# 实验 0（PA0）实验前的准备

## 一. 实验环境

NEMU 基于 Linux/GNU 实验环境，所需要的环境包括：

1. 天津大学虚拟仿真实验平台（使用说明见附件 1）。注意：推荐大家直接在该平台上进行实验开发，最后**实验均会在该平台上进行测试验收**。如果大家使用自己的环境进行实验，最后仍需要把实验提交到该平台之上。

2. 操作系统系统：Ubuntu18.04；编译器：GCC-4.4.7。实验所需的所有库和可能会用到的工具 (包括：编辑器 Vim、版本控制 Git、代码阅读辅助工具 Ctags、终端分屏工具 tmux) 都以在虚拟仿真实验平台安装好，可直接配置使用。注意：如果同学有意愿自己配置实验环境（附件 2），为保证实验的正常进行，**OS 及编译器版本需要与上面的一致**，其他版本理论没有问题，但并未经过测试。

## 二. NEMU 课程设计所需要的参考资料：

### 2.1 必读资料

(1). **NEMU 实验指导书**（已上传至虚拟仿真实验平台）

(2). **i386 程序员手册**（已上传至虚拟仿真实验平台）

(3). 参考书“袁春风，**计算机系统基础**，机械工业出版社”（第 1、2 版均可）

### 2.2 其它资料

(1). Linux 入门教程（附件 3）

(2). git 入门教程（附件 4）

(3). man 入门教程（附件 5）

(4). i386 手册勘误（附件 6）

(4). make 手册

## 三. 实验基本步骤

### 3.1 获取 NEMU 工程代码

(1). 使用自己的账户/密码登录到虚拟仿真实验平台。

(2). 进入实验系统

(3). 双击终端图标（或使用快捷键“Ctrl+Alt+t”）打开终端 terminal。

(4). 输入如下命令，进入 home 目录（注意：不要使用 root 账户进行实验）。

```
cd ~
```

(5). 获取 NEMU 工程源代码，命令如下：

```
git clone https://gitee.com/wjztju/NEMU2021.git
```

在当前目录下会创建一个名为“NEMU2021”的目录，即 NEMU 工程目录。

### 3.2 Git 配置

(1). 进入 NEMU2021 目录，完成 Git 基本配置，命令如下：

```
cd NEMU2020
git config --global user.name "2018216008-Zhang San" # 学号和姓名
git config --global user.email "zhangsan@foo.com" # 邮箱
git config --global core.editor vim # 选用的编辑器（可更换为 gedit）
git config --global color.ui true
```

注：更多有关 Git 的使用请参考附件 4，或进行百度。

(2). 编辑 config/Makefile.git 文件，修改“STU_ID”为自己的学号。(**注意：必须修改学号，提交打包时会使用该学号，否则影响成绩后果自负!!!**)

```
vim config/Makefile.git
```

(3). 输入如下命令完成第一次 git 提交。

```
git add .
git commit -m “modified my STU_ID”
```

### 3.3 编译、运行 NEMU

(1). NEMU 实现的是一台基于 IA-32 指令集的虚拟机，在终端中输入如下命令完成 NEMU 的编译和运行：

**make**

**make run**

此时，终端中显示如下错误信息：

```
nemu: nemu/src/cpu/reg.c:21: reg_test: Assertion `(cpu.gpr[check_reg_index(i)]._16) == (sample[i] & 0xffff)' failed.
```

这条信息显示程序在“**nemu/src/cpu/reg.c**”文件的第 21 行触发了一个断言错误（提示：若不清楚什么是断言，请查询相关程序设计书籍或网上资料）。这个断言错误是故意触发的，它用来对 IA-32 的寄存器组织结构进行测试，目前，NEMU 中实现的寄存器结构是错误的。我们将在 PA1 中来修复这个错误，目前可以忽略这个错误。**现在，NEMU 及相关开发环境已经安装配置成功。**

