实验 1：Linux 编程基础实验

一、实验目的

1. 学会自己安装 Linux 系统；
2. 学会配置简单的 Linux 开发环境；
3. 在 Linux 下完成简单编程练习并熟悉各种命令行工具的使用方法。

二、实验内容

1. Linux 安装和配置

i. 请分别按照 Linux 安装和 Linux 配置按步骤完成 VirtualBox 和 Linux 系统安装和环境配置。

要求 1：安装 32 位版本的 Ubuntu 并配置好所有工具。

ii. 请使用 man 查询 Vim/Git/GCC/AS/OBJDUMP/GDB 版本的命令，然后使用查到的命令打印出对应版本，将版本截图贴在实验报告中。

iii. 写下你在安装过程中遇到的问题，并说明你是如何解决的（没有可不写）。

iv. 机房的计算机中已有 Ubuntu 平台，也可以直接在 Ubuntu 平台下实验（用户名 ubuntu，密码 6 个 1）。

2. Linux 下的编程实践

i. 使用文件管理器或者 mkdir 在主目录下创建工作目录 workspace。

```
cd ~
mkdir workspace
```

ii. 在 workspace 目录下创建目录 lab01，在 lab01 下创建以自己学号为名的目录（如 151220000，后面的同理）。

```
cd workspace
mkdir lab01
cd lab01
mkdir 学号（如 151220000）
```

iii. 将工作目录切换为~/workspace/lab01/学号（如 151220000），然后使用 Gedit 或 Vim 编辑器编写汇编语言的表白程序（可以直接修改下图中的代码让它打印出一个爱心），并保存为 heart.S（程序代码参考**汇编编译工具 as**，或下图）。

```
.text
    .global  main

main:
    movl $len, %edx
    movl $msg, %ecx
    movl $1, %ebx
    movl $4, %eax
    int $0x80

    movl $0, %ebx
    movl $1, %eax
    int $0x80

.data
msg:
    .asciz "hello, world\n"
    len = . - msg
```

要求：爱心形状如下，前后不要有多余的空格，爱心之后紧跟自己的学号，最后一行为空行（请严格遵守格式，助教的自动批改脚本非常不智能）。

touch heart.S # 创建文件

编辑 heart.S

as -o heart.o heart.S # 编译

gcc -o heart heart.o # 链接

./heart # 运行

TIPS：汇编的编译链接命令的详细介绍在教程汇编编译工具 as 中。**请提交修改的 heart.S 代码文件，而不是反汇编文件。正确的输出样例如下：**

```
 *     *
*******
 *****
   *
151220000
```

3. 熟悉工具：

i. 使用 objdump 的-D 选项反汇编 heart.o 文件，找到你学号的位置并截图（提示：ASCII 码数字的 16 进制）。

objdump –D heart.o > dog

cat dog

要求：在实验报告中贴出截图，并说明哪里是你的学号。

ii. 编写简单的 C 语言源程序 hello.c，通过预处理、编译、汇编、链接四个步骤将 C 语言源程序转换为可执行文件，即 hello.c -> hello.i -> hello.s -> hello.o -> hello

要求：在实验报告中贴出截图，每个步骤的命令。

iii. 通过创建或修改~/.vimrc 文件，使你的 vim 支持显示行号。

set nu

要求：请自行查找如何修改，无需写在实验报告。

4. 数据的表示范围及不同类型的数据长度实验。

代码：

sqr.c

```
#include <stdio.h>
int main()
{
    int i,j;
    i=40000;
    j=i*i;
    printf("The 40000*40000 is %d\n", j);
    i=50000;
    j=i*i;
    printf("The 50000*50000 is %d\n", j);
    return 0;
}
```

gcc –o sqr sqr.c

i. 将输出结果导出，说明发生这种现象的原因？
ii. 寻找在该程序中保证结果正确的最大整数值？（不需要提交这部分的代码，请在实验报告中写答案）
iii. 应如何修改程序，才能保证结果都正确？（请提交修改后的程序代码）

5. 矩阵运算执行时间比较

i. 比较两个矩阵复制函数的执行时间。（请提交可正确编译运行的代码，并将实验结果截图贴在实验报告中）
ii. 说明为什么会出现这个差别。

代码：

matrix.c

```
#include <stdio.h>
#include <time.h>
void copyij(int src[2048][2048], int dst[2048][2048]){
    int i, j;
    for (i = 0; i < 2048; i++){
        for (j = 0; j < 2048; j++){
            dst[i][j] = src[i][j];
        }
    }
}
void copyji(int src[2048][2048], int dst[2048][2048]){
    int i, j;
    for (j = 0; j < 2048; j++){
        for (i = 0; i < 2048; i++){
            dst[i][j] = src[i][j];
        }
    }
}
int src[2048][2048];
int dstij[2048][2048];
int dstji[2048][2048];
int main(){
    int t, m;
    for (t = 0; t < 2048; t++){
        for (m = 0; m < 2048; m++){
            src[t][m] = t + m;
        }
    }
    clock_t startij, finishij, startji, finishji;

    startij = clock();
    copyij(src, dstij);
    finishij = clock();
```

```
    double durationij = (double)(finishij - startij) / CLOCKS_PER_SEC;
    printf("copyij %f s\n", durationij);

    startji = clock();
    copyji(src, dstji);
    finishji = clock();
    double durationji = (double)(finishji - startji) / CLOCKS_PER_SEC;
    printf("copyji %f s\n", durationji);
    return 0;
}
# gcc -o matrix matrix.c
```

【选做】Git 相关选做（不加分也不扣分）

i. 注册并成为 GitHub 的一员，为 Ubuntu 系统配置 git 的 name 和 Email，并在 github 上添加 Ubuntu 系统的 ssh-key，Git 的用法可以通过上文的教程学会（推荐廖雪峰的教程，通读教程并学会 Git 的基本操作大约需要 2 小时）。

ii. 使用 Git 对实验 1 进行代码版本管理：

1. 使用 git init 初始化~/workspace/lab01/151220000 为版本库，然后使用 git 记录你认为有意义的代码改动（推荐每实现一个功能进行一次提交）。

```
cd ~/workspace/lab01/151220000
git init
git add <filename>
git commit –m '<what you changed>'
```

2. 学习使用.gitignore 将不需要进行版本追踪的文件（例如 pdf，word 等）从 Git 追踪中排除。

提交要求：

请在规定时间内提交一个以学号为名的压缩文件，如 151220000.zip 到课程网站（注意修改学号和压缩格式，不接受过期提交）。压缩包内部应该是一个目录。

压缩文件解压后获得目录内容如下（注意文件名大小写和每一个文件的提交要求）：

```
151220000
    |----heart.S
    |----sqr.c
    |----matrix.c
    |----report.pdf
```