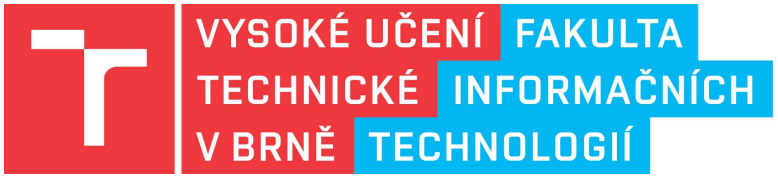
Vývojové nástroje

Základy programování (IZP 2023/2024)

Aleš Smrčka - smrcka@fit.vut.cz



Vývojové nástroje

- Pracovní nástroje programátora
- Verzovací systémy

Kde probíhá vývoj

- Vývoj na lokálním počítači vs. vývoj na vzdáleném počítači
  - Kde jsou uložené / upravované zdrojové kódy
  - Kde je kód spuštěn
- Příklad 1: Skript v jazyku Python
  - Editace a spuštění souboru na lokálním počítači
- Příklad 2: Program v jazyce C
  - Editace souborů, překlad a spuštění na lokálním počítači
- Příklad 3: E-shop v PHP
  - Editace souborů lokálně, přesunutí souboru na server, spuštění na serveru
- Příklad 4: Webový chat v Node.js
  - Editace souborů lokálně, transfer na server, spuštění na serveru i na klientech

Jak probíhá spuštění

- Překladač (compiler) = nástroj pro překlad textových zdrojových souborů (v programovacím jazyce) do binárního jazyka instrukcí
  - Instrukce = operace prováděná buď nativně přímo CPU (HW) nebo jiným SW
- Interpret = nástroj, který „interpretuje“ program - spouští textové zdrojové soubory přímo
- Příklad překládaných jazyků: C, Go, Rust, C++, Java, Ocaml
  - Některé jazyky jsou překládané do bajtkódu, vyžadují SW pro spuštění
- Příklad interpretovaných jazyků: Javascript, Python, PHP, Ruby

Struktura projektu

- Softwarový projekt (zjednodušeně!) = sada souborů, dokumentace, požadavků řešící softwarový problém.
  - V předmětu Základy programování si vystačíme s definicí: projekt = soubory
- Struktura souborů:
  - Soubory jsou strukturované ve složkách.
  - Moduly projektu buď v samostatných složkách nebo vedle sebe.
  - Konkrétní struktura je vyžadovaná pouze u některých programovacích jazyků, avšak platí best-practice.
- Pro začátek:
  - README.md – základní popis
  - src/ - adresář se soubory
  - tests/ - adresář s testy (více viz přednášky o ladění a testování)

Pracovní nástroje programátora

- Editor zdrojových kódů
- Manuál, nápověda, web
- Překladač
- Automatizace překladu a spuštění testů

