Linux 平台下的反汇编及程序链接的在线授课方案

--基于 EMACS 编辑器的 ORG 表格及艺术家模式

一、基本信息

课程名称：计算机系统基础

课程类型：□通识教育课　　□公共基础课　　+专业课

□创新创业课程　□实验课

开课年级：本科二年级

面向专业：软件工程

教学章节：第三章 程序的机器级表示、第四章 程序的链接

授课学时：两章共 26 理论学时

主讲教师：李沁

授课形式：zoom 在线直播，学习通，中国大学 MOOC spoc.

选用平台及课程链接：

https://www.icourse163.org/learn/NJU-1001625001?tid=1001698002#/learn/announce

二、案例背景

本课程是软件工程的专业基础课，从程序员的角度，基于 intel X86 的硬件平台和 Linux，介绍与计算机系统相关的核心概念，解释这些概念如何相互关联并最终影响程序执行的结果和性能。可以使程序员明确程序设计语言中的语句、数据是如何在计算机系统中实现和运行的。主要内容包括：数据的表示和运算、程序的转换及机器级表示、程序的链接等。

课程目标包括：

1. 理解数制与编码在计算机和 C 编译器处理类型转换时的机制，并能据此发现存在问题代码缺陷的能力；能够根据汇编语言和高级语言的转换机制推理程序运行时栈帧的工作机制并能对常见语法构件的汇编代码进行向高级语言的还原；能够利用链接的知识分析多文件链接结构的分析。
2. 能够使用 Linux 环境下开源的编译与调试工具对可执行程序进行反汇编，并能进行简单的缓冲区溢出。

课程内容包括：

（一）计算机系统概述

1. 掌握计算机基本工作原理；
2. 掌握程序开发过程中编译器、链接器的作用；
3. 理解计算机系统的层次架构。

（二）数据的表示

1. 回顾数制与编码；
2. 掌握整数的补码表示机器在C语言中的类型转换；
3. 掌握浮点数的754标准与C语言中的类型；
4. 分析C语言中移位、扩展、整数运算、浮点数运算和底层表示相关的可能错误。

（三）程序的转换与机器级表示

1. 掌握IA32指令系统的主要指令结构；
2. 掌握C程序的机器级表示以及复杂数据结构的分配与访问；
3. 熟练掌握Linux的反汇编工具与技术；
4. 理解缓冲区溢出原理并能进行简单的缓冲区溢出攻击。

（四）程序的链接

1. 掌握ELF的结构与构造原理；
2. 掌握符号表结构与解析原理；
3. 理解重定位，能够对静态链接文件进行人工重定位分析；
4. 理解动态重定位、位置无关代码的原理，能定性解释动态链接程序的运行流程。

软件工程专业大二的学生已经学过C语言和数据结构，需要对C语言从源代码与可执行程序的对应关系有清楚的理解，尚无Linux系统开发的经验。

疫情期间，只要学生在家中有电脑和网络，可以通过安装Linux虚拟机完成实验部分的学习。

疫情期间整体的教学规划包括：

1. 联系出版社和作者，获得教材的电子版。
2. 借用作者在中国大学MOOC上的spoc资源，让学生观看南京大学袁春风老师的教学视频，并完成每周在线测验。
3. 将本人收集的教学电子版资料和本人制作的课件放在泛雅平台上。
4. 按照教学日历在前9周完成理论课教学；理论课教学以zoom平台的直播进行，直播同时录屏，将录屏视频上传至QQ群中供学生复习观看。
5. 实验部分的代码按教学进度在学习通上发布，让有能力的学生同步完成，同时准备好返校后的实验安排。

三、案例设计思路

拟解决的主要问题与相应的教学方法：

1. 如何在晦涩难懂的汇编代码与程序的动态执行之间形成一个直观性强、容易让学生接受的表现形式。我采取的教学方法是利用EMACS编辑器的org模式中的表格来模拟汇编程序执行的动态变化，用一种静态动画的方式，让学生能够直观地观察到程序执行时CPU寄存器和内存状态的变化。

