NEMU 框架选讲 (1): 编译运行

蒋炎岩

南京大学　　计算机科学与技术系　　计算机软件研究所

本讲概述

本次课程

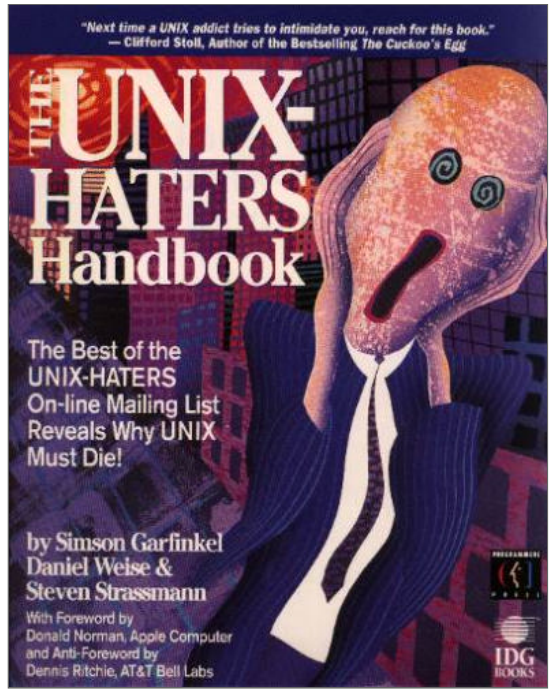
- Git, GitHub 与代码仓库
  - Git 选讲
  - Git 在实验课中的应用
- 项目的构建
  - Lab
  - NEMU
  - AbstractMachine

Git, GitHub 与代码仓库

The UNIX-Hater's Handbook (and Beyond)

写于 1994 年

- Simson Garfinkel 的主页有**电子版**
  - 说有道理也有道理
  - 说没道理也没道理
- 至少指出了 UNIX 的一些缺陷
  - user friendly
  - 命令行/系统工具的缺陷
- 但今天 UNIX/Linux 已经成熟多了！



The Community Way

开源社区 (百度百科): 根据相应的开源软件许可证协议公布软件源代码的网络平台，同时也为网络成员提供一个自由学习交流的空间。

从 GitHub 获取代码

- 传统工具链 + “现代” 编程体验
  - 其实我们很想用 Github Classroom
    - ~~但国内网络实在不靠谱……~~


```
git clone -b 2020 https://github.com/NJU-ProjectN/ics-pa ics2020
git clone https://github.com/NJU-ProjectN/ics-workbench
```

The Community Way (cont'd)

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers. (不愧为全球最大的~~同性~~交友网站)

