

## 实验 1 Linux 编程基础实验

姓名: 林荣恩 学号: 191220060 院系: 计算机科学与技术系

### 1. Linux 安装和配置

用 man 查询 Vim/Git/GCC/AS/OBJDUMP/GDB 版本的命令, 打印出对应版本。

由于查询内容过长, 截屏只取版本的部分内容。

Vim:

```
linrongen@linrongen: ~/workspace/lab01/191220060
linrongen@linrongen:~/workspace/lab01/191220060$ vim --version
VIM - Vi IMproved 8.1 (2018 May 18, compiled Apr 15 2020 06:40:31)
包含补丁: 1-2269
修改者 team+vim@tracker.debian.org
编译器 team+vim@tracker.debian.org
巨型版本 无图形界面。 可使用(+)与不可使用(-)的功能:
+acl                -farsi                -mouse_sysmouse    -tag_any_white
+arabic              +file_in_path          +mouse_urxvt       -tcl
+autocmd              +find_in_path          +mouse_xterm       +termguicolors
+autochdir            +float                 +multi_byte        +terminal
+autoservername       +folding               +multi_lang        +terminfo
-balloon_eval         -footer                -mzscheme          +termresponse
+balloon_eval_term   +fork()                +netbeans_intg     +textobjects
-browse               +gettext               +num64             +textprop
++builtin_terms       -hangul_input          +packages          +timers
+byte_offset          +iconv                 +path_extra        +title
+channel              +insert_expand         -perl              -toolbar
+cindent              +job                   +persistent_undo   +user_commands
-clientserver         +jumplist              +postscript        +varargs
-clipboard            +keymap                +printer           +vertsplit
+cmdline_compl        +lambda                +profile           +virtualedit
+cmdline_hist         +langmap               -python            +visual
+cmdline_info         +libcall               +python3           +visualextra
+comments             +linebreak             +quickfix          +viminfo
+conceal              +lispindent            +reltime           +vreplace
+cryptv               +listcmds              +rightleft         +wildignore
+cscope               +localmap              -ruby              +wildmenu
+cursorbind           -lua                   +scrollbind        +windows
+cursorshape          +menu                  +signs             +writebackup
+dialog_con           +mksession             +smartindent       -X11
+diff                 +modify_fname          +sound             -xfontset
+digraphs             +mouse                 +spell             -xim
```

Git:

```
linrongen@linrongen:~/workspace/lab01/191220060$ git --version
git version 2.25.1
```

GCC:

```
linrongen@linrongen:~/workspace/lab01/191220060$ gcc --version
gcc (Ubuntu 9.3.0-10ubuntu2) 9.3.0
Copyright (C) 2019 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.
```

AS:

```
linrongen@linrongen:~/workspace/lab01/191220060$ as --version
GNU 汇编器 (GNU Binutils for Ubuntu) 2.34
Copyright (C) 2020 Free Software Foundation, Inc.
本软件是自由软件; 您可以使用 GNU General Public License 版本 3 或更新版本重新发放。
本软件完全不带有任何保证。
为目标"x86_64-linux-gnu"配置汇编程序。
```

OBJDUMP:

```
linrongen@linrongen:~/workspace/lab01/191220060$ objdump --version
GNU objdump (GNU Binutils for Ubuntu) 2.34
Copyright (C) 2020 Free Software Foundation, Inc.
这个程序是自由软件; 您可以遵循 GNU 通用公共授权版本 3 或
(您自行选择的) 稍后版本再发布它。
这个程序不含任何担保。
```

GDB:

```
linrongen@linrongen:~/workspace/lab01/191220060$ gdb --version
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
Copyright (C) 2020 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.
```

