

# PA0 实验报告 开发环境配置

宁晨然 17307130178

## 一、内容概述

PA0 主要涵盖了安装 ubuntu、配置环境、熟悉 linux 操作等几个任务。我之前对 linux 已经有部分了解，所以这次开发环境配置任务相对轻松，花费时间 8 小时左右。最后的成果与预期成果相符，我也成功熟悉了 linux 的命令行操作与各种工具的使用，也学会了自我解决遇到的困难问题。

此报告侧重讲解我探索 linux 系统的过程和遇到困难解决的方法，我会分成几个方面来描述。

## 二、学习过程

与其说 PA0 是一次 homework，更像是一种探索的过程。学习新知识，打破墨守成规的常识操作，尝试用新的知识与方法运用到新的系统、新的工具使用当中。我以前对 linux 本身比较陌生，但上学期我学习过 linux 部分内容，所以在 linux 命令行操作和部分工具使用中比较得心应手。这次作业中遇到的最大的问题是在 vim 和网络配置。

### (1)、安装与设置

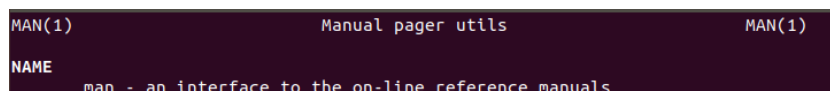
PA0 安装了各种各样的东西，我总结了一下：

①**软件安装与 linux 虚拟机：**之前已经安装过 VMware，在学校官网中下载了正版软件，并且下载了 ubuntu18.04.2（之前使用的是 14 版本），没遇到问题。

**A.Linux：**熟悉 linux 的操作，在以前我已经非常清楚了，这次花了一点时间回忆起来，重点就是命令行的操作。所有的文件操作都可以通过命令行执行。ls/pwd/cd/touch/cp/mv/mkdir/rm/man 操作基本用一次就会记住。这里花了很多时间研究 man，这个命令非常的简单，用起来极其方便。因为可以直接查看手册，所有 RTFM 的东西基本可以靠 man 和百度 google 解决。后来我试着阅读了几个 man 的解释，发现规律还蛮明显的，其实以后只要看不懂就查一下 man 和常用用法就知道了，类似于英语词典查语法一般。

这里其实有个难懂的就是正则表达式 (grep 操作)，我没有太多花心思放在这个上面看教程，之后如果有时间我会补上这里的教程。之后尝试了一下“du -sc /usr/share/\* | sort -nr | more”等命令。掌握了 tap 的联想功能后，确实再也不用担心我输错文件名啦！（p.s. 我还发现当有几个名称类似的比如 gap.c 和 gap.h，用 tap 联想时，它会补全相同部分 gap，很智能！）

用 vim 写了 hello.c 并体验了一下重定向，重定向部分有些不太明白，如果之后有更多操作会更加理解吧。



```
MAN(1)                                Manual pager utils                                MAN(1)
NAME
man - an interface to the on-line reference manuals
```

**B.VMWARE：**虚拟机的设置项目不多，对于这个 linux 的系统设置，如下。重点应该是在教程中讲到的网络适配器。vmware 有三种网络模式，桥接、NAT、仅主机模式，然后教程中想利用 ssh，所以讲述了 host-only（仅主机模式）的用法。由于 vmware 和 virtualbox 有一点区别，在设置上我找了很多网络资料，可惜最后还是没能在仅主机模式下将 linux 连上网络（差点还把网络弄没了）。最后我还是采用了 NAT 模式，更加方便简单。所以在文件传输上，我是采用的直接把文件拷贝或者拖拽，即可实现文件的传输（感觉更加方便，个人

没有看懂 ssh 的用处)。

设备	摘要
内存	2 GB
处理器	4
硬盘 (SCSI)	20 GB
CD/DVD (SATA)	自动检测
网络适配器	NAT
USB 控制器	存在
声卡	自动检测
打印机	存在
显示器	自动检测

