分子生物计算
(*Perl 语言编程*)

天津医科大学
生物医学工程与技术学院

2016-2017 学年上学期（秋）
2014 级生信班

第三章　编程的艺术

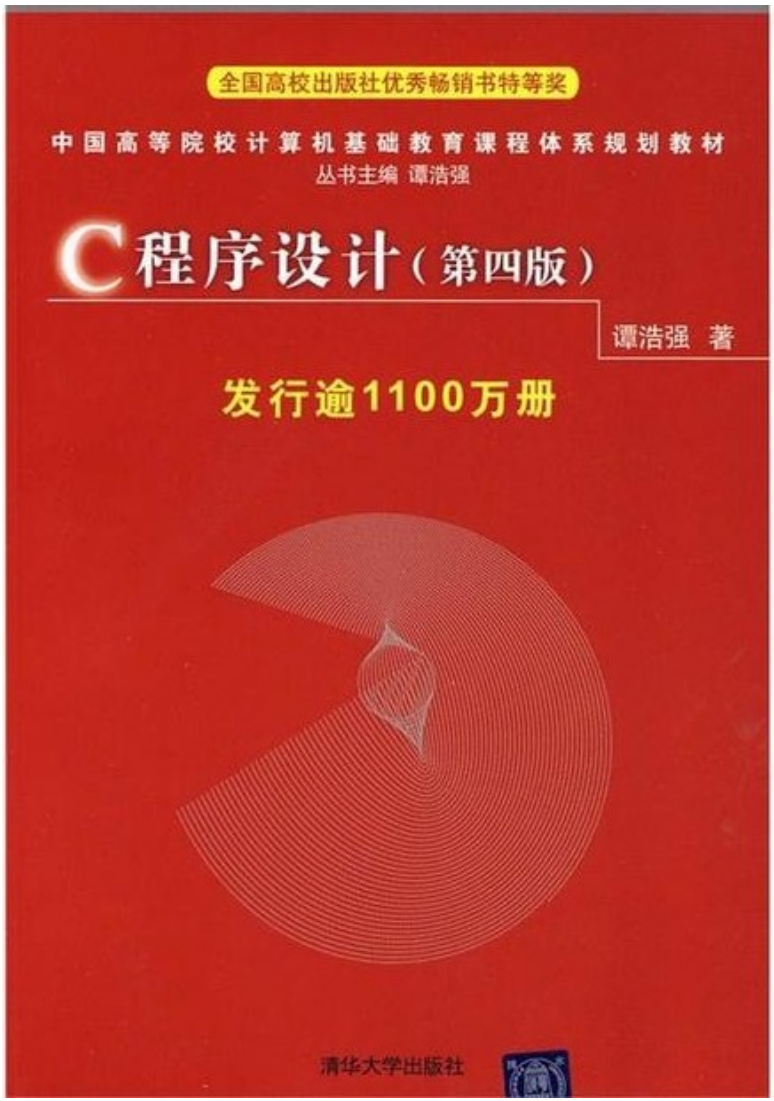
伊现富（Yi Xianfu）

天津医科大学（TIJMU）
生物医学工程与技术学院

2016 年 11 月

教学提纲

教学提纲


全国高校出版社优秀畅销书特等奖
中国高等院校计算机基础教育课程体系规划教材
丛书主编 谭浩强
C程序设计（第四版）
谭浩强 著
发行逾1100万册
清华大学出版社


PROGRAMMER TO PROGRAMMER™
Beginning Unix
Unix 入门经典
（美）Paul Love
Joe Merlino 等著
张楚雄 许文昭 译
wrox
清华大学出版社

教学提纲

提问

学习编程的最佳方法是什么?

回答

- 取决于你要完成的任务
- 取决于你打算如何学习编程
- 取决于……

提问

学习编程的最佳方法是什么?

