GIT

O que é

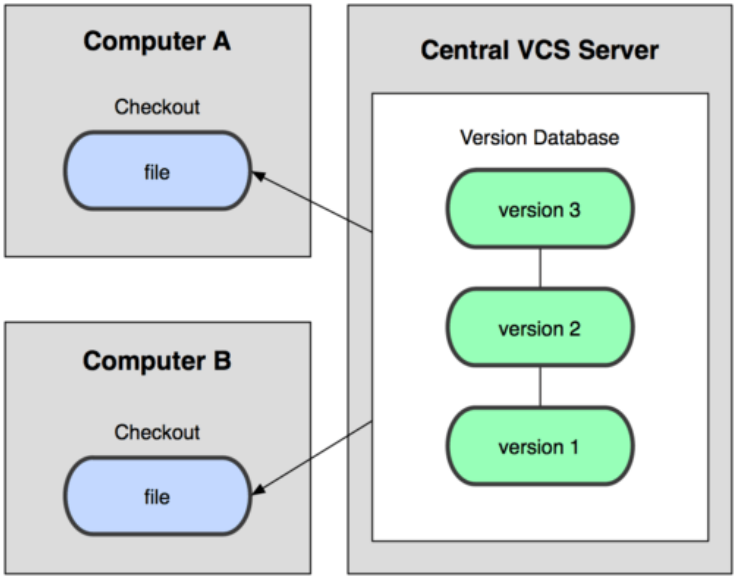
- “a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.”
  - Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- “created by Linus Torvalds in 2005 for development of the Linux kernel,”

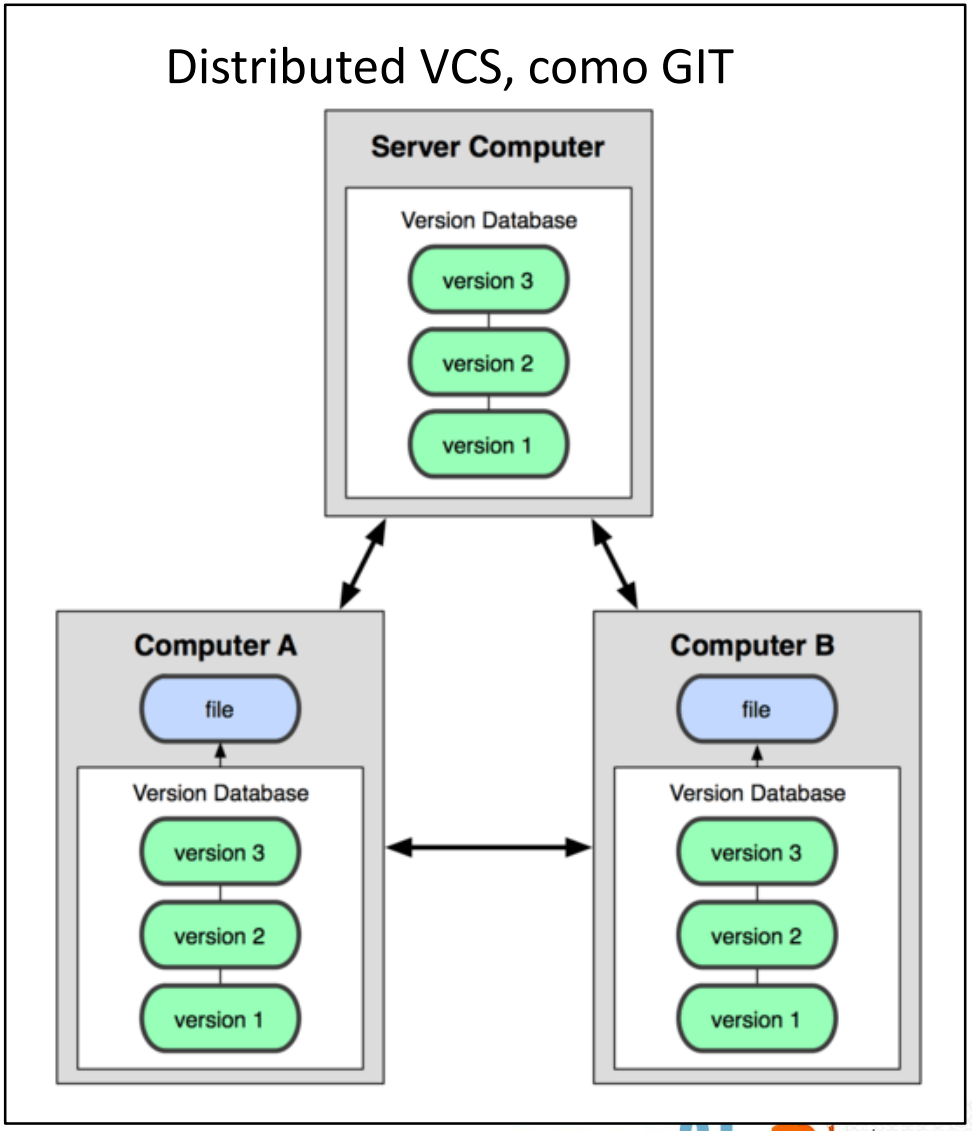
- **Documentação:**
  - https://git-scm.com/book/en/v1/Getting-Started

