“造轮子”的方法和乐趣

王慧妍
why@nju.edu.cn

南京大学 计算机科学与技术系 计算机软件研究所

本讲概述

《计算机系统基础》课到底学了啥？

- 本讲内容
  - 一个关于编译运行的问题
  - 那些年我们造过的轮子

一个关于编译运行的问题

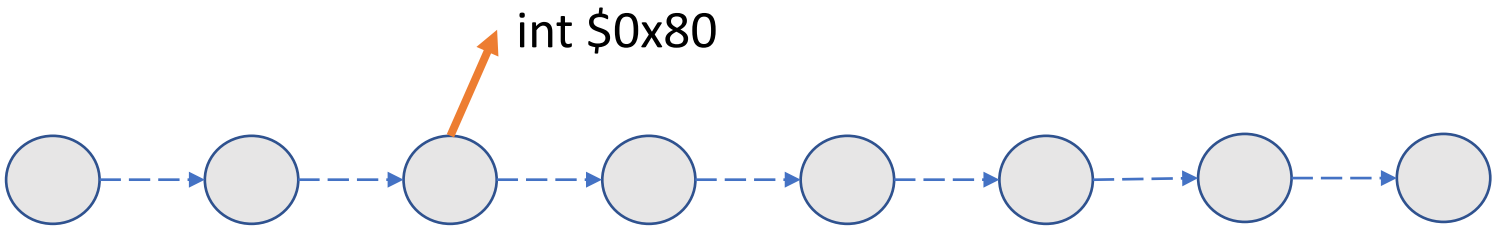
回到第一次C语言课

- 在IDE里，[illegible]
  - 编译、链[illegible]
    - .c → [illegible]接 → a.out
  - 加载执行
    - ./a.o[illegible]

```
$ ls -l
total 4
-rw-rw-r-- 1 why why 14 12月  16 08:19 a.c
$ gcc a.c
$ ls -l
total 20
-rw-rw-r-- 1 why why    14 12月  16 08:19 a.c
-rwxrwxr-x 1 why why 15912 12月  16 08:19 a.out
```

- 现在，一位同学对这个过程提出了质疑
  - 我不信！我就觉得是 gcc 一个程序直接搞定的
    - 道理上完全可以这么实现
  - 如何说服这位同学?

所有程序的执行都是状态机




```
$ ls -l
total 4
-rw-rw-r-- 1 why why 14 12月  16 08:19 a.c
$ gcc a.c
$ ls -l
total 20
-rw-rw-r-- 1 why why    14 12月  16 08:19 a.c
-rwxrwxr-x 1 why why 15912 12月  16 08:19 a.out
```

答案：用工具!

- 观察 trace: 查看 gcc 是否调用了其他命令
  - `strace -f -qq gcc a.c 2>&1 | vim -`
    - 可以使用 grep (shell) 或 :g! (vim) 筛选感兴趣的系统调用

```
STRACE(1)                  General Commands Manual                 STRACE(1)

NAME
       strace - trace system calls and signals

SYNOPSIS
       strace [-ACdffhikqqrtttTvVwxxyyzZ] [-I n] [-b execve] [-e expr]...
              [-O overhead] [-S sortby] [-U columns] [-a column] [-o file]
              [-s strsize] [-X format] [-P path]... [-p pid]...
              [--seccomp-bpf] { -p pid | [-DDD] [-E var[=val]]...
              [-u username] command [args] }

       strace -c [-dfwzZ] [-I n] [-b execve] [-e expr]... [-O overhead]
              [-S sortby] [-U columns] [-P path]... [-p pid]...
              [--seccomp-bpf] { -p pid | [-DDD] [-E var[=val]]...
              [-u username] command [args] }

DESCRIPTION
       In the simplest case strace runs the specified command until it  ex-
       its.  It intercepts and records the system calls which are called by
       a process and the signals which are received by a process.  The name
       of  each system call, its arguments and its return value are printed
       on standard error or to the file specified with the -o option.

       strace is a useful diagnostic, instructional,  and  debugging  tool.
       System administrators, diagnosticians and trouble-shooters will find
 Manual page strace(1) line 1 (press h for help or q to quit)
```

管理 控制 视图 热键 设备 帮助

```
$ ls
a.c  a.out  gdb-internals.txt
$
```

管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
$ strace gcc a.c 2>&1 | vim -
```

Right Shift + Right Alt

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
201 readlink("/usr/bin/gcc-10", "x86_64-linux-gnu-gcc-10", 1023) = 23
202 readlink("/usr/bin/x86_64-linux-gnu-gcc-10", 0x7ffe8a430820, 1023) = -1 EINVAL (I
203 openat(AT_FDCWD, "/tmp/ccIx8ZMR.s", O_RDWR|O_CREAT|O_EXCL, 0600) = 4
204 close(4)                                = 0
```

