1. Úvod do správy verzií

git

Roderik Ploszek

Písanie dokumentov a správa verzií

16.9.2025

Outline

Prednášky a cvičenia

Prednášky: utorok 15:00 – 16:40 v CD-150
Roderik Ploszek
A-208, klapka 459 (konzultácie dohodou)
`roderik.ploszek@stuba.sk`

Cvičenia: streda 17:00 – 17:50 v C-117
Ing. Juraj Paška
D-515 (konzultácie dohodou)
`juraj.paska@stuba.sk`

Miestnosť C-117 je počas rekonštrukcie fakulty presunutá na blok D, druhé poschodie vpravo. Keďže je vstup do miestnosti regulovaný, príďte prosím na cvičenie načas.

Písanie dokumentov a správa verzií

Predmet má tri nesmelé ciele:

- naučiť vás správu verzií pomocou nástroja git
- naučiť vás robiť sadzbu dokumentov systémom LaTeX
- naučiť vás písať technické dokumenty

- Predmet je za 3 kredity. 1 kredit predstavuje 25 až 30 hodín štúdia.
  - 2 hodiny prednášok
  - 1 hodina cvičení
  - cca 4 hodiny týždenne domáce štúdium a príprava
  - cca 10 hodín príprava na skúšku

Komunikačné kanály

Materiály k predmetu `https://uim.fei.stuba.sk/predmet/b-pdsv/`

Rýchla spoločná komunikácia Discord server PDSV (pozvánka na stránke predmetu)

Pomalšia komunikácia `{roderik.ploszek,juraj.paska}@stuba.sk`

Osobne na prednáškach/cvičeniach

Harmonogram semestra

Zmeny harmonogramu sú vyhradené a budú vopred oznámené!

Týždeň	Prednáška	Dátum prednášky	Cvičenie	Dátum cvičenia
1.	1.	16.09.2025	1.	17.09.2025
2.	2.	23.09.2025	2.	24.09.2025
3.	3.	30.09.2025	3.	01.10.2025
4.	4.	07.10.2025	4.	08.10.2025
5.	5.	14.10.2025	5.	15.10.2025
6.	6.	21.10.2025	6.	22.10.2025
7.	7.	28.10.2025	7.	29.10.2025
8.	8.	04.11.2025	8.	05.11.2025
9.	9.	11.11.2025	9.	12.11.2025
10.		výučba ako v pondelok	10.	19.11.2025
11.	10.	25.11.2025	11.	26.11.2025
12.	11.	02.12.2025	12.	03.12.2025
13.	12.	09.12.2025		

Hodnotenie

- semester
  - správa verzií: 2 testy, spolu 30 bodov
    - v 3. týždni semestra za 12 bodov
    - 21.10.2025 na prednáške za 18 bodov
  - písanie dokumentov
    - 19.11.2025 test za 10 bodov
    - veľké zadanie za 20 bodov – odovzdanie do konca semestra
  - na získanie predpokladu na vykonanie skúšky je potrebné zo semestra získať 24 bodov
- skúška:
  - písomná časť skúšky: 20 bodov
  - ústna časť skúšky: 20 bodov

Úvod do vysokoškolského štúdia

Vysokoškolské štúdium 1

Všeobecné rady, neplatí iba pre tento predmet!

- Navštevujte prednášky.[1] Odznejú tam veci, ktoré v prezentáciách nemusia byť.
- Vysoká škola nie je stredná škola. Nemôžete očakávať priebežné skúšanie, ktoré vás *donúti* učiť sa. Musíte sa učiť priebežne, za jeden večer sa na skúšku/test nepripravíte dostatočne.

  Ideálne je po školskom dni si prejsť prebrané učivo, venovať tomu napr. 20 minút.
- Komunikujte s vyučujúcimi. Ak niečo nie je jasné, dohodnite si konzultácie, zostaňte po prednáške sa porozprávať. Ak niečo nie je v poriadku, okamžite to povedzte vyučujúcim, nenechávajte si to až na evaluáciu predmetu.

[1] Za štúdium platíte (alebo platí niekto za vás) 1200 eur ročne.

Vysokoškolské štúdium 2

Sústreďte sa na zisk bodov počas semestra (započítavajú sa do *predpokladu na vykonanie skúšky*, kedysi známeho ako zápočet).

Ak máte dostatok bodov zo semestra, skúšku nie je problém urobiť. Opačne to je problém.