(2). 如果想使用 gdb 对 NEMU 进行调试，可在工程目录下输入如下命令：

**make gdb**

(3). 若对 NEMU 中的源代码进行了修改，则在重新编译前，输入如下命令，删除之前编译所生成的各类中间文件。

**make clean**

### 3.4 开发过程的跟踪

NEMU 工程的开发过程会被 git 所跟踪。同学每次的编译、运行、调试、测试动作均会被记录在本地 git 日志中，在工程路径下可通过如下命令查看：

**git log**

如果可看到类似如下信息，则说明跟踪成功，日志被记录。

```
commit 4072d39e5b6c6b6837077f2d673cb0b5014e6ef9
Author: tracer-NEMU2020 <tjuics2020@163.com>
Date: Sun Aug 16 11:30:31 2020 +0800
> compile NEMU
20192160000
user
Linux debian 3.16.0-4-686-pae #1 SMP Debian 3.16.7-3 i686 GNU/Linux
14:30:31 up 3:44, 2 users, load average: 0.28, 0.09, 0.07
3860572d5cc66412bf85332837c381c5c8c1009f
```

否则，如果看到如下信息，则说明日志没有别成功记录

```
fatal: Unable to create '/home/user/NEMU2020/.git/index.lock': File exists.
If no other git process is currently running, this probably means a
git process crashed in this repository earlier. Make sure no other git
process is running and remove the file manually to continue.
```

此时，可尝试清空之前的编译结果，再次重新编译，命令如下。如果这类问题总是出现，请尽快联系老师。

**make clean**

**make**

**特别说明（重要！重要！重要！）：开发跟踪**

我们使用 Git 对你的实验过程进行跟踪，不合理的跟踪记录会影响你的成绩。如果我们发现 git log 没有任何更新，将被视为没有完成实验。git log 是独立完成实验的最有力证据，完成了实验内容却缺少合理的 git log，不仅会损失大量分数，还会给抄袭判定提供最有力的证据。因此，请你注意以下事项：

- 请你不定期查看自己的 git log，检查是否与自己的开发过程相符。
- 提交其它同学代码或网上找到的答案将被视为没有提交。
- 不要把你的代码上传到公开的地方。
- 总是在工程目录下进行开发，不要在其它地方进行开发，然后一次性将代码复制到工程目录下，这样 git 将不能正确记录你的开发过程。
- 不要修改 Makefile 中与开发跟踪相关的内容.
- 不要清除 git log。
- 偶然的跟踪失败不会影响你的成绩。如果上文中的错误信息总是出现，请尽快联系我们。

## 3.5 版本控制

虽然 git 可以跟踪你的开发痕迹，但这些痕迹记录还不足以支撑你的开发过程。在开发过程中你的代码总是存在 bug 的，你几乎不可能编写一次代码就成功完成所有实验任务。因此，最好的方法就是进行阶段性测试和阶段性的版本保

存。每当你成功完成一个子任务，你都可以利用 git 将这个阶段性版本提交至本地仓库，这就是一种简单的版本控制。这样做的好处就是你可以在任何需要的时候回溯到你之前的某个工程代码状态。上述工作需要你手工完成，命令如下：

**git add .**

**git commit --allow-empty**

输入“commit”命令后会弹出已设定的编辑器窗口（例如，vim），你可以在其中输入对当前版本的描述信息，以便于今后查找。如果版本描述信息比较简单，也可以在“commit”命令后使用-m 选项，直接输入描述信息。此外，“--allow-empty”是一个可选的选项。因为，有些时候你对代码的改变已经被 git 系统自动记录了，此时如果不使用“--allow-empty”选项，git 是无法完成本地提交的。如果最终提交成功，通过如下“log”命令你可以找到使用你的学号和姓名标注的日志信息。

**git log**

如果你想在众多日志信息中定位你手工提交的信息，可以在 git 的“log”命令中添加“author”选项。更多使用细节请阅读附件 4 或百度查询。

注意：版本控制是复杂软件系统开发必备技能。虽然在 NEMU 课程设计中我们不对其进行强制要求，但鉴于 NEMU 本身的复杂程度，强烈推荐大家对自己的工程进行版本控制，会大大节省你维护代码的成本，提高代码开发效率。相信你坚持下来的话，会受益匪浅！