名称	类型	外部连接	主机连接	DHCP	子网地址
VMnet0	桥接模式	自动桥接	-	-	-
VMnet1	仅主机...	-	已连接	已启用	192.168.138.0
VMnet8	NAT 模式	NAT 模式	已连接	已启用	192.168.13.0

**C.Makefile:** make 命令上学期数据结构 pj 就用过, 所以明白它的基本操作, 也比较清楚它的作用, 用于编译出可执行文件。我用 vim 写了一个 makefile 文件, 执行了 hello 程序, 也能够体会到 make 的强大。

## ②系统基本工具:

**A.sudo:** sudo 是最常用用来安装、root 的工具了, 很多安装需要暂时使用 root 的状态, 一般套路就是“sudo apt-get install xxx”, 用习惯了就很方便了。还有一个“su root”用于切换到 root 用户。比较神奇的是, linux 系统输入密码时没有显示个数, 第一次我以为是卡住了还是出了什么 bug, 后来用习惯了发现这样的安全性确实更高。

```
chty627@ubuntu:~$ su root
Password: 
```

**B.gcc:** gcc 的安装让我有些难受。使用直接安装(sudo)的 gcc 版本特别高, 然而要求使用的 gcc 的版本是 4.8/4.7, 所以我搜索了很多降级和切换版本的教程, 下了很多的功夫才把 gcc 降级成 4.8, 并删除高级版本。

```
gcc version 4.8.5 (Ubuntu 4.8.5-4ubuntu8)
```

```
chty627@ubuntu:~$ sudo update-alternatives --config gcc
[sudo] password for chty627:
There is only one alternative in link group gcc (providing /usr/bin/gcc): /usr/bin/gcc-4.8
Nothing to configure.
```

**C.vim:** vim 的编辑基础应该是重中之重。我阅读的教程是 vintutor。vintutor 最大的好处就是可以一边看教程一边练习, 虽然现在也没能把整个教程记下来, 但是许多常用的操作已经熟练掌握, 因为后面很多文本编辑、修改等操作都需要使用 vim。感觉 vim 是一个万能的编辑器, 远远超出 windows 下的 txt (每次打开就是一堆排版贼乱的乱码)

vim 的操作很规整化, 最大的感受就是利用肌肉记忆进行键盘操作。在方向键 hjkl 中, 其实一开始还是很习惯, 总是会回到上下左右键, 后来感觉只要用熟悉之后, hjkl 似乎还更加方便, 可以配合数字进行一系列删除、移动操作。vim 花费的时间是比较多的, 很多时间花在了阅读教程上面。这里也不可能一一列举我学到的 vim 操作, 我会把 vim 操作放在我使用过程中讲述。这里是用 vim 写了 hello.c。(换了主题色好看!)

```
File Edit View Search Terminal Help
1 #include <stdio.h>
2 int main()
3 {
4     printf("hello,world");
5 }
```

**D.SSH:** ssh 的教程我看的时间是最多的, 最后半天一直在看 ssh。但是我没有看懂 ssh 到底是用来干什么、用处真的很大之类的。然后教程中的 ssh 也并没有用于虚拟机之间的文件传输和 remote 文件传输, 只适用于虚拟机和主机文件传输, 让我觉得有些多次一举。后来也尝试设置了网络, 尝试多次并没有成功最后改为 NAT 模式。