在EMACS编辑器的org模式中，可以很方便的建立表格和调整表格结构，所有操作都可以通过键盘快捷键来完成，利用这个特点，教师可以利用快捷键来操纵表格，以模拟汇编代码执行过程中内存布局的变化。相对于教室中的面授过程学习，在线学习的过程中，可以直接看到代码的模拟执行过程，理解的过程更为直观，反而可以体现在线教学的优势。

教师可以在EMACS中打开两个窗口，一侧是汇编代码，一侧是内存布局与寄存器状态的表格演示

2. 如何将抽象的ELF文件格式的链接过程(包括静态链接与动态链接）形象的展示给学生，尤其是动态链接，不但牵涉到学生对ELF格式的理解，也牵涉到程序运行时共享库中全局变量和库函数的装载与寻址，无论是相关的数据结构，还是寻址回填的过程，其复杂性都达到了一个相当的水平，对于大二的学生来说是一个智力上的挑战。

我们在这个章节采用的教学方法是利用EMACS的艺术家模式，用字符串绘图。使用EMACS的艺术家模式绘图的好处在于我们可以在代码中以注释的形式将数据结构、内存布局等概念模型清楚的表达出来，不再需要在课件、代码之间来回切换，更为适合在线屏幕分享。

四、教学目标

1.知识与能力目标

(1) 理解汇编程序与相应的c语言源代码之间的对应关系，能够根据反汇编的代码还原c语言代码，即对汇编代码进行程序理解；

(2) 将静态链接过程中的重定位以图形化的方式展现给学生，使其能够准确理解静态链接器在链接可重定位文件时如何进行地址计算，并能够完成重定位符号地址的手动计算；对于动态链接，将其过程以动态绘图的方式呈现给学生，辅助以ELF反汇编结构的讲解，让学生能够理解Linux平台下动态链接间接寻址和延迟绑定的实现原理，可以对已有的ELF包含的重定位信息进行解析。

2.育人目标

培养学生不畏困难、勇于挑战，在学习计算机系统中复杂原理的同时体会探索的愉悦；在疫情期间加强自我管理以及对自己的要求，培养自律精神；体验开源文化。

五、教学过程

1. 反汇编课程教学设计

本节以两个具有代表性的例子来说明反汇编的教学设计。

1.1 缓冲区溢出代码的教学设计

图 1 中，左下是 C 的缓冲区溢出源代码；右侧是 C 程序编译后再进行反汇编得到的汇编指令，包含了进程虚拟内存地址信息；左上是对应汇编指令绘制出的 ORG 模式表格。**表格的第 1 列用于表示寄存器中的值或某些变量的标记；表格第 2 列表达程序运行时内存中栈帧的状态，每行 4 字节。**这个缓冲区攻击的例子中，关键指令在 0x8048434 和 0x8048452 这两个地址之间，程序员通过计算，根据指针变量 buffer 的地址计算出变量 ret 的地址，直接修改 ret 的值为函数 function 栈帧的返回地址的存储地址，最后将 ret 的值加 10，使得返回地址跳过了 0x804848a 和 0x804848d 两处的指令(共 10 字节)，导致 main 函数中赋值语句 x=1 不能被执行，从而在执行 printf 语句时输出 0，而不是预期的 1.

教师在讲解这个例子的分享屏幕过程中，沿着汇编指令逐条修改左上的内存栈帧示意图，让学生可以清楚的追踪返回地址是如何随着汇编指令的执行变化为 0x804a494 的全过程。**所有汇编指令执行带来的栈帧变化都是用 EMACS 表格操纵快捷键来完成的，给学生造成的视觉效果是一个节奏比较慢的动画演示，直观的领会汇编指令的执行和内存寄存器内容的变化之间的关系。**

1.2 二维数组的一维存储

图 2 中分享屏幕右上的内存栈帧绘制了一个 3×3 的二维数组是如何在内存中一维存放的；左侧的汇编指令主要用来说明：二维数组时如何如何通过汇编指令存入栈帧中(7-3f)；二维数组中对元素的访问需要如何进行其地址的计算(7e-8a)。