回答

- 取决于你要完成的任务
- 取决于你打算如何学习编程
- 取决于……

常见方法

- 参加（XXX 新手）培训班
- 阅读（XXX 入门、30 天学会 XXX）书籍
- 死啃手册
- 拜师学艺
- 研究经典程序
- ……
- 组合多种方法

五字真言

- 实践出真知！
- Experience is the best teacher.
- 不要只读书/看手册/读源代码，一定要亲自动手去编写、调试程序。

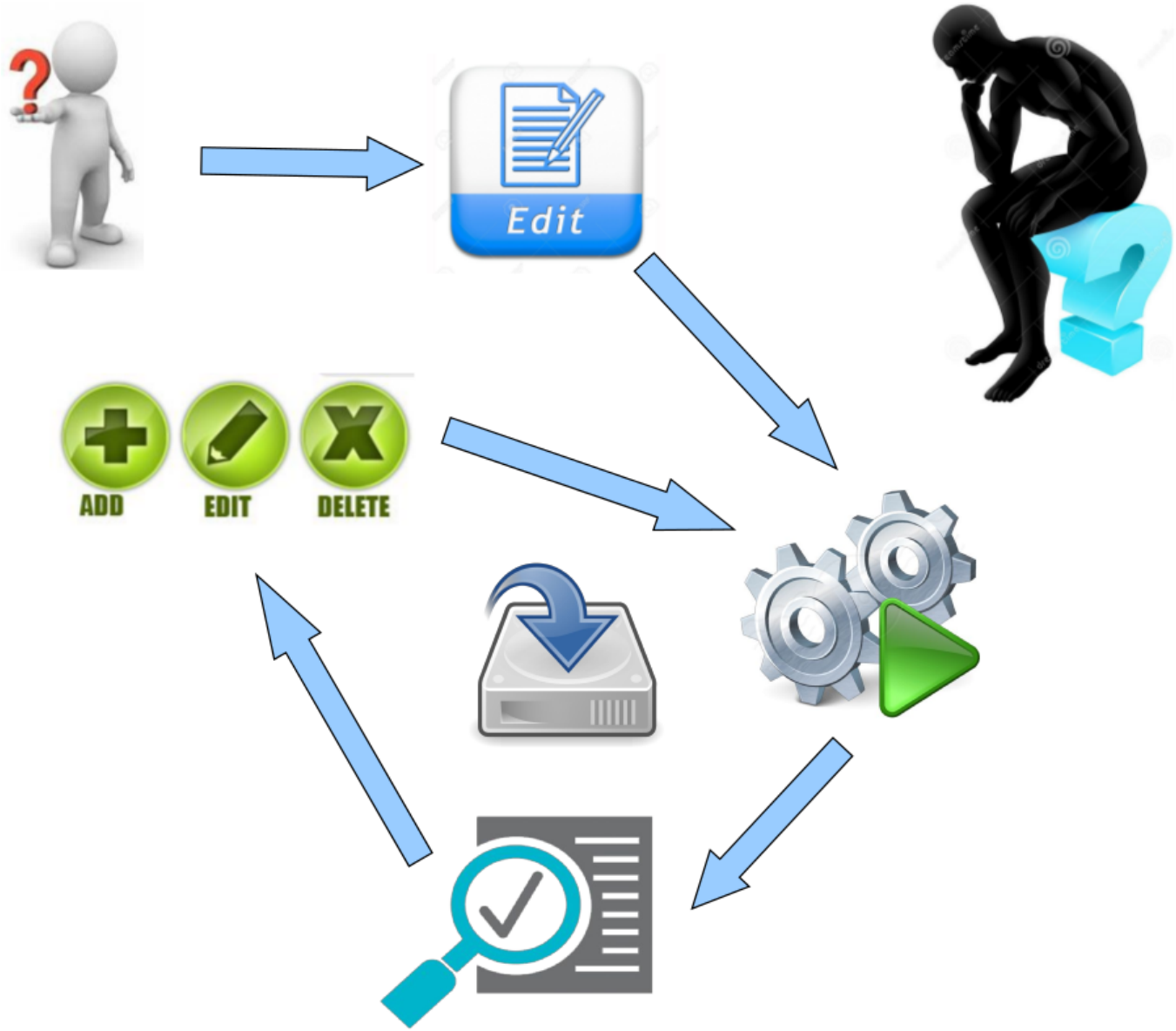
常见方法

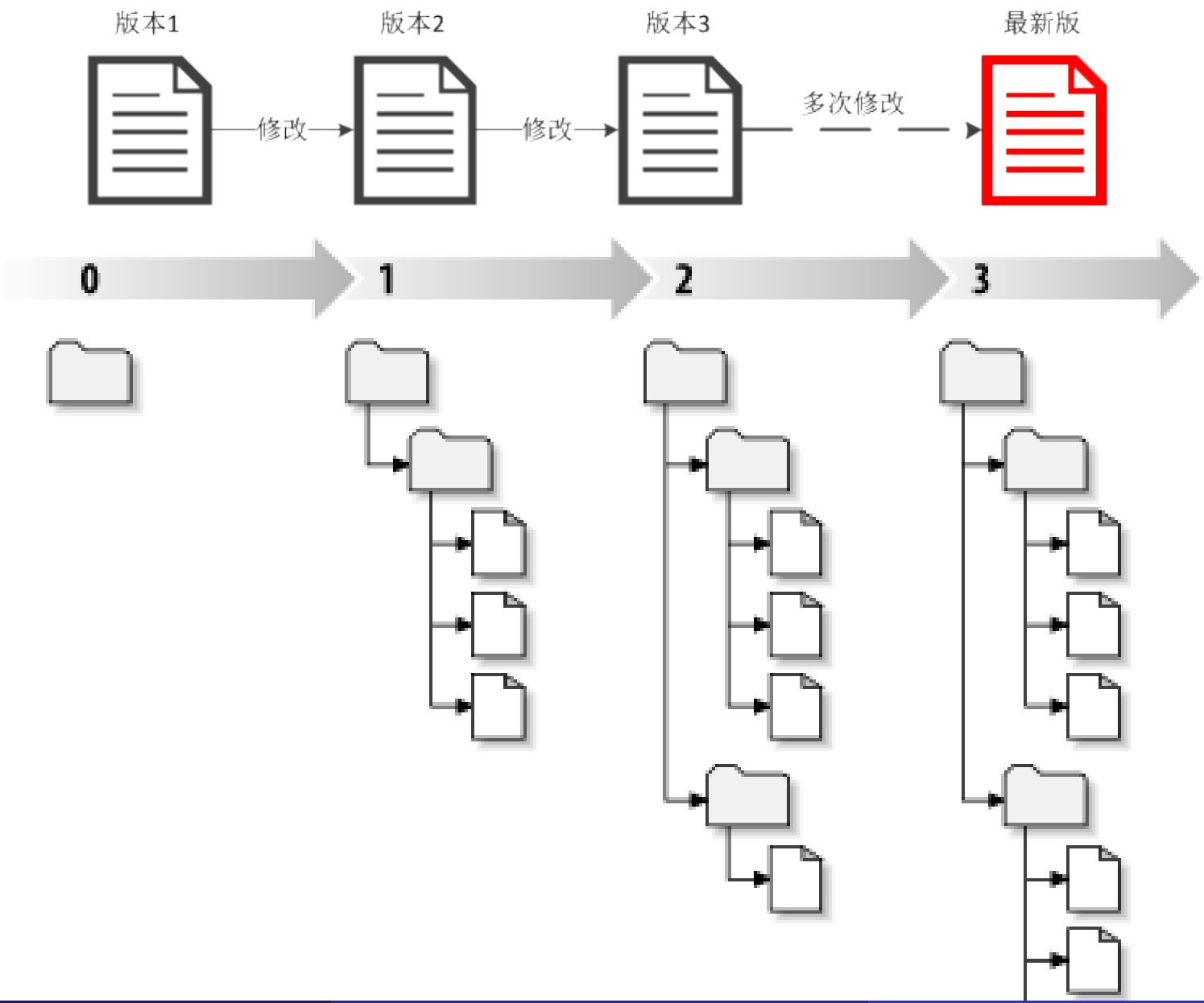
- 参加（XXX 新手）培训班
- 阅读（XXX 入门、30 天学会 XXX）书籍
- 死啃手册
- 拜师学艺
- 研究经典程序
- ……
- 组合多种方法

五字真言

- 实践出真知！
- Experience is the best teacher.
- 不要只读书/看手册/读源代码，一定要亲自动手去编写、调试程序。

教学提纲


Edit
ADD
EDIT
DELETE


版本1
版本2
版本3
最新版
修改
修改
多次修改
0
1
2
3

什么是版本控制？我真的需要吗？版本控制是一种记录若干文件内容变化，以便将来查阅特定版本修订情况的系统。

如果你是位图形或网页设计师，可能会需要保存某一幅图片或页面布局文件的所有修订版本（这或许是你非常渴望拥有的功能）。采用版本控制系统（VCS，Version Control System）是个明智的选择。有了它你就可以将某个文件回溯到之前的状态，甚至将整个项目都回退到过去某个时间点的状态。你可以比较文件的变化细节，查出最后是谁修改了哪个地方，从而导致出现怪异问题，又是谁在何时报告了某个功能缺陷等等。使用版本控制系统通常还意味着，就算你乱来一气把整个项目中的文件改的改删的删，你也照样可以轻松恢复到原先的样子。但额外增加的工作量却微乎其微。

