Systemy Kontroli Wersji

Grzegorz Kulewski Katarzyna Macioszek
Wanda Niemyska Aleksander Zabłocki

ZPP 2005/06
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytetu Warszawskiego

14 listopada 2006 r.

Spis treści

Spis treści

Spis treści

Spis treści

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Krótka historia oprogramowania

- 1969 r. — w Bell Labs powstaje UNIX
- 1972 r. — powstaje język C
- 1973 r. — UNIX przepisano w C
- 1976 r. — powstaje diff
- 1982 r. — powstaje RCS
- 1983 r. — powstaje projekt GNU
- 1985 r. — Larry Wall tworzy patch
- 1989 r. — powstaje CVS
- 1991 r. — powstaje Linux
- 2001 r. — powstaje subversion
- 2003 r. — powstaje wersja Linuksa 2.6
- 2005 r. — powstaje GIT

Wniosek

Wniosek

Systemy kontroli wersji i zarządzania kodem źródłowym powstały kiedy programy przestały być tworzone przez jedną osobę lub przez małe zespoły.
Od tej pory nieodłącznie towarzyszą procesowi powstawania oprogramowania.

Spis treści

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Do czego służy system kontroli wersji?

- przechowuje obecny stan projektu,
- synchronizuje programistów,
- „pokazuje co się zmieniło”,
- automatyzuje wiele czynności programisty,
- zawiera i pozwala przeglądać historię projektu,
- ułatwia zarządzanie wydaniami i wersjami projektu,
- umożliwia tworzenie niezależnych gałęzi projektu,
- pozwala łatwiej znaleźć błędy.

Spis treści

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy open-source

- diff + patch (+ wiggle),
- quilt,
- RCS,
- CVS,
- Subversion,
- GNU Arch,
- Darcs,
- Bazaar,
- Monotone,
- Mercurial,
- GIT.

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Dostępne systemy komercyjne

- ClearCase (IBM Rational),
- Perforce,
- StarTeam (Borland),
- Visual Source Safe (Microsoft),
- Bitkeeper,
- ...

Spis treści

diff

diff

Program diff służy do wygenerowania różnicy dwóch plików lub katalogów.

Przykład

W katalogu `skw` wykonaj następujące polecenie:

```
$ diff -Nru hello hello1
```

albo

```
$ diff -Nru hello hello1 | colordiff
```

diff

diff

Program diff służy do wygenerowania różnicy dwóch plików lub katalogów.

Przykład

W katalogu `skw` wykonaj następujące polecenie:

```
$ diff -Nru hello hello1
```

albo

```
$ diff -Nru hello hello1 | colordiff
```

Spis treści

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —–dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —–dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —–dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

patch

Program patch służy do zakładania patchy.

Przydatne opcje

- –p 1 — decyduje o tym ile elementów obciąć ze ścieżki,
- —dry–run — nic nie robi tylko wypisuje co by zrobił,
- –R — nakłada patcha „odwrotnie”.

Przykład

W katalogu `linux` wykonaj następujące polecenia:

```
$ tar xf linux-2.6.18.tar; cd linux-2.6.18
$ patch -p1 --dry-run < ../patch-2.6.18.1
$ patch -p1 < ../patch-2.6.18.1
```

patch

Przykład 2.

Jeśli dojdziemy do wniosku, że jakiś patch jest nam niepotrzebny:

```
$ patch -R -p1 < ../patch-2.6.18.1
```

Zaaplikuj innego patcha:

```
$ patch -p1 --dry-run < ../patch-2.6.18.2
$ patch -p1 < ../patch-2.6.18.2
```

I „na wierzch” jeszcze innego:

```
$ patch -p1 --dry-run < ../patch-2.6.18-ck1
```

(oj, chyba będą kłopoty...)

Ale zaaplikuj i tak:

```
$ patch -p1 < ../patch-2.6.18-ck1
```

I co powstało?

Spis treści

wiggle

wiggle

Program wiggle służy do nakładania patchy, które nie chcą sie założyć.

Przykład

Jeśli jakiś fragment patcha się nie założył to można użyć programu wiggle. Jednym z możliwych sposobów jego wywołania jest:

```
$ wiggle -r Makefile Makefile.rej
```

wiggle

wiggle

Program wiggle służy do nakładania patchy, które nie chcą sie założyć.

Przykład

Jeśli jakiś fragment patcha się nie założył to można użyć programu wiggle. Jednym z możliwych sposobów jego wywołania jest:

```
$ wiggle -r Makefile Makefile.rej
```

wiggle

Konflikty

Jeżeli wiggle też nie będzie umiał założyć patcha to przynajmniej wskaże w wygodny sposób miejsce, w którym powinien się on znaleźć:

```
<<<<<<<
Jakaś część pliku
|||||||
Tekst do zastąpienia
=======
Tekst, którym należy go zastąpić
>>>>>>>
```

Spis treści

quilt

quilt

Program quilt służy do zarządzania stosami patchy.

Przykład

Będziemy poprawiali program znajdujący się w katalogu `hello`.

```
$ quilt new add-return.patch
$ quilt edit main.c
```

(dodaj `return 0;` do funkcji `main()`)

```
$ quilt diff
$ quilt refresh --diffstat
$ quilt pop
$ quilt push
```

quilt

quilt

Program quilt służy do zarządzania stosami patchy.

Przykład

Będziemy poprawiali program znajdujący się w katalogu `hello`.

```
$ quilt new add-return.patch
$ quilt edit main.c
```

(dodaj `return 0;` do funkcji `main()`)

```
$ quilt diff
$ quilt refresh --diffstat
$ quilt pop
$ quilt push
```

Spis treści

CVS

CVS

Concurrent Version System — zestaw programów do zarządzania wersjami.

Wywołanie

```
$ cvs nazwa-polecenia parametry
```

CVS

CVS

Concurrent Version System — zestaw programów do zarządzania wersjami.

Wywołanie

```
$ cvs nazwa-polecenia parametry
```

CVS

Przykład

Z katalogu `cvs` wywołaj:

```
$ export CVSROOT=`pwd`
$ export EDITOR=vim
$ cvs init
```

CVS

Przykład c.d.

```
$ cd ../hello
$ cvs import hello main start
$ cd ../test
```

CVS

Przykład c.d.

```
$ cvs checkout hello
$ vim hello/hello.c
$ cvs status
$ cvs commit
```

CVS

Przykład c.d.

```
$ cvs tag v1
$ vim hello/plik.c
$ cvs add plik.c
$ cvs remove plik.c
$ cvs update
```

Spis treści

subversion

subversion

Zestaw programów `subversion` to poprawiony CVS.
Używa się praktyczynie tak samo, na ogół wystarczy zmienic `cvs` na `svn`. :-)