无所不能的代码聚集地

- 有整个计算机系统世界的代码
  - 硬件、操作系统、分布式系统、库函数、应用程序……

学习各种技术的最佳平台

- 海量的文档、学习资料、博客 (新世界的大门)
  - 提供友好的搜索
    - 例子：awesome C

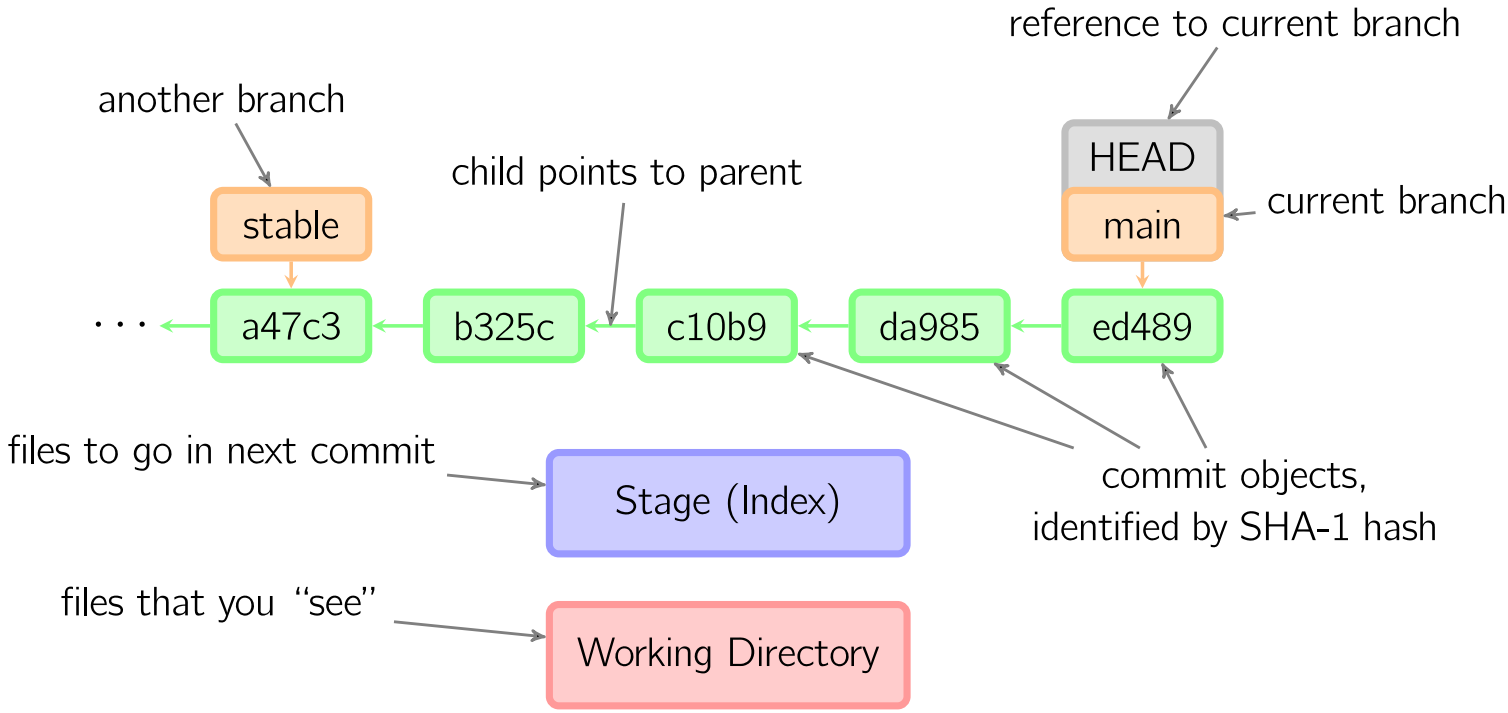
学习 Git？

> 内心独白：~~我[redacted]，又要我学一个新东西，劳资不要~~

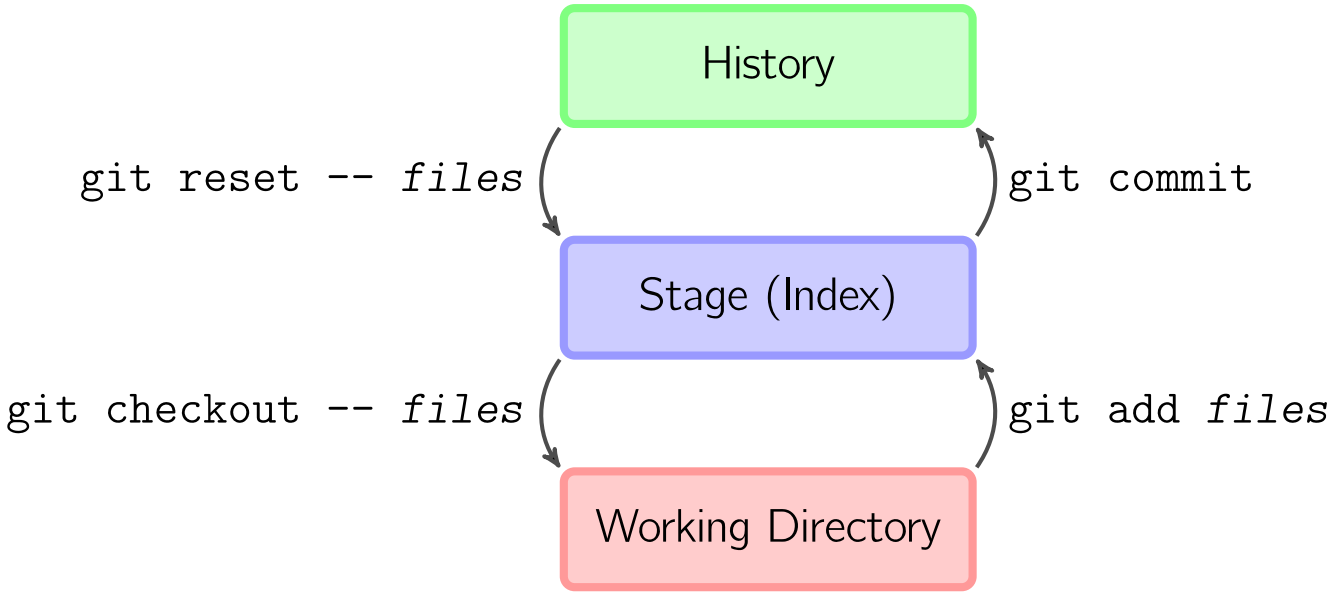
RTFM? STFW!

- 百度：得到一堆不太靠谱的教程
  - ~~请修改 hosts 永久屏蔽百度~~
- 大家已经见识过开源社区的力量了
  - **A Visual Git Reference**
  - 好的文档是存在的
    - 还记得 tldr 吗？

A Visual Git Reference



A Visual Git Reference (cont'd)



一些 Comments

有趣的 “`--`”

- UNIX 的设计缺陷 (UGH 中点名批评)
  - 虽然是编程语言，但 Shell 更贴近自然语言
  - 有很多 corner cases
    - 如果有一个文件叫 “`-rf`”……怎么删除它？？？
    - best practice: 文件名不以 “-” 开头、不含空格/符号……

体验 Git

- 创建一个新的 repo，自由探索
  - 为什么 “预计完成时间 XX 小时” 是骗人的？
    - ~~预计完成时间是假设你在大一开始就用 Git 的~~
- **Visualizing Git Concepts with D3**

