CVS

(Concurrent Versions System)

Łukasz Chmielewski
Dariusz Działak
Michał Malinowski
Maciej Osiński
(XPlagiator Development Team)

Plan prezentacji

Wstęp czym jest system CVS

Jak to działa jak CVS wygląda od środka

Jak to obsługiwać polecenia i opcje

Przykłady przykłady używania CVS'a

Praktyka ćwiczenia z obsługi CVS'a

Podsumowanie przypomnienie, odnośniki do dalszej dokumentacji

Wstęp

- CVS - co to jest?!
- Czym nie jest CVS
- Co Nam daje CVS
- Inne metody i dlaczego CVS jest lepszy
- Omówienie

CVS - co to jest?!

CVS Concurrent Version System = System Kontroli Wersji

CVS jest narzędziem pomagającym w organizacji tworzenia projektu

CVS pomaga aktualizować dane wielu użytkownikom jednocześnie

CVS sprawnie kontroluje wprowadzanie zmian

CVS ułatwia prowadzenie zespołowego projektu programistycznego

Czym nie jest CVS

- systemem zarządzającym przechowywaniem plików
- twoim kierownikiem
- programem testującym
- programem komunikacyjnym między programistami
- bazą trzymającą dane o błędach programistycznych w projekcie
- narzędziem budującym

Co Nam daje CVS?

- CVS pamięta historię zmian dokonanych podczas realizowania projektu
- Minimalizuje konieczność porozumiewania między członkami zespołu dokonującymi zmiany
- Odporność na wprowadzenia błędnych danych (plików) i usunięcie danych
- Umożliwia pracę grupie ludzi, którzy nie muszą się znać
- Umożliwia sprawną pracę bardziej aktywnym współudziałowcom projektu

Inne metody i dlaczego CVS jest lepszy

- Koordynacja mail'ami
- Trzymanie plików na komputerze szefa grupy (wszystko zależy od niego)
- Trzymanie plików „każdy u siebie", a potem metodą „burzy mózgów" łączenie wszystkiego w całość

Jak to działa?

co to jest repozytorium?

praca z CVS

równoczesne zmiany, łączenie zmian, konflikty

wersje: rewizje, wydania

odgałęzienia - tworzenie, praca, łączenie

Repozytorium

- sposób przechowywania plików, katalogów, prawa dostępu
- specjalne - *CVS, modules, blokady*, inne
- przenoszenie repozytorium, kopie zapasowe
- sposoby dostępu do repozytorium
- zdalny dostęp:
  - rsh
  - bezpośredni dostęp
  - GSSAPI
  - kerbedos

Praca z systemem CVS

- założenie repozytorium, stworzenie modułów
- ściągnięcie modułu
- naniesienie zmian do repozytorium
- uaktualnianie swojej kopii
- dodawanie/usuwanie plików
- dostęp w trybie read-only

Kilka osób zmieniających dany plik

- łączenie zmian (lokalnie lub w repozytorium)
- pamiętaj, że łączenie zmian jest **głupie**
- konflikty - gdy zmiany są blisko siebie (tekstowo)
- aktualizowanie kilku plików nie jest **operacją atomową**
- blokowanie, obserwowanie czy liczenie na szczęście