## 四. 实验中会使用的相关工具

### 4.1 编辑工具

vim 是 Linux 系统下首选编辑工具，被誉为“编辑器之神”。你可以使用 vim 完成 NEMU 所有的设计任务。也许你更喜欢带有 GUI 的编程环境（例如 Visual Studio），然而，在一些情况下你是无法使用 GUI 的，特别是当你登录到一个远程服务器进行程序开发工作的时候。

- 远程服务器上没有安装带有 GUI 的编程环境。
- 网络条件不允许你使用任何 GUI 工具。

在这些情况下，vim 是则是最好的选择（熟练掌握 vim 后，其编辑效率仍然高于

GUI 工具)。如果你更喜欢使用 emacs，你也可以下载安装 emacs。

大多数同学都是第一次使用 vim。不可否认，vim 相比其它编辑器学习难度更大。因此，你需要一个入门教程，下面推荐两种获取教程的方式：

- 在终端中输入命令“vimtutor”，即可打开一个 vim 教程。推荐大家使用这种方式，因为你可以一遍阅读教程一遍进行联系。
- 在网上输入关键词“vim 教程”，你会找到大量有关 vim 的使用介绍。

**请大家记住反复练习是最关键的。仅仅阅读文档是学不会任何东西的。**为了增加学习的趣味性，下面给出了一些很有意思的小游戏帮助你掌握 vim 的基本操作。

- Vim Snake
- Open Vim Tutorials
- Vim Genius

**vim 很强大**

第一次使用 vim 的你一定很困惑，如此“糟糕”的编辑器能做什么。让我们来看这样一个例子：产生如下的文件

```
1
2
3
.....
98
99
100
```

这个文件有 100 行，每行为一个数字。你应该如何编辑？对于 vim 而言，这简直是小菜一碟。首先让 vim 处于普通模式(vim 打开时默认就是普通模式)，然后在键盘上敲击如下按键：

```
i1<ESC>q1yyp<C-a>q98@1
```

其中，<ESC>表示 ESC 按键，<C-a>表示“Ctrl”和“a”按键同时按下。你仅仅敲击了 15 个按键就生成这样一个文件，是不是很神奇？如果文件有 1000 行呢？你只需要再多按一个按键，将“98”变为“998”。这个例子背后所隐藏的就是 vim 的记录和回放功能。你初始化文件的第 1 行，然后如何产生第 2 行，接着就是回访这个记录 998 次即可。虽然这是一个人为设计的例子，但它显示了 vim 十分强大的编辑功能，远超过你用过的其它编辑器。

由于和虚拟仿真平台快捷键冲突，本实验 Vim 中的“**Esc**”键由“**Ctrl+c**”替代。

## 4.2 代码阅读辅助工具

NEMU 是一个规模比较大，复杂程度较高的工程，由几十个文件组成。在阅读源代码的过程中，你势必要寻找某个语法要素，如变量、结构体、函数等，在哪个文件的什么位置。此时，你需要专门的代码阅读辅助工具。在 Linux 环境下，结合 vim，推荐使用 ctag 工具。目前虚拟仿真平台已经安装好了 ctag，大家可以直接使用。亦可以通过如下命令自行安装。ctag 的使用方法可参考教程，也可上网自行搜索。

```
sudo apt-get install ctags
```