教学过程中，根据对指令的分析，学生可以观察到四个寄存器 EAX、EBX、ECX 和 EDX 内容的变化，从而理解二位数组元素地址的计算。红色箭头所指的三个地方分别是数组存放的汇编指令、数组在栈帧中起始位置的图示以及在 C 源代码中对数组元素的访问。

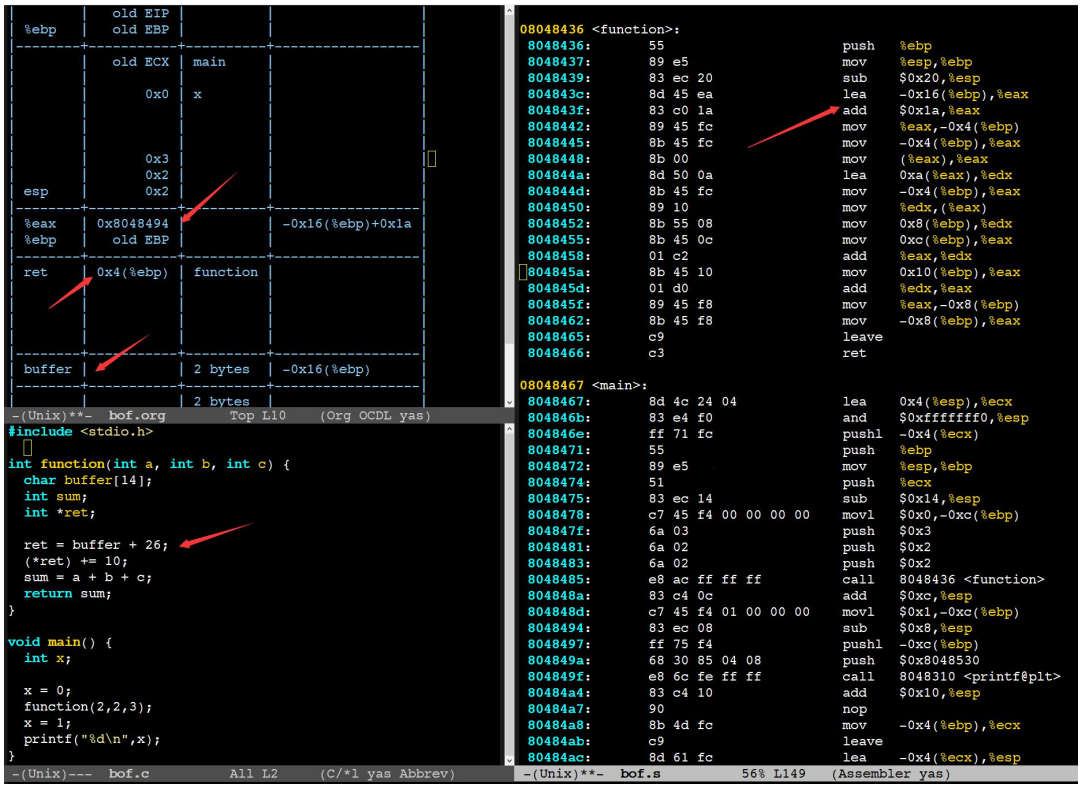


图 1 缓冲区溢出的屏幕分享

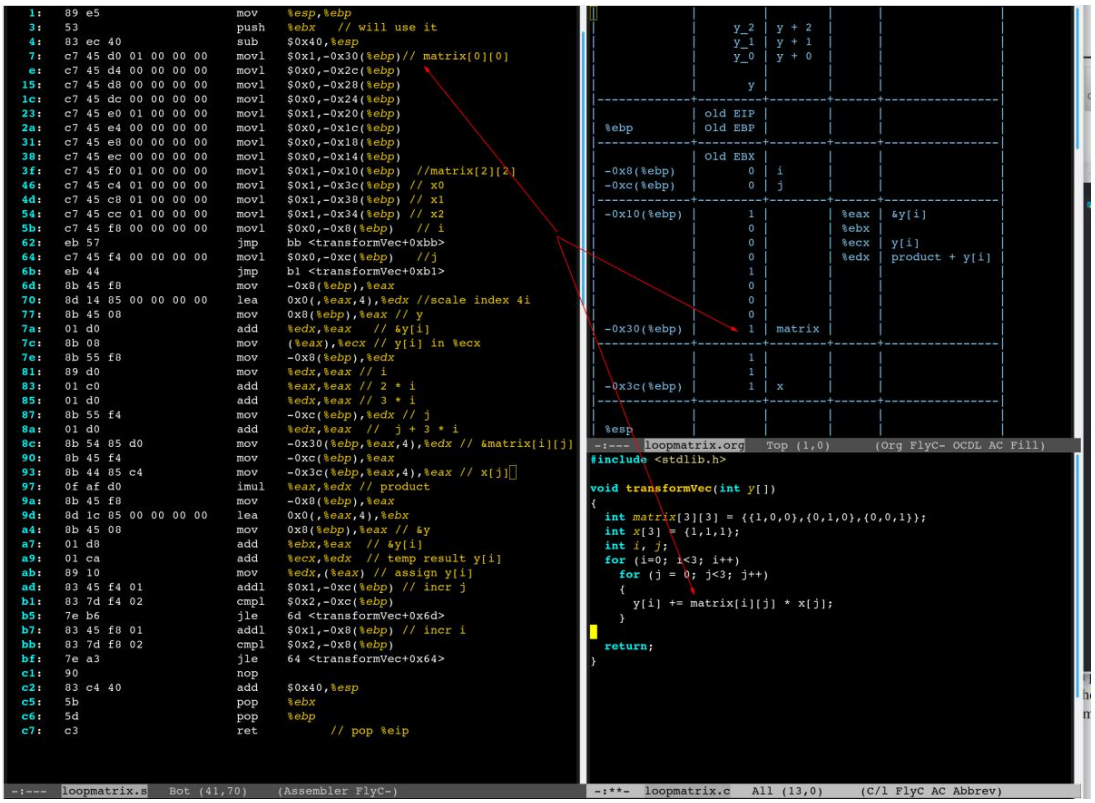


图 2 二维数组行主序存储的屏幕分享

2. 程序链接的教学设计

2.1 静态链接的符号重定位

图 3 中，左侧两段反汇编指令分别是需要被重定位的两个 ROF 文件，右侧是根据重定位的需求绘制出的 ROF 文件的结构图和重定位之后链接起来的 ELF 文件结构图。此图着重说明当两个 ROF 文件中的函数指令被合并为 elf 中代码区的时，如何进行 PC32 相对寻址的地址计算，以及对全局变量的绝对寻址。

在每一个 ROF 文件结构图中绘制出了代码区变量区以及重定位信息，重定位信息可于右上的重定位结构体定义相对应。中间 main 的 ELF 结构包含了合并后的数据区.data 和代码区.text，最下方的箭头指出了重定位之后函数 swap 的起始指令地址，由此可以算出在其 18 字节之前的地址恰为 main.o 的重定位偏移量所指地址，然后用该处的-4(0xFFFFFFFC)与 18 相加，得到 14 既为调用 swap 第一条指令需要的相对 PC 寄存器地址的偏移量.18 字节可以通过对左上的汇编代码从 07 开始向下数 18 个字节到达 main 函数的末尾得到。

另外，在右侧 swap 的 ROF 文件结构图中，可以发现 bufp1 和 bufp2 的值是未定的。在合并以后，中间 main 的 ELF 文件结构图中，两个指针变量的值已经被确定下来了，分别指向 buf[0]和 buf[1]的地址，同时数组 buf 也获得了存储地址 0x804a018。

