ICS PA 0

201300035 方盛俊

0. 目录

1. Installing GNU/Linux

安装环境永远是最头疼的东西之一.

1.1 第一个坑: WSL 不支持 systemctl 与 service 等命令

尽管在 pa0 的部分更推荐使用 vim 进行代码的编辑, 我仍然决定尝试用 VSCode 来编辑 (部分原因是已经习惯了 VSCode 的很多快捷键).

并且还想尝试一下, 用 Windows 下的 WSL2 (以下均简称为 WSL) 代替真正的 Linux 来开发, 这样就省去了在两个系统中切换的麻烦, 也能够更加方便地使用 VSCode 来开发.

因为 WSL 是 Windows 的一个子系统, 所以默认并不是以 systemd 作为初始化进程, 也就无法使用 systemctl, 如果用到了, 就会报错:

```
System has not been booted with systemd as init system (PID 1). Can't operate
```

经过初步的 STFW 发现, 很多人是在 Docker 时遇到了这个问题, 对应的解决方案对我来说没有多大用处. 再经过一番痛苦的 STFW 之后, 发现了一个相对靠谱且简单的解决方案.

这里我用的 WSL 系统是 Ubuntu, 对应的解决方案为 `ubuntu-wsl2-systemd-script`:

```
sudo apt install git
git clone https://github.com/DamionGans/ubuntu-wsl2-systemd-script.git
cd ubuntu-wsl2-systemd-script/
bash ubuntu-wsl2-systemd-script.sh
```

解决方案来自 这里.

1.2 第二个坑: WSL 原生不支持图形界面和音频播放

在尝试运行 FCEUX 中的 mario 的时候, 果不其然, 报错了.

报错信息大概如下:

```
ALSA lib confmisc.c:768:(parse_card) cannot find card '0'
ALSA lib conf.c:4292:(_snd_config_evaluate) function snd_func_card_driver retu
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
```

经过搜寻, 大概是 WSL 中 Linux 自带的音频功能 ALSA 工作不正常, 不能使用.

尝试解决问题, 无果, 迫不得已先放弃 WSL, 果然 WSL 当成普通的 Linux 开发环境还行, 但是过于底层的开发, 或者说涉及到图形界面开发, 还是用正常一点的 Linux 环境吧.

1.3 安装 Kubuntu Linux 与 Windows 双系统

Windows 下有很多资料, 也安装了很多软件, 不太想直接重装. 如果开 Linux 虚拟机的话, 又感觉没那么快乐 (奇怪的说法), 还是试试双系统吧.

先是 Manjaro, 好不容易配好了, 却发现包管理的问题很难解决, 尝试了一番, 还是打算用回 Debian 系的 Ubuntu.

但是我又不太喜 Ubuntu 默认选用的 Gnome 桌面环境, 所以最终选择了以 KDE 为桌面环境的 Kubuntu, 这样比较好看.

经过一番折腾, 终于配好了双系统到环境, 没有造成数据损失, 真是可喜可贺可喜可贺.

2. First Exploration with GNU/Linux

2.1 Why Windows is quite "fat"?

> **Question:** Installing a Windows operating system usually requires much more disk space as well as memory. Can you figure out why the Debian operating system can be so "slim"?
>
> **Answer:** There are GUI in Windows and many other things in it, which are reserved for "compatibility". Besides, the users who use Windows usually know little knowledge about computers so that there are many functions in Windows for helping they use computer simply. However Debian is designed for professionals and many functions are used in CUI, so Debian is quite "slim".

2.2 Why executing the "poweroff" command requires superuser privilege?

> **Question:** Can you provide a scene where bad thing will happen if the poweroff command does not require superuser privilege?
>
> **Answer:** It is possible that many users are using the same Linux system in the same machine. If a normal user can poweroff the machine, other users will face the danger that their programs exit and data lost.

3. 《提问的智慧》读后感

每一个人, 特别是学计算机的人, 都要学会通过 STFW 和 RTFM 独立解决问题. 这是由多方原因决定的.

编程的复杂性

编程在一般情况下并不是一件非常困难的工作, 但编程一定是一件复杂的工作.

软件的本质是它的复杂性, 没有足够复杂度的软件, 是无法应对用户多样性的需求的. 既然软件成品尚且如此复杂, 在编程的过程自然更加地复杂, 你必须面对多种多样的情况, 处理数不胜数的 bugs, 在极大的努力下才能做出成果.

编程的复杂性往往又会导致下面几种结果.

首先, 编程的复杂性意味着人们必须共同协作, 共同编程. 我编写的程序, 往往建立在你建立的平台上, 使用着他提供的第三方库, 最后部署在她维护的不同系统的服务器上. 这个过程往

往是充满了纰漏的: 我并不知道你的平台是否稳定, 他提供的第三方库里是否存在着 bugs 甚至是后门, 她使用的服务器也不一定完全兼容我写的代码.

因为没有人能够真正完全掌握这浩如烟海的细节, 每个人能够掌控并了解的只有自己所写的那一部分, 将不同部分组合起来, 往往就会带来各种问题. 每个人写的代码不同, 遇到的问题自然也有所不同, 别人也并不了解你写的代码, 你也很难依靠别人给你一个详细的解答, 这时候, 请 STFW 和 RTFM 吧.