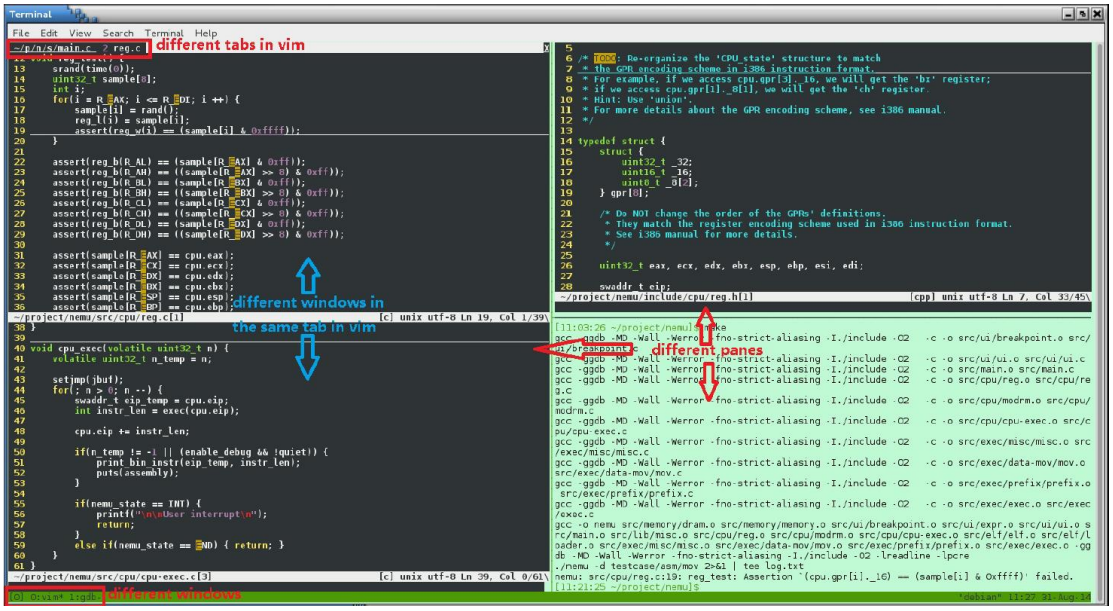
此外，Source Insight 也是一个推荐的代码阅读工具（从这里下载，密码：8k3i），但该工具只有 window 版本，如果想在 Linux 系统下使用，需要先安装 wine 工具，命令如下：

```
sudo apt-get install wine
```

安装成功后就可以在 Linux 系统下使用 Windows 的软件了。大家可以根据自己的需要选用上述两款代码阅读辅助工具。

## 4.3 终端复用器 tmux

tmux 可以将一个终端切分成多个窗口，便于同时编辑、操作多个文件。例如，你可以在一个窗口编辑代码，另一个窗口编译、运行，如下图所示。




目前虚拟仿真平台已经安装好了 tmux，大家可以直接使用。亦可以通过如下命令自行安装。tmux 的常用命令可参考教程，也可上网自行搜索。

**sudo apt-get install tmux**

## 实验小贴士

1. NEMU 实验对于大家是一个不小的挑战。在实验过程中你会遇到各种问题，其中最主要的问题就是不熟悉 GNU/Linux 开发环境，包括：新的 shell 命令、新的库函数等。为了克服这个困难，请大家记住 GNU/Linux 下最常用到的命令“man”——在线手册(见附件 3)。man 可以告诉你其它的命令是如何使用的。记住，**学会了使用 man，你就学会了任何东西**。因此，如果你想知道有关 GNU/Linux 下的事情（例如，shell 命令、库函数、系统调用、设备文件、配置文件...），读“该死”的手册。

2. 善于使用互联网搜索各种问题答案，例如工具使用、库函数的使用等。常用的一些问答网站或技术网站包括：stack overflow、CSDN 等。

3. 切记不要使用 root 账户做实验!!! 如果你仍然不理解为什么要这样做，你可以阅读这个页面: Why is it bad to login as root? 正确的做法是: 永远使用你的普通账号做那些安分守己的事情(例如写代码)，当你需要进行一些需要 root 权限才能进行的操作时，使用 sudo。

4. 在阅读 NEMU 代码的时候，如果你无法找到某个函数的定义，它很有可能是一个库函数。你可以通过 man 或上网查询获取相关解释。

5. 如果你无法理解代码所模拟的硬件细节，那么请认真阅读 i386 手册，**读“该死”的手册**。

6. NEMU 的全部实验均是使用 **C 语言**完成。这里是一个十分优秀的 C 语言入门手册，它不仅包含了 C 语言的介绍（例如如何使用 printf()和 scanf()），还包括了有关计算机系统的相关知识（例如数据结构、计算机系统结构、汇编、链接、操作系统、网络等）。强烈推荐你阅读这个入门手册。

最后，希望大家享受 NEMU 之旅，通过整个实验最终你会发现受益良多，你会发现计算机系统不再神秘、不再难以理解，你会发现计算机系统是如此精妙！但是，请牢记一下三个原则：

- **机器永远是对的！**
- **未经过测试的每行代码永远是错误的！**
- **读“该死”的手册！**