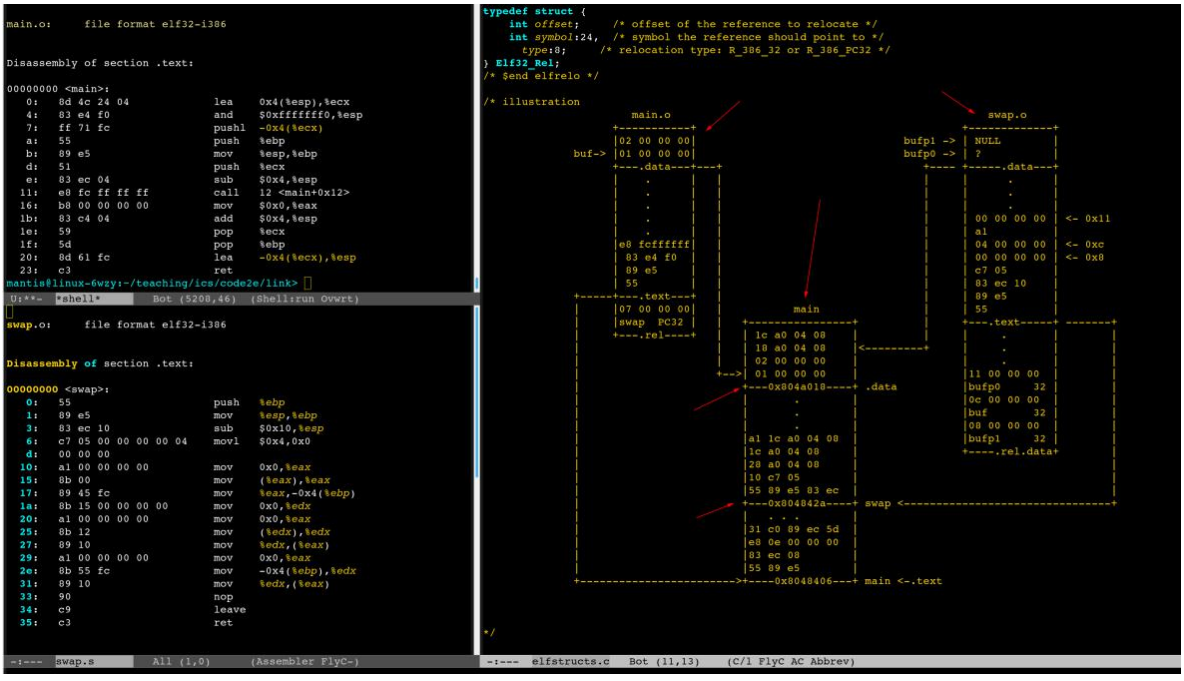
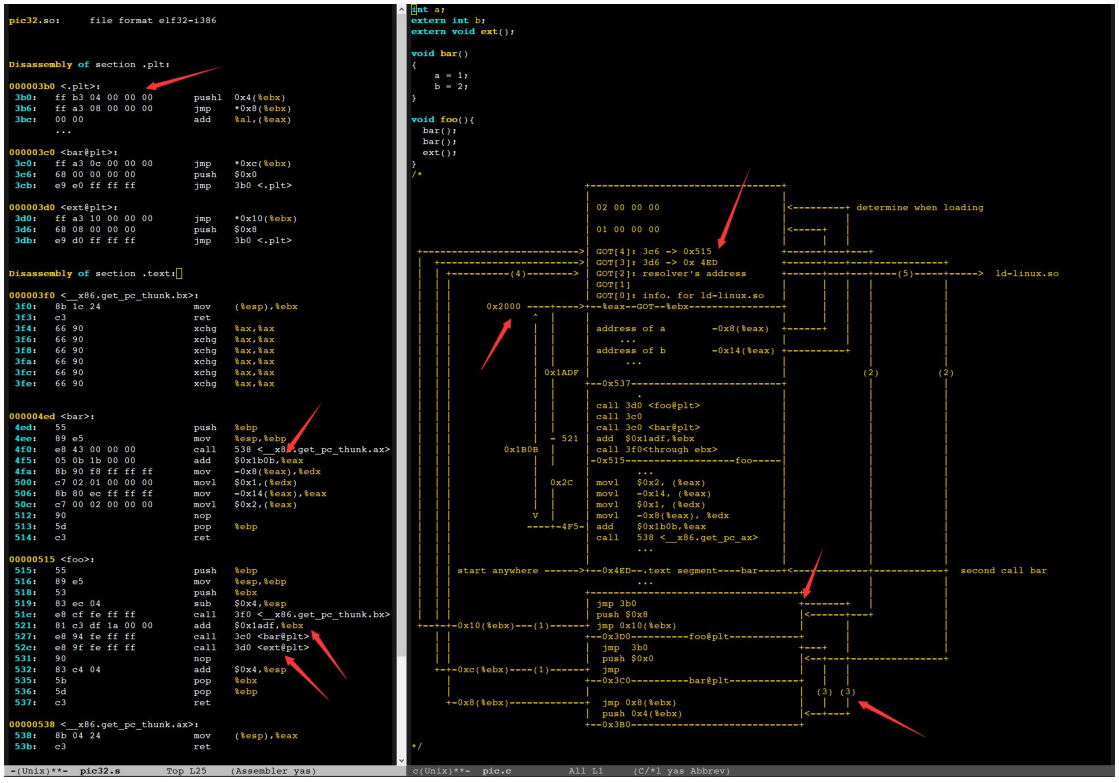