Príklad

Na predmete je 60 b zo semestra a 40 b skúška. Študent sa nepripravoval dobre a zo semestra získal iba 20 b, čo je 33 %. Aby urobil skúšku, musí získať 36 bodov zo skúšky, čo je až 90 %!

Podľa vety o nemennosti výkonu študenta[2] nemôže študent, ktorý podáva počas semestra 33 % výkon (Fx) zrazu na konci zažiť osvietenie a urobiť skúšku na 90 % (B).

Práve kvôli tomu existuje PPVS, teda minimálny počet bodov zo semestra.

[2] možno som si ju vymyslel, ale jej dôsledky boli experimentálne pozorované

Vysokoškolské štúdium 3

- Píšte si poznámky z toho, čo hovorí prednášajúci. Prezentácie sú iba osnova k tomu, čo chce povedať.

  Navyše, písanie poznámok perom zvyšuje mieru zapamätania/porozumenia preberanej látky.[3]
- Používajte literatúru k predmetu. Neznamená to, aby ste prečítali celú knihu, ale aspoň si prelistujte kapitolu k preberanej problematike.

[3] Link na zaujímavú štúdiu k tejto téme:
`https://e-iji.net/ats/index.php/pub/article/view/630`

Správa verzií

Motivácia

- Čo rieši systém riadenia verzií?
- Projekt na počítači – textový dokument, referát, prezentácia, grafický projekt, kresba, 3D model, zvukový projekt, skladba, partitúra, zdrojový kód, softvérový projekt.
- Ako riešite priebežné ukladanie verzií?
- V projekte potrebujete veľkú časť zmazať/prerobiť, ale zároveň chcete zachovať – aktuálny stav pre neskoršie použitie. Ako budete postupovať?

https://pinterest.com/pin/blog--6685099438564178

História projektu

- Mať možnosť uložiť míľniky alebo snímky projektu.
- Každý snímok môže mať metadáta
  - Čas, keby bol urobený
  - Komentár, prečo bol urobený
- Možnosť kedykoľvek sa vrátiť do predchádzajúcej verzie
- Možnosť pozrieť si rozdiely medzi verziami

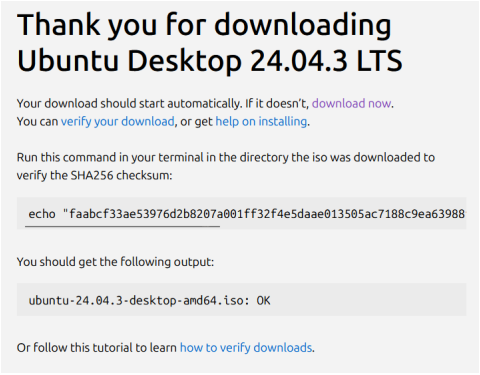
Spolupráca

- Google Docs, Office 365 umožňuje pracovať na jednom dokumente viacerým prispievateľom naraz
- Ako to aplikovať na celý softvérový projekt, ktorý môže mať stovky, tisícky súborov?

Zachovanie integrity

- Chyba prenosu
- Ako viete, že súbor je nepoškodený, keď ho stiahnete z internetu?
- Kontrola cyklickým kódom (CRC), hašovacia funkcia
- Vďaka kontrolným súčtom viete, že všetky dáta v repozitári sú nepoškodené

Po stiahnutí inštalačného média pre *Ubuntu* viete overiť konzistenciu média pomocou hašovacej funkcie SHA256:



Zhrnutie motivácie

- Uloženie histórie s metadátami
- Spolupráca medzi vývojármi
- Integrita projektu

Literatúra

Štandardná literatúra

- Kniha Pro Git: `https://git-scm.com/book/en/v2`
- Používateľský manuál: `https://git.github.io/htmldocs/user-manual.html`

Príručky

Manuálové stránky: `https://git.github.io/htmldocs/git.html`

História správy verzií

- Prehistória
  - SCCS (1972)
  - RCS (Revision Control System, 1982)
  - CVS (Concurrent Versions System, 1986)
- Prelom milénia
  - svn (Subversion, 2000)
  - BitKeeper (2000)
- Nadvláda gitu
  - Git (2005)
  - hg (Mercurial, 2005)
  - bzr (GNU Bazaar, 2005)
  - brz (Breezy, 2017)
- Budúcnosť?
  - Pijul (2016)

- Distribuovaný systém pre správu verzií
- Vytvoril ho v roku 2005 *Linus Torvalds* na správu kódu jadra Linux
- V súčasnosti sa jedná o de-facto štandardný systém pre správu verzií
- Prispela k tomu aj online služba GitHub