其次, 编程的复杂性意味着我们的工作总是"创造性"的, 即使不是"创造性"的, 至少也是"有差异的".

编程不像数学题, 大部分解答起来不困难的数学题都已经有了现成的答案, 剩下未解决的数学题往往都是非常困难的, 只有少数数学家敢于去挑战. 而编程不一样, 每一个程序, 每一个软件, 每一个应用, 都是不一样的. 即使看起来功能上大体相同, 也会在很多细节上, 或者底层实现上有相当大的差异. 再次提及这个观点: 绝大多数的编程任务并不困难, 但依然复杂.

大部分编程任务并不困难, 这意味可以让更多不能成为顶尖数学家的普通人参与进编程任务中; 编程依然复杂, 这意味着不能简单通过寻找现成的"答案"来完成一个程序的编写, 即不能自动化地完成, 也就需要许多拥有一定编程知识和经验的程序员来完成一个个的编程任务. 这些编程任务都是"创造性的"或"有差异的", 这自然使得每一个程序员都要像"创作"一样, 独立解决问题.

第三, 编程的复杂性意味着容易出现 bugs. 我们都是人类, 不是机器. 机器也许能够保证不犯错且二十四小时工作, 但是人类一定不行.

每个人都会犯错, 这个事实, 在我们上中学学习数学并参加数学考试以来, 就已经人尽皆知了. 我们做数学题会算错数, 写语文作文时会写错字, 自然在编程的时候也很可能会写下许多的 bugs. 而且往往我们自己也很难发觉为什么会引起这种 bugs. 既然最了解自己写的代码的我们也很难找出 bugs, 自然也很难指望别人帮我们找出 bugs, 最终还是得靠自己 STFW 和 RTFM.

用好互联网

STFW 的意思是用好网页搜索 (Search The Friendly Web). 我们生活在互联网的时代, 我们可以很方便地在网络上搜索, 分享信息. 也许你不能在网络上找到你丢失的一双袜子, 但是对于我们犯的蠢, 很有可能在网络上发现别人也曾经犯过相同的蠢.

要善于利用出 bugs 时的报错信息, 有一些网站可以很容易地根据报错信息找出可能犯的错误, 例如网站 Stack Overflow. 即使没有现成的解答, 你也可以在相应的论坛, 或用 Github Issues 功能找到一些相关的信息.

想要更广泛, 方便地找到这些信息, 用好搜索引擎也很重要. 我们要注意: 互联网上 90% 的内容是用英语书写的, 所以我们最好用英文来搜索, 不要用中文. 而且搜索引擎最好使用

Google 这些有口皆碑的搜索引擎. 最好不要用 Baidu 搜中文信息, 更不要用 Baidu 搜英文信息, 它对信息搜集的召回率和准确率都远低于 Google. 并且, 要用好一些额外的搜索功能, 例如"用双引号括住关键词"代表了搜索出来的网页一定要包括该关键词, 用 "site:" 功能实现对指定网站内容的搜索等等.

除了搜索网页外, 我们还可以 RTFM, 即查看官方文档 (Read The Friendly Manual). 我们可以通过互联网找到相应的资源. 学会自己看文档, 对于编程来说, 非常重要.

提问的方式

如果经过 STFW 和 RTFM 之后, 还是找不到解决方案, 也许你可以考虑去询问他人了.

但是, 请注意提问的方式!

别人没有义务回答你的问题, 所以你必须想办法激起别人的兴趣, 让别人乐意回答.

首先, 最重要的是准确描述问题. 准确描述问题非常重要, 你不能以一种模棱两可的方式进行提问, 例如 "为什么我的电脑突然死机了" 这种问题, 没有提供任何有用的信息, 自然也别想指望别人帮助你. 为了更好的描述问题, 你还需要讲述你为了解决这个问题所做出的努力, 例如加上 "我尝试了网上搜索出来的方案一, 问题并没有解决", 这样, 别人至少能排除一种可能的情况, 也可以知道你并非坐等其成的人. 除此之外, 最好描述你中遇到问题之前做的操作, 提供复现的方式 (例如可以复现的代码 repo), 这样能给别人帮你解决问题提供很大的便利.

其次, 要注意询问的方式. 时刻牢记, 别人没有义务为你解答, 所以不要趾高气扬地提问, 也不要抱着一种理所应当的态度. 在解决问题之后, 至少要表示你的感谢, 否则别人很可能因为你的态度, 而在下一次你提问的时候拒绝帮助你. 还有, 尽量少用私人联系方式向他人提问, 更应该中论坛, 问答网站或 Github Issues 这类公共的地点提问, 这是为了方便帮助以后再次碰到类似问题的人, 这样他们就可以通过 STFW 找到答案, 避免了重复提问与重复回答.

总结

提问的智慧很重要, 而学会通过 STFW 和 RTFM 独立解决问题, 是避免发起不必要提问的好办法. 先学会独立解决问题, 这才是最好的提问的智慧.