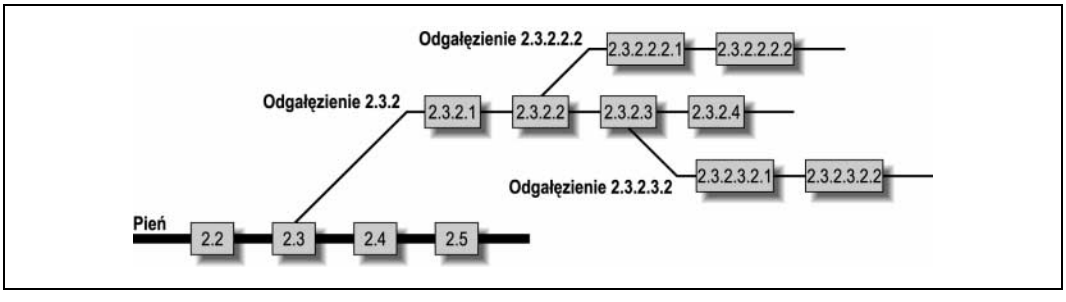


Rysunek 4.9. Gałęzie krótkie

Gałęzie zagnieżdżone

CVS pozwala również na tworzenie odgałęzień z odgałęzień. Gałąź główną traktuje się przy tym jako kod główny, a do zarządzania podgałęzią stosuje się jedną z opisanych już wcześniej technik. Gałęzie zagnieżdżone są najbardziej przydatne do zarządzania zmianami konfiguracji. Aby uniknąć zamieszania, należy wystrzegać się zbytniego rozbudowywania gałęzi zagnieżdżonych.

W przypadku, gdy w trakcie opracowywania nowego zestawu funkcji w danej gałęzi powstaje potrzeba przekazania jednej z tych funkcji do testowania przy równoczesnym zachowaniu ciągłości prac nad odgałęzieniem, można utworzyć odgałęzienie testowe, składające się tylko z testowanej funkcji. Na rysunku 4.10 przedstawiono zestaw gałęzi zagnieżdżonych, odłączonych od głównego kodu projektu.



Rysunek 4.10. Gałęzie zagnieżdżone

Polityka rozgałęziania

Jednolite zasady mogą pomóc zapewnić, że odgałęzianie projektu będzie rzeczywiście stanowiło dla niego znaczącą pomoc. Bez tego, może okazać się, że jedyne co nam zostało to trudny do ogarnięcia bałagan w odgałęzieniach, gdzie nie sposób prześledzić ich rozkładu.

Dzięki określonym i wprowadzonym w życie jednolitym zasadom można przeprowadzać operację scalania w bardzo prosty sposób. Minimalizacja ilości scaleń wymaga większej komunikacji pomiędzy deweloperami pracującymi nad odgałęzieniem i tymi zajmującymi się kodem głównym.

Najlepiej wypracować takie zasady, które będą przynosiły wymierne korzyści zespołowi projektowemu. Poniższe zasady okazały się pomocne dla autorki i jej zespołu:

1. Określenie ogólnych założeń projektu.
2. Zminimalizowanie ilości aktualnie aktywnych odgałęzień oraz upewnienie się, czy każde z nich ma określone zadanie.
3. Zminimalizowanie złożoności systemu odgałęzień. CVS umożliwia tworzenie odgałęzień zagnieżdżonych, jednak złożone systemy odgałęzień bardzo rzadko okazywały się pomocne.
4. Używanie jednolitego schematu nazewnictwa odgałęzień, który odzwierciedlałby równocześnie przeznaczenie każdego odgałęzienia.
5. Uświadomienie sobie semantycznych różnic pomiędzy odgałęzieniem i kodem głównym; dobry projekt jak również sprawna komunikacja mogą przyczynić się do zminimalizowania tych różnic.

Przykładem różnicy semantycznej jest sytuacja, gdy jeden z deweloperów zmienił liczbę (lub, co gorsza, kolejność) parametrów funkcji, podczas gdy inny próbuje wywołać tę funkcję ze starymi parametrami.

6. Unikanie używania plików binarnych tam, gdzie się tylko da, aby umożliwić systemowi CVS automatyczne dokonywanie operacji scalania.
7. Częste scalanie zmian. Im mniej zmian do jednorazowego scalenia z kodem głównym, tym łatwiej jest je scalić.
8. Oznaczanie odgałęzienia przy każdym scalaniu, co pozwala uniknąć ponownego scalania tych samych zmian.
9. Oznaczanie kodu głównego w momencie tworzenia odgałęzienia, jak również tuż przed i po wykonaniu operacji scalenia, na wypadek gdy powstałaby potrzeba ponownego odtworzenia zawartości kodu głównego dla tych momentów.

Używanie jednolitego schematu nazewnictwa znaczników powiązanych z odgałęzieniami.