许多人习惯用复制整个项目目录的方式来保存不同的版本，或许还会改名加上备份时间以示区别。这么做唯一的好处就是简单。不过坏处也不少：有时候会混淆所在的工作目录，一旦弄错文件丢了数据就没法撤销恢复。

Git 是一个分散式版本控制软件，最初由林纳斯·托瓦兹（Linus Torvalds）创作，于 2005 年以 GPL 释出。最初目的是为更好地管理 Linux 内核开发而设计。

Git 是用于 Linux 内核开发的版本控制工具。与 CVS、Subversion 一类的集中式版本控制工具不同，它采用了分布式版本库的做法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。Git 的速度很快，这对于诸如 Linux 内核这样的大项目来说自然很重要。Git 最为出色的是它的合并追踪（merge tracing）能力。

在 Git 中的绝大多数操作都只需要访问本地文件和资源，不用连网。因为 Git 在本地磁盘上就保存着所有当前项目的历史更新，所以处理起来速度飞快。

Git 和其他版本控制系统的主要差别在于，Git 只关心文件数据的整体是否发生变化，而大多数其他系统则只关心文件内容的具体差异。

这类系统（CVS，Subversion，Perforce，Bazaar 等等）每次记录有哪些文件作了更新，以及都更新了哪些行的什么内容。

Git 并不保存这些前后变化的差异数据。实际上，Git 更像是把变化的文件作快照后，记录在一个微型的文件系统中。每次提交更新时，它会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照的索引。为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一链接。

```
1  # 安装Git
2  sudo apt-get install git-core
3
4  # 使用帮助
5  man git
6  git --help
7  git help CMD
8
9  # 创建项目目录
10 mkdir ~/project
11 cd ~/project
```

```
1  # 创建Git仓库（启动版本控制）
2  git init
3
4  # 创建编辑文件
5  vim script.pl
6  #print "Hello, world!";
7
8  # 添加需要进行版本控制的文件
9  git add script.pl
10 #git add .
11
12 # 提交改动信息
13 git commit -m "Say hello to the world."
```

```
1  # 修改文件
2  vim script.pl
3  #把Hello替换成Bye
4
5  # 添加改动信息
6  git add .
7
8  # 提交改动信息
9  git commit -m "Bye to the world."
10
11 # 查看提交历史
12 git log
13
14 # 版本回退
15 git reset --hard ID #不需要全部ID，只需要有区分度
     的前几位即可
```

```
1  # 创建test分支并切换过去
2  git checkout -b test
3  #相当于两步: git branch test; git checkout test
4
5  # 修改文件
6  vim script.pl
7  #添加一行: print "Bye, world!";
8
9  # 添加改动信息
10 git add .
11
12 # 提交改动信息
13 git commit -m "And bye to the world."
```

```
1 # 切换回主分支
2 git checkout master
3
4 # 把test分支合并到主分支
5 git merge test
6 #可能需要手动修改后执行git add和git commit命令
7
8 # 删除test分支
9 git branch -d test
```

```
1  # 查看状态
2  git status
3
4  # 查看提交日志
5  git log
6
7  # 查看修改内容
8  git diff
9
10 # 删除文件
11 git rm
12
13 # 查看/创建标签
14 git tag
```

```
1  # 配置Git
2
3  #查看配置信息
4  git config --list
5
6  #彩色的 git 输出:
7  git config color.ui true
8  #显示历史记录时, 只显示一行注释信息
9  git config format.pretty oneline
10
11 #配置个人信息等
12 git config --global user.name "Yixf"
13 git config --global user.email "yixf@example.
     com"
14 git config --global core.editor vim
```

```
1 # 内置的图形化Git
2 gitk
3
4 # 忽略文件/文件夹
5 #.gitignore
6 videos/
7 *.pdf
8 *.doc
```

```
1  # 克隆仓库
2  #克隆本地仓库
3  git clone /path/to/repository
4  #克隆远程服务器上的仓库到本地
5  git clone username@host:/path/to/repository
6
7  # 把本地已有的仓库和服务器上的仓库关联起来
8  git remote add origin <server>
9
10 # 把本地库的内容推送到远程库
11 git push origin master
12
13 # 把远程库的内容更新到本地库
14 git pull
```