图 3 静态链接重定位的屏幕分享

2.2 动态链接的延迟绑定

图 4 中，左侧是对右上 C 程序反汇编后的汇编代码，汇编代码试图在程序运行时临时决全局变量 a、b 的地址，并且在调用 bar、foo 和 ext 函数时才确定它们的起始地址。右侧是其对应的动态链接后的内存布局。右侧图中最左箭头所指为全局偏移表 GOT 的起始地址，下方箭头所指为延迟绑定表 PLT。用(1)、(2)、(3)、(4)、(5)标记了动态链接过程中 GOT 表关于函数 foo 和 bar 的装载地址的回填过程，首先给出了计算 GOT 表地址 0x2000 的指令(4f5-4fa,521-527)，然后在调用函数时(2)跳转到自己的 PLT 表项，(3)跳转至 PLT[0]所在地址，继而(4)跳转到 GOT[2]动态链接器 ld-linux.so 的首地址，完成(5)GOT 表项的回填，确定两个函数的起始地址.



图表 4 动态链接延迟绑定的屏幕分享

六、教学效果与特色创新

图 5 与图 6 是学生在上完函数栈帧之后做作业时绘制的递归阶乘函数栈帧变化图；图 7 是在理论课结束之后，本人在课程群中发起的对教学方式改革的意见统计投票。投票说明，本人在这门课中在线教学方式改革还是得到了大部分同学的认可。当然在这种模式下，如何使教学能够更加直观与生动，还有待后续的改进。

总体上来说，本课程的在线教学方式的特色创新之处在于：**充分利用在线教学的特点，拉近了学生与实际代码分析的距离，让枯燥的代码分析更加生动与直观；强调实战；在课程传授的同时让学生体验开源文化。**

