Linux 和 C语言拾遗

王慧妍

why@nju.edu.cn

南京大学 软件学院 计算机软件研究所

讲前提醒

- 下周课前自动发放token到选课同学smail邮箱
- 未成功选课如需token请登记：
  https://table.nju.edu.cn/dtable/links/bbda8ba45ece40439310（密码：software）

Windows世界

- 任务驱动下的世界设计
  - 每个任务都可以找到一个复杂但容易上手的工具/软件
    - 写文档、刷网页、画图等
  - 点鼠标能做的事情很多
    - 易入门、易使用

- 但是，PA要求Linux环境配置
  - Why？

Linux世界

- Unix（Linux祖先）是通过命令行搞定一切
  - 能想到什么命令
    - `ping, wc, fdisk, grep, apt-get, wget, curl, tar, iconv, netstat…`
  - 点鼠标好像在windows也能做

  - Windows有时候有些简单的事情反而做不好
    - Exp1:比较两个文件是否完全相同
      - `vimdiff, diff, md5sum`
      - 鼠标20下 vs. 命令行3秒
    - Exp2: 批量重命名文件名称
    - Exp3: 列出项目中所有被包含的头文件

入门门槛高、使用自由度大

Unix哲学：

1. 每个小工具只做一件事情，但做到极致
2. 小工具统一文本输入输出，易于使用
3. 小工具直接通过管道进行组合，协作解决复杂任务

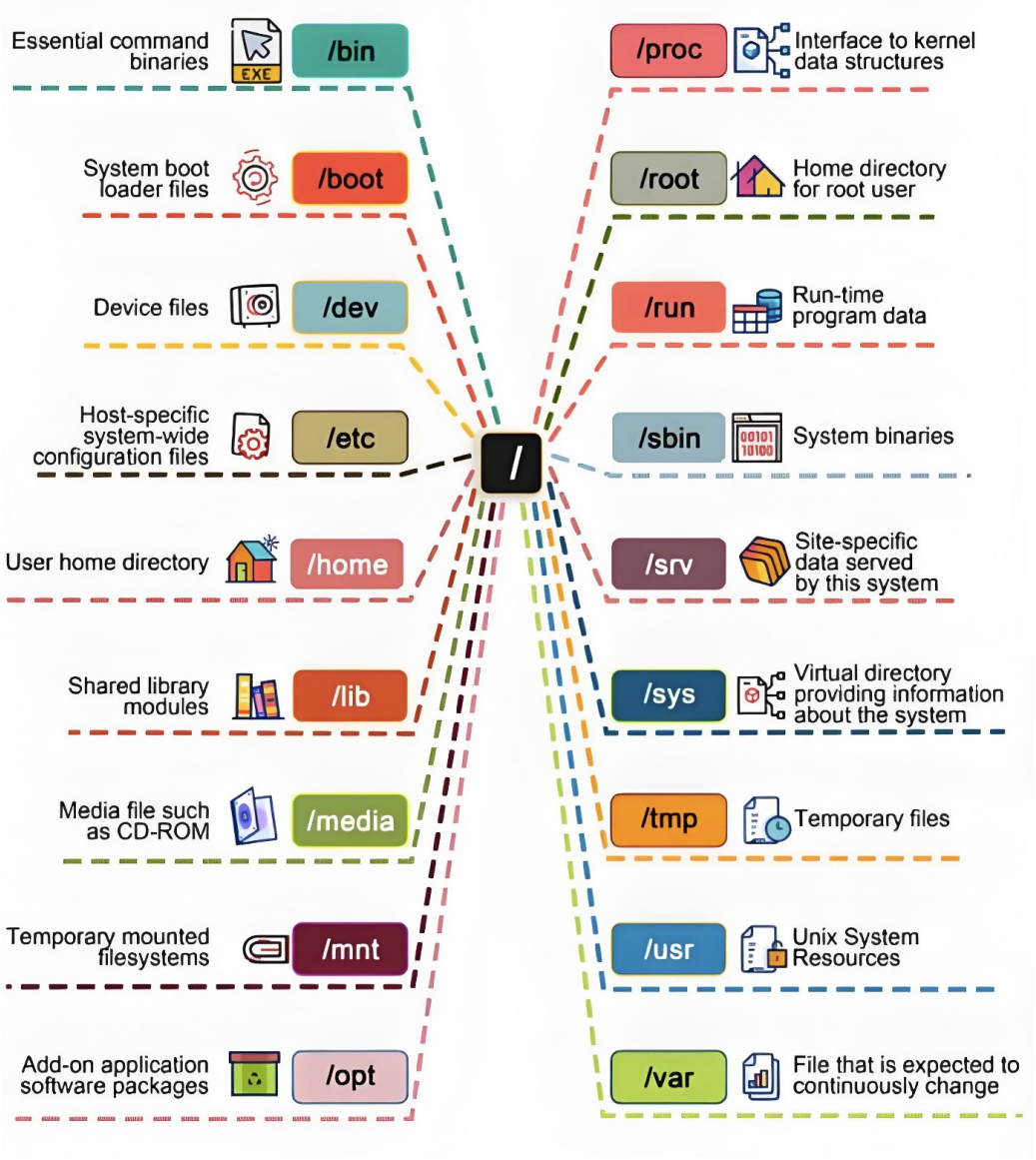
为什么Linux更适合程序员？

- 关键：使用Linux就是在编程
  - 命令行工具→shell编程
  - grep/awk/sed→正则表达式编程
  - 管道→模块编程

- Linux开源
  - 有机会知道计算机到底如何运行的？
  - Everything is a file in Linux.
    - 提供一种简单、统一和灵活的访问方式
    - /bin, /user, /etc, .bashrc, …

```
why@why-VirtualBox:~/Documents/ICS2024/ics2024$ ls -l
total 28
-rw-rw-r-- 1 why why 1148  9月   5 15:08 2405016.pdf
drwxrwxr-x 6 why why 4096  9月   5 14:44 abstract-machine
drwxrwxr-x 5 why why 4096  9月   5 14:44 fceux-am
-rw-rw-r-- 1 why why 1634  9月   5 14:44 init.sh
-rw-rw-r-- 1 why why  615  9月   5 14:44 Makefile
drwxrwxr-x 9 why why 4096  9月   5 14:47 nemu
-rw-rw-r-- 1 why why  727  9月   5 14:44 README.md
```

Filesystem Hierarchy Standard



命令行工具

- 命令格式

  命令　　参数1　　参数2　　...

```
LS(1)                                   User Commands                                   LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the cur
       nor --sort is specified.

       Mandatory arguments to long options are manc

       -a, --all
              do not ignore entries starting with

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each fi

       -b, --escape
              print C-style escapes for nongraphic

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with -lt: sort by, and show, ctime (time of last modification of file status information); with  -l:  show  ctime
              and sort by name; otherwise: sort by ctime, newest first

       -C     list entries by columns

       --color[=WHEN]
 Manual page ls(1) line 1 (press h for help or q to quit)
```

```
why@why-VirtualBox:~/Documents/ICS2024/ics2024/nemu$ ls -l -a
total 68
drwxrwxr-x 9 why why 4096  9月   5 14:47 .
drwxrwxr-x 6 why why 4096  9月   5 15:08 ..
drwxrwxr-x 4 why why 4096  9月   5 14:47 build
-rw-rw-r-- 1 why why 1452  9月   5 14:47 .config
drwxrwxr-x 2 why why 4096  9月   5 14:44 configs
-rw-rw-r-- 1 why why  106  9月   5 14:44 .gitignore
drwxrwxr-x 7 why why 4096  9月   5 14:47 include
-rw-rw-r-- 1 why why 4068  9月   5 14:44 Kconfig
-rw-rw-r-- 1 why why 9540  9月   5 14:44 LICENSE
-rw-rw-r-- 1 why why 2393  9月   5 14:44 Makefile
-rw-rw-r-- 1 why why 1138  9月   5 14:44 README.md
drwxrwxr-x 5 why why 4096  9月   5 14:44 resource
drwxrwxr-x 2 why why 4096  9月   5 14:44 scripts
drwxrwxr-x 9 why why 4096  9月   5 14:44 src
drwxrwxr-x 9 why why 4096  9月   5 14:44 tools
```

更多工具

- Busybox套件（包含常用Linux命令行工具）
  - Coreutils基本工具、编辑器、压缩归档、Linux系统编程等

```
/ # busybox
BusyBox v1.20.0 (2012-04-22 12:29:58 CEST) multi-call binary.
Copyright (C) 1998-2011 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

Currently defined functions:
        [, [[, acpid, add-shell, addgroup, adduser, adjtimex, arp, arping, ash,
        awk, base64, basename, beep, blkid, blockdev, bootchartd, brctl,
        bunzip2, bzcat, bzip2, cal, cat, catv, chat, chattr, chgrp, chmod,
        chown, chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm,
        conspy, cp, cpio, crond, crontab, cryptpw, cttyhack, cut, date, dc, dd,
        deallocvt, delgroup, deluser, depmod, devmem, df, dhcprelay, diff,
        dirname, dmesg, dnsd, dnsdomainname, dos2unix, du, dumpkmap,
        dumpleases, echo, ed, egrep, eject, env, envdir, envuidgid, ether-wake,
        expand, expr, fakeidentd, false, fbset, fbsplash, fdflush, fdformat,
        fdisk, fgconsole, fgrep, find, findfs, flock, fold, free, freeramdisk,
        fsck, fsck.minix, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty,
        grep, groups, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid,
        hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave,
        ifplugd, ifup, inetd, init, insmod, install, ionice, iostat, ip,
        ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iprule, iptunnel,
        kbd_mode, kill, killall, killall5, klogd, last, less, linux32, linux64,
        linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread,
        losetup, lpd, lpq, lpr, ls, lsattr, lsmod, lspci, lsusb, lzcat, lzma,
        lzop, lzopcat, makedevs, makemime, man, md5sum, mdev, mesg, microcom,
        mkdir, mkdosfs, mke2fs, mkfifo, mkfs.ext2, mkfs.minix, mkfs.vfat,
        mknod, mkpasswd, mkswap, mktemp, modinfo, modprobe, more, mount,
        mountpoint, mpstat, mt, mv, nameif, nanddump, nandwrite, nbd-client,
        nc, netstat, nice, nmeter, nohup, nslookup, ntpd, od, openvt, passwd,
        patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill,
        pmap, popmaildir, poweroff, powertop, printenv, printf, ps, pscan,
        pstree, pwd, pwdx, raidautorun, rdate, rdev, readahead, readlink,
        readprofile, realpath, reboot, reformime, remove-shell, renice, reset,
        resize, rev, rm, rmdir, rmmod, route, rpm, rpm2cpio, rtcwake,
        run-parts, runlevel, runsv, runsvdir, rx, script, scriptreplay, sed,
        sendmail, seq, setarch, setconsole, setfont, setkeycodes, setlogcons,
        setserial, setsid, setuidgid, sh, sha1sum, sha256sum, sha512sum,
        showkey, slattach, sleep, smemcap, softlimit, sort, split,
        start-stop-daemon, stat, strings, stty, su, sulogin, sum, sv, svlogd,
        swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar,
        tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top,
        touch, tr, traceroute, traceroute6, true, tty, ttysize, tunctl,
        ubiattach, ubidetach, ubimkvol, ubirmvol, ubirsvol, ubiupdatevol,
        udhcpc, udhcpd, udpsvd, umount, uname, unexpand, uniq, unix2dos,
        unlzma, unlzop, unxz, unzip, uptime, users, usleep, uudecode, uuencode,
        vconfig, vi, vlock, volname, wall, watch, watchdog, wc, wget, which,
        who, whoami, whois, xargs, xz, xzcat, yes, zcat, zcip
```

