https://github.com/lazyparser/becoming-a-compiler-engineer

方舟·编译技术入门与实战

第九课：语法制导翻译

吴伟（@lazyparser）

2020-01-12

PLCT Group, part of ISRC, ISCAS & HelloGCC Community & HelloLLVM Community

本课程所有资料都是开源免费的（更新中）

- 课程配套代码及幻灯片地址（有不明白的地方就开issues问）
  - https://github.com/lazyparser/becoming-a-compiler-engineer
  - **https://github.com/lazyparser/becoming-a-compiler-engineer-codes**
- 课程视频回看（包含所有直播及录播视频）
  - https://space.bilibili.com/296494084
- 课程直播地址（可以弹幕或评论区互动）
  - https://live.bilibili.com/10339607

后续课程开始分叉到《编译器设计》（EaC）

- 两本教材：《编译器设计》（EaC）和《现代编译原理》（虎书/tiger）
- 后续默认使用《编译器设计》（EaC）的技术内容讲解
- 虎书内容作为辅助，穿插进行讲解
- 两本教材最好都购买一本放在手边

第二次问卷调查的结果（不要停下来啊！）

- 填写问卷人数比第一次问卷调查减少了1/3（同学们不要走啊～～）
- 接近八成受访者看完了前四课，不到两成完成了语法部分编程作业
- **习题课对于超过1/3的同学有帮助作用（感谢助教）**
- 一半同学没写编程作业，做了的同学普遍觉得写起来吃力
- 刚开始做觉得吃力是很正常的：即使小如PL/0，语法也有点复杂的。

如何掌握编译器开发技术？or 如何应对挫败感？

- 这是一门工匠技术，也就是说，一定要动手自己写代码实现
- 编译器即使很小的语言，组合起来**复杂度**也是可观的，所以
- 动手之前先建立一定的计划/预期（时间，任务阶段，试错）
- 测试用例驱动开发，先能解析一个最小的程序示例开始
- 试错是多步骤的，随手记录自己的尝试路线，让自己能回溯很重要

刚开始吃力是正常的，参考下课代表们的作业

- https://github.com/lazyparser/becoming-a-compiler-engineer-codes
- **提供6个分支，分别对应六份不同的编程实现**

```
git clone https://github.com/lazyparser/becoming-a-compiler-engineer-codes
# 可以看到很多不同助教独立的代码提交
git branch -av
# 切入某一个助教的代码，开始查看，例如
git checkout sunyueying
# 或者开始你自己的代码
git checkout -b your-name-or-feature # Let's Hack!
```

这一课讲什么内容？作业是什么？

- 自学如何在语法分析的算法过程中，进行属性计算或语法树构建
- 理论部分 EaC 讲解的多一些， Tiger 就直接上 yacc 的代码了
- **作业：在语法分析作业的基础上，尝试构建一个语法树出来**
- **作业：将自己生成的语法树用 dot/graphviz 生成图出来（成就感）**

什么，这就没了？

- 对。现在开始，主要就是编程练习了。后续课程形式改为编程讲解

一定要自己写作业，自己敲出来的代码才有可能学会

fork https://github.com/lazyparser/becoming-a-compiler-engineer-codes

新同学：可以加入本课程的微信学习群

- 群有人数限制，请在HelloGCC微信公众号输入「旁听」

志愿者征集：B站支持字幕了

- 用户可以自己编辑字幕并上传，有在线的字幕编辑器可以使用
- 需要UP主（我）和B站内容管理员进行审核
  - 我一般不出差的话当天会审核通过（或退回修改）
- 请大家有空时候为课程添加一点字幕，方便静音状态下学习观看
  - 这样可以在地铁或其他碎片时间观看视频了 ☺

方舟·编译技术入门与实战

视频可以碎片时间看，编程作业一定要沉下心来写才行的

有不明白的地方就及时开issues提问☺

https://github.com/lazyparser/becoming-a-compiler-engineer

PLCT Group, part of ISRC, ISCAS & HelloGCC Community & HelloLLVM Community