1. Linux 系统安装与配置

```
river@ubuntu:~$ vim --version
VIM - Vi IMproved 8.0 (2016 Sep 12, compiled Mar 18 2020 18:29:15)
Included patches: 1-1453
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Compiled by pkg-vim-maintainers@lists.alioth.debian.org
Huge version without GUI.  Features included (+) or not (-):
+acl               +farsi             +mouse_sgr         -tag_any_white
+arabic            +file_in_path      -mouse_sysmouse    -tcl
+autocmd           +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              +vertsplit
-clipboard         +jumplist          +persistent_undo   +virtualedit
+cmdline_compl     +keymap            +postscript        +visual
+cmdline_hist      +lambda            +printer           +visualextra
+cmdline_info      +langmap           +profile           +viminfo
+comments          +libcall           -python            +vreplace
+conceal           +linebreak         +python3           +wildignore
+cryptv            +lispindent        +quickfix          +wildmenu
+cscope            +listcmds          +reltime           +windows
+cursorbind        +localmap          +rightleft         +writebackup
+cursorshape       -lua               -ruby              -X11
+dialog_con        +menu              +scrollbind        -xfontset
+diff              +mksession         +signs             -xim
+digraphs          +modify_fname      +smartindent       -xpm
-dnd               +mouse             +startuptime       -xsmp
-ebcdic            -mouseshape        +statusline        -xterm_clipboard
+emacs_tags        +mouse_dec         -sun_workshop      -xterm_save
+eval              +mouse_gpm         +syntax
+ex_extra          -mouse_jsbterm     +tag_binary
+extra_search      +mouse_netterm     +tag_old_static
   system vimrc file: "$VIM/vimrc"
     user vimrc file: "$HOME/.vimrc"
 2nd user vimrc file: "~/.vim/vimrc"
      user exrc file: "$HOME/.exrc"
       defaults file: "$VIMRUNTIME/defaults.vim"
  fall-back for $VIM: "/usr/share/vim"
Compilation: gcc -c -I. -Iproto -DHAVE_CONFIG_H   -Wdate-time  -g -O2 -fdebug-prefix-map=/build/vim-dfCUyu/vim-8.0.1453=. -fstack-protector-strong -Wformat -Werro
r=format-security -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=1
Linking: gcc   -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-z,now -Wl,--as-needed -o vim        -lm -ltinfo -lnsl  -lselinux  -lacl -lattr -lgpm -ldl     -L/usr/lib
/python3.6/config-3.6m-x86_64-linux-gnu -lpython3.6m -lpthread -ldl -lutil -lm
```

```
river@ubuntu:~$ git --version
git version 2.17.1
```

```
river@ubuntu:~$ gcc --version
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 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.
```

```
river@ubuntu:~$ as --version
GNU assembler (GNU Binutils for Ubuntu) 2.30
Copyright (C) 2018 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'.
```

```
river@ubuntu:~$ objdump -V
GNU objdump (GNU Binutils for Ubuntu) 2.30
Copyright (C) 2018 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.
```

```
river@ubuntu:~$ gdb --version
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 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.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
```

2. Linux 编程实践

```
river@ubuntu:~$ cd Desktop
river@ubuntu:~/Desktop$ mkdir workspace
river@ubuntu:~/Desktop$ ls
workspace
river@ubuntu:~/Desktop$ cd workspace
river@ubuntu:~/Desktop/workspace$ mkdir lab01
river@ubuntu:~/Desktop/workspace$ cd lab01
river@ubuntu:~/Desktop/workspace/lab01$ mkdir 191220154
river@ubuntu:~/Desktop/workspace/lab01$ cd 191220154
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi heart.s
```

```
.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   *\n191220154\n"
    len = . - msg
```