Využívanie systémov správy verzií podľa dotazníka Stack Overflow:

Name	2015	2017	2018	2022
Git	69,3 %	69,2 %	87,2 %	93,9 %
Subversion	36,9 %	9,1 %	16,1 %	5,2 %
Team Foundation V. C.	12,2 %	7,3 %	10,9 %	
Mercurial	7,9 %	1,9 %	3,6 %	1,1 %
Concurrent Versions System	4,2 %			
Perforce	3,3 %			
Microsoft Visual SourceSafe		0,6 %		
IBM DevOps Code ClearCase		0,4 %		
Zip file backups		2,0 %	7,9 %	
Raw network sharing		1,7 %	7,9 %	
Other	5,8 %	3,0 %		
None	9,3 %	4,8 %	4,8 %	4,3 %

`https://en.wikipedia.org/wiki/Git#Adoption`

Inštalácia nástrojov

Git for Windows

`https://git-scm.com/downloads/win` – klik na Git for Windows/x64 Setup

Textový editor

Použite svoj obľúbený. Neskôr v semestri budeme používať Visual Studio Code: `https://code.visualstudio.com/`. Môže sa vám hodiť aj neskôr.

Úvodné nastavenie

Príkazy zadávame do príkazového riadku. Použite napr. PowerShell.

Kontrola, či je git nainštalovaný:

```
git --version
```

Prvotné nastavenie, meno a e-mail sú použité na „podpísanie“ commitov, ktoré vytvoríme:

```
git config --global user.name "Meno Priezvisko"
git config --global user.email meno.priezvisko@stuba.sk
```

Vytvorenie repozitára

`git init [<priecinok>]`

`git init` vytvorí nový repozitár v aktuálnom priečinku

`git init repo` vytvorí nový repozitár v priečinku `repo`. Ak priečinok neexistuje, bude vytvorený.

`git init` neprepíše už existujúci repozitár

Vytvorenie repozitára – ukážka

```
$ git init repo
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/Roderik/PDSV/repo/.git/
$ cd repo

$ ls -la
total 4
drwxr-xr-x 1 Roderik None 0 Jul 26 22:50 .
drwxr-xr-x 1 Roderik None 0 Jul 26 22:50 ..
drwxr-xr-x 1 Roderik None 0 Jul 26 22:50 .git
```

Vytvorenie repozitára – vysvetlenie

- Vyvorením repozitára sa vytvorí aj hlavná vetva s názvom master. Pomocný text informuje používateľa o možnosti zmenu názvy vetvy, ako aj zmeny predvoleného názvu hlavnej vetvy.
- Po prejdení do priečinka `repo` a vypísaní jeho obsahu vidíme priečinok `.git`. Sú tam uložené všetky informácie týkajúce sa repozitára, aj obsah súborov, ktoré repozitár sleduje.

git status

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

- Príkaz `git status` zobrazí aktuálny stav repozitára.
  - Aktuálna vetva je `master`.
  - Zatiaľ v repozitárii neexistuje ani jeden *commit*.
  - Nemáme ani čo *commitovať*, pretože žiadne súbory nie sú sledované (*tracked*).

Pridávanie obsahu do repozitára – *staging area*

Aktuálny stav priečinka, v ktorom sú zdrojové súbory

Časti súborov, ktoré sú pripravené na uloženie do histórie

Uchováva všetky zmeny v projekte

Index

Vďaka indexu môžete mať kód v rozpracovanom stave a do histórie uložiť iba podmnožinu z tohto rozpracovaného stavu.

Index má riadkovú granularitu. Do indexu môžete pridať iba časť súboru.

Index sa tiež nazýva staging area a pridanie súborov do indexu staging.

git add

`git add` pridáva do indexu obsah súborov. Inými slovami, pripravujete ním súbory na *commit*.

```
git add [--interactive | -i] [--patch | -p] [--all | -A | [--update | -u]]
        [<pathspec>...]
```

`<pathspec>...` špecifikuje, ktoré súbory budú pridané do indexu

`--interactive | -i` Aktivuje interaktívny mód (ovládaný číslami/príkazmi z klávesnice).

`--patch | -p` Umožní pridať do indexu iba časti súborov

`--all | -A` Pridá do indexu všetky súbory z pracovného priečinka.

`--update | -u` Pridá do indexu iba sledované (*tracked*) súbory (tie, ktoré v histórii už sú).