更多工具

- Busybox套件（包含常用Linux命令行工具）
  - Coreutils基本工具、编辑器、压缩归档、Linux系统编程等
- 标准Linux工具：默认/bin目录下

```
why@why-VirtualBox:/bin$ ll | wc
   1935   18588  138715
```

- 查找更多工具
  - Modern unix

```
LS(1)                                   User Commands                                   LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).  Sort entries alphabetically if none of -cftuvSUX
       nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with -lt: sort by, and show, ctime (time of last modification of file status information); with  -l:  show  ctime
              and sort by name; otherwise: sort by ctime, newest first

       -C     list entries by columns

       --color[=WHEN]
 Manual page ls(1) line 1 (press h for help or q to quit)
```

强大的shell界面

- 作为和操作系统交互的界面，shell功能丰富
  - 通过Tab自动补全
  - 通过上下检索命令
  - 通过cd – 返回上一个工作目录
  - 通过history查看历史命令
  - 特殊符号
    - 通配符*(任意长度任意字符串)
    - ？(任意一个字符)
    - […]集合中任何一个字符
    - …
  - 括号扩展{…}
  - 更多可man readline

```
2034  make submit
2035  ls
2036  git add 2405016.pdf
2037  vim .gitignore
2038  git add -f 2405016.pdf
2039  git commit -m "add pdf"
2040  make submit
```

```
find . -name "*.[ch]"
```

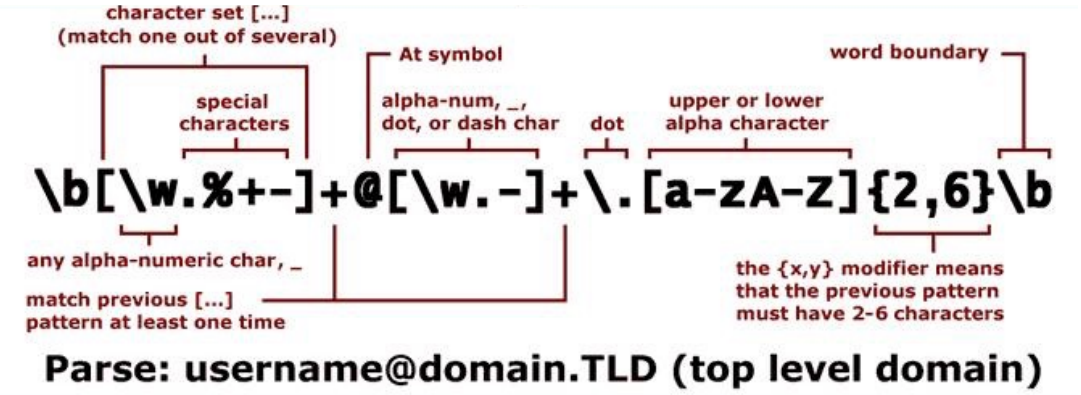
```
why@why-VirtualBox:~$ echo ICS-{pa1,pa2}-{id1,id2}
ICS-pa1-id1 ICS-pa1-id2 ICS-pa2-id1 ICS-pa2-id2
```

正则表达式

- 一般配合grep/awk/sed/vim，PA1紧密关联

```
30 static struct rule {
31   const char *regex;
32   int token_type;
33 } rules[] = {
34
35   /* TODO: Add more rules.
36    * Pay attention to the precedence leve
37    */
38
39   {" +", TK_NOTYPE},    // spaces
40   {"\\+", '+'},         // plus
41   {"==", TK_EQ},        // equal
42 };
```

- 一种强大的字符匹配的语言
  - E.g., [^abc], \d



匹配email

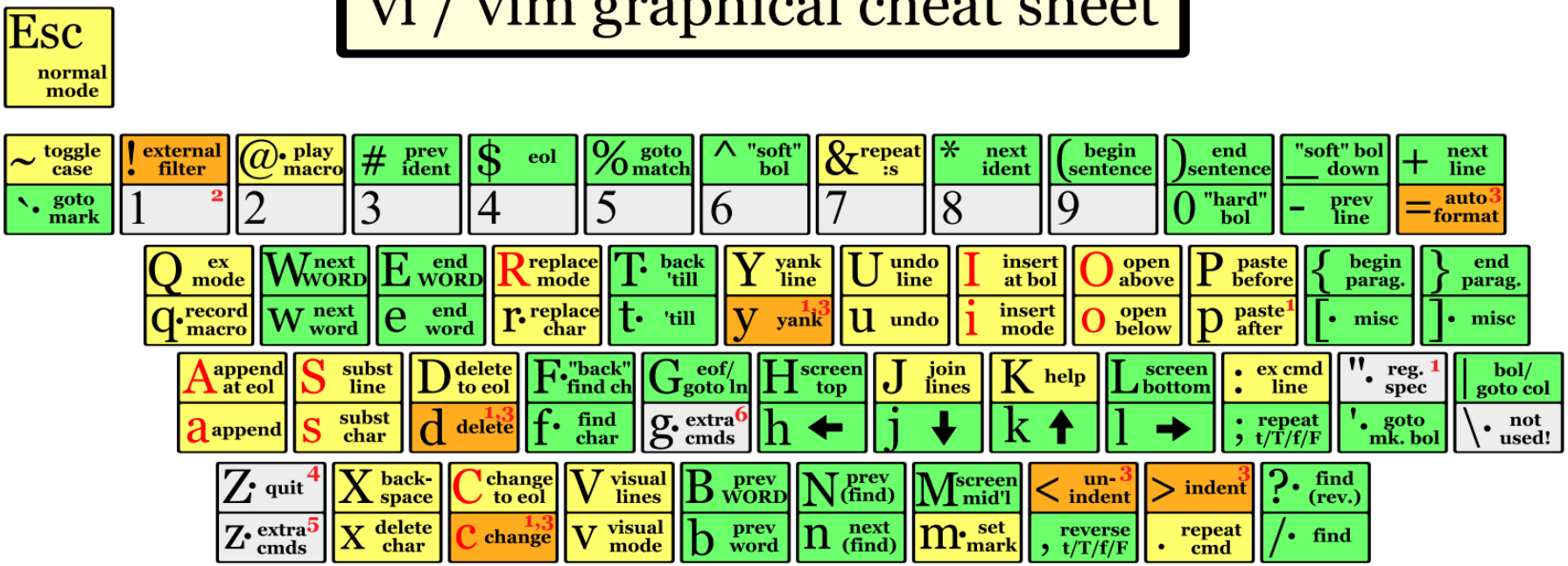
Vim：这都搞不定还引发什么编辑器圣战

- Marks (文件内标记)
  - ma, 'a, mA, 'A, ...
- Tags (在位置之间跳转)
  - :jumps, C-], C-i, C-o, :tjump, ...
- Tabs/Windows (管理多文件)
  - :tabnew, gt, gT, ...
- Folding (浏览大代码)
  - zc, zo, zR, ...

- 更多的功能/插件：(RTFM, STFW)
  - vimtutor

version 1.1
April 1st, 06

vi / vim graphical cheat sheet



motion — moves the cursor, or defines the range for an operator

command — direct action command, if red, it enters insert mode

operator — requires a motion afterwards, operates between cursor & destination

extra — special functions, requires extra input

q• — commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: `quux((foo, bar, baz);`
WORDs: `quux(foo, bar, baz);`

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important comands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