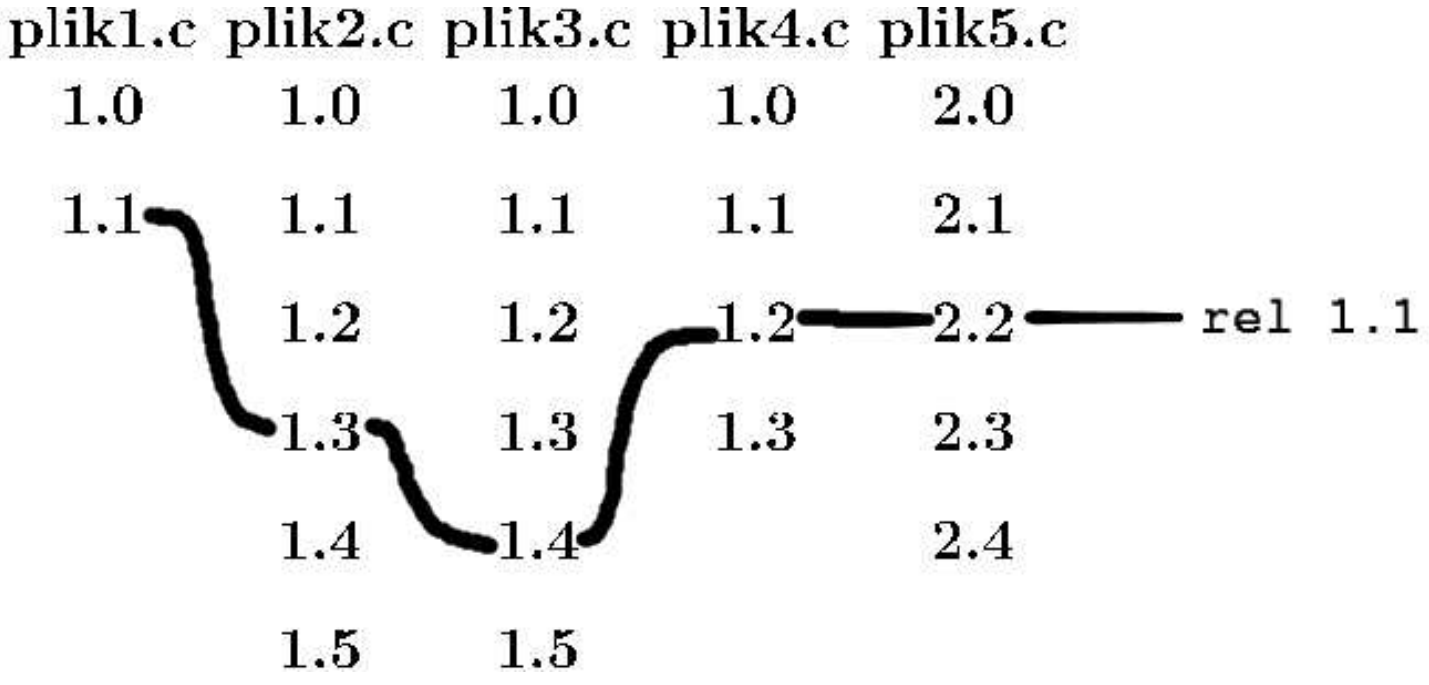
Wersje?

rewizje (revision) - wersje plików

tag - symboliczna nazwa dla rewizji

wydania (release) - „linia łącząca rewizje”

Wydanie (release)