Špecifikácia cesty (*pathspec*)

- Používa sa nielen v `git add`, ale vo všetkých príkazoch, ktoré špecifikujú cesty k súborom.
- V najjednoduchšej forme sa jedná o doslovnú cestu.
- Súčasťou cesty môžu byť aj špeciálne znaky `*` a `?`.
- `*` predstavuje ľubovoľné znaky, aj lomky. `?` predstavuje práve jeden znak, aj lomku.

Príklady

`git add README.txt` pridá súbor `README.txt`

`git add src/ docs/` pridá priečinky `src` a `docs` (fungovalo by aj bez koncovej lomky)

`git add '*.py'` pridá všetky súbory, ktoré končia na `.py`. Budú pridané všetky súbory bez ohľadu na to, v ktorom priečinku sa nachádzajú. Jednoduché úvodzovky sú dôležité.

`git add 'docs/*.jpg'` pridá všetky obrázky typu `jpg` v podstrome priečinkov `docs`.

git status

Vypíše aktuálny stav repozitára – porovná stav pracovného priečinka, indexu a najnovšieho commitu v histórii.

```
git status [<prepínače>] [--] [<pathspec>...]
```

`<pathspec>...` špecifikuje podmnožinu súborov, pre ktoré sa má status vypísať

`-s | --short` krátky tvar výpisu

`-v | --verbose` vypíše aj zmeny, ktoré sú uložené v indexe

`-u[<mód>] | --untracked-files[=<mód>]` nastavuje výpis nesledovaných súborov. `mód` má tri možné hodnoty:

`no`	nevypíše nesledované súbory
`normal`	vypíše nesledované priečinky a súbory v koreňovom adresári repozitára. Takto funguje `git status` bez prepínača `-u`.
`all`	vypíše úplne všetky nesledované súbory

git commit

Vytvorí nový commit podľa stavu indexu a pridá ho do histórie.

`-a | --all` automaticky pridá do indexu modifikované alebo zmazané súbory (ignoruje nesledované súbory)

`-p | --patch` umožní pridať do indexu iba časti súborov

`--date=<dátum>` umožní zmeniť dátum a čas commitu (štandardne sa použije aktuálny)

`-m <msg> | --message=<msg>` správa ku commitu, popisujúca zmeny

`--amend` nevytvorí nový commit, ale upraví posledný commit podľa zmien v indexe

`--no-edit` spoločne s `--amend`, ak nechceme upraviť správu commitu

Správa ku commitu – *message*

- Správa ku commitu by mala byť krátka a výstižná, aby vývojár vedel rýchlo pochopiť históriu projektu.
- Niekedy ale chceme veľmi podrobne popísať zmenu.
- Preto sa používa takýto formát:

```
krátky popis (do 54 znakov)

dlhý popis
jeden riadok v dlhom popise by nemal prekročiť 72 znakov
...
ľubovoľne dlhá správa
...
```

Je to iba odporúčanie! Každý projekt má svoj vlastný štýl, ktorý je potrebné dodržať.

Tim Pope štýl

```
Capitalized, short (50 chars or less) summary

More detailed explanatory text, if necessary.  Wrap it to about 72
characters or so.  In some contexts, the first line is treated as the
subject of an email and the rest of the text as the body.  The blank
line separating the summary from the body is critical (unless you omit
the body entirely); tools like rebase will confuse you if you run the
two together.

Write your commit message in the imperative: "Fix bug" and not "Fixed bug"
or "Fixes bug."  This convention matches up with commit messages generated
by commands like git merge and git revert.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, followed by a
  single space, with blank lines in between, but conventions vary here

- Use a hanging indent
```

https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project.html#_commit_guidelines

Conventional commits

Populárny formát správ s formálnou špecifikáciou je Conventional Commits.

```
<typ>[voliteľný scope]: <krátky popis> ()

[voliteľné telo commitu] --> tu pôjde dlhý text, 72 znakov na riadok

[voliteľné päty]
```

git log

`git log`

Vypíše zoznam commitov od najnovšieho po najstarší.

`git log <commit1>..<commit2>`

Vypíše zoznam commitov od (`commit1`; `commit2`]

`--oneline` skrátený výpis na jeden riadok

Opakovanie

Čo by ste mali vedieť:

- vysvetliť motiváciu za správou verzií
- čo je to správa verzií a čo je git
- vytvoriť nový repozitár
- ovládať špecifikáciu ciest v gite (pathspec)
- pridať súbory do indexu
- vytvoriť nový commit
- popísať formát git správ
- ovládať `git log`

Domáca úloha

Nainštalujte si nástroje na svoje PC!