实验1：Linux 编程基础实验

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

实验内容

1.Linux安装和配置

- 关于安装：因为我在课前安装了Ubuntu19.04真机，所以不再安装虚拟机，安装过程没有遇到什么问题。
- 查询命令：（以vim为例)
  在终端中使用 `man vim` 查看手册命令，搜索version， 找到如下说明

```
 --version   Print version information and exit.
```

所以使用`--version`命令来查看vim版本，下面贴上vim/git/gcc/as/objdump/gdb版本截图

```
katherine@ubuntu:~$ vim --version
VIM - Vi IMproved 8.1 (2018 May 18, compiled Nov 03 2018 00:15:14)
Included patches: 1-320
Modified by team+vim@tracker.debian.org
Compiled by team+vim@tracker.debian.org
Huge version with GTK3 GUI.  Features included (+) or not (-):
+acl               +extra_search      +mouse_netterm     +tag_old_static
+arabic            +farsi             +mouse_sgr         -tag_any_white
+autocmd           +file_in_path      -mouse_sysmouse    +tcl
+autochdir         +find_in_path      +mouse_urxvt       +termguicolors
-autoservername    +float             +mouse_xterm       +terminal
+balloon_eval      +folding           +multi_byte        +terminfo
+balloon_eval_term -footer            +multi_lang        +termresponse
+browse            +fork()            -mzscheme          +textobjects
++builtin_terms    +gettext           +netbeans_intg     +timers
+byte_offset       -hangul_input      +num64             +title
+channel           +iconv             +packages          +toolbar
+cindent           +insert_expand     +path_extra        +user_commands
+clientserver      +job               +perl              +vartabs
+clipboard         +jumplist          +persistent_undo   +vertsplit
+cmdline_compl     +keymap            +postscript        +virtualedit
+cmdline_hist      +lambda            +printer           +visual
+cmdline_info      +langmap           +profile           +visualextra
+comments          +libcall           -python            +viminfo
+conceal           +linebreak         +python3           +vreplace
+cryptv            +lispindent        +quickfix          +wildignore
+cscope            +listcmds          +reltime           +wildmenu
+cursorbind        +localmap          +rightleft         +windows
+cursorshape       +lua               -ruby              +writebackup
+dialog_con_gui    +menu              +scrollbind        +X11
+diff              +mksession         +signs             -xfontset
+digraphs          +modify_fname      +smartindent       +xim
+dnd               +mouse             +startuptime       +xpm
-ebcdic            +mouseshape        +statusline        +xsmp_interact
+emacs_tags        +mouse_dec         -sun_workshop      +xterm_clipboard
+eval              +mouse_gpm         +syntax            -xterm_save
+ex_extra          -mouse_jsbterm     +tag_binary
```

```
katherine@ubuntu:~$ git --version
git version 2.20.1
katherine@ubuntu:~$ gcc --version
gcc (Ubuntu 8.3.0-6ubuntu1) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

katherine@ubuntu:~$ as --version
GNU assembler (GNU Binutils for Ubuntu) 2.32
Copyright (C) 2019 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or later.
This program has absolutely no warranty.
This assembler was configured for a target of `x86_64-linux-gnu'.
katherine@ubuntu:~$ objdump --version
GNU objdump (GNU Binutils for Ubuntu) 2.32
Copyright (C) 2019 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.
This program has absolutely no warranty.
katherine@ubuntu:~$ gdb --version
GNU gdb (Ubuntu 8.2.91.20190405-0ubuntu3) 8.2.91.20190405-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

2.Linux下的编程实践

- 修改后的代码为

heart.S

```
.code32
.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 " *   *\n*******\n *****\n   *\n181840326\n\n"
    len = . - msg
```

- 输出为

```
katherine@ubuntu:~/workspace/lab01/181840326$ as --32 -o heart.o heart.S
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -o heart heart.o
katherine@ubuntu:~/workspace/lab01/181840326$ ./heart
 *   *
*******
 *****
   *
181840326
```

3.熟悉工具

- 首先查阅ASCII码，0-9的十六进制分别为0x30-0x39

0x30	0
0x31	1
0x32	2
0x33	3
0x34	4
0x35	5
0x36	6
0x37	7
0x38	8
0x39	9

找到学号（181840326）的位置，是图中的31-36。根据我的代码推断0a是换行符。

```
 1a:   0a 31                   or     (%ecx),%dh
 1c:   38 31                   cmp    %dh,(%ecx)
 1e:   38 34 30                cmp    %dh,(%eax,%esi,1)
 21:   33 32                   xor    (%edx),%esi
 23:   36 0a 0a                or     %ss:(%edx),%cl
```

- 编写简单的c语言源程序hello.c,并预处理、编译、汇编、链接。

```
katherine@ubuntu:~/workspace/lab01/181840326$ vim hello.c
katherine@ubuntu:~/workspace/lab01/181840326$ cat hello.c
#include <stdio.h>
int main(){
    printf("hello, world\n");
    return 0;
}
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -E hello.c -o hello.i
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -S hello.i
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -c hello.s
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 hello.o -o hello
katherine@ubuntu:~/workspace/lab01/181840326$ ./hello
hello, world
```

- 使vim支持显示行号

4.数据的表示范围及不同类型的数据长度实验

- 导出输出结果,发现当i为50000时结果变成了负数。

```
katherine@ubuntu:~/workspace/lab01/181840326$ touch sqr.c
katherine@ubuntu:~/workspace/lab01/181840326$ vim sqr.c
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -o sqr sqr.c
katherine@ubuntu:~/workspace/lab01/181840326$ ./sqr
The 40000*40000 is 1600000000
The 50000*50000 is -1794967296
```

原因是计算机的表示法是用有限数量的位来对一个数字编码，因此，当结果太大以至不能表示时，某些运算就会溢出。int型用四个字节表示，而50000*50000的结果超出了int 所能表示的范围。

- 在该程序中，int型能表示的最大数字是

$$2^{31} - 1 = 2147483647$$

要保证结果正确，i*i<=2147483647,i最大为46340

- 要保证结果都正确，只需将数据类型改为long long。

```
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -o sqr sqr.c
katherine@ubuntu:~/workspace/lab01/181840326$ ./sqr
The 40000*40000 is 1600000000
The 50000*50000 is 2500000000
```

5.矩阵运算执行时间比较

- 比较两个矩阵复制函数的执行时间

```
katherine@ubuntu:~/workspace/lab01/181840326$ vim matrix.c
katherine@ubuntu:~/workspace/lab01/181840326$ gcc -m32 -o matrix matrix.c
katherine@ubuntu:~/workspace/lab01/181840326$ ./matrix
copyij 0.019156s
copyji 0.171906s
```

- 从程序的执行结果可以看出，第一种复制方法即copyij所用时间远小于copyji,即按行复制所用时间远小于按列复制。因为二维数组的本质是一维数组，其在内存中的存放是按照行存放的。copyij只需遍历内存一次便完成全部复制，而copyji每复制一次要跳过一行，大大增加了遍历内存的次数，故时间也大大增加。

选做：用git进行版本管理