一些 Comments (cont'd)

我们使用了“白名单” `.gitignore` 文件

- 只在 Git repo 里管理 `.c`, `.h` 和 `Makefile`
  - 基本原则：一切**生成**的文件都不放在 Git 仓库中

```
*        # 忽略一切文件
!*/      # 除了目录
!*.c     # .c
!*.h     # ...
!Makefile*
!.gitignore
```

- 为什么 `ls` 看不到这个文件？
  - 怎么还有一个 `.git`

提交脚本

make submit 会下载执行 http://jyywiki.cn/static/submit.sh

```
bash -c "$(curl -s http://jyywiki.cn/static/submit.sh)"
```

```
# submit.sh (服务器)
COURSE=ICS2020
MODULE=$(git rev-parse --abbrev-ref HEAD | tr '[a-z]' '[A-Z]')
NAME=$(basename $(realpath .))
FILE=/tmp/upload.tar.bz2

cd .. && \
tar caf "$FILE" "$NAME/.git" $(find $NAME -maxdepth 1 -name "*.pdf'
curl -F "stuid=$STUID" -F "stuname=$STUNAME" \
  -F "course=$COURSE" -F "module=$MODULE" \
  -F "file=@$FILE" http://jyywiki.cn/upload
```

- 我们做了什么?