Basic Git Workflow Example

Initialize a new git repository, then stage all the files in the directory and finally commit the initial snapshot.

```
$ git init
$ git add .
$ git commit -m 'initial commit'
```

Create a new branch named featureA, then check it out so it is the active branch. then edit and stage some files and finally commit the new snapshot.

```
$ git branch featureA
$ git checkout featureA
$ (edit files)
$ git add (files)
$ git commit -m 'add feature A'
```

Switch back to the master branch, reverting the featureA changes you just made, then edit some files and commit your new changes directly in the master branch context.

```
$ git checkout master
$ (edit files)
$ git commit -a -m 'change files'
```

Merge the featureA changes into the master branch context, combining all your work. Finally delete the featureA branch.

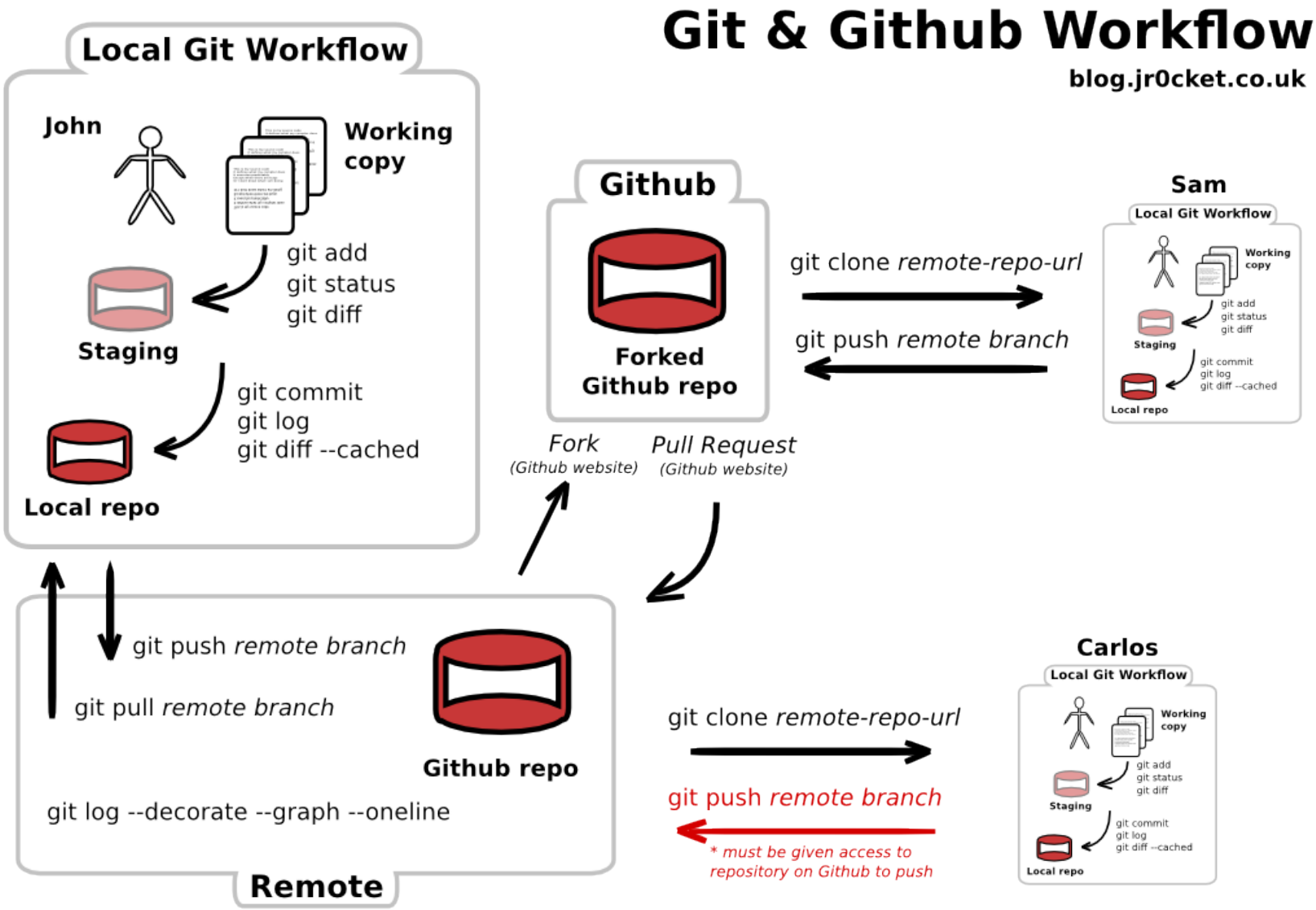
```
$ git merge featureA
$ git branch -d featureA
```