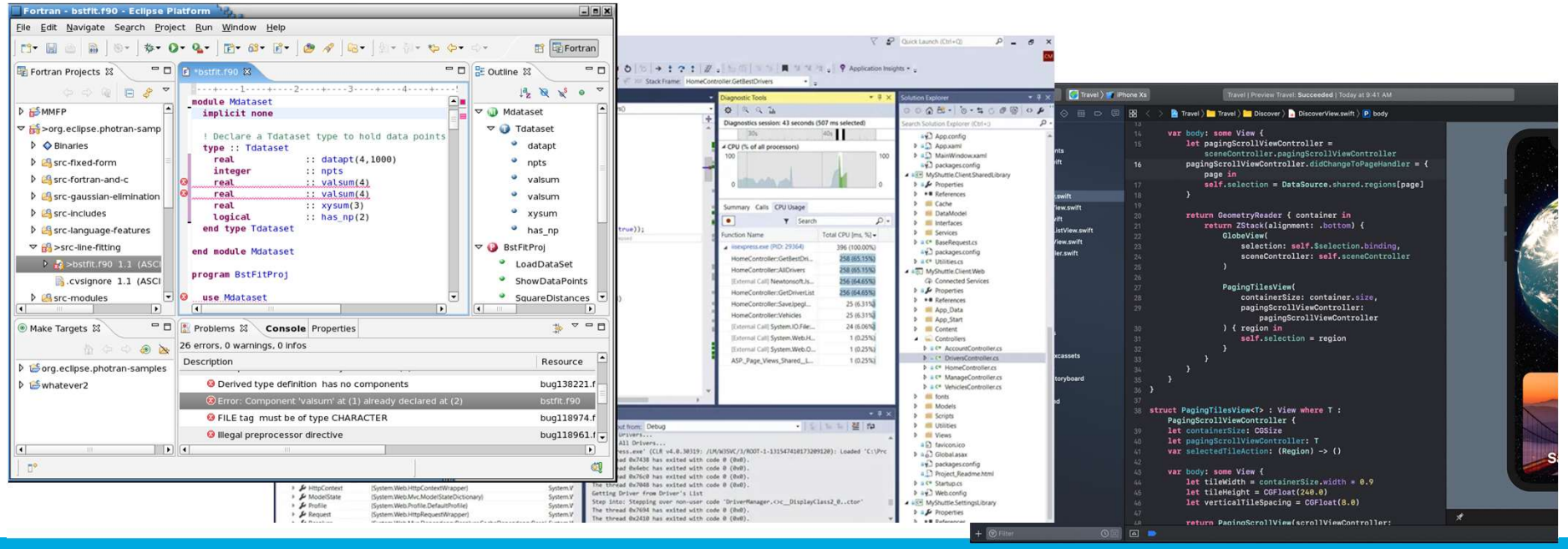
- Hlavní vlastnosti editorů:
  - Zvýraznění syntaxe
  - Automatické zarovnání kódu
  - Úprava více souborů zaráz
  - Zvýraznění chyb (před nebo po překladu)
  - Automatické doplňování (identifikátorů)
  - Skoky (“Jít na“) v kódu (mezi definicí a použitím)

Editory zdrojových kódů

- Ne! notepad, nano, …
- Vim (Vi Improved)
  - Textový editor, rozšířen v unixových systémech.
  - Bez GUI, běží v terminálu (příkazové řádce).
  - Je téměř na všech unixových serverech.
  - Lze upravovat zdrojové kódy na dálku (bez zatížení síťového spojení).
- Notepad++ (MS Windows)
- Visual Studio Code, Atom, …
  - Zvětšující se popularita
  - Modulární, možnost integrace s překladači, interprety, ladicími nástroji

Integrované vývojové prostředí

- Integrované vývojové prostředí:
  - „Velký“ editor, který má integrovanou (úzce spjatou) úpravu zdrojových kódů, překlad, spuštění, ladění, manuál, správu SW projektů, případně i nasazení (deploy).
  - Full-time vývojář stráví v takovém programu klidně celou svoji pracovní dobu.
- Příklady: Eclipse, Visual Studio, Xcode, Code::blocks



