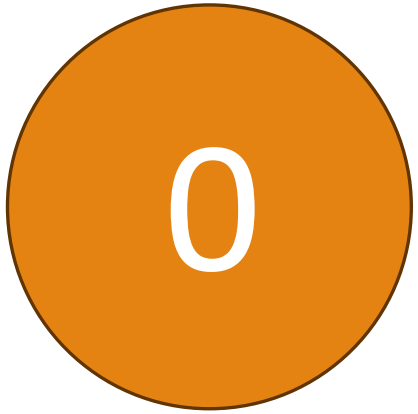
PA生存纪实

胡皓明 211250182



环境配置

我们花在配置环境的时间上可能比写代码的时间都多，为什么我们不属于环境学院呢

VSCode

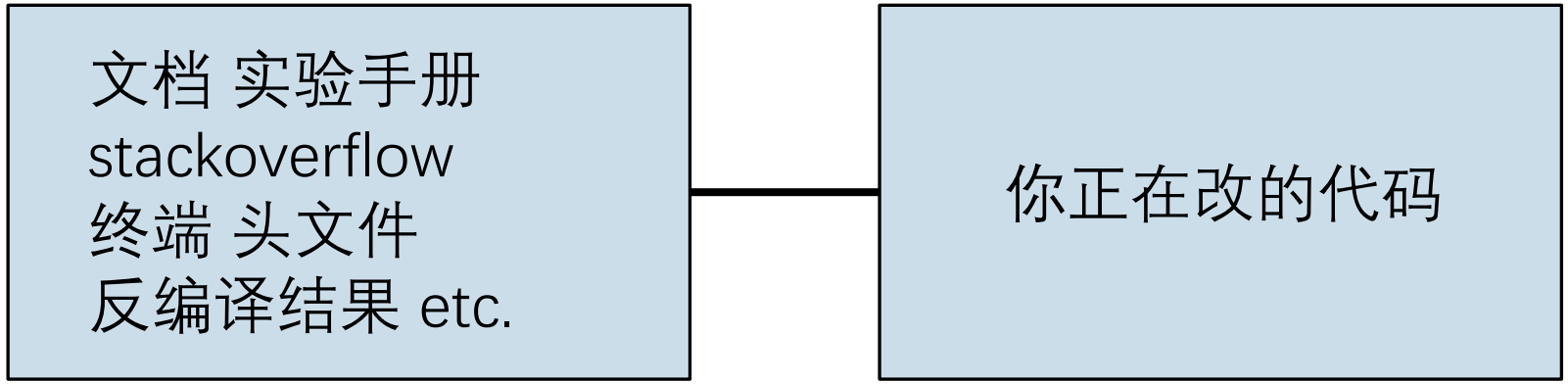
同学们没有必要非得用vim，vscode挺好的

通常而言，vim只是用来改那些不好用图形化编辑器打开的配置文件的

因此，大家需要掌握关于vim的只有以下这些：
1. 按I进入编辑模式，按Esc进入命令模式
2. :wq保存并退出，:q退出，:q!不保存并退出

第二块屏幕

只要预算允许，你应该尽快购置一块额外的屏幕。就算现在不买，总有一天在你写大项目的时候你会想要的。



写PA时连接电源

由于绝大多数同学使用笔记本+虚拟机来写PA，这使得获得一个相对高性能的和可靠性的环境很重要。因此，连接电源以获得更好的性能，并且避免因为突然没电导致虚拟机损坏。

配置

同学们执行git命令和调整配置文件的时候，除非你是高手或者整个做好了备份（指复制而非git备份），执行和保存前一定一定要确认清楚。
去年有同学

1. 配置改烂了最终不得不整个重做。
2. 还有一位同学因为git操作失误把PA1中实现好的sdb删除了然后在后续的PA中需要调试时才发现。

PA究竟在干什么

Abstract Machine PA的灵魂

阶段	NEMU	AM	Nano-OS
1	调试器		
2	CPU&外设	TRM&IOE	
3	中断	CTE	系统调用 文件系统 —系列用户库
4	内存映射	VME	分页

PA与其他实验

1. 完成PA之后，数电实验大作业的软件部分就不需要费太大力气了
2. 大班OS实验的内容没超出PA太多
3. 顺利完成PA可以为你选择jyy OS提供充分的信心。事实上，jyy OSLab=调用AM接口，编写一个支持多核的大型C程序。

PA生存指南

我们能做到

PA1

~~没那么困难~~

如果你不适应手册中给出的求值方式，你大可以清空expr.c，然后用你的方式实现。但如果正确性出了问题，请自行解决。

你可以试着把expr.c复制出来单独调试，然后再粘贴回去，查重没有那么严格。

简易调试器的每个功能几乎都有用。

PA1

很快你就会发现在简易调试器中应该添加一条以十六进制输出的指令，而且这条指令的使用频率甚至比以十进制输出高。

监视点很少用到，但是当需要排查奇奇怪怪的内存破坏时，这是你的救命稻草。

PA1的测试并不难，但是比较综合，不要钻牛角尖，检查一下是不是每条命令都能正确执行。

PA2

强烈建议在实现指令的时候压制一下copy-paste的欲望。这很有可能引起某天某个程序跑不起来。通过diff-test测试，最终发现时copy-paste的时候某个R忘记改成了I的。

AM是直接运行在NEMU上的，这意味着你不能调用libc，而只能通过硬件来获取相关数据。

PA2

AM的头文件包含和宏定义异常复杂（为了能通过较为简单的方式支持多种架构），这不仅会让你在寻找需要修改的代码时感到困惑，并且会使得VSCode产生一堆报错。

PA2中的大部分tracer都很难派上实际的用场，也难以加以测试，但是ftrace还是要写一下，这对大家理解ELF文件有巨大的帮助。

PA2

DiffTest非常重要。

从PA2后半段开始，简易调试器能提供的帮助就非常有限了，输出调试开始变得愈发重要。

PA3&4

PA3&4的难度没有比PA2高出太多。如果你在PA2中积攒了足够多的经验，自己完成PA3&4并不是异常困难的任务。

杂项排障

1. make失败可以试试先make clean
2. 程序行为没更新先检查是不是有些东西没有重新编译。
3. ……

最后，欢迎向助教提问，但是要遵守最基本的提问原则。

祝各位同学能够顺利的从PA和ICS中生存下来