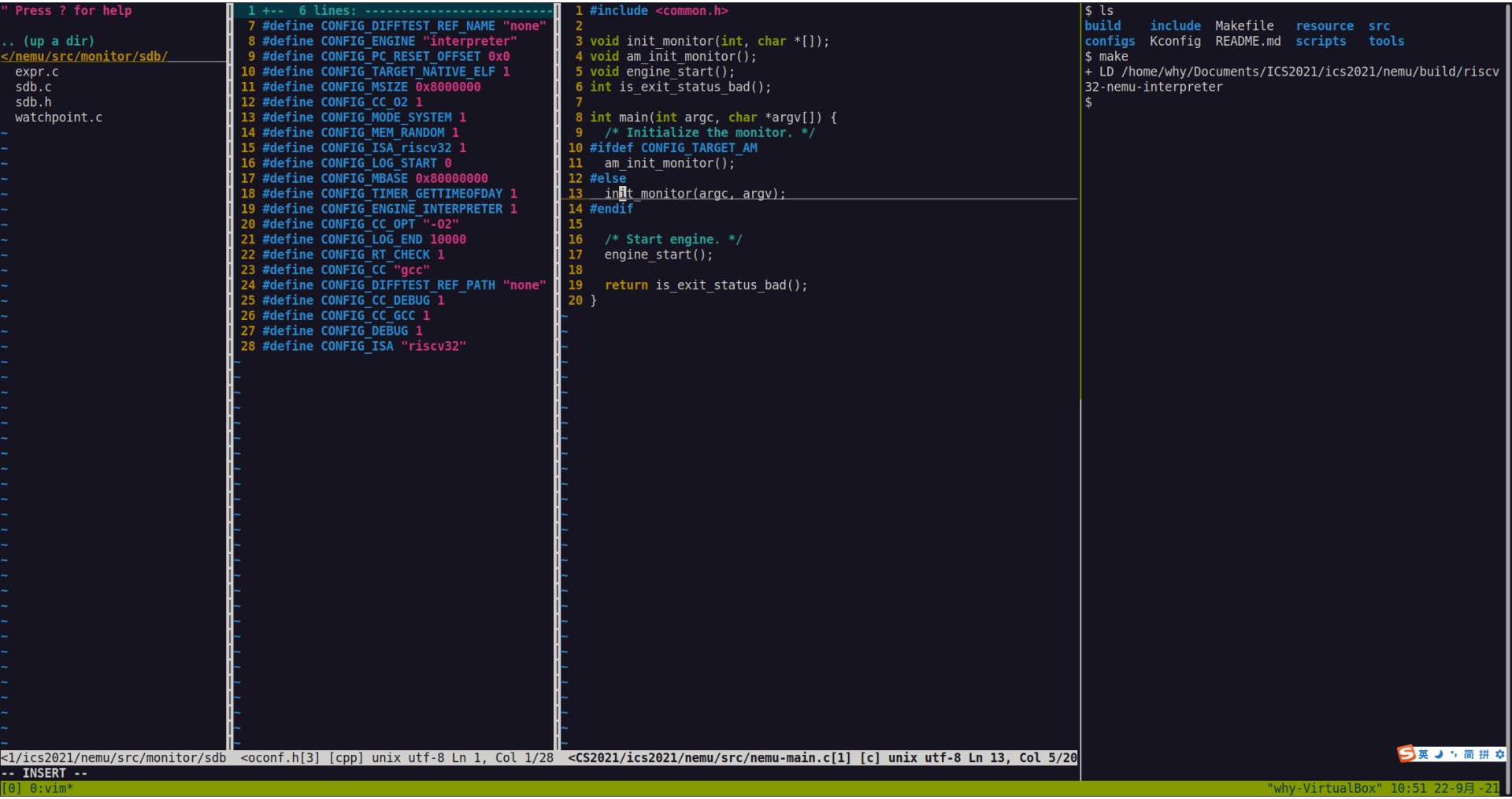
(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top, zb: bottom, zz: center

(6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

如果你有块更大的屏幕


" Press ? for help
.. (up a dir)
</nemu/src/monitor/sdb/
expr.c
sdb.c
sdb.h
watchpoint.c
1 +-- 6 lines: ----------
7 #define CONFIG_DIFFTEST_REF_NAME "none"
8 #define CONFIG_ENGINE "interpreter"
9 #define CONFIG_PC_RESET_OFFSET 0x0
10 #define CONFIG_TARGET_NATIVE_ELF 1
11 #define CONFIG_MSIZE 0x8000000
12 #define CONFIG_CC_O2 1
13 #define CONFIG_MODE_SYSTEM 1
14 #define CONFIG_MEM_RANDOM 1
15 #define CONFIG_ISA_riscv32 1
16 #define CONFIG_LOG_START 0
17 #define CONFIG_MBASE 0x80000000
18 #define CONFIG_TIMER_GETTIMEOFDAY 1
19 #define CONFIG_ENGINE_INTERPRETER 1
20 #define CONFIG_CC_OPT "-O2"
21 #define CONFIG_LOG_END 10000
22 #define CONFIG_RT_CHECK 1
23 #define CONFIG_CC "gcc"
24 #define CONFIG_DIFFTEST_REF_PATH "none"
25 #define CONFIG_CC_DEBUG 1
26 #define CONFIG_CC_GCC 1
27 #define CONFIG_DEBUG 1
28 #define CONFIG_ISA "riscv32"
1 #include <common.h>
2
3 void init_monitor(int, char *[]);
4 void am_init_monitor();
5 void engine_start();
6 int is_exit_status_bad();
7
8 int main(int argc, char *argv[]) {
9 /* Initialize the monitor. */
10 #ifdef CONFIG_TARGET_AM
11 am_init_monitor();
12 #else
13 init_monitor(argc, argv);
14 #endif
15
16 /* Start engine. */
17 engine_start();
18
19 return is_exit_status_bad();
20 }
$ ls
build include Makefile resource src
configs Kconfig README.md scripts tools
$ make
+ LD /home/why/Documents/ICS2021/ics2021/nemu/build/riscv32-nemu-interpreter
$
<1/ics2021/nemu/src/monitor/sdb <oconf.h[3] [cpp] unix utf-8 Ln 1, Col 1/28 <CS2021/ics2021/nemu/src/nemu-main.c[1] [c] unix utf-8 Ln 13, Col 5/20
-- INSERT --
[0] 0:vim*
"why-VirtualBox" 10:51 22-9月 -21

RTFM, STFW

- 最重要的Linux命令：man（ sometimes tldr ）
  - 查阅命令/库函数/系统文件等内容的手册
  - man ls – 查看如何使用ls命令
  - man 3 printf – 查看如何使用printf

```
PRINTF(3)                  Linux Programmer's Manual                 PRINTF(3)

NAME
       printf,  fprintf,  dprintf, sprintf, snprintf, vprintf, vfprintf, vdprintf,
       vsprintf, vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int dprintf(int fd, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
       int snprintf(char *str, size_t size, const char *format, ...);

       #include <stdarg.h>

       int vprintf(const char *format, va_list ap);
       int vfprintf(FILE *stream, const char *format, va_list ap);
       int vdprintf(int fd, const char *format, va_list ap);
       int vsprintf(char *str, const char *format, va_list ap);
       int vsnprintf(char *str, size_t size, const char *format, va_list ap);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       snprintf(), vsnprintf():
           _XOPEN_SOURCE >= 500 || _ISOC99_SOURCE ||
               || /* Glibc versions <= 2.19: */ _BSD_SOURCE

       dprintf(), vdprintf():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
               _GNU_SOURCE

DESCRIPTION
       The functions in the printf() family produce output according to  a  format
       as  described  below.  The functions printf() and vprintf() write output to
 Manual page printf(3) line 1 (press h for help or q to quit)
```

RTFM, STFW

- 最重要的Linux命令：man（sometimes tldr）
  - 查
  - m
  - m
  - m

```
man(1)      General Commands Manual     man(1)

NAME
       man - format and display the on-line
       manual pages

SYNOPSIS
       man [-acdfFhkKtwW] [--path] [-m system]
       [-p string] [-C config_file] [-M
       pathlist] [-P pager] [-B browser] [-H
       htmlpager] [-S section_list] [section]
       name ...

DESCRIPTION
       man formats and displays the on-line
       manual pages.  If you specify section,
       man only looks in that section of the
       manual.  name is normally the name of
```

```
strace-log-merge (1) - merge strace -ff -tt output
tracepath (8)        - traces path to a network host discovering MTU along this path
xtables-monitor (8)  - show changes to rule set and trace-events
```

```
$ man -k trace
```

好消息

- 助教正在准备指南（期待中）：
  - 如何在Win VSCode+WSL上完成PA？
- Linux环境各种命令依旧有用且非常关键
- VSCode+快捷切换/跳转+Vim编码速度up！

C语言拾遗

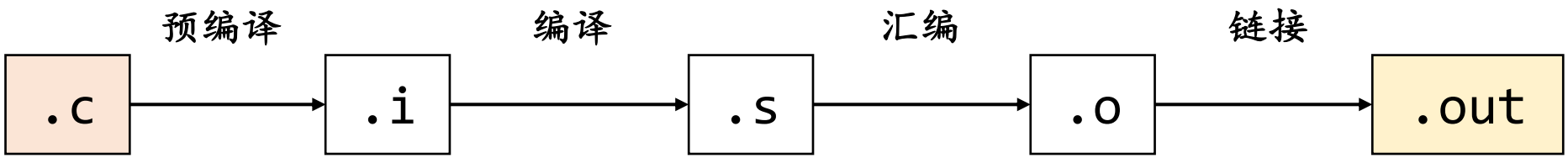
- 在IDE里，为什么按一个键，就能够编译运行？
  - 编译、链接
    - `.c` → 预编译 → `.i` → 编译 → `.s` → 汇编 → `.o` → 链接 → `a.out`
  - 加载执行
    - `./a.out`

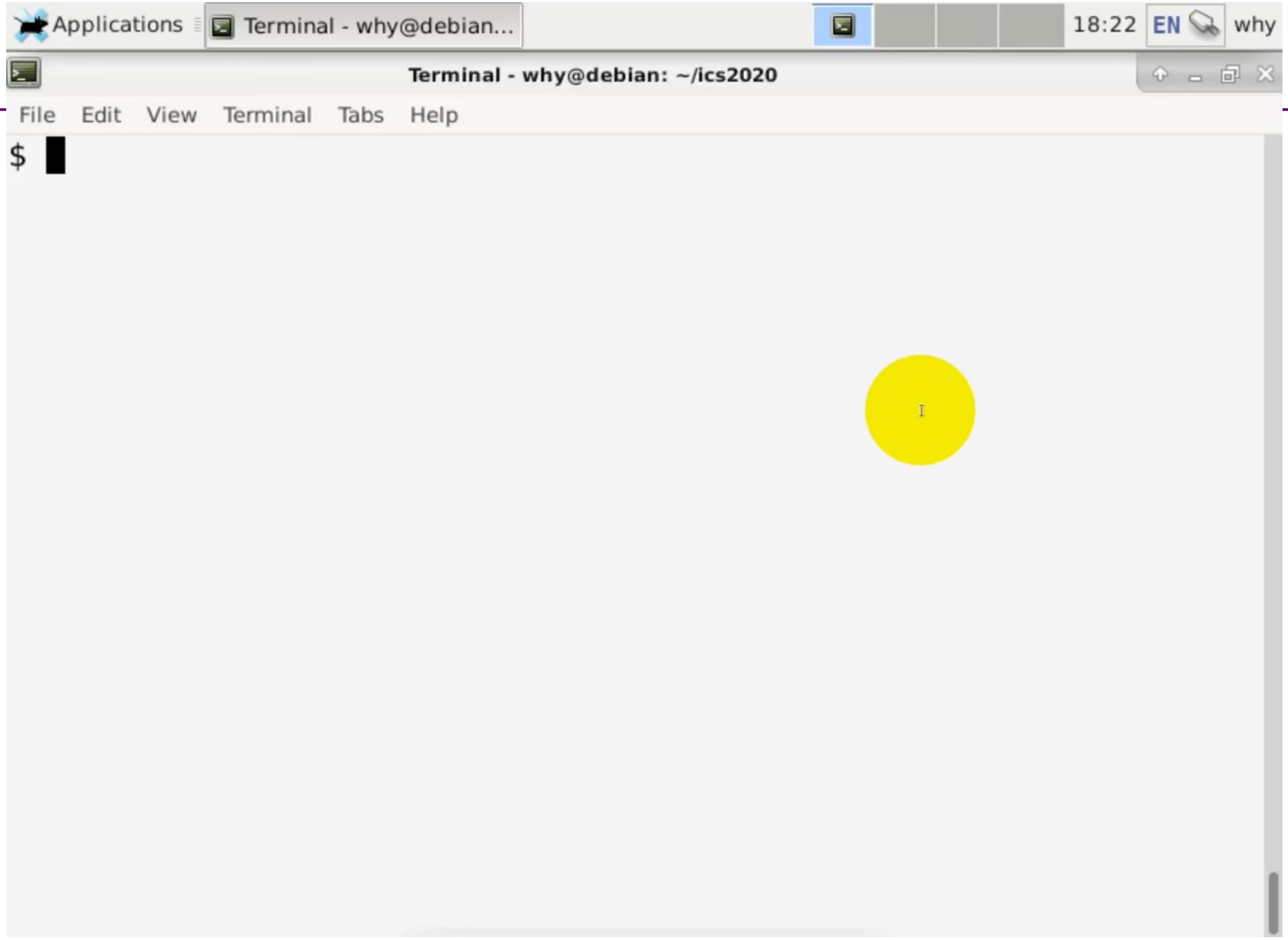
- 背后是通过调用命令行工具完成的
  - RTFM： `man gcc; gcc –help; tldr gcc`
    - 控制行为的三个选项：`-E`，`-S`，`-c`