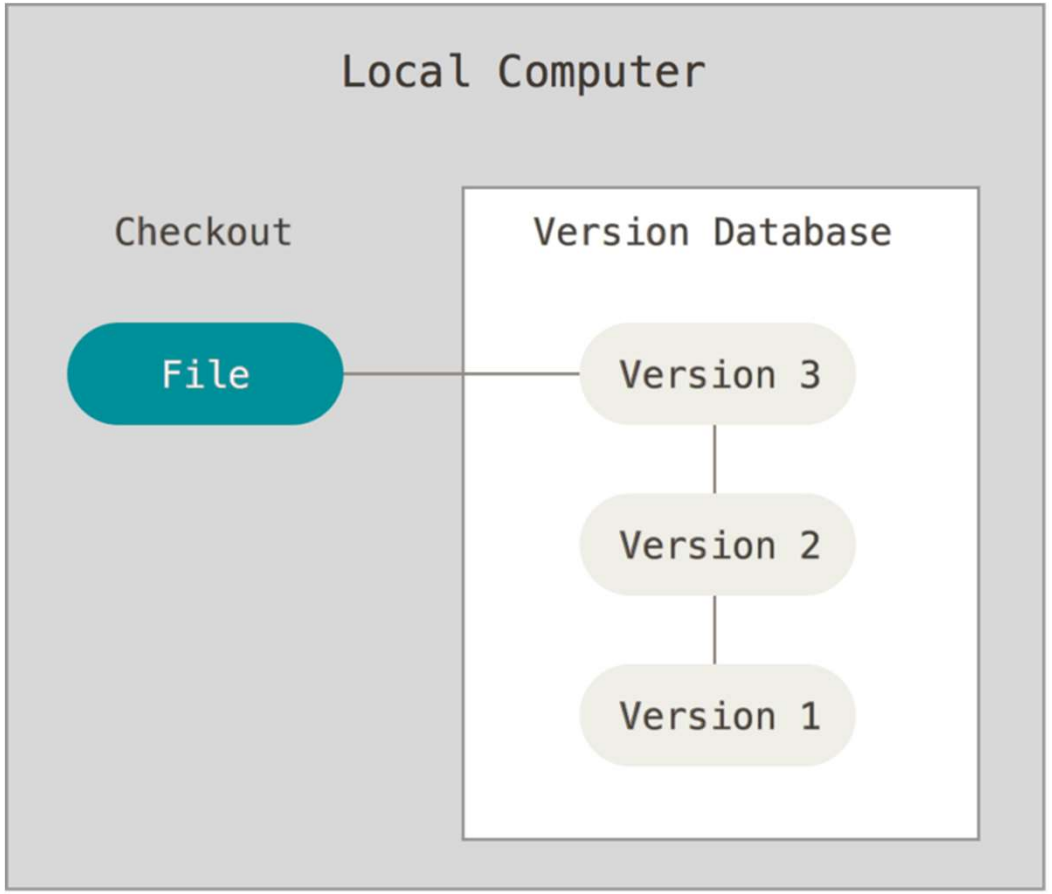
Jak vybrat správný editor

- Neexistuje „ten nejlepší“. Je třeba brát ohled na:
  - Typ projektu
  - Vývojový tým
  - Zásady / strukturu projektu
  - Teprve potom vlastní preference:
    - Zkušenost (znalost nastavení, možností, klávesových zkratek, …)
    - Modularita (ohebnost editoru pro nové vlastnosti související s programovacím jazykem a dalšími typy souborů)
    - Integrovanost (provázanost s překladačem a dalšími podpůrnými nástroji)