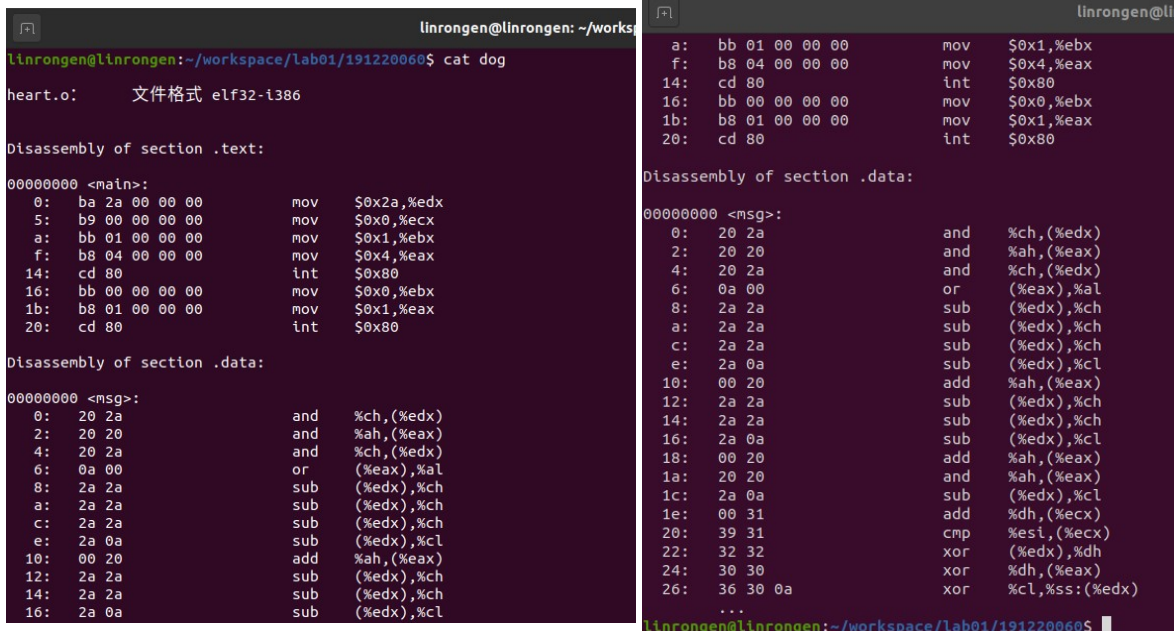
## 2. Linux 下的编程实践

见 heart.S.

## 3. 熟悉工具

I. 使用 objdump 的 -D 选项反汇编 heart.o 文件，找到学号的位置并截图。

dog 的截图如下。其中，第二张图中，从 “1e:” 开始的行到 “26:” 开始的行中，连续的 “31 39 31 32 32 30 30 36 30” 为学号。



```
linrongen@linrongen:~/workspace/lab01/191220060$ cat dog
heart.o:      文件格式 elf32-i386

Disassembly of section .text:

00000000 <main>:
0:  ba 2a 00 00 00      mov     $0x2a,%edx
5:  b9 00 00 00 00      mov     $0x0,%ecx
a:  bb 01 00 00 00      mov     $0x1,%ebx
f:  b8 04 00 00 00      mov     $0x4,%eax
14: cd 80              int     $0x80
16:  bb 00 00 00 00      mov     $0x0,%ebx
1b:  b8 01 00 00 00      mov     $0x1,%eax
20:  cd 80              int     $0x80

Disassembly of section .data:

00000000 <msg>:
0:  20 2a              and     %ch,(%edx)
2:  20 20              and     %ah,(%eax)
4:  20 2a              and     %ch,(%edx)
6:  0a 00              or      (%eax),%al
8:  2a 2a              sub     (%edx),%ch
a:  2a 2a              sub     (%edx),%ch
c:  2a 2a              sub     (%edx),%ch
e:  2a 0a              sub     (%edx),%cl
10: 00 20              add     %ah,(%eax)
12: 2a 2a              sub     (%edx),%ch
14: 2a 0a              sub     (%edx),%cl
18: 00 20              add     %ah,(%eax)
1a: 20 20              and     %ah,(%eax)
1c: 2a 0a              sub     (%edx),%cl
1e: 00 31              add     %dh,(%ecx)
20: 39 31              cmp     %esi,(%ecx)
22: 32 32              xor     (%edx),%dh
24: 30 30              xor     %dh,(%eax)
26: 36 30 0a          xor     %cl,%ss:(%edx)
...
```

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.

```
linrongen@linrongen:~/workspace/lab01/191220060$ gcc -m32 -E hello.c -o hello.i
linrongen@linrongen:~/workspace/lab01/191220060$ gcc -m32 -S hello.i
linrongen@linrongen:~/workspace/lab01/191220060$ gcc -m32 -c hello.s
linrongen@linrongen:~/workspace/lab01/191220060$ gcc -m32 hello.o -o hello
linrongen@linrongen:~/workspace/lab01/191220060$ ./hello
hello, world
```

## 4. 数据的表示范围及不同类型的数据长度实验

I. 将输出结果导出，说明发生这种现象的原因？

输出结果：

```
The 40000*40000 is 1600000000
The 50000*50000 is -1794967296
```

原因：此处 int 类型能表示的范围为 -2147483648~2147483647. 对 40000\*40000 而言，结果落在了可表示的范围内，输出正常；对 50000\*50000 而言，乘法结果溢出。

II. 寻找在该程序中保证结果正确的最大整数值。

最大数值为  $\text{int}(\text{sqrt}(2147483647))=46340$ .

III. 应如何修改程序,才能保证结果都正确?

见 `sqr.c`.

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。

```
copyij 0.022430 s
copyji 0.226088 s
```

对相同的矩阵, `copyij` 的执行时间显著小于 `copyji`.

II. 出现这一差别的原因:

由于二维数组的存储方式是线性的 (`a[0][0]`, `a[0][1]`, ..., `a[0][m]`, `a[1][0]`, ..., `a[1][m]`, ...) , 在 `copyij` 中 `a[i][j]` 与 `a[i][j+1]` 的存储地址相邻, 而 `copyji` 中由 `a[j][i]` 到 `a[j+1][i]` 的距离则更远, 单次寻址需要花费的时间更长。故后者执行时间更长。