GitHub 是一个共享虚拟主机服务，用于存放使用 Git 版本控制的软件代码和内容项目。

GitHub 同时提供付费账户和免费账户。这两种账户都可以建立公开的代码仓库，但是付费账户也可以建立私有的代码仓库。除了允许个人和组织建立和存取代码库以外，它也提供了一些方便社会化软件开发的功能，包括允许用户跟踪其他用户、组织、软件库的动态，对软件代码的改动和 Bug 提出评论等。GitHub 也提供了图表功能，用于显示开发者们怎样在代码库上工作以及软件的开发活跃程度。

截止到 2015 年，GitHub 已经有超过九百万注册用户和 2110 万代码仓库。事实上已经成为了世界上最大的代码存放网站。

GitHub 里面的项目可以通过标准的 Git 命令进行访问和操作。同时，所有的 Git 命令都可以用到 GitHub 项目上面。


Git & Github Workflow
blog.jr0cket.co.uk
Local Git Workflow
John
Working copy
git add
git status
git diff
Staging
git commit
git log
git diff --cached
Local repo
Github
Forked Github repo
git clone remote-repo-url
git push remote branch
Sam
Local Git Workflow
Working copy
git add
git status
git diff
Staging
git commit
git log
git diff --cached
Local repo
Fork
(Github website)
Pull Request
(Github website)
git push remote branch
git pull remote branch
Github repo
git log --decorate --graph --oneline
Remote
git clone remote-repo-url
git push remote branch
* must be given access to repository on Github to push
Carlos
Local Git Workflow
Working copy
git add
git status
git diff
Staging
git commit
git log
git diff --cached
Local repo

- 出错并不可怕！（【编程初期】出错是非常正常的。）
- 一定不要对错误信息视而不见！
- 从第一个错误开始，逐个进行修复。
- 必要时进行一定的猜测。

- 使用 Perl 调试器：`perl -d script.pl`。
- 在程序中加入 print 语句，输出中间值。
- 选择性地注释掉部分代码。
- 使用相关的模块：Benchmark，Data::Dumper，Smart::Comments，……
- 组合使用多种调试方法
- ……

教学提纲

1. 寻找现成的（免费/收费）程序：避免“重复发明轮子”
2. 自己编写程序
   1. 修改现成的程序（平时注意收集、整理程序）
   2. 充分利用已有模块，快速“拼凑”程序
   3. 从头编写完整的程序
3. 请其他专家（无偿/有偿）编写程序