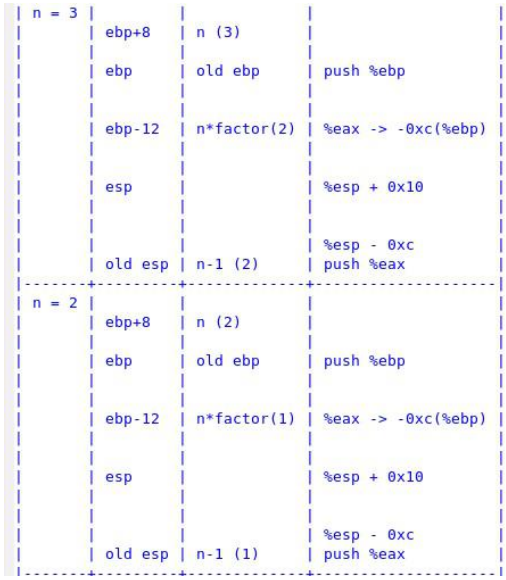


图 5 学生作业绘制的栈帧变化图 1

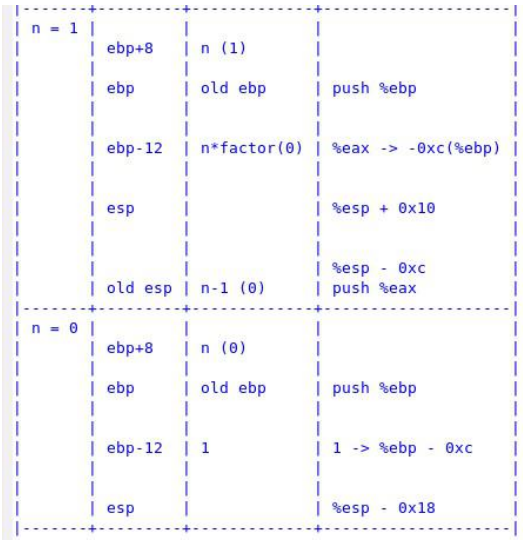


图 6 学生作业绘制的栈帧变化图 2

图 7 对在线教学方式改革的意见统计

图 8 是本人在课余时间在 QQ 群内告诫学生疫情期间好好学习，用自己的行动向一线的抗疫医护人员致敬，共同为国分忧.

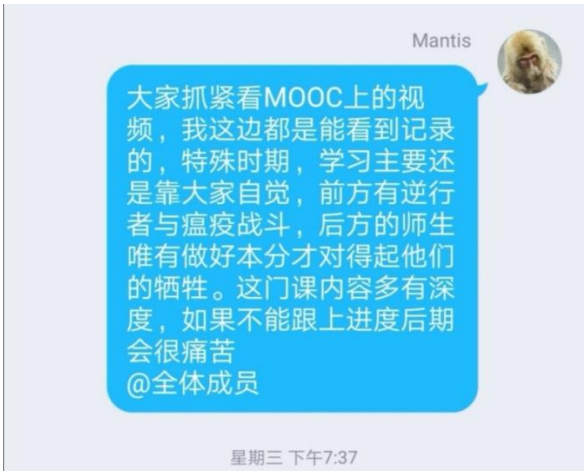
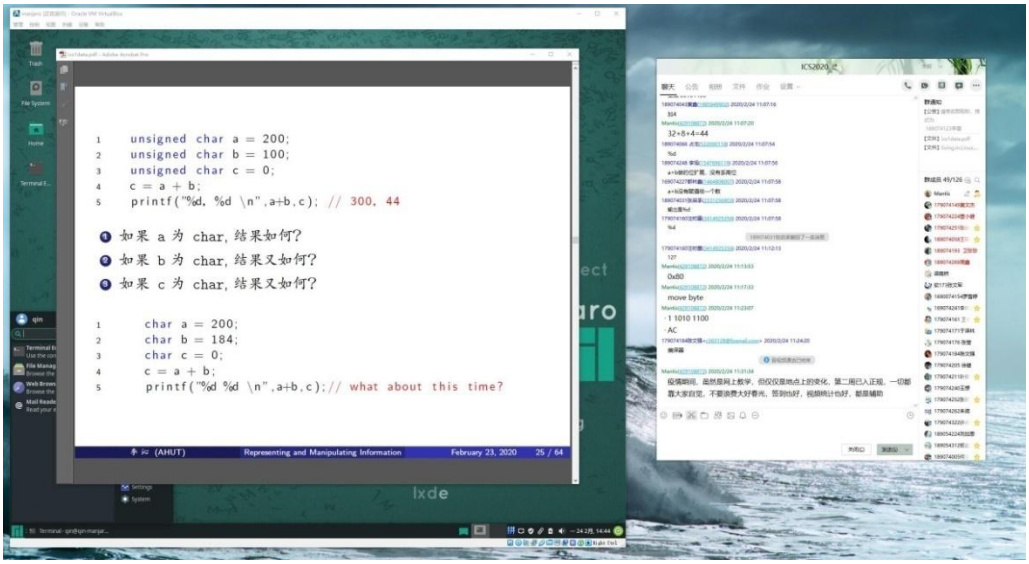