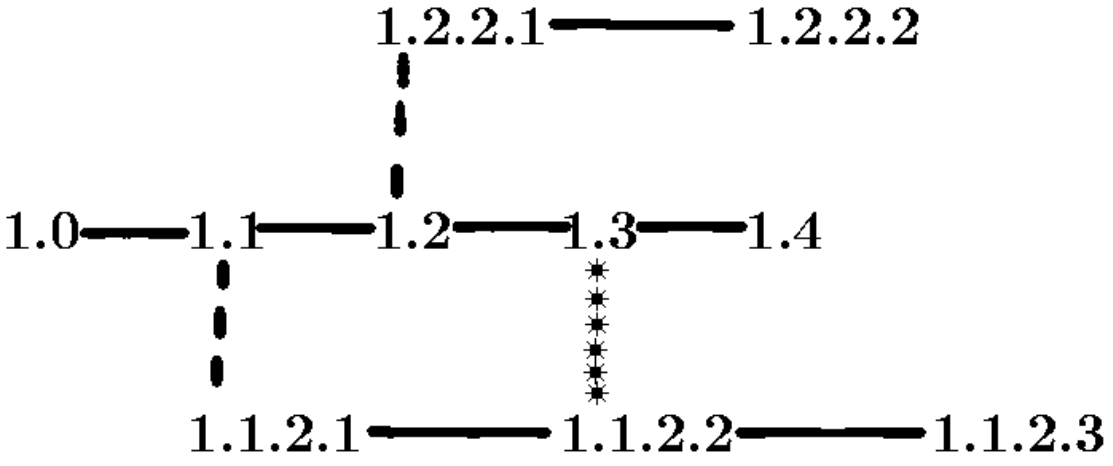
Niezależne ścieżki rozwoju programu

Naprawianie błędów - jeśli w poprzednim wydaniu odnaleziono błędy, a nowe nie jest jeszcze gotowe

Wsparcie dla starych wersji - musimy pielęgnować starszą wersję, choć pracujemy już nad nowszą

Eksperymenty - część zespołu próbuje wdrożyć do systemu innowacyjne zmiany, ale nie chcemy ryzykować uszkodzenia całości; w przyszłości być może włączymy te zmiany

Jak działają odgałęzienia?



Zauważ jak zmieniają się numery rewizji przy tworzeniu odgałęzień.

Obsługa CVS'u

- Prowadzenie projektu przy pomocy CVS'u
- Wysyłanie poprawionych wersji
- Dodawanie i usuwanie plików
- Ściąganie i Sprawdzanie
- Słowa kluczowe
- Przechowywanie plików binarnych

Wstęp

- Podstawowym programem obsługi CVS'u jest program o nazwie *cvs*.
- *cvs* [opcje ogólne] *komenda* [opcje danej komendy] [argumenty]
- Ma on wiele opcji globalnych (niezależnych od komendy), np.: -help, -z gzip-level

Ściąganie plików z repozytorium

cvs checkout opcje moduły pobieranie plików (modułów) z repozytorium

Aliasy: *co* albo *get*

cvs checkout -z9 moduły pobieranie z repozytorium w formie skompresowanej (samo przesyłanie)

cvs checkout -D data moduły pobieranie najnowszej wersji, ale nie nowszej niż zadana data

cvs checkout -p moduły przekierowanie ściągniętych plików na standartowe wyjście

cvs export opcje moduły pobieranie plików z repozytorium (bez tworzenia katalogu roboczego)

Ściąganie plików (uaktualnianie) z repozytorium

cvs update opcje pliki uaktualnianie plików (ściąganie nowszej wersji)

cvs update opcje domyślnie uaktualniane są wszystkie pliki z katalogu bieżącego

cvs update -I nazwa pliki uaktualnienie ignorujące pliki pasujące do nazwy

cvs update -C pliki uaktualnienie wymuszające tworzenie pliku #... (chcemy zgłosić nasze poprawki, ale chcemy zobaczyć oryginalne zmiany innych osób)

wyjścia update i checkout U, P, A, R, M, C, ? nazwa pliku

Dodawanie i usuwanie plików

cvs add pliki/katalogi dodaje pliki lub katalogi do repozytorium

cvs remove opcje pliki usuwa pliki z repozytorium z podanymi opcjami

cvs remove -f pliki nie trzeba usunąć plików (zanim się użyje tej komendy)

cvs remove -P pliki usuwanie również pustych katalogów

cvs import katalog opcje moduły zaimportowanie modułu od zera (nie tworzy katalogu roboczego)

Wysyłanie poprawionych wersji

cvs commit opcje pliki „ogłasza” - zatwierdza nasze zmiany w repozytorium

Aliasy: *ci*

cvs opcje commit domyślnie jest brany bieżący katalog

cvs -m commit "tekst"pliki opis zmian to tekst - nie ma interakcji

cvs -p commit "tekst"plik opis zmian to tekst wzięty z pliku

Konflikt występuje, gdy twoje wersja różni się od tej z repozytorium

Sprawdzanie logów

cvs log opcje pliki wypisuje dane o pliku, m.in. te które wpisuje się przy „commitowaniu”

cvs log -H lub -R pliki okrojone informacje (-H mniej okrojone niż -R)