- 过去忽略但是至关重要的知识

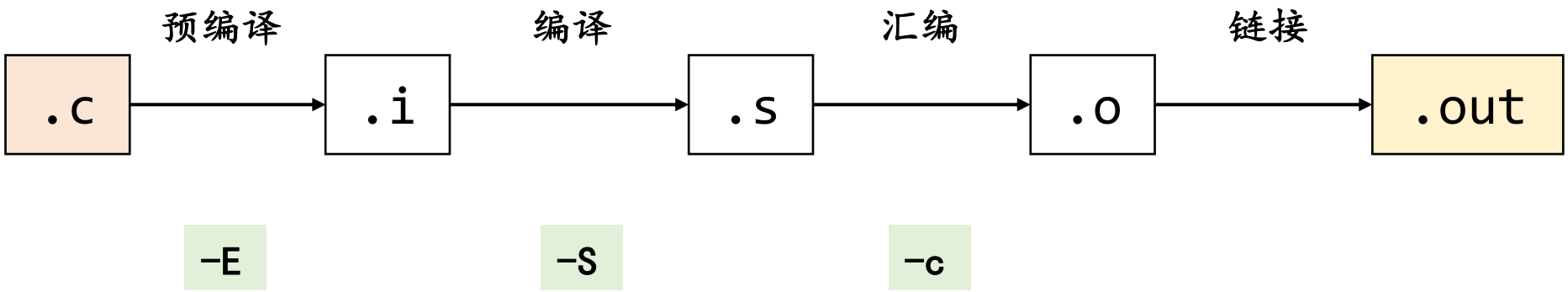
IDE的一个键到底发生了什么？

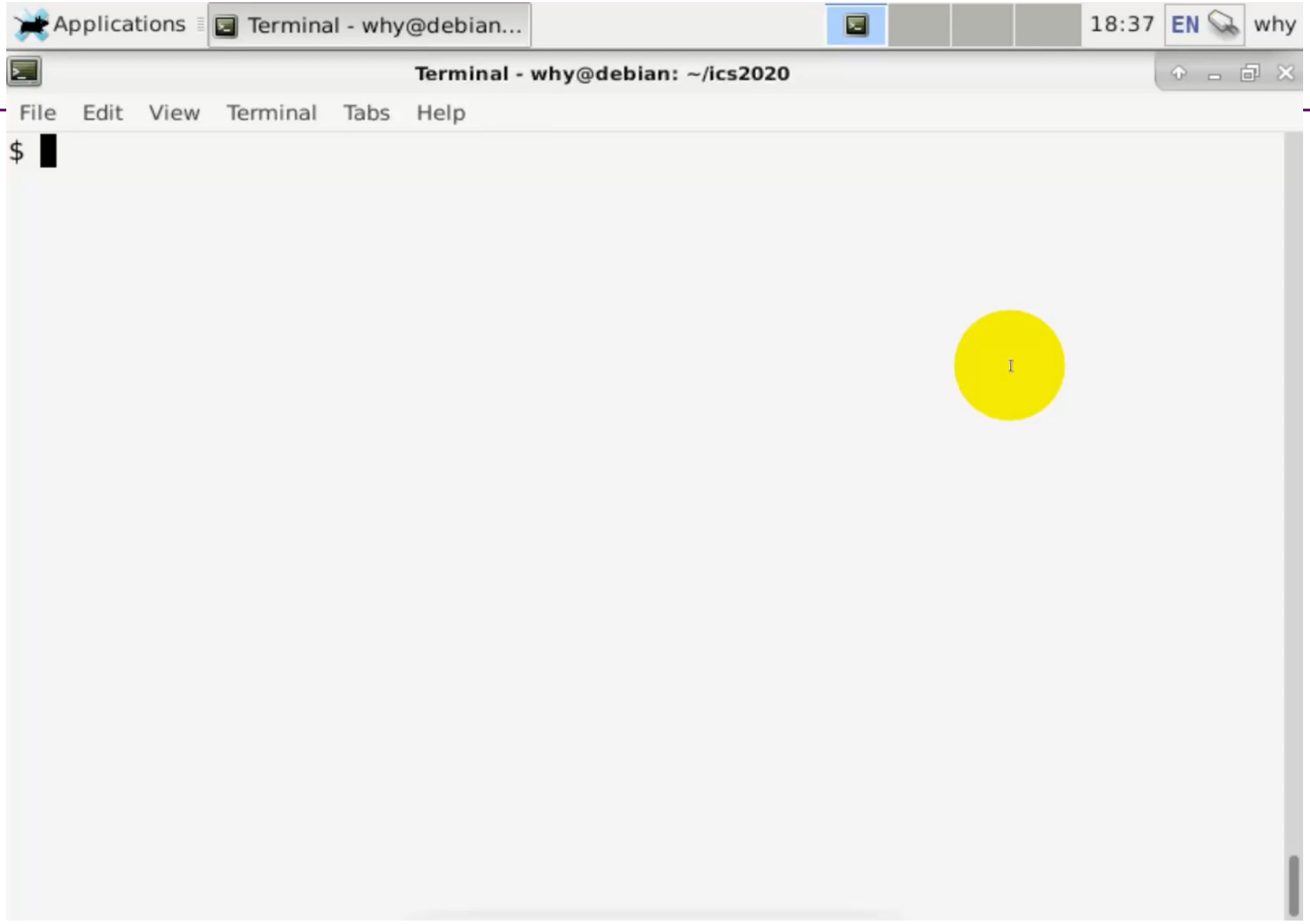
IDE的一个键到底发生了什么？




Applications
Terminal - why@debian...
18:22
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$

从源代码到可执行文件




Applications
Terminal - why@debian...
18:37
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$

进入C语言之前：预编译

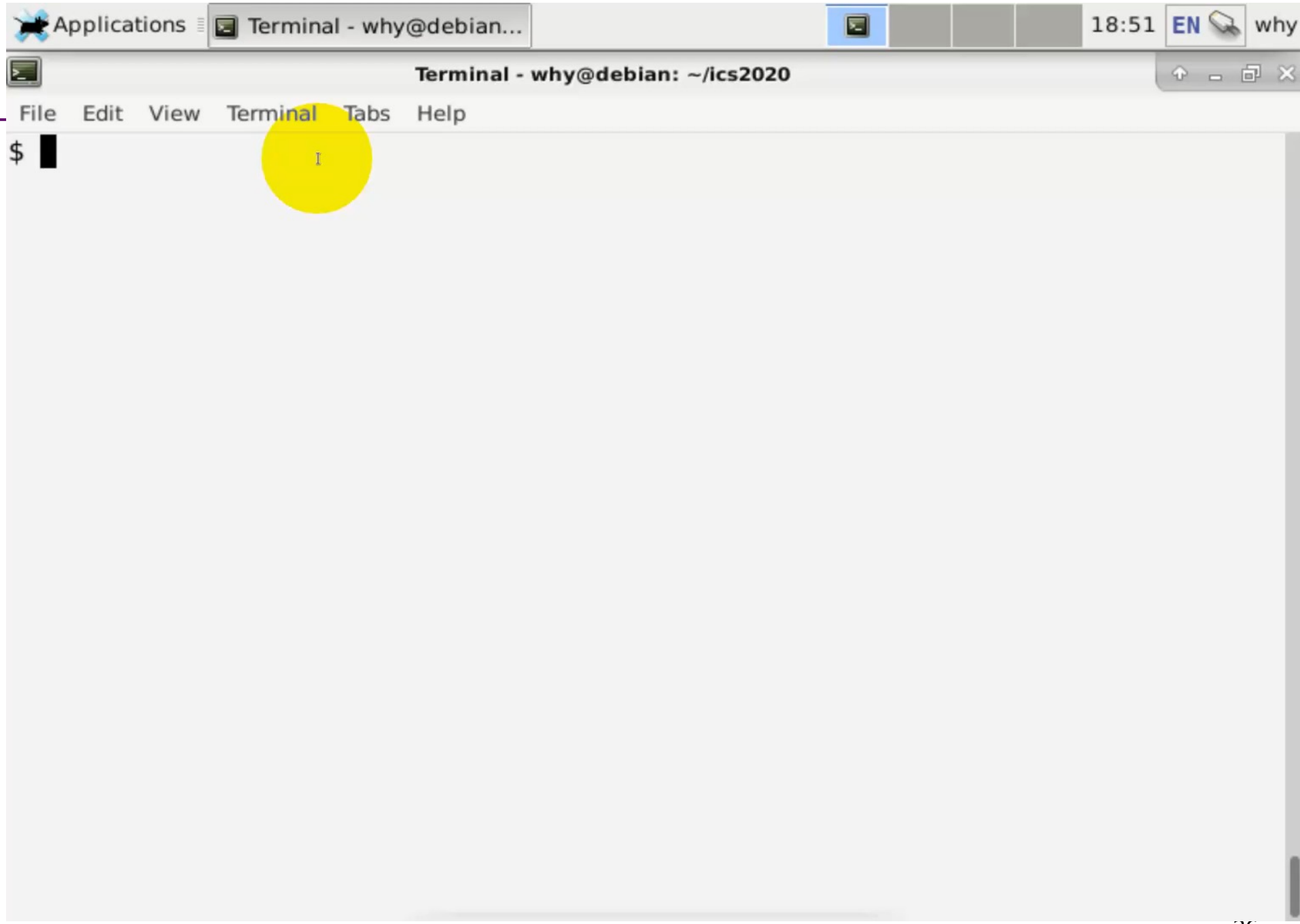
```
#include
#define
#
##
#ifdef
```

#include指令

- 什么是#include？

- 怎么理解？

```
1  //#include <stdio.h>
2  int printf(const char * __restrict, ...);
3  int main() {
4      printf("Hello, World!\n");
5      return 0;
6  }
```