Git 追踪

另一段神秘代码 (来自 Makefile.lab)

```
git:
  @git add $(shell find . -name "*.c") \
    $(shell find . -name "*.h") -A --ignore-errors
  @while (test -e .git/index.lock); do sleep 0.1; done
  @(hostnamectl && uptime) | \
    git commit -F - -q --author=... --no-verify --allow-empty
  @sync # 《操作系统》课程为大家揭秘
```

在每次 make 执行时

- git 目标都会被执行
  - 将 .c 和 .h 添加、强制提交到 Git repo

这是什么？

对抄袭代码的一种威慑

- **如何抓抄袭**
  - 旧方法；但至今仍在使用
- Git 追踪是“一劳永逸的新方法”
  - 透过它，我们看到南大学子的人生百态
    - 记得 academic integrity
    - 我们留了足够多的分数给努力尝试过的同学通过
- ~~我们正在研制终极加强版~~

小结：现代化的项目管理方式

Git: 代码快照管理工具

- 是一种“可持久化数据结构”
- 拓展阅读：**Pro Git**

框架代码中的两处非常规 Git 使用

- 提交脚本
  - 仅上传 `.git`；在服务器执行 `git reset`
  - 减少提交大小 (仅源文件)
- Git 追踪
  - 编译时强制提交，获取同学编码的过程

思考题：如何管理自己的代码快照？

- 提示：分支/HEAD/... 只是指向快照的指针 (references)

项目构建

Make 工具

回顾：**YEMU 模拟器**

- Makefile 是一段 “declarative” 的代码
  - 描述了构建目标之间的依赖关系和更新方法
  - 同时也是和 Shell 结合紧密的编程语言
    - 能够生成各种字符串
    - 支持 “元编程” (`#include`, `#define`, ...)

Lab 代码的构建

顶层 (top-level) Makefile:

```
# := -> C #define
       NAME   := $(shell basename $(PWD))
export MODULE := Lab1

# 变量 -> 字面替换
all: $(NAME)-64 $(NAME)-32

# include -> C #include
include ../Makefile
```

Lab 代码的构建 (cont'd)

构建目标

- 总目标
  - `.DEFAULT_GOAL := commit-and-make`
  - `commit-and-make: git all` (all 在顶层 Makefile 中定义)
- 可执行文件
  - `multimod-64:gcc -m64`
  - `multimod-32:gcc -m32`
- 共享库 (之后的 lab 使用)
  - `multimod-64.so:gcc -fPIC -shared -m64`
  - `multimod-32.so:gcc -fPIC -shared -m32`
- `clean`
  - 删除构建的代码

NEMU 代码构建

Makefile 真复杂

- ~~放弃~~
  一个小诀窍
- 先观察 make 命令实际执行了什么 (trace)
- RTFM/STFW: make 提供的两个有用的选项
  - `-n` 只打印命令不运行
  - `-B` 强制 make 所有目标

```
make -nB \
  | grep -ve '^\(\#\|echo\|mkdir\)' \
  | vim -
```

NEMU 代码构建 (cont'd)

嘿！其实没那么复杂

- 就是一堆 `gcc -c` (编译) 和一个 gcc (链接) 诶
  - 原来大部分 Makefile 都是编译选项

Read the friendly source code!

AbstractMachine 代码构建

更长，更难读

- 但我们给大家留了一个小彩蛋
- “现代”的文档编写方式
  - “docs as code”
  - 例子：LLVM 使用 Doxygen **自动生成文档**

```
### *Get a more readable version of this Makefile* by `make html` (
html:
	cat Makefile | sed 's/^\([^#]\)/    \1/g' | \
	  markdown_py > Makefile.html
.PHONY: html
```

总结

关于《计算机系统基础》习题课

教会大家“计算机的正确打开方式”

- 编程 ≠ 闷头写代码
- 使用工具也是编程的一部分
  - version-control systems: git, svn, ...
  - build systems: make, cmake (C++), maven (Java), ...
  - shell: bash, zsh, ...

基本原则：任何感到不爽的事情都一定有工具能帮你

- 如果真的没有，自己造一个的就会就来了
  - (不太可能是真的)
  - 但这将会是一份非常好的研究工作

End.

(RTFM & RTFSC)