**E.build-essential / gcc-doc / time**

**F.gdb / pwndbg:** 调试神器 gdb, 只需要输入 gdb 命令即可进入调试阶段。用于

在运行.c等文件时可以查看栈情况、反汇编等信息。我这里安装了一个更有用的 pwndbg 配合 gdb 使用，可以更方便地查看栈、汇编等内容。之前已经对汇编语言有很多了解，也清楚二进制炸弹需要使用 gdb 找清楚程序执行具体内容，所以更加侧重的看了 gdb 关于调试部分的教程。比如 run/break/delete/continue 等基本操作，还有 pwndbg 中的可以使用 stack n 查看栈情况。比如下图 hello.c 的调试。

```
pwndbg> file hello
Reading symbols from hello...(no debugging symbols found)...done
pwndbg> disassemble main
Dump of assembler code for function main:
0x000000004004fd <+0>: push rbp
0x000000004004fe <+1>: mov rbp, rsp
0x00000000400501 <+4>: mov edi, 0x4005a4
0x00000000400506 <+9>: mov eax, 0x0
0x0000000040050b <+14>: call 0x4003f0 <printf@plt>
0x00000000400510 <+19>: pop rbp
0x00000000400511 <+20>: ret
End of assembler dump.
pwndbg> break *0x40050b
Breakpoint 1 at 0x40050b

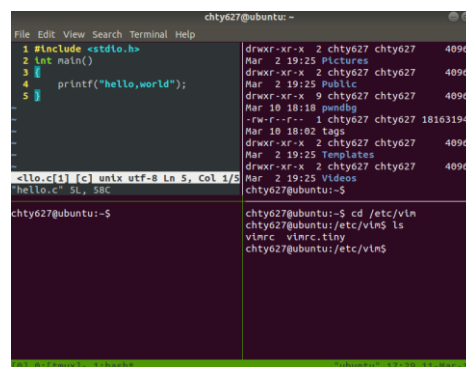
Breakpoint 1, 0x0000000040050b in main ()
LEGEND: STACK | HEAP | CODE | DATA | RMX | RODATA

[ REGISTERS ]
RAX 0x0
RBX 0x0
RCX 0x400520 (libc_csu_init) ← push r15
RDX 0x7fffffff00e0 → 0x7fffffff405 ← 'CLUTTER:IM_MODULE=xlm'
RDI 0x00000000 ← push 0x6f6cc05 /* 'hello,world' */
RSI 0x7fffffff00d0 → 0x7fffffff3f1 ← '/hone/cht627/hello'
R8 0x7ffff7dd0d80 (initial) ← 0x0
R9 0x7ffff7dd0d80 (initial) ← 0x0
R10 0x0
R11 0x1
R12 0x400400 (.start) ← xor ebp, ebp
R13 0x7fffffff00d0 ← 0x1
R14 0x0
R15 0x0
RBP 0x7fffffff00d0 → 0x400520 (libc_csu_init) ← push r15
RSP 0x7fffffff00d0 → 0x400520 (libc_csu_init) ← push r15
RIP 0x40050b (main+14) ← call 0x4003f0

[ DISASM ]
> 0x40050b <main+14>
call printf@plt <0x4003f0>
format: 0x4005a4 ← 'hello,world'
vararg: 0x7fffffff00d0 → 0x7fffffff3f1 ← '/hone/cht627/hello'

0x400510 <main+19> pop rbp
0x400511 <main+20> ret
0x400512 pop word ptr cs:[rax + rax]
```

G. tmux: 多个 terminal 同时运行, 互相不干涉, 但是应该在之后需要对照 terminal 进行 debug 或者调试操作的时候很有用。



H. git: 接下来实验经常用到的记录工具，用 branches 来记录文件发展，有些像一颗树。操作比较简单，git log/git commit 等，只需要掌握几个常规的保存传送版本控制操作就可以在之后起到保存的作用。最后修改 Makefile.git 文件中的学号。

```
commit 2b9fb58a077b1cd03aca850cd9aa31bc971bcc19 (HEAD -> master)
Author: tracer-ics2015 <tracer@njuics.org>
Date: Mon Mar 11 00:12:48 2019 -0700

> compile NEMU
141220000
cht627
Linux ubuntu 4.18.0-16-generic #17-18.04.1-Ubuntu SMP Tue Feb 12 13:35:51 U
TC 2019 x86_64 x86_64 x86_64 GNU/Linux
00:12:48 up 32 min, 2 users, load average: 0.16, 0.07, 0.03
77dec8909ed2d09169dc194b1a2d7b77e10ca493
```

```
File Edit View Search Terminal Help
1 STU_ID = 17307130178
2
3 # DO NOT modify the following code!!!
4
5 GITFLAGS = -q --author='tracer-ics2015 <tracer@njuics.org>' --no-verify -
llow-empty
6
7 # prototype: git commit(msg)
8 define git_commit
9     @git add . -A --ignore-errors
10     @while (test -e .git/index.lock); do sleep 0.1; done
11     @echo "> $1" && echo $(STU_ID) && id -un && uname -a && uptime &&
head -c 20 /dev/urandom | hexdump -v -e '%02x' && echo | git commit -f
- $(GITFLAGS)
12 endef
13
```

