PA0 实验报告

翟笑晨

2024 年 9 月 13 日

1 Installing GNU/Linux

在安装了 Windows 11 专业版之后，我用 windows 自带的 Hyper-V 装载了 Ubuntu22.04 的虚拟机。

2 First Exploration with GNU/Linux

这部分我完成了所有内容。

2.1 遇到的问题和思考：

1、运行某些指令会显示 “Permission denied.”，可以通过在指令之前加上 sudo 运行，或者通过指令 “su -” 进入管理员身份，用 root 用户运行完成指令。

2.2 回答蓝框问题：

1、为什么关机指令要管理员权限?
防止计算机被无意或被渗透工具如 metasploit 恶意关机，导致未保存的文件和数据受损丢失, 避免恶意攻击

3 Installing Tools

这部分我完成了所有内容

3.1 遇到的问题和思考：

1、在使用 hyper-v 虚拟机时，发现占用主机 cpu 和内存比较高，经过排查发现是虚拟 cpu 的数量过多，消耗软件资源过多，下调数量之后有所缓解。
2、直接使用指令安装教程上最后一个工具包时会报错:The following packages have unmet dependencies，发现 libsdl2-dev 包的版本与 Ubuntu 的版本不匹配。经过搜索，通过安装 aptitude 包，运行 “sudo aptitude apt-get libsdl2-dev” 就可以顺利安装。

3.2 想法和心得：

想配置声卡驱动配置失败，流程极其复杂，到时候只能玩没有声音的 Mario 了 (T_T)。

4 Configuring Vim

这部分我完成了全部内容

4.1 想法和心得：

1、安装完 Vim 之后，我按照指令体验了 Vim 编辑文件的强大能力。通过学习基本的操作命令，我发现 Vim 的模式化编辑可以显著提升文本处理的效率。普通模式、插入模式和命令模式的切换让我能够更加灵活地操作文件。利用 Vim 的强大插件系统和自定义配置，我逐步深入探索了其高级功能，比如自动补全、代码折叠和宏录制，这些都极大地丰富了我的编辑体验。
2、在 Vim 编辑器中运行了.c 文件后，我对其集成的开发功能有了更深入的了解。通过使用 Vim 的编译命令，我可以直接在编辑器中构建和运行代码，而不需要切换到终端。这种一体化的工作流程极大地提升了开发效率。此外，我还学会了如何配置 Vim 来支持自动编译和错误提示功能，这使得代码调试变得更加流畅。随着对 Vim 的不断探索，我逐渐掌握了更多的技巧和插件，进一步优化了我的编程环境。

5 More exploration

这部分我完成了全部内容

5.1 想法和心得：

0、通过学习 Linux 的初级教程，我逐渐体验到了在终端中使用命令行操作计算机的便利。相比于图形化界面，终端操作不仅可以更高效地完成任务，而且还允许更精细的控制。使用命令行时，我可以快速执行复杂的操作，自动化重复的任务，并通过脚本提高工作效率。我相信，随着对终端操作的进一步学习和实践，我将更加深刻地体会到它在系统管理、开发和其他任务中的独特优势。
1、在学习如何使用 gdb 进行调试时，我体验了其强大的功能。虽然教程中的 test.c 文件比较简单，可能无法完全展示 gdb 的潜力，但我对其调试能力有了初步的了解。gdb 提供的 frame 和 print 指令，使得在调试过程中能够方便地查看函数调用栈和变量值。通过这些工具，我可以逐步追踪程序的执行流程，识别并解决潜在的问题。预计在处理更复杂的程序时，gdb 将成为一个非常有价值的调试助手，帮助我深入理解程序的行为并定位错误。
2、在初步学习 Makefile 的编辑格式后，我发现这是一种强大而灵活的工具，用于简化和自动化构建过程。通过 Makefile，我可以方便地定义文件之间的依赖关系，并用简洁的语法描述如何从源代码生成目标文件。比如，使用 Makefile 可以自动编译.c 文件、链接生成可执行文件，并且进行清理操作。Makefile 的规则和指令使得整个构建过程更加高效和可维护。接下来，我计划深入探索 Makefile 的更多高级功能，例如变量和条件判断，以便更好地应对复杂的构建需求。

6 Getting Source code for PAs

这部分我完成了所有内容

6.1 遇到的问题和思考：

1、make menuconfig 遇到 warning：.config does not exists。根据教程安装了 flex 和 bison。

6.2 想法和心得：

1. 按照 PA 讲义中的步骤，我成功克隆了 PA 的代码库。在这个过程中，我结合之前学到的 git 机制和语法，进一步加深了对 git 工具的理解。我熟悉了如何使用 git 进行版本控制，包括创建分支、提交更改以及合并代码等操作。通过 git，我可以有效地追踪代码的修改历史，管理不同版本的代码，并与团队成员协作。这些技能不仅帮助我更好地管理自己的开发进度，也为未来的工作提供了重要的支持，使得代码开发和维护变得更加高效和有条理。

6.3 对“好的提问”以及“通过 STFW 和 RTFM 独立解决问题”的看法

在学习计算机及其他技术领域时，遇到问题几乎是不可避免的。处理问题的能力不仅仅是对技术的掌握，更涉及到如何有效地提问和如何独立解决问题。通过学习和反思，我发现“好的提问”与“独立解决问题”的技巧对于个人成长和学习进步都至关重要。

首先，我们面对的问题通常可以分为两类：普遍问题和复杂问题。普遍问题往往比较简单，且已有大量资源可供参考。这类问题通常描述清晰，触发条件明确，容易在互联网上找到答案。例如，“如何在 Vim 中删除一行”这种问题，因其普遍性，网上有大量的解决方案和教程。通过精准的搜索，通常能够快速找到问题的答案。我们可以借助搜索引擎，利用关键词来迅速获取答案，这种方法称为 STFW（Search The F***ing Web）。

然而，当问题变得复杂且难以描述时，情况就会有所不同。复杂问题可能涉及到多种因素，单纯的搜索可能无济于事。这时候，提问的艺术便显得尤为重要。在提问时，需要确保问题描述详细而准确。简单地描述问题现象是不够的，我们还应提供操作步骤、相关的环境信息以及出现问题的背景。这样做不仅能帮助他人更快地理解和解决问题，还能在整理问题的过程中，自己对问题有更深刻的认识。

提问的过程实际上是一个自我整理和思考的过程。好的问题描述不仅能帮助他人理解问题，更能帮助自己理清思路。通过清晰的描述，我们可能会发现问题的根源，从而自己找到解决方案。计算机系统的复杂性意味着问

题往往具有多样性，准确的描述和详细的背景信息能大大提高得到有效回答的概率。

与此同时，盲目依赖单一的教程或指南并不可取。虽然许多教程提供了宝贵的知识，但它们可能并不总是适用于每一种情况。解决问题时，我们需要结合多个资源，综合分析，才能找到最适合自己的解决方案。对于复杂问题，单纯依赖别人的解决方案可能无法完全适用，因此综合多种方法和技巧，通过自己的判断和实践，才能真正掌握解决问题的技巧。

总之，学会提问是一种重要的技能，它不仅能帮助我们更高效地解决问题，还能促进个人的思维和理解能力的提升。通过精准的提问和独立的思考，我们不仅可以找到解决方案，还能在过程中提升自己的技能和知识水平。正如《上善若水》所言，学会提问看似简单，却蕴含了丰富的内涵。好的提问和独立解决问题的能力，既是学习的艺术，也是成长的必经之路。