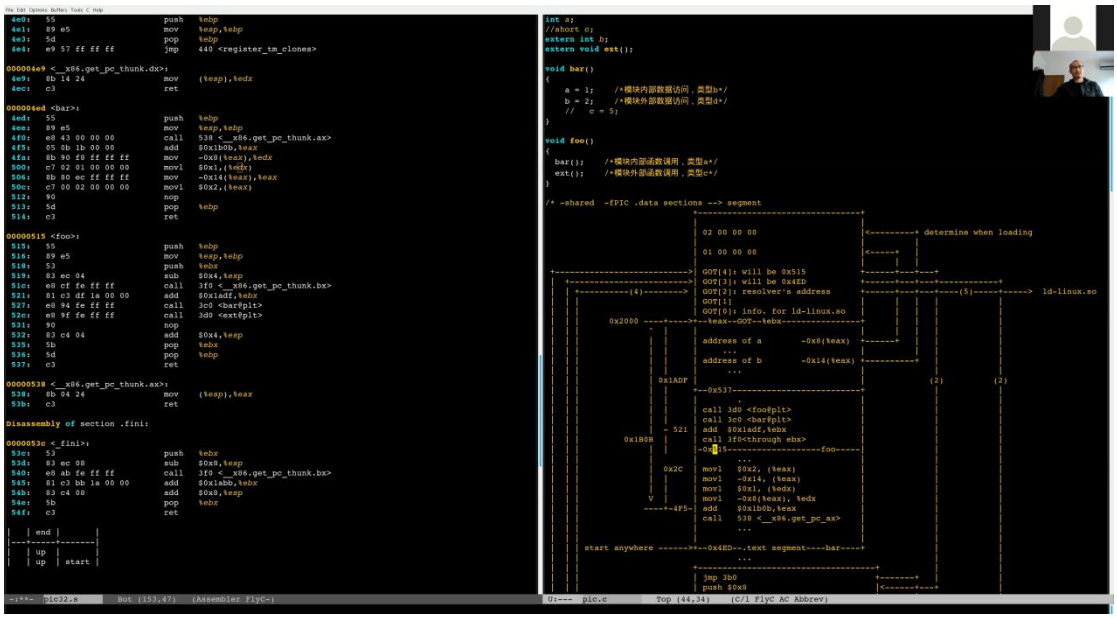
图 8 QQ 群内的思政滴灌

以下为学习通上部分签到截图，除去免听人数后，到课率保持在 95%以上。

以下为直播上课的现场截图，内容是第二张数据的机器级表示与操纵，右侧是 QQ 群内与学生互动。



以下为 Linux zoom 直播上课的现场录屏截图，内容是动态链接的延迟绑定。



七、教学反思

本学期疫情期间的在线教学的尝试与过程中自己做的教学改革让我认识到，**就本科教学而言，在线教学的教学效果可以达到甚至超越线下的教学效果，关键还是教师能否结合所承担课程的特点去主动地调整教学方式，充分发挥线上直播的优势，淡化不利的地方，这是未来几年内每一位高校教师都应该认真考虑并付诸实践的。**对本人承担的两门专业课来说，计算机系统基础和 UNIX 网络编程，在教学过程中存在大量的代码分析，过去在教室里，学生和投影屏幕之间距离比较远，在讲解代码的时候，只能抓住代码最主要的部分，细节展示的不够充分，而且可操作的余地也很小，讲解还是围绕着课件来进行的，这并不有助于对学生实战能力的培养。**转成在线直播以后，我采用了在 Linux 平台下的 zoom 直播，分享 EMACS 编辑器的屏幕，把代码的运行、分析、图示或者动画讲解，直接呈现在学生眼前，拉近了学生和代码的距离，枯燥的代码分析变得更为直观与生动。**同时也能够让学生深刻的体会到开源软件的力量，也就是说，当学生看到老师可以在一个编辑器内利用各种开源工具完成不同类型任务的时候，他自然而然会对这个编辑器及其背后的开源文化产生浓厚的兴趣。就我个人的观察，很多学生在上完这两门课以后，都把自己编程的开发环境转换到 EMACS 下。

八、教学资料

以下是学习通泛雅平台上的教材电子版、课件以及供学生阅读的课外参考书。