Conceito

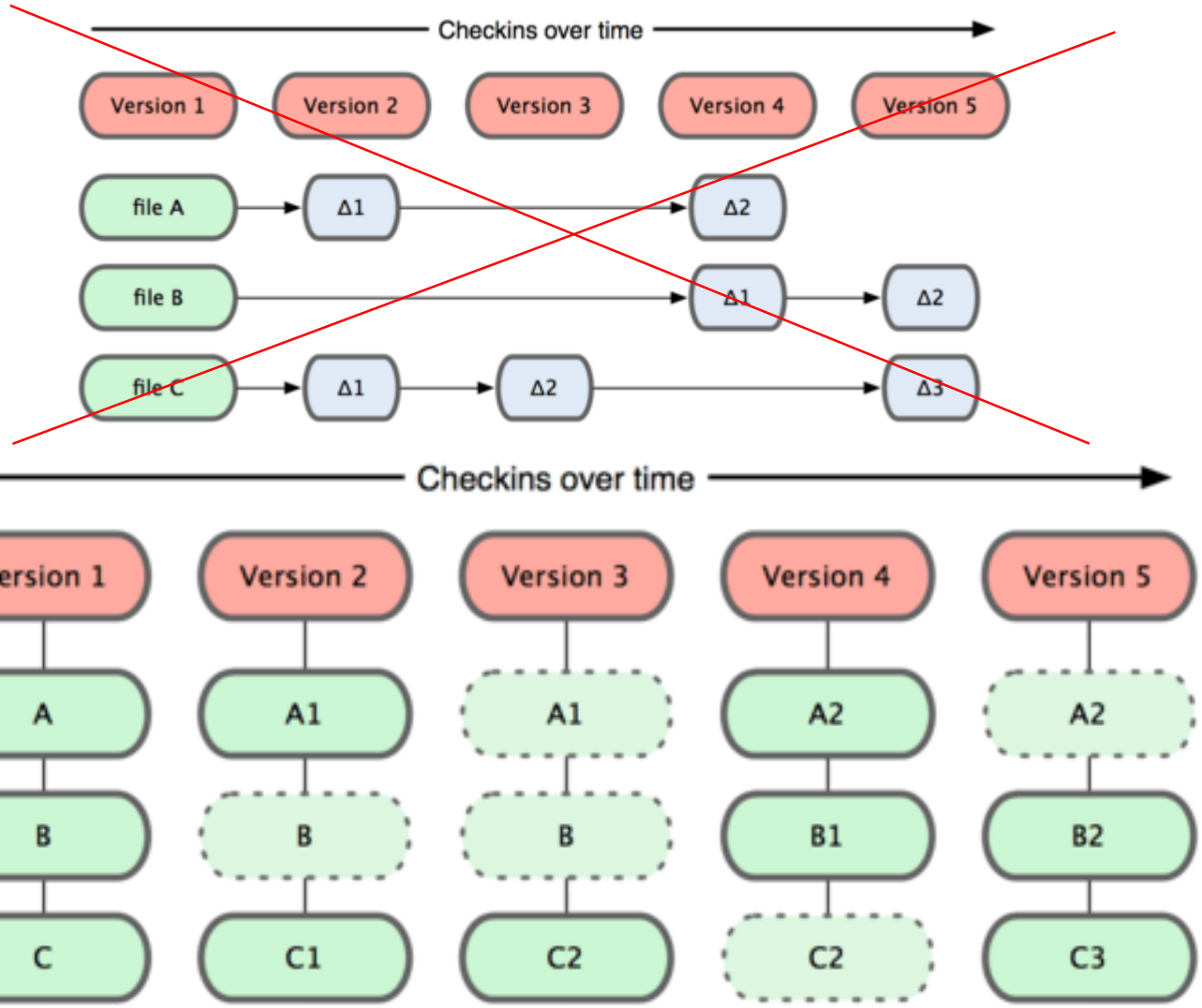
Centralized VCS, como CVS, SVN



Distributed VCS, como GIT

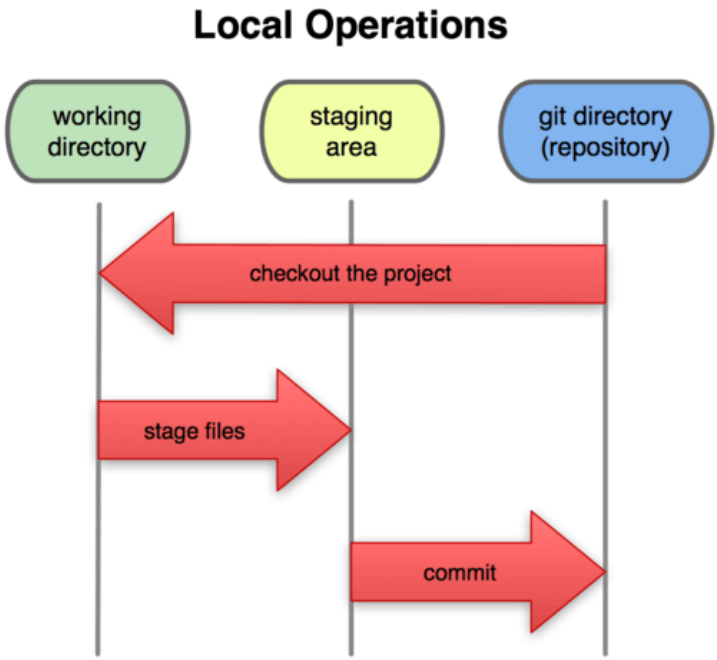


Snapshots, not differences



Local Operations

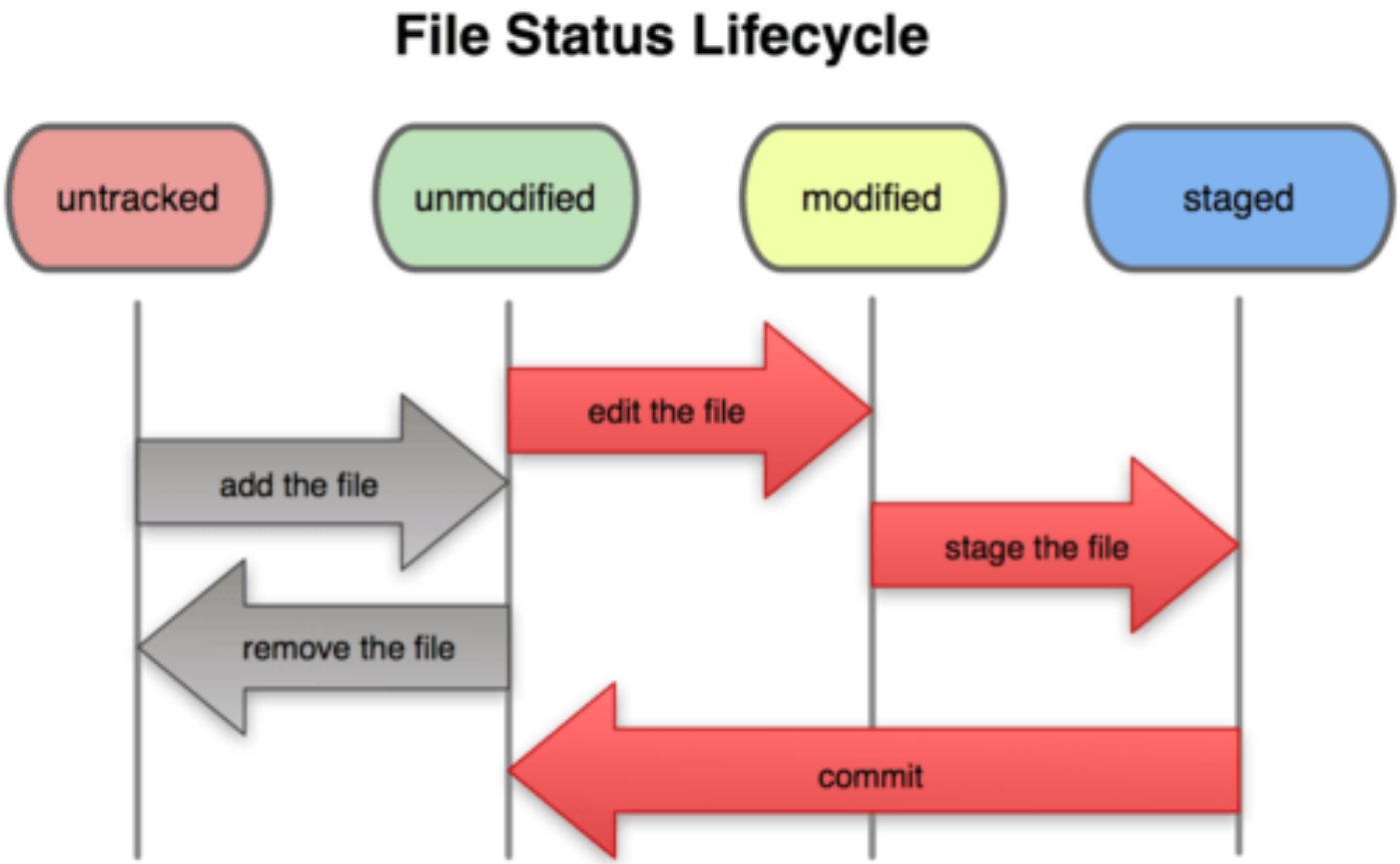
- Nearly Every Operation Is Local
- Git has three main states that your files can reside in:
  - Committed means that the data is safely stored in your local database.
  - Modified means that you have changed the file but have not committed it to your database yet.
  - Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- basic Git workflow goes something like this:
  - You modify files in your working directory.
  - You stage the files, adding snapshots of them to your staging area.
  - You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.