Applications
Terminal - why@debian...
18:51
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$

#include<>指令

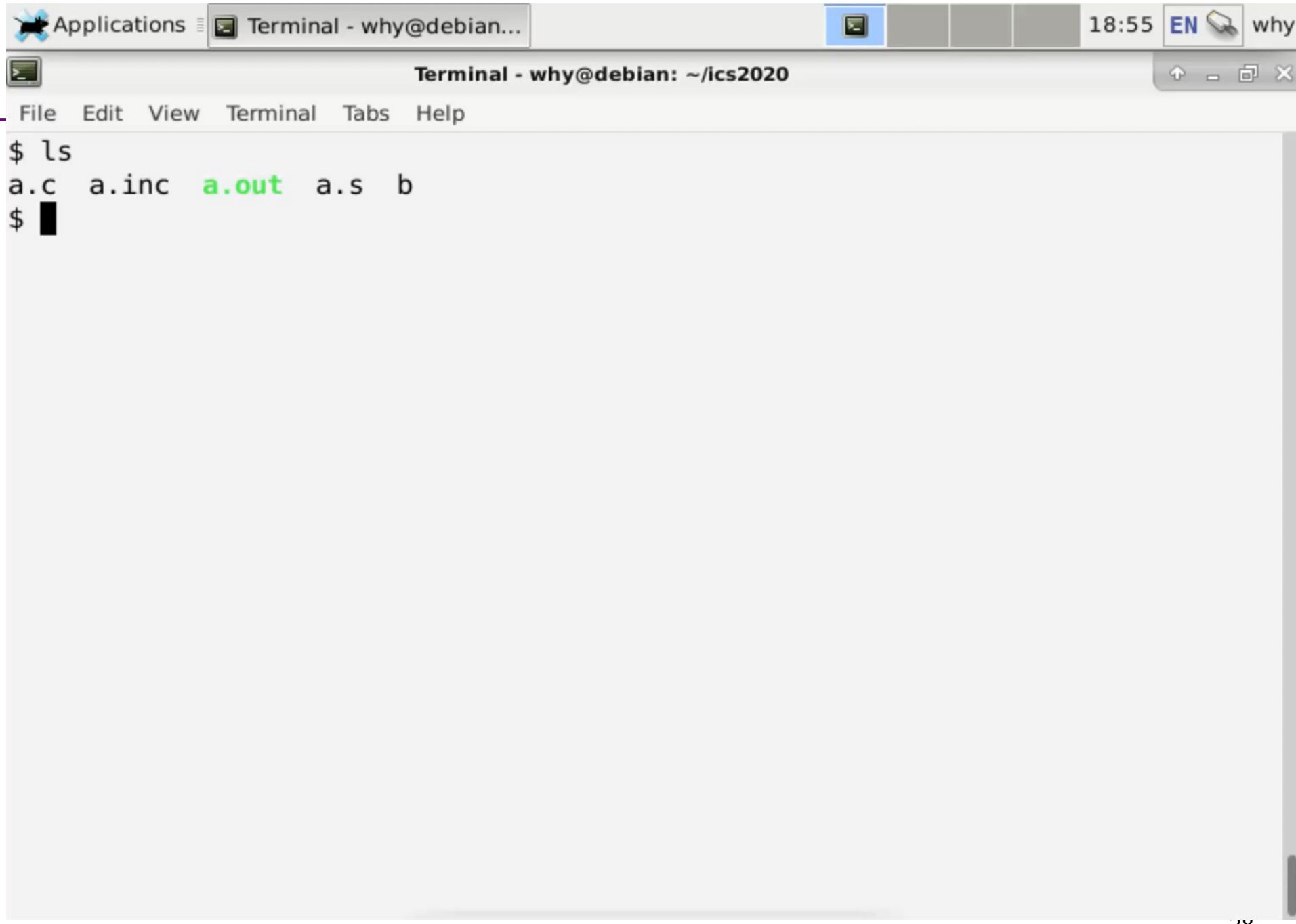
- 以下代码有什么区别？

```
#include <stdio.h>
#include “stdio.h”
```

- 为什么在没有安装库时会发生错误？

```
#include <SDL/SDL2.h>
```

- 你可能在书/阅读材料上了解过一些相关的知识
  - 但更好的办法是阅读命令的日志
  - `gcc --verbose a.c`


Applications
Terminal - why@debian...
18:55
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$ ls
a.c a.inc a.out a.s b
$

#include<>指令

```
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/x86_64-linux-gnu/8/include
 /usr/local/include
 /usr/lib/gcc/x86_64-linux-gnu/8/include-fixed
 /usr/include/x86_64-linux-gnu
 /usr/include
```

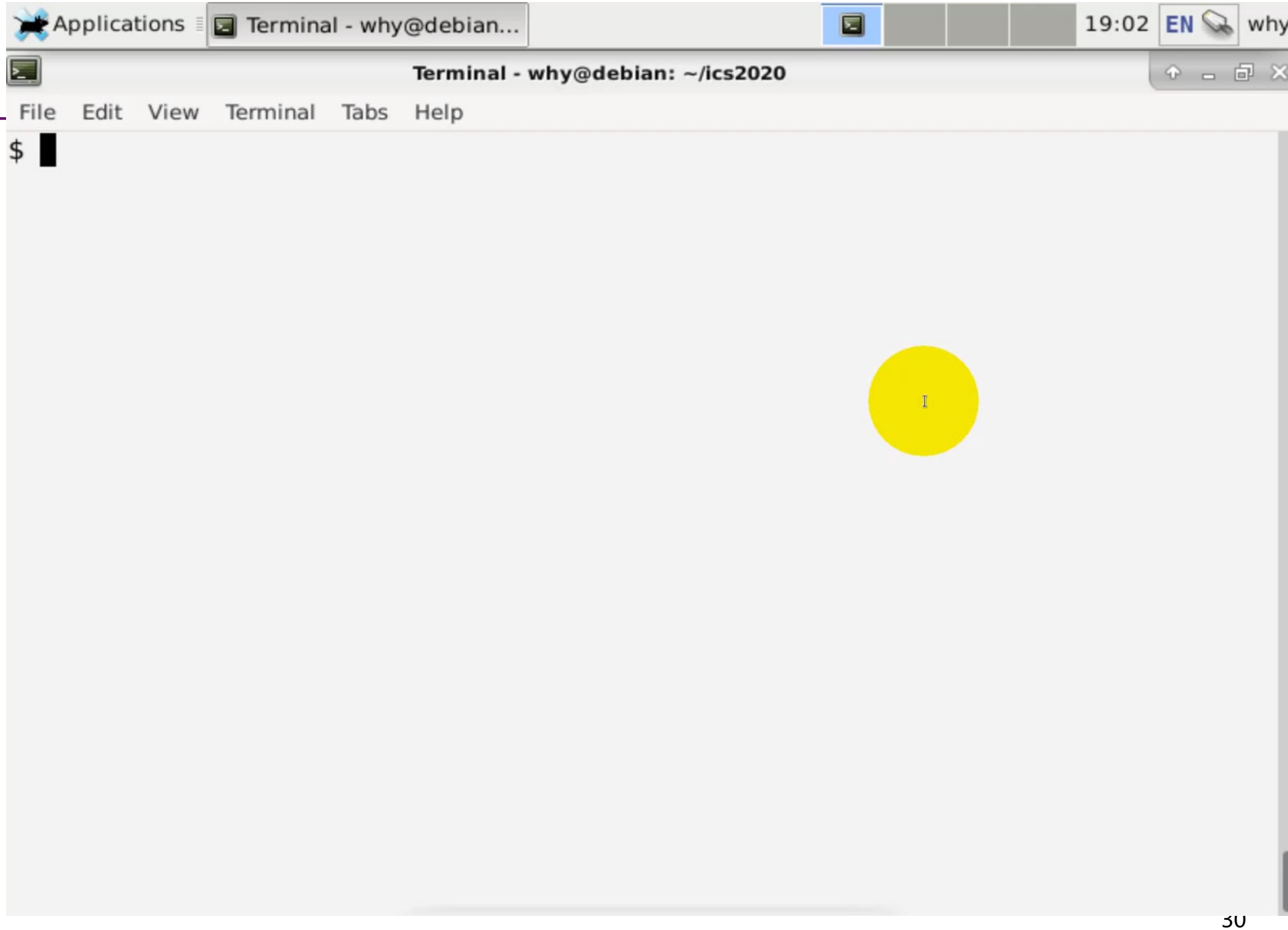
- 以下代码有什么区别？

```
#include <stdio.h>
#include “stdio.h”
```