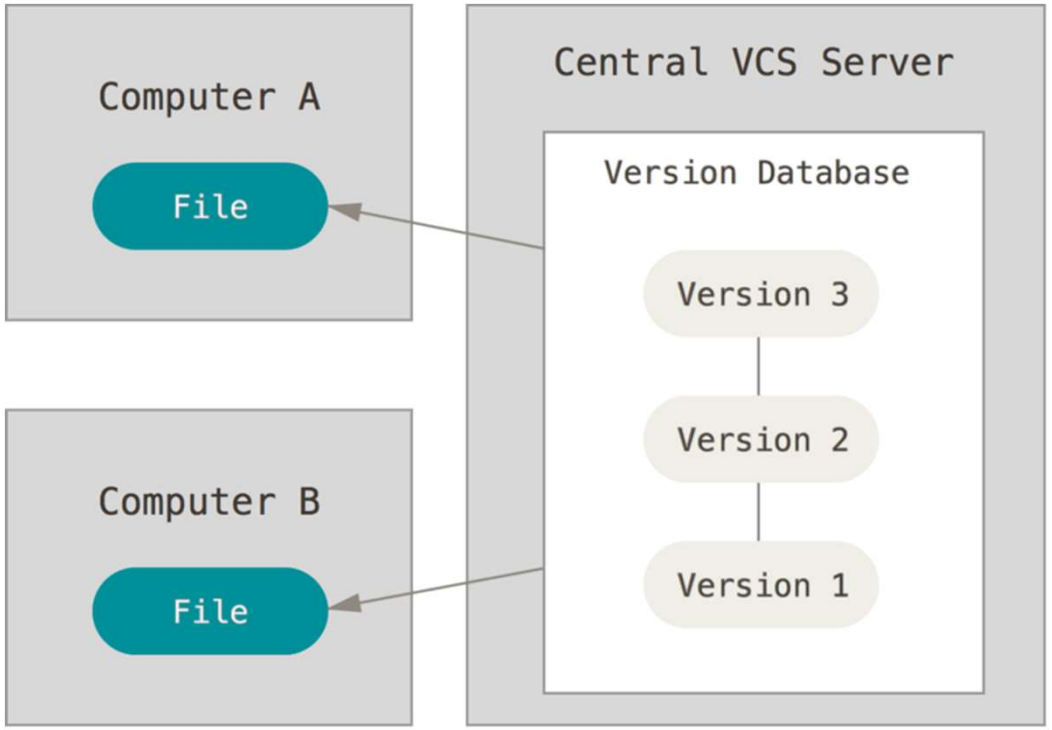
Postupný vývoj projektu

- Postupný vývoj projektu je třeba uchovávat, tvořit tzv. verze nebo revize.
  - Záloha – když se cokoliv pokazí, můžeme obnovit starou verzi
  - Porovnání změn – když se něco nového pokazí, lehce zjistit, proč
  - Různé varianty – nově implementované vlastnosti, různé běhové prostředí, různé závislosti
- Nejjednodušší je *kopie (souborů) projektu*:
  - Příklad: `projekt/*` `projekt-2022-09-15/*` `projekt-2020-07-01/*`
  - Nekopírovat jednotlivé soubory, protože projekt soubory provazuje. Změna jednoho souboru by mohla poškodit celý zbytek projektu.
  - Výhody: rychle provedeno, rychle obnoveno
  - Nevýhody:
    - Nepřehlednost – lze řešit adresářem archiv/
    - Úprava ve starém projektu způsobí zmatek – lze řešit právy
    - Pouze lineární vývoj (bez možností variant s další vývojovou větví), nekolaborativní

- Lokální vývoj probíhá v jednom adresáři projektu.
- Verzovací systém ví o různých verzích projektu.
- Lokální:
  - V adresáři projektu jsou ve speciálním podadresáři uloženy různé verze souborů.
  - Příklad: RCS

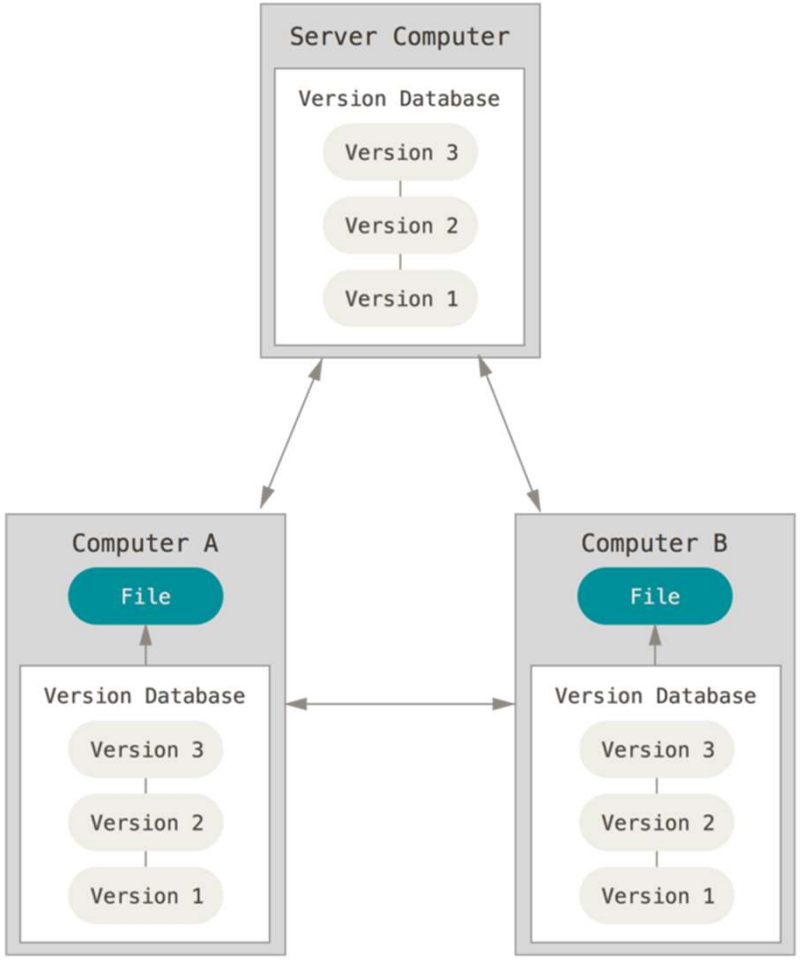


- Centralizovaná správa verzí:
  - Verze jsou ukládány vzdáleně na verzovací server
  - Příklad: CSV, Subversion



- Distribuovaná správa verzí:
  - Každý počítač má uložen klon celé verze.
  - Příklad: **Git**, Mercurial

- Vše o gitu:
  - Scott Chacon, Ben Straub: Pro Git, online (CC BY-NC-CA), https://git-scm.com/book/en/v2



Patch = rozdílové soubory

- Rozdílový soubor (tzv. patch, patch file) slouží k poslání a záznamu rozdílu dvou verzí.
- Používá se k:
  - Návrhu změn, oprav chyb, záplat apod.
  - Přehledné revizi, co se v kódu změnilo, pro schválení a zahrnutí změn,
  - Úspornému uložení několika navázaných verzí souboru.
- Patch **Δ** = Stará_verze – Nová_verze
- Jeden patch soubor může zaznamenat úpravu jednoho nebo více souborů nebo dokonce celého adresářového stromu projektu.