I. ctags: 以后准备慢慢研究!

### ③修改设置:

**A.APT sources:** 添加下载的镜像网站, 多添加了几个库。这里重点在于 vim 和 ubuntu 命令行的运用, 在使用 ls -l 可以查看详细文件信息后, 可以发现有些文件是只读模式, 需要 root 才能有修改权限, 这也是为什么要强调不能习惯用 root 模式操作系统, 很容易发生把重要文件修改删除后引起 bug 的情况。使用“sudo vim xxx”命令打开这个只读文件, 就可以进行“:w!”的重写操作了。这里的操作是在原来的库中新添几个下载源地址。最后使用“apt-get update”就可以更新源库了。

**B.configure vim:** 按照教程的做法, 我使用 sudo vim /etc/vim/vimrc 修改了 vim 的设置, 主题和教程改的有些不一样, 也确实好看些!

**C.Host-only Network:** 这是没有尝试成功的部分, 上面讲 ssh 也提到了。我的网络设置如下。我最后选择使用了 NAT 模式链接网络。安装了一下 net-tools。

```
chty627@ubuntu:~$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.138.133 netmask 255.255.255.0 broadcast 192.168.138.255
    inet6 fe80::3f82:3176:5a24:8466 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:24:a2:db txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 1568 (1.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 322 bytes 22181 (22.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 661 bytes 56277 (56.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 661 bytes 56277 (56.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

## (2)、观点与更新

PA0 中能够领悟到很多新的观点:

①**RTFM:**这个应该是印象最深刻的观点了。PA0 的教程写得很生动有趣, 有些地方让人不禁一笑。Read The Fucking Manual, 很多时候别人问题的时候我也会陷入这种僵局, 又不想讲又不得不讲、但明明答案只需要翻翻书或者百度 google 一下就出的来的事情, 让人抓狂地狂想着“这人怎么不会看书”“为什么 ta 就不能直接搜索百度”, 我相信以后的学习也要避免问这种白痴问题, 先阅读手册, 利用掌握好的 man 操作, 如果看不懂再用百度 google, 再问同学、老师, 应该是这样的一个循环学习的方法。当然一些重点部分也应该先倾听助教和老师的意见, 以免陷入学习误入僵局的情况。

②**linux 系统的强大:** linux 和 windows 和 OS X 几个系统, 侧重点不同, 像 OS X 感觉是以用户为主的系统, 内置 UI 实在是更符合主流审美观, 内核也比 windows 简化很多。windows 实在是太拥挤、臃肿了, win10 的 UI 其实已经优化很多了, 但是作为一个拥挤的系统, 安装各种了盗版、垃圾捆绑软件后, windows 还是变得肮脏不堪。linux 作为轻量级的系统, 最方便的就是 terminal 的用处比 windows 和 OS X 都要高级, 还有安装各个工具只需要 sudo apt-get install 短短一行代码, 也是方便了好多好多。目前体会来说, linux 的安全性比 windows 和 OS X 要好一点。UI 方面 linux 更像是以前的安卓机, 可能是用习惯了苹果和 win10 的 UI 界面, 18.0.4 版本的 UI 已经算是做的比较智能了。

## 三、总结

总体来说, 我认为我的 PA0 的完成度很高, 并且有一定的知识扩展性。这次实验让我更加深刻的了解认识到了 linux 系统及部分工具的使用, 一下子掌握了好多工具的使用方法。当然, 之后也需要加强练习才能把这些所有的知识学以致用, 成功融会贯通。

期待下一次与你见面。