- 为什么在没有安装库时会发生错误？

```
#include <SDL/SDL2.h>
```

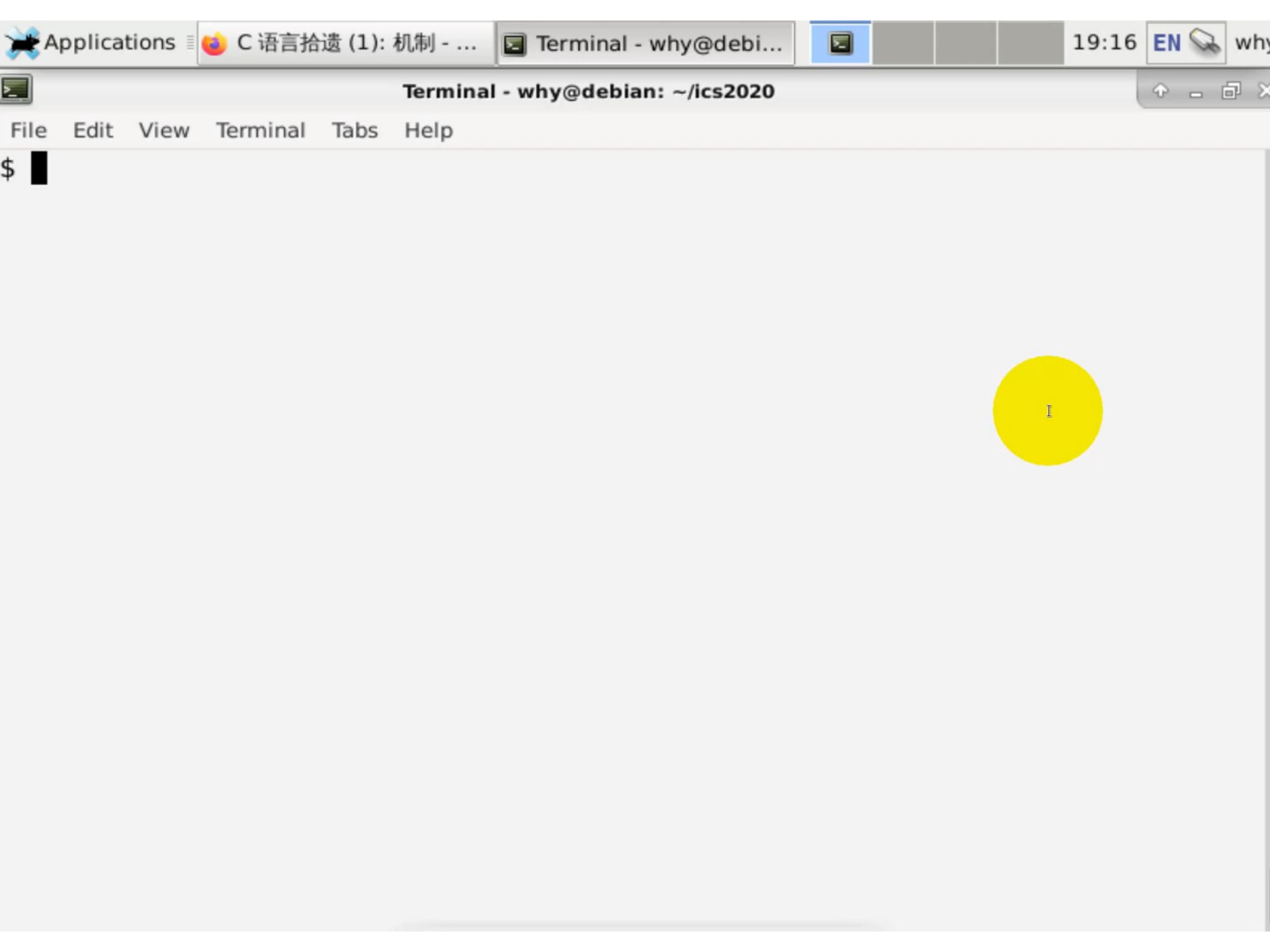
- 你可能在书/阅读材料上了解过一些相关的知识
  - 但更好的办法是阅读命令的日志
  - `gcc --verbose a.c`


Applications
Terminal - why@debian...
19:02
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$

有趣的预编译

- 以下代码会输出什么？
  - 为什么？

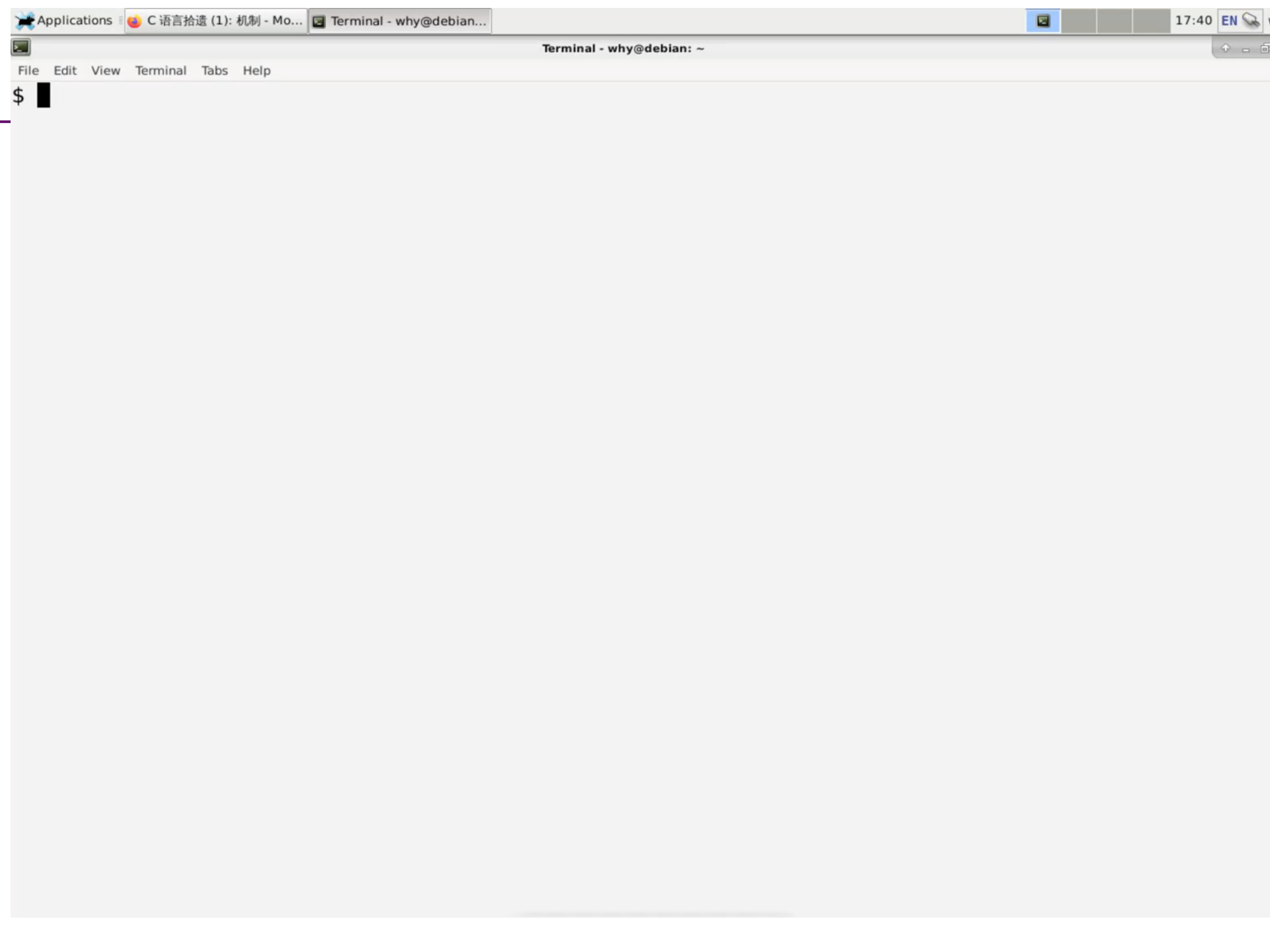
```c
#include <stdio.h>

int main() {
#if aa == bb
  printf("Yes\n");
#else
  printf("No\n");
#endif
}
```


Applications
C 语言拾遗 (1): 机制 - ...
Terminal - why@debi...
19:16
EN
why
Terminal - why@debian: ~/ics2020
File Edit View Terminal Tabs Help
$

宏定义与展开

- 宏展开：通过复制/粘贴改变代码的形态
  - #include →粘贴文件
  - aa, bb →粘贴符号
- 知乎问题：如何搞垮一个OJ?

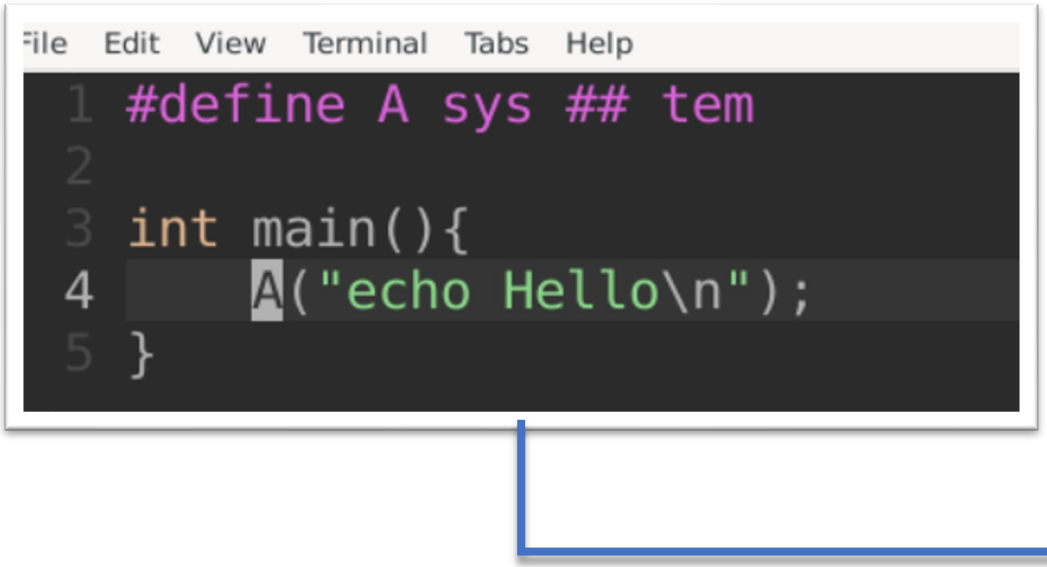
```
#define A "aaaaaaaaaa"
#define TEN(A) A A A A A A A A A A
#define B TEN(A)
#define C TEN(B)
#define D TEN(C)
#define E TEN(D)
#define F TEN(E)
#define G TEN(F)
int main() { puts(G);}
```