注意

有时修改现成的程序可能会比从头编写一个完整的程序还要困难！

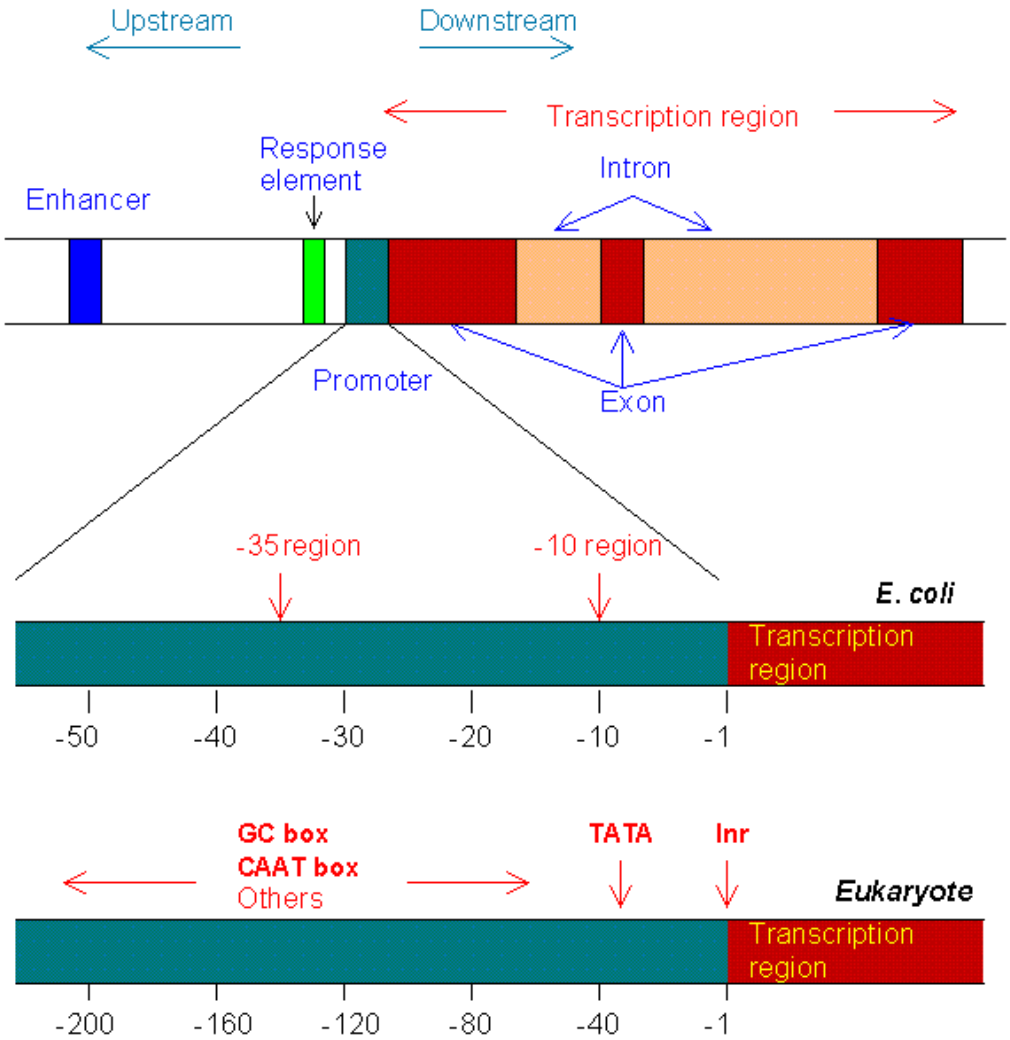
1. 寻找现成的（免费/收费）程序：避免“重复发明轮子”
2. 自己编写程序
   1. 修改现成的程序（平时注意收集、整理程序）
   2. 充分利用已有模块，快速“拼凑”程序
   3. 从头编写完整的程序
3. 请其他专家（无偿/有偿）编写程序

注意

有时修改现成的程序可能会比从头编写一个完整的程序还要困难！

教学提纲

计算一个 DNA 序列中调控元件的数目。



- 分析任务属性，对其进行充分的理解（数量、频率、时间等）
- 确定输入数据，对其进行充分理解（数据量、格式等）
- 对程序进行整理构思（算法、数据结构等）
- 确定输出数据，包括输出方法（文件、图形化展示、管道）、格式等
- 进一步改善整体构思，根据输入、输出等信息添加细节内容
- 必要时编写伪代码整理思路
- 最后才是动手编写程序代码

- 程序构思：在实际编程前首先要完成的关键步骤
- 分析任务属性：任务数量、任务频率、解决任务的时间限制等
- 确定输入数据：数据来源（文件、输入等）、数据数量、数据校验（文件存不存在、格式对不对）等
- 选择正确/合适的算法（速度、优劣）：针对每一个调控元件，在 DNA 序列中从头到尾进行查找；针对 DNA 序列的每一个位置，对每个调控元件进行查找
- 确定输出数据：输出形式、数据格式、人性化输出（用户提供文件名、易于解读……）等
- 选择编程范式，命令式编程（把一个大的问题/程序分割成多个微小、但却相互关联配合的部分/子程序）：程序式编程，面向对象编程
- 编写伪代码：整理思路、优化构思、调整细节……

在数学和计算机科学/算学之中，算法（algorithm）为一个计算的具体步骤，常用于计算、数据处理和自动推理。精确而言，算法是一个表示为有限长列表的有效方法。算法应包含清晰定义的指令用于计算函数。

算法中的指令描述的是一个计算，当其运行时能从一个初始状态和初始输入（可能为空）开始，经过一系列有限而清晰定义的状态最终产生输出并停止于一个终态。

程序所做的事情：获取文件、打开文件、读入数据、进行计算、输出结果；而算法就是此过程中计算的思路。

伪代码（pseudocode），又称为虚拟代码，是高层次描述算法的一种方法。它不是一种现实存在的编程语言；它可能综合使用多种编程语言的语法、保留字，甚至会用到自然语言。