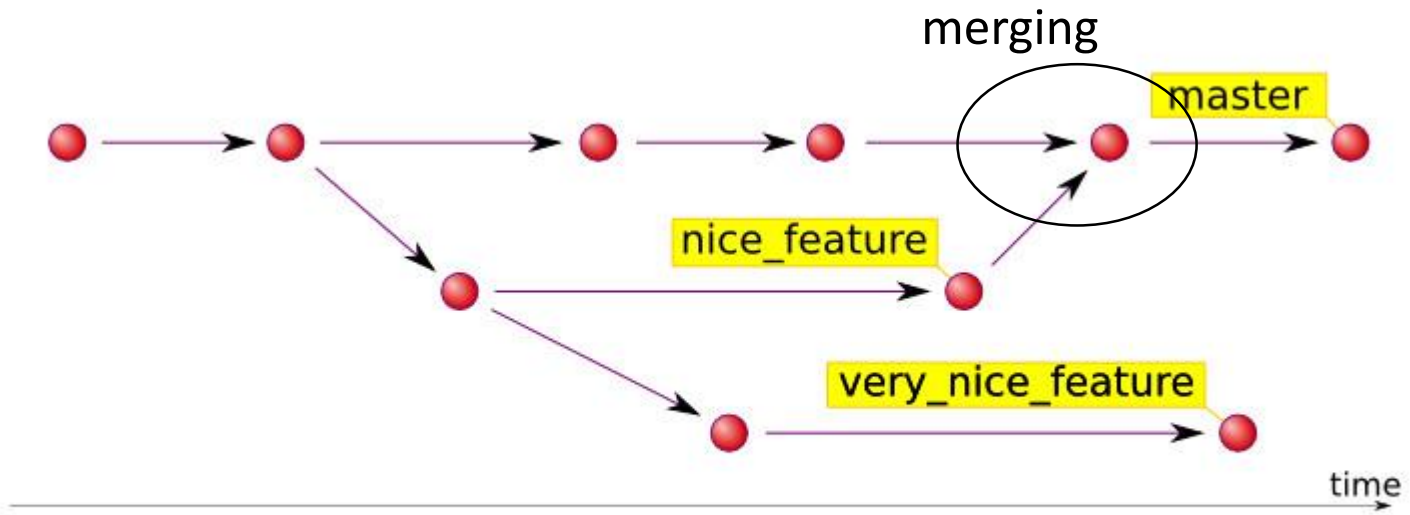
Cloning

- Cloning an Existing Repository
- If you want to get a copy of an existing Git repository
- $ git clone https://github.com/ppgee-simulacoes/intro.git

File Lifecycle



Branching



- Cada aluno deve criar seu “branch”
- Nomenclatura:
  - Dev_nome_sobrenome

Style Guide

Code style guide

- This project adheres to:
  - PEP 8: Style Guide for Python Code
  - PEP 257: Docstring Conventions
- Along with Style Guide, the following useful reading is suggested:
  - https://google.github.io/styleguide/pyguide.html
- Example: naming

```
module_name, package_name, ClassName, method_name,
ExceptionName, function_name, GLOBAL_CONSTANT_NAME,
global_var_name, instance_var_name,
function_parameter_name, local_var_name
```

Example on how to document source code

```
def __insert_text(self, source: str, text: str):
        """
        This method can be called to display a message on the console. The same
        message will be written to the log file and the file will also include
        the name of the class that called the method. By default, all logging
        messages will be written in INFO level.

        Parameters
        ----------
            source : Class name that called the method
            text : Message that will be displayed on console and on log file
        """
        self.__scrolledtext.insert(tkinter.INSERT, text + "\n")
        self.__scrolledtext.see(tkinter.END)
        logger = logging.getLogger(source)
        logger.info(text)
```

Regras Gerais

- Não usar “magic numbers” – criar variáveis
- Nomes de variáveis devem ser compreensíveis
  - Evitar i, k, aux, tmp, ...
  - Nomes em inglês:
    - Snr_index,

Números Pseudo-Aleatórios

Aspectos Gerais de Simulação de Sistemas de Comunicação
Números Aleatórios

- Comportamento dos sistemas reais são aleatórios, com múltiplas variáveis:
  - Começo e duração de chamada/conexão/sessão
  - Canal Rádio móvel (fading)
  - Mobilidade
  - Perfil de tráfego, etc.

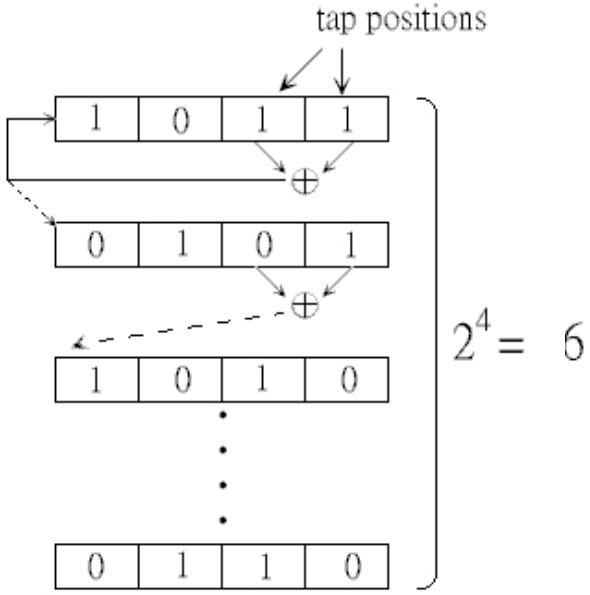
- Em simulações, essas variáveis são modeladas probabilisticamente