Applications
C 语言拾遗 (1): 机制 - Mo...
Terminal - why@debian...
17:40
EN
Terminal - why@debian: ~
File Edit View Terminal Tabs Help
$

宏定义与展开

- 如何躲过Online Judge的关键字过滤？

```
#define SYSTEM sys ## tem
```



宏定义与展开

- 如何毁掉一个身边的同学？

```
#define true (__LINE__ % 2 != 0)
```

File Edit View Terminal Tabs Help

```
1  #define true (__LINE__ % 2 != 0)
2
3  #include <cstdio>
4
5  int main(){
6      if (true) printf("yes %d\n",__LINE__);
7      if (true) printf("yes %d\n",__LINE__);
8      if (true) printf("yes %d\n",__LINE__);
9      if (true) printf("yes %d\n",__LINE__);
10     if (true) printf("yes %d\n",__LINE__);
11     if (true) printf("yes %d\n",__LINE__);
12     if (true) printf("yes %d\n",__LINE__);
13     if (true) printf("yes %d\n",__LINE__);
14     if (true) printf("yes %d\n",__LINE__);
15     if (true) printf("yes %d\n",__LINE__);
16     if (true) printf("yes %d\n",__LINE__);
17     if (true) printf("yes %d\n",__LINE__);
18     if (true) printf("yes %d\n",__LINE__);
19     if (true) printf("yes %d\n",__LINE__);
20     if (true) printf("yes %d\n",__LINE__);
21 }
```

```
$ g++ a.c
$ ./a.out
yes 7
yes 9
yes 11
yes 13
yes 15
yes 17
yes 19
```

宏定义与展开

```
#define s ((((((((((((((( 0
#define _ * 2)
#define X * 2 + 1)
static unsigned short stopwatch[] = {
  s _ _ _ _ _ X X X X X _ _ _ X X _ ,
  s _ _ _ X X X X X X X X X _ X X X ,
  s _ _ X X X _ _ _ _ _ X X X _ X X ,
  s _ X X _ _ _ _ _ _ _ _ _ X X _ _ ,
  s X X _ _ _ _ _ _ _ _ _ _ X X _ ,
  s X X _ X X X X X _ _ _ _ _ X X _ ,
  s X X _ _ _ _ _ X _ _ _ _ _ X X _ ,
  s X X _ _ _ _ _ X _ _ _ _ _ X X _ ,
  s _ X X _ _ _ _ X _ _ _ _ X X _ _ ,
  s _ _ X X X _ _ _ _ _ X X X _ _ _ ,
  s _ _ _ X X X X X X X X X _ _ _ _ ,
  s _ _ _ _ _ X X X X X _ _ _ _ _ _ , };
```

宏定义与展开

```
#define s ((((((((((((((( 0
#define _ * 2)
#define X * 2 + 1)
static unsigned short stopwatch[] = {
  s _ _ _ _ _ X X X X X _ _ _ X X _ ,
  s _ _ _ X X X X X X X X X _ X X X ,
  s _ _ X X X _ _ _ _ _ X X X _ X X ,
  s _ X X _ _ _ _ _ _ _ _ _ X X _ _ ,
  s X X _ _ _ _ _ _ _ _ _ _ X X _ ,
  s X X _ X X X X X _ _ _ _ _ X X _ ,
  s X X _ _ _ _ _ X _ _ _ _ _ X X _ ,
  s X X _ _ _ _ _ X _ _ _ _ _ X X _ ,
  s _ X X _ _ _ _ X _ _ _ _ X X _ _ ,
  s _ _ X X X _ _ _ _ _ X X X _ _ _ ,
  s _ _ _ X X X X X X X X X _ _ _ _ ,
  s _ _ _ _ _ X X X X X _ _ _ _ _ _ , };
```

gcc -E a.c

```
static unsigned short stopwatch[] = {
  ((((((((((((((( 0 * 2) * 2) * 2) * 2) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2) * 2
) * 2) * 2 + 1) * 2 + 1) * 2) ,
  ((((((((((((((( 0 * 2) * 2) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1)
* 2 + 1) * 2 + 1) * 2) * 2 + 1) * 2 + 1) * 2 + 1) ,
  ((((((((((((((( 0 * 2) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2 + 1) * 2 + 1
) * 2 + 1) * 2) * 2 + 1) * 2 + 1) ,
  ((((((((((((((( 0 * 2) * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2 + 1) *
 2 + 1) * 2) * 2) ,
  ((((((((((((((( 0 * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2) * 2 +
 1) * 2 + 1) * 2) ,
  ((((((((((((((( 0 * 2 + 1) * 2 + 1) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2) * 2)
* 2) * 2) * 2) * 2 + 1) * 2 + 1) * 2) ,
  ((((((((((((((( 0 * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2 + 1) * 2) * 2) * 2) * 2) * 2) *
 2 + 1) * 2 + 1) * 2) ,
  ((((((((((((((( 0 * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2 + 1) * 2) * 2) * 2) * 2) * 2) *
 2 + 1) * 2 + 1) * 2) ,
  ((((((((((((((( 0 * 2) * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2 + 1) * 2) * 2) * 2) * 2) * 2 +
1) * 2 + 1) * 2) * 2) ,
  ((((((((((((((( 0 * 2) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2) * 2) * 2) * 2) * 2) * 2 + 1) * 2 + 1
) * 2 + 1) * 2) * 2) * 2) ,
  ((((((((((((((( 0 * 2) * 2) * 2) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1) * 2 + 1)
```

```
$ ./a.out
1990
8183
14395
24588
49158
57094
49414
49414
24844
14392
8176
1984
```

X-Macros

- 宏展开：通过复制/粘贴改变代码的形态
  - 反复粘贴，直到没有宏可以展开为止

- 例子：X-macro

```
#define NAMES(X) \
   X(Tom) X(Jerry) X(Tyke) X(Spike)

int main() {
   #define PRINT(x) puts("Hello, " #x "!");
   NAMES(PRINT)
}
```

```
$ ./a.out
Hello, Tom!
Hello, Jerry!
Hello, Tyke!
Hello, Spike!
```

```
PRINT(TOM) PRINT(Jerry) PRINT(Tyke) PRINT(Spike)
```

X-Macros

```
$ ./a.out
Hello, Tom!
Hello, Jerry!
Hello, Tyke!
Hello, Spike!
Goodbye, Tom!
Goodbye, Jerry!
Goodbye, Tyke!
Goodbye, Spike!
```

```
#define NAMES(X) \
   X(Tom) X(Jerry) X(Tyke) X(Spike)

int main() {
   #define PRINT(x) puts("Hello, " #x "!");
   NAMES(PRINT)
   #define PRINT2(x) puts("Goodbye, " #x "!");
   NAMES(PRINT2)
}
```

```
   PRINT(TOM) PRINT(Jerry) PRINT(Tyke) PRINT(Spike)
```

```
   PRINT2(TOM) PRINT2(Jerry) PRINT2(Tyke) PRINT2(Spike)
```

PA中有趣的预编译

- PA头文件遵循标准头文件结构，避免被重复引用
  - in nemu/include/isa.h

```
#ifndef __ISA_H__
#define __ISA_H__
......
#endif
```

  - in nemu/include/common.h
    - 实际上是include哪一些头文件？
    - CONFIG_TARGET_AM是否有定义，在哪里?
      - make menuconfig做了什么？

```
#ifdef CONFIG_TARGET_AM
#include <klib.h>
#else
#include <assert.h>
#include <stdlib.h>
#endif
```

PA中有趣的预编译

```
#ifdef __cplusplus
extern "C" {
#endif

// ------------------------ TRM: Turing Machine -----------------------
extern   Area        heap;
void     putch       (char ch);
void     halt        (int code) __attribute__((__noreturn__));

// -------------------- IOE: Input/Output Devices --------------------
bool     ioe_init    (void);
void     ioe_read    (int reg, void *buf);
void     ioe_write   (int reg, void *buf);
#include "amdev.h"

// ---------- CTE: Interrupt Handling and Context Switching ----------
bool     cte_init    (Context *(*handler)(Event ev, Context *ctx));
void     yield       (void);
bool     ienabled    (void);
void     iset        (bool enable);
Context *kcontext    (Area kstack, void (*entry)(void *), void *arg);

// ----------------------- VME: Virtual Memory -----------------------
bool     vme_init    (void *(*pgalloc)(int), void (*pgfree)(void *));
void     protect     (AddrSpace *as);
void     unprotect   (AddrSpace *as);
void     map         (AddrSpace *as, void *vaddr, void *paddr, int prot);
Context *ucontext    (AddrSpace *as, Area kstack, void *entry);

// --------------------- MPE: Multi-Processing ---------------------
bool     mpe_init    (void (*entry)());
int      cpu_count   (void);
int      cpu_current (void);
int      atomic_xchg (int *addr, int newval);

#ifdef __cplusplus
}
#endif
```

in abstract-machine/am/include/am.h

PA中有趣的预编译

```c
#define _Log(...) \
  do { \
    printf(__VA_ARGS__); \
    log_write(__VA_ARGS__); \
  } while (0)
```

为什么要用do{...} while(0)？

```c
#if !defined(likely)
#define likely(cond)   __builtin_expect(cond, 1)
#define unlikely(cond) __builtin_expect(cond, 0)
#endif
```

likely和unlikely？

```c
#define ANSI_FG_BLACK   "\33[1;30m"
#define ANSI_FG_RED     "\33[1;31m"
#define ANSI_FG_GREEN   "\33[1;32m"
#define ANSI_FG_YELLOW  "\33[1;33m"
#define ANSI_FG_BLUE    "\33[1;34m"
#define ANSI_FG_MAGENTA "\33[1;35m"
#define ANSI_FG_CYAN    "\33[1;36m"
#define ANSI_FG_WHITE   "\33[1;37m"
#define ANSI_BG_BLACK   "\33[1;40m"
#define ANSI_BG_RED     "\33[1;41m"
#define ANSI_BG_GREEN   "\33[1;42m"
#define ANSI_BG_YELLOW  "\33[1;43m"
#define ANSI_BG_BLUE    "\33[1;44m"
#define ANSI_BG_MAGENTA "\33[1;45m"
#define ANSI_BG_CYAN    "\33[1;46m"
#define ANSI_BG_WHITE   "\33[1;47m"
#define ANSI_NONE       "\33[0m"
```