Příklad patch

- Moje soubory:
  - santa.v1/santa.c - originální
  - santa.v2/santa.c - navržená úprava
- Tvorba patch:
  - diff –u santa.c \ ../santa.v2/santa.c >santa.c.patch

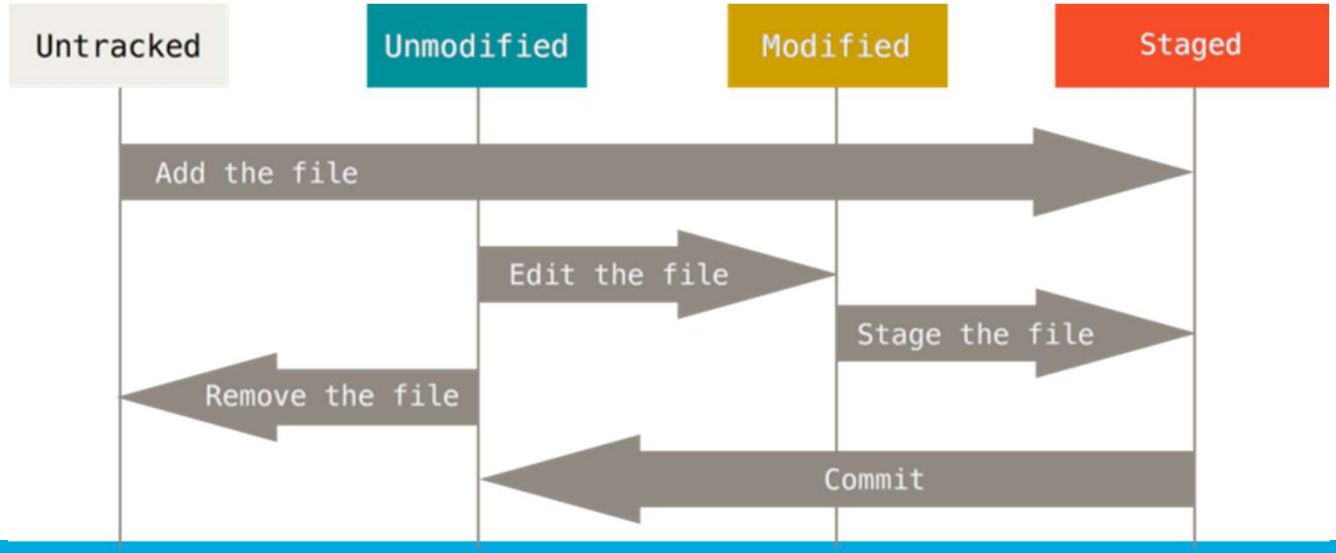
```
diff -u ../santa.v1/santa.c ./santa.c
--- ../santa.v1/santa.c 2022-09-19 10:07:53.725883336 +0200
+++ ./santa.c   2022-09-19 10:08:02.920788825 +0200
@@ -17,6 +17,7 @@
 int *elf_in;
 int *christmass;
 int *get_help;
+int *xmas_time;

 void init_shared_objects() {
    /* init shared memory */
```

- Jeho soubory
  - santa.v1/santa.c
- Aplikace patch:
  - patch <santa.c.patch

Začátky s Git

- Tvorba README.md
  (více v přednášce o dokumentaci)
- Inicializace adresáře .git:
  `git init`
- Naplánování commit:
  `git add README.md`
- Vložení nové revize:
  `git commit -m “first commit“`

- Stáhnutí vzdáleného repa:
  `git clone https://../pr`
- Přidání vzdáleného repa k mému:
  `git remote add origin https://../pr`
- Stažení poslední verze z remote:
  `git pull`
- Nahrání mých verzí na remote:
  `git push [-u origin main]`



Další užitečné příklady

- git config
- git status
- git diff
- git reset
- git branch
- git checkout

- SSH klíče
  - ssh-keygen
    - $HOME/.ssh/id_rsa.pub
  - ssh-copy-id