- Geradores de números aleatórios são usados para criar valores de acordo com uma dado distribuição de probabilidade (PDF)

Aspectos Gerais de Simulação de Sistemas de Comunicação
Números Aleatórios

- Computadores são máquinas determinísticas

- Na verdade, dispomos de números pseudo-aleatórios
  - Partindo de uma semente (seed) inicial, uma sequência de números pode ser determinado, mas com características aleatórias
  - O uso de uma semente fixa permite que os resultados sejam reproduzíveis
  - Uma semente aleatória pode ser obtida por exemplo a partir do clock do computador
  - E.g., **Geradores de Registradores de Deslocamento:** consideram os bits de uma palavra de computador como os elementos de um vetor binário, na qual, iterativamente, através de transformações lineares geram sequência de vetores binários interpretadas como sequências de números inteiros aleatórios uniformes.



Aspectos Gerais de Simulação de Sistemas de Comunicação
Números Aleatórios

- Partindo de um gerador de números pseudo-aleatórios podemos obter outras distribuições
  - Normal, Exponencial, Poisson podem ser obtidas por transformação de uma distribuição uniforme.

- Existem um conjunto de características que uma seqüência de números pseudo-aleatório deve satisfazer
  - Amostras equiprováveis: igualdade de probabilidade entre os diversos valores;
  - Boas propriedade de autocorrelação: independência, o valor corrente da variável aleatória não tem qualquer relação com os valores anteriores;
  - Seqüência muito longas: o comprimento da seqüência de números aleatórios, antes que números anteriores comecem a repetir-se a si próprios pela ordem anterior;
  - Implementação Eficiente
  - Elas usualmente não é fácil de obtê-las de uma vez!

- Vários algoritmos são encontrados na literatura
  - e.g., in *Press et al., "Numerical Recipes In xxx"* or in *Jeruchim et al., "simulation of Communication Systems"*
  - Nem sempre confie na função `rand()` disponível nas bibliotecas C (é bom verificar se as características da implementação satisfazem os estudos pretendidos)
  - Os números aleatórios do Python são confiáveis.

- RANDOM.ORG offers true random numbers to anyone on the Internet
- Bibliotecas:
  - Numpy.random
  - random

Bibliotecas

- Numpy.random

- Numpy.random.random([size])
  - return random floats in the half-open interval [0.0, 1.0)
  - Funções parecidas em outras linguagem
  - Distribuição uniforme
  - Obtido diretamente do registro de deslocamento

- Outras distribuições podem ser obtidas a partir de uma uniforme
  - Várias outras distribuições disponíveis, lista em
  - https://docs.scipy.org/doc/numpy/reference/routines.random.html

  - Qualquer distribuição pode ser obtida a partir da CDF

Docs

- https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/random-vs-pseudorandom-number-generators
- https://en.wikipedia.org/wiki/Pseudorandom_number_generator

Tarefas

Tarefas

- Criar branch de desenvolvimento
- Implementar gerador de números aleatórios com distribuição normal
  - Usando transformada de Box-Muller
  - Gerar sequência de N=100, 1000 e 10000 amostras
  - Plotar histograma e comparar com PDF esperada
  - Plotar CDF obtida e comparar com CDF esperada
  - Usar sementes fixas diferentes e plotar gráficos

- Implementar gerador de números pseudo-aleatórios para distribuição qualquer e repetir item acima
  - Por exemplo
  - $f_X(x) = \begin{cases} Ax, & 0 < x \leq 10 \\ 0, & \text{caso contrário} \end{cases}$
  - Dica:
    - http://matlabtricks.com/post-44/generate-random-numbers-with-a-given-distribution