这个是什么？

PA中有趣的预编译

- 宏维护按键名称列表

```
#define AM_KEYS(_) \
  _(ESCAPE) _(F1) _(F2) _(F3) _(F4) _(F5) _(F6) _(F7) _(F8) _(F9) _(F10) _(F11) _(F12) \
  _(GRAVE) _(1) _(2) _(3) _(4) _(5) _(6) _(7) _(8) _(9) _(0) _(MINUS) _(EQUALS) _(BACKSPACE) \
  _(TAB) _(Q) _(W) _(E) _(R) _(T) _(Y) _(U) _(I) _(O) _(P) _(LEFTBRACKET) _(RIGHTBRACKET) _(BACKSLASH) \
  _(CAPSLOCK) _(A) _(S) _(D) _(F) _(G) _(H) _(J) _(K) _(L) _(SEMICOLON) _(APOSTROPHE) _(RETURN) \
  _(LSHIFT) _(Z) _(X) _(C) _(V) _(B) _(N) _(M) _(COMMA) _(PERIOD) _(SLASH) _(RSHIFT) \
  _(LCTRL) _(APPLICATION) _(LALT) _(SPACE) _(RALT) _(RCTRL) \
  _(UP) _(DOWN) _(LEFT) _(RIGHT) _(INSERT) _(DELETE) _(HOME) _(END) _(PAGEUP) _(PAGEDOWN)

#define AM_KEY_NAMES(key) AM_KEY_##key,
enum {
  AM_KEY_NONE = 0,
  AM_KEYS(AM_KEY_NAMES)
};
```

可以使用类似的宏技巧来批量定义变量。以下是几种常见的写法：

1. 批量定义整型变量

```c
#define DEFINE_INT_VARS(_) \
    _(var1) _(var2) _(var3) _(var4) _(var5)

#define DECLARE_INT(var) int var;

// 使用方式
DEFINE_INT_VARS(DECLARE_INT)
// 展开为：int var1; int var2; int var3; int var4; int var5;
```

PA中有趣的预编译

- 阅读PA代码的第一个困难(in nemu/src/nemu-main.c)
  - am_init_monitor()?
  - 还是init_monitor()?

```
int main(int argc, char*argv[]) {
  /* Initialize the monitor. */
#ifdef CONFIG_TARGET_AM
  am_init_monitor();
#else
  init_monitor(argc, argv);
#endif
  /* Start engine. */
  engine_start();
  returnis_exit_status_bad();
}
```

PA1阶段相关的宏

- in nemu/src/monitor/monitor.c

```
#ifndef CONFIG_TARGET_AM
#include <getopt.h>

void sdb_set_batch_mode();

static char *log_file = NULL;
static char *diff_so_file = NULL;
static char *img_file = NULL;
static int difftest_port = 1234;

static long load_img() {…

static int parse_args(int argc, char *argv[]) {…

void init_monitor(int argc, char *argv[]) {…
#else // CONFIG_TARGET_AM…
#endif
```

```
/* Initialize the simple debugger. */
init_sdb();

IFDEF(CONFIG_ITRACE, init_disasm());
```

PA2阶段最重要的宏

- 看看能不能读懂？
  - 可变长宏参数（ ）

```
// --- pattern matching wrappers for decode ---
#define INSTPAT(pattern, ...) do { \
  uint64_t key, mask, shift; \
  pattern_decode(pattern, STRLEN(pattern), &key, &mask, &shift); \
  if ((((uint64_t)INSTPAT_INST(s) >> shift) & mask) == key) { \
    INSTPAT_MATCH(s, ##__VA_ARGS__); \
    goto *(__instpat_end); \
  } \
} while (0)

#define INSTPAT_START(name) { const void * __instpat_end = &&concat(__instpat_end_, name);
#define INSTPAT_END(name)   concat(__instpat_end_, name): ; }
```

PA中丰富可用的宏

cpu.h ● macro.h 2 ×

ics2025 > nemu > include > C macro.h > MAP(c, f)

```
16  #ifndef __MACRO_H__
27
28  // calculate the length of an array
29  #define ARRLEN(arr) (int)(sizeof(arr) / sizeof(arr[0]))
30
31  // macro concatenation
32  #define concat_temp(x, y) x ## y
33  #define concat(x, y) concat_temp(x, y)
34  #define concat3(x, y, z) concat(concat(x, y),
35  #define concat4(x, y, z, w) concat3(concat(x,
36  #define concat5(x, y, z, v, w) concat4(concat(
37
38  // macro testing
39  // See https://stackoverflow.com/questions/266
40  #define CHOOSE2nd(a, b, ...) b
41  #define MUX_WITH_COMMA(contain_comma, a, b) CH
42  #define MUX_MACRO_PROPERTY(p, macro, a, b) MUX
43  // define placeholders for some property
44  #define __P_DEF_0  X,
45  #define __P_DEF_1  X,
46  #define __P_ONE_1  X,
47  #define __P_ZERO_0 X,
48  // define some selection functions based on th
49  #define MUXDEF(macro, X, Y)  MUX_MACRO_PROPERT
50  #define MUXNDEF(macro, X, Y) MUX_MACRO_PROPERT
51  #define MUXONE(macro, X, Y)  MUX_MACRO_PROPERT
52  #define MUXZERO(macro, X, Y) MUX_MACRO_PROPERT
53
54  // test if a boolean macro is defined
55  #define ISDEF(macro) MUXDEF(macro, 1, 0)
56  // test if a boolean macro is undefined
57  #define ISNDEF(macro) MUXNDEF(macro, 1, 0)
58  // test if a boolean macro is defined to 1
59  #define ISONE(macro) MUXONE(macro, 1, 0)
60  // test if a boolean macro is defined to 0
61  #define ISZERO(macro) MUXZERO(macro, 1, 0)
62  // test if a macro of ANY type is defined
63  // NOTE1: it ONLY works inside a function, since it calls `strcmp()`
64  // NOTE2: macros defined to themselves (#define A A) will get wrong results
65  #define isdef(macro) (strcmp("" #macro, "" str(macro)) != 0)
```

```
#define Log(format, ...) \
    _Log(ANSI_FMT("[%s:%d %s] " format, ANSI_FG_BLUE) "\n", \
         __FILE__, __LINE__, __func__, ## __VA_ARGS__)

#define Assert(cond, format, ...) \
  do { \
    if (!(cond)) { \
      MUXDEF(CONFIG_TARGET_AM, printf(ANSI_FMT(format, ANSI_FG_RED) "\n", ## __VA_ARGS__), \
        (fflush(stdout), fprintf(stderr, ANSI_FMT(format, ANSI_FG_RED) "\n", ##  __VA_ARGS__))); \
      IFNDEF(CONFIG_TARGET_AM, extern FILE* log_fp; fflush(log_fp)); \
      extern void assert_fail_msg(); \
      assert_fail_msg(); \
      assert(cond); \
    } \
  } while (0)

#define panic(format, ...) Assert(0, format, ## __VA_ARGS__)

#define TODO() panic("please implement me")
```

多路线支持的背后

- ISA_H
  - CFLAGS维护，影响不同路线选择（有兴趣去看看构建过程？）

```
#if defined(CONFIG_ISA_x86)
# define DIFFTEST_REG_SIZE (sizeof(uint32_t) * 9) // GPRs + pc
#elif defined(CONFIG_ISA_mips32)
# define DIFFTEST_REG_SIZE (sizeof(uint32_t) * 38) // GPRs + status + lo + hi + badvaddr
+ cause + pc
#elif defined(CONFIG_ISA_riscv)
#define RISCV_GPR_TYPE MUXDEF(CONFIG_RV64, uint64_t, uint32_t)
#define RISCV_GPR_NUM  MUXDEF(CONFIG_RVE , 16, 32)
#define DIFFTEST_REG_SIZE (sizeof(RISCV_GPR_TYPE) * (RISCV_GPR_NUM
#elif defined(CONFIG_ISA_loongarch32r)
# define DIFFTEST_REG_SIZE (sizeof(uint32_t) * 33) // GPRs + pc
#else
# error Unsupport ISA
#endif
```

```
#if defined(__ARCH_X86_NEMU)
# define DEVICE_BASE 0x0
#else
# define DEVICE_BASE 0xa0000000
#endif

#define MMIO_BASE 0xa0000000

#define SERIAL_PORT     (DEVICE_BASE + 0x00003f8)
#define KBD_ADDR        (DEVICE_BASE + 0x0000060)
#define RTC_ADDR        (DEVICE_BASE + 0x0000048)
#define VGACTL_ADDR     (DEVICE_BASE + 0x0000100)
#define AUDIO_ADDR      (DEVICE_BASE + 0x0000200)
#define DISK_ADDR       (DEVICE_BASE + 0x0000300)
#define FB_ADDR         (MMIO_BASE   + 0x1000000)
#define AUDIO_SBUF_ADDR (MMIO_BASE   + 0x1200000)
```

```
#if defined(__ISA_X86__)
# define nemu_trap(code) asm volatile ("int3" : :"a"(code))
#elif defined(__ISA_MIPS32__)
# define nemu_trap(code) asm volatile ("move $v0, %0; sdbbp" :
#elif defined(__riscv)
# define nemu_trap(code) asm volatile("mv a0, %0; ebreak" : :"
#elif defined(__ISA_LOONGARCH32R__)
# define nemu_trap(code) asm volatile("move $a0, %0; break 0" : :"r"(code))
#else
# error unsupported ISA __ISA__
#endif
```

有趣的预编译

- 发生在实际编译之前
  - 也称为元编程（meta-programming）
    - Gcc的预处理器同样可以处理汇编代码
    - C++中的模板元编程；Rust中的macros；…
- Pros
  - 提供灵活的用法（X-macros）
  - 接近自然语言的写法
- Cons
  - 破坏可读性IOCCC、程序分析（补全）、……

```
#define L (
int main L ) { puts L "Hello, World" ); }
```

编译与链接

（先行剧透本学期的主要内容）

编译、链接

.c → 预编译 → .i → 编译 → .s → 汇编 → .o → 链接 → .out

End.

- C语言简单（在可控时间成本里可以精通）
- C语言通用（大量系统是C语言编写的）
- C语言实现对底层机器的精准控制（鸿蒙）
- 推荐阅读：The Art of Readable Code