RCS pliki administracyjne

UTC Coordinated Universal Time

Sprawdzanie róznic pomiędzy plikami

cvs diff opcje pliki pokazuje ścisłe różnice między wersjami

diff przyjmuje większość opcji programu GNU diff

cvs diff -u pliki czytelniejszy format podawanych różnic

cvs diff -r pliki porównywanie podanych wersji rewizji

cvs diff -kk pliki słowa kluczowe nie będą rozwijane

Sprawdzanie statusu plików

status opcje pliki sprawdza status plików

Up-to-date

Locally Modified, Added, Removed

Needs Checkout, Patch, Merge

Unknown

File had conflicts on merge

Słowa kluczowe

$klucz$ lub $klucz:wartość$

Id rozwijane w najważniejsze informacje

Log to co pokazuje komenda *log* (uwaga)

Author autor (osoba, która dodała)

Date data utworzenia

Revision numer wersji

Header wiele danych, np: numer wersji

Przechowywanie plików binarnych

- CVS umożliwia trzymanie plików binarnych
- CVS nie rozwiązuje konfliktów pomiędzy plikami binarnymi
- Należy używać przy wszystkich komendach opcji -kb

Tworzenie odgałęzień

cvs tag -b nazwa tworzy odgałęzienie (wewnątrz repozytorium) o danej nazwie z rewizji z kopii lokalnej

cvs rtag -b baza nowa moduł tworzenie odgałęzienia z rewizji o nazwie *baza* w repozytorium

Pobieranie gałęzi

cvs checkout -r gałąź moduł - ściągnięcie gałęzi modułu

cvs update -r gałąź - aktualizacja z katalogu z modułem (włącza zmiany, jeśli lokalny moduł z innej gałęzi)

Scalanie gałęzi

update -j gałąź plik dołączenie zmian (od stworzenia odgałęzienia) z danej gałęzi do kopii lokalnej pliku

update -j od -j do plik dołączenie zmian pomiędzy danymi rewizjami odgałęzienia

update -j gałąź:data -j gałąź plik jeśli nie pamiętamy numeru rewizji, to może pamiętamy datę ostatniego scalania

update -j 1.5 -j 1.3 plik cofnięcie dołączonych zmian z rewizji 1.3–1.5

update -kk .. keyword substitution

Dodatkowe polecenia

cvs watch on pliki założenie „alarmu” na pliki

cvs watch off pliki usunięcie „alarmu” z plików

cvs edit plik zgłoszenie edycji pliku

cvs unedit plik zgłoszenie zakończenia edycji pliku

cvs realse katalog kończy działalność z danym katalogiem repozytorium

cvs realse -d katalog dodatkowo kasuje dany katalog

Zakładanie repozytorium

mkdir katalog tu powstanie repozytorium

export CVSROOT="katalog" ułatwia dostęp do repozytorium

cvs init inicjalizacja

Dodawanie modułów

cvs import katalog vendor release importuj strukturę katalogów do repozytorium

cvs ckeckout CVSROOT

vim modules dodaj nazwę modułu i katalog w repozytorium

cvs commit modules

Prowadzenie projektu przy pomocy CVS'a

- Przemyślenie struktury katalogów
- Kiedy zatwierdzać zmiany (*commit*) ?
- Używanie *tagów* dla wydań systemu i ważniejszych rewizji plików
- Używanie odgałęzień
- Nie ufać systemowemu algorytmowi scalania
- Uważać na pliki binarne

Podsumowanie

- Program *cvs*
- Wiele opcji poleceń nie zostało zaprezentowanych
- Szczegóły można znaleźć w dokumentacji podanej w podsumowaniu prezentacji

Prosta sesja CVS

```
cvs -d :pserver:ktos@cos.tu.i.tam:/cvsroot login
CVS password:
cvs checkout -r whatever_xp_patch whatever
cd whatever
vim whatever.c
cvs commit -m "Super-Hiper poprawki" whatever.c
```