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ as -o heart.o heart.s
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -no-pie -o heart heart.o
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./heart
 *   *
*******
 *****
   *
191220154
```

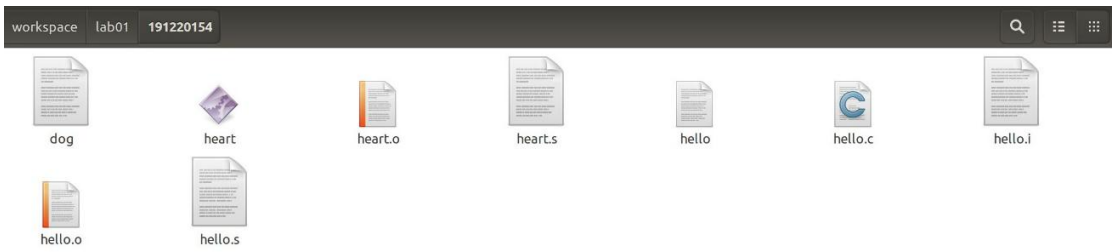
3. 熟悉工具

使用 objdump 反汇编

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ objdump -D heart.o>dog
river@ubuntu:~/Desktop/workspace/lab01/191220154$ cat dog

heart.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   ba 26 00 00 00          mov    $0x26,%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:

0000000000000000 <msg>:
   0:   20 2a                   and    %ch,(%rdx)
   2:   20 20                   and    %ah,(%rax)
   4:   20 2a                   and    %ch,(%rdx)
   6:   0a 2a                   or     (%rdx),%ch
   8:   2a 2a                   sub    (%rdx),%ch
   a:   2a 2a                   sub    (%rdx),%ch
   c:   2a 2a                   sub    (%rdx),%ch
   e:   0a 20                   or     (%rax),%ah
  10:   2a 2a                   sub    (%rdx),%ch
  12:   2a 2a                   sub    (%rdx),%ch
  14:   2a 0a                   sub    (%rdx),%cl
  16:   20 20                   and    %ah,(%rax)
  18:   20 2a                   and    %ch,(%rdx)
  1a:   0a 31                   or     (%rcx),%dh
  1c:   39 31                   cmp    %esi,(%rcx)
  1e:   32 32                   xor    (%rdx),%dh
  20:   30 31                   xor    %dh,(%rcx)
  22:   35                      .byte 0x35
  23:   34 0a                   xor    $0xa,%al
```

C 语言 hello world->可执行文件的过程

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi hello.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -E hello.c -o hello.i
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -S hello.i
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -c hello.s
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc hello.o -o hello
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ls
dog  heart  heart.o  heart.s  hello  hello.c  hello.i  hello.o  hello.s
```



```c
#include <stdio.h>

int main()
{
    printf("hello,world\n");
}
```

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./hello
hello,world
```

通过创建或修改~/.vimrc 文件，使你的 vim 支持显示行号

```
river@ubuntu:~$ vim ~/.vimrc
river@ubuntu:~$ vim test
```

```
set number
```

```c
 1 #include <stdio.h>
 2
 3 int main()
 4 {
 5     int i, j;
 6     i = 40000;
 7     j = i * i;
 8     printf("The 40000*40000 is %d\n", j);
 9     i = 50000;
10     j = i * i;
11     printf("The 50000*50000 is %d\n", j);
12     return 0;
13 }
```

4. 数据的表示

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi sqr.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -o sqr sqr.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./sqr
The 40000*40000 is 1600000000
The 50000*50000 is -1794967296
```

40000*40000=1600000000，50000*50000=2500000000

int 型数据的表示范围是-2147483638~2147483647，$(50000)_2$ = 1100 0011 0101 0000

$(2500000000)_2$ = 1001 0101 0000 0010 1111 1001 0000 0000

最高位为 1，转化为十进制数时按照负数来读取，即为-1794967296

```c
#include <stdio.h>

int main()
{
    int i, j;
    i = 46340;
    j = i * i;
    printf("The 46340*46340 is %d\n", j);
    i = 46341;
    j = i * i;
    printf("The 46341*46341 is %d\n", j);
    return 0;
}
```

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi newfile.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -o newfile newfile.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./newfile
The 46340*46340 is 2147395600
The 46341*46341 is -2147479015
```

在该程序中保证结果正确的最大整数值为 46340

```c
#include <stdio.h>

int main()
{
    long int i, j;
    i = 40000;
    j = i * i;
    printf("The 40000*40000 is %ld\n", j);
    i = 50000;
    j = i * i;
    printf("The 50000*50000 is %ld\n", j);
    return 0;
}
```

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi sqr.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -o sqr sqr.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./sqr
The 40000*40000 is 1600000000
The 50000*50000 is 2500000000
```

5. 矩阵运算执行时间比较

```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;
}
```

```
river@ubuntu:~/Desktop/workspace/lab01/191220154$ vi matrix.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ gcc -o matrix matrix.c
river@ubuntu:~/Desktop/workspace/lab01/191220154$ ./matrix
copyij 0.018730 s
copyji 0.144052 s
```

原因：二维数组的内存地址连续，前一行的尾与后一行的头相邻。访问数组元素时，每次读取一个区域。当数组元素较多时，CPU 缓存一次只读取不到一行的数据，因此按行访问可以每几个元素访问一次内存，而按列访问每一个元素都要访问内存，运行速度较慢。