它以编程语言的书写形式指明算法的职能。相比于程序语言，它更类似自然语言。它是半形式化、不标准的语言。我们可以将整个算法运行过程的结构用接近自然语言的形式（这里可以使用任何一种作者熟悉的文字，例如中文、英文，重点是将程序的意思表达出来）描述出来。使用伪代码，可以帮助我们更好得表述算法，不用拘泥于具体的实现。

人们在用不同的编程语言实现同一个算法时意识到，他们做出来的实现（而非功能）很不同。程序员要理解一个用他并不熟悉的编程语言编写的程序，可能是很困难的，因为程序语言的形式限制了程序员对程序关键部分的理解。伪代码就这样应运而生了。

当考虑算法功能（而不是其语言实现）时，伪代码常常得到应用。计算机科学在教学中通常使用伪代码，以使得所有的程序员都能理解。

伪代码是介于自然语言和编程语言之间的一种“中间语言”。

代码

```
1 sub getanswer {
2   print "Type in your answer here :";
3   my $answer = <STDIN>;
4   chomp $answer;
5   return $answer;
6 }
```

伪代码

```
1 getanswer
```

伪代码是介于自然语言和编程语言之间的一种“中间语言”。

代码

```
1 sub getanswer {
2   print "Type in your answer here :";
3   my $answer = <STDIN>;
4   chomp $answer;
5   return $answer;
6 }
```

伪代码

```
1 getanswer
```

```
1 get the name of DNAfile from the user
2
3 read in the DNA from the DNAfile
4
5 for each regulatory element
6   if element is in DNA, then
7     add one to the count
8
9 print count
```

- 注释是源代码的一部分，旨在帮助用户/程序员理解程序
- 从# 开始到行末的所有内容都被看做是注释，会被 Perl 解释器忽略掉
- 首行的`#!/usr/bin/perl` 不是注释，不会被 Perl 解释器忽略掉
- 注释内容：程序的目的、整体构思、使用实例、细节注释等
- 牢记：代码不止是被计算机看的，也会被人查看
- 可以通过注释掉伪代码把它们保留在程序中

教学提纲

教学提纲

知识点

- 学习编程的方法：培训班、读书、看手册、拜师、研究程序、……
- 编程的基本流程：编辑-运行-修正
- 版本控制：Git，GitHub
- 调试程序：调试器、print、注释、模块、……
- 编程策略：找现成的程序、自己编写程序、请别人帮忙、……
- 编程的基本步骤：构思（输入、算法、输出）、编程
- 编程前：伪代码；编程中：注释

技能

- 熟练应用编程的基本策略、步骤和流程
- 能够使用 Git 和 GitHub 进行版本控制
- 能够用不同的方法调试程序

知识点

- 学习编程的方法：培训班、读书、看手册、拜师、研究程序、……
- 编程的基本流程：编辑-运行-修正
- 版本控制：Git，GitHub
- 调试程序：调试器、print、注释、模块、……
- 编程策略：找现成的程序、自己编写程序、请别人帮忙、……
- 编程的基本步骤：构思（输入、算法、输出）、编程
- 编程前：伪代码；编程中：注释

技能

- 熟练应用编程的基本策略、步骤和流程
- 能够使用 Git 和 GitHub 进行版本控制
- 能够用不同的方法调试程序

教学提纲

1. 总结学习编程语言的方法。
2. 编写程序的基本流程是什么?
3. 如何使用 Git 进行版本控制?
4. 总结调试程序的方法。
5. 总结常用的编程策略。
6. 编程的基本步骤是什么? 需要构思哪些内容?
7. 使用伪代码和注释有哪些优势?

下节预告

编程相关

回顾 shell/Perl 中的以下知识点：

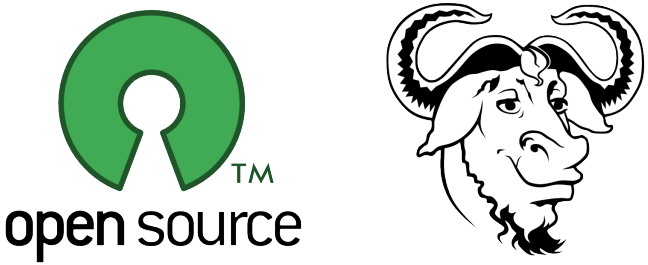
- 变量
- 数组

生物学相关

回顾生物学中的以下知识点：

- DNA 的组成
- DNA 的转录
- DNA 的反向互补
- 蛋白质的组成

Powered by



TEX LATEX XETEX Beamer