Trudniejsza sesja CVS

```
cvs checkout whatever
cd whatever
cvs log whatever.c
cvs status whatever.c
rcs2log | less
vim whatever.c
cvs commit whatever.c   ! BŁĄD !
cvs diff whatever.c > whatever.c.diff
cvs update whatever.c
patch < whatever.c.diff
cvs commit -m "Jestem Genialny, już działa" whatever.c
```

Trochę praktyki

- przygotowanie środowiska pracy
- pobieranie plików (modułów) i praca z nimi
- uaktualnianie zmodyfikowanych plików
- blokady i rozwiązywanie konfliktów
- importowanie modułów
- inicjowanie repozytorium *CVS'a* i zakładanie nowych modułów

Przygotowanie środowiska pracy

export CVSROOT=”/home/inf/m/mm189420/cvs” -
ustawiamy ścieżkę repozytorium w którym będziemy pracować (to samo można osiągnąć dodając odpcje: *-d katalog*). Warto dodać inicjację zmiennej *CVSROOT* do skryptów startowych.

Ściąganie plików

pobieramy moduły: *tex* oraz *ala* - dla przypomnienia komenda: *cvs checkout* (lub *cvs get* lub *cvs ci*)

Praca z plikami

sciaga.tex - należy dodać słowa kluczowe:

- *Id*
- *Author*
- *Date*
- *Revision*
- oraz pod tekstem - *Log*

Wprowadzone zmiany zapisać w nowym pliku i dodać ten plik.

ala.txt - dokończyć zdanie „ala ma kota”

Uaktualnianie modułów

zatwierdzamy zmiany w module *tex* - dla przypomnienia komenda: *cvs commit* (lub *cvs ci*)

uaktualniamy moduł *tex*

Rozwiązywanie konfliktów

- zatwierdzamy zmiany w module *ala*
- sprawdzamy różnice między wersją znajdującą się w repozytorium a naszą
- poprawiamy plik *ala.txt*
- ponownie zatwierdzamy zmiany

Importowanie modułów

- pliki w katalogu *tex* importujemy do modułu *latex* - dla przypomnienia komenda: *cvs import*

Inicjowanie repozytorium

- tworzymy katalog *cvs* w domowym katalogu
- export CVSROOT=”$HOME/cvs”
- inicjujemy repozytorium - *cvs init*
- dodajemy katalogi do repozytorium

Organizacja pracy

- ustalić sposób korzystania z *CVS'a*
- stworzyć katalog *CVSROOT* (np. na jednym z kont studenckich)
- założyć moduły (można na bieżąco)
- wszystkie pliki przechowywać w repozytorium
- wszelkie zmiany na bieżąco zatwierdzać
- prowadzić historię zmian

Podsumowanie - cechy systemu

CVS System Zarządzania Wersjami

system scentralizowany - jedno repozytorium na „serwerze”

zwykłe pliki - dane przechowywane w postaci plików

wersje - rewizje plików lub wydania (*release*) programu

odgałęzienia - do równoległego rozwijania kilku wersji programu

„inteligentne” scalanie - system stara się sam łączyć zmiany

Podsumowanie - podstawowe komendy

cvs init założenie repozytorium

cvs add plik dodanie pliku

cvs checkout moduł ściągnięcie modułu

cvs diff plik zmiany pomiędzy wersją w repozytorium a roboczą

cvs update plik zaktualizowanie danych i włączenie zmian

cvs commit -m "komentarz"plik zatwierdzenie zmian

Materiały, dokumentacja

strona domowa http://www.cvshome.org/

po polsku,PLD http://www.pld.org.pl/devel/doc/cvs/

po angielsku http://pl.qmail.org/~ser/doc/cvs.html

nasze materiały http://rainbow.mimuw.edu.pl/~os/cvs/

info cvs w dystrybucjach CVS'a

To już koniec :)-