STRACE(1) General Commands Manual STRACE(1)

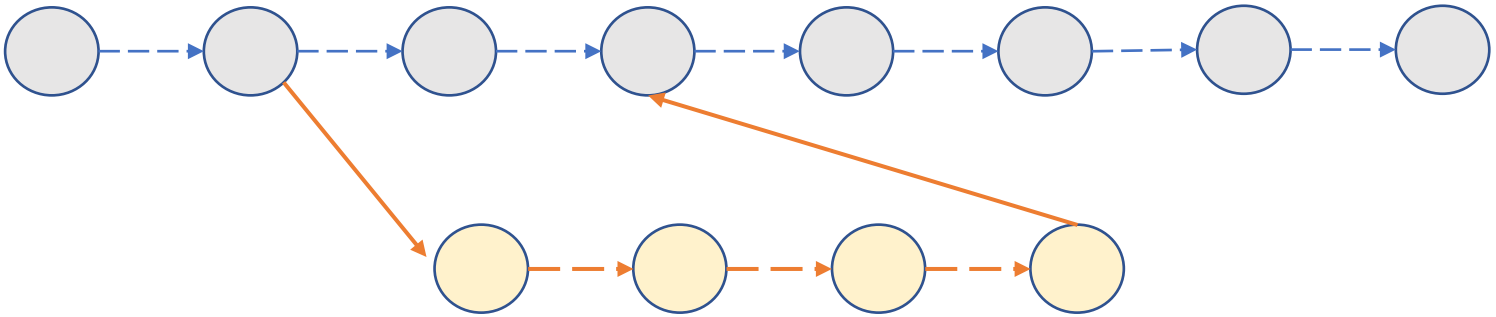
NAME

strace - trace system calls and signals

-f
--follow-forks

Trace child processes as they are created by currently traced processes as a result of the **fork**(2), **vfork**(2) and **clone**(2) system calls. Note that **-p** PID **-f** will attach all threads of process PID if it is multi-threaded, not only thread with thread_id = PID.

```
224 vfork()                              = 17244
225 close(5)                             = 0
[No Name][+1]                    [strace] unix utf-8 Ln 201, Col 1/234
```



管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
$ man strace
```

Right Shift + Right Alt

管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
249 newfstatat(AT_FDCWD, "/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper", {st_mode=S_
    IFREG|0755, st_size=655432, ...}, 0) = 0
250 access("/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper", X_OK) = 0
251 access("/tmp", R_OK|W_OK|X_OK)           = 0
252 openat(AT_FDCWD, "/tmp/ccFnrVKs.s", O_RDWR|O_CREAT|O_EXCL, 0600) = 4
253 close(4)                                 = 0
254 newfstatat(AT_FDCWD, "/usr/lib/gcc/x86_64-linux-gnu/10/cc1", {st_mode=S_IFREG|07
    55, st_size=25697696, ...}, 0) = 0
255 access("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", X_OK) = 0
256 pipe2([4, 5], O_CLOEXEC)                 = 0
257 vfork( <unfinished ...>
258 [pid 17480] close(4)                     = 0
259 [pid 17480] execve("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", ["/usr/lib/gcc/x86_64
    -linux-gnu/10"..., "-quiet", "-imultiarch", "x86_64-linux-gnu", "a.c", "-quiet",
     "-dumpbase", "a.c", "-mtune=generic", "-march=x86-64", "-auxbase", "a", "-fasyn
    chronous-unwind-tables", "-fstack-protector-strong", "-Wformat", "-Wformat-secur
    ity", "-fstack-clash-protection", "-fcf-protection", "-o", "/tmp/ccFnrVKs.s"], 0
    x18ec000 /* 66 vars */ <unfinished ...>
260 [pid 17479] <... vfork resumed>)          = 17480
261 [pid 17479] close(5)                     = 0
262 [pid 17479] read(4, "", 16)              = 0
263 [pid 17480] <... execve resumed>)        = 0
264 [pid 17479] close(4)                     = 0
265 [pid 17480] brk(NULL <unfinished ...>
266 [pid 17479] wait4(17480,  <unfinished ...>
[No Name][+1]                                [strace] unix utf-8 Ln 266, Col 1/2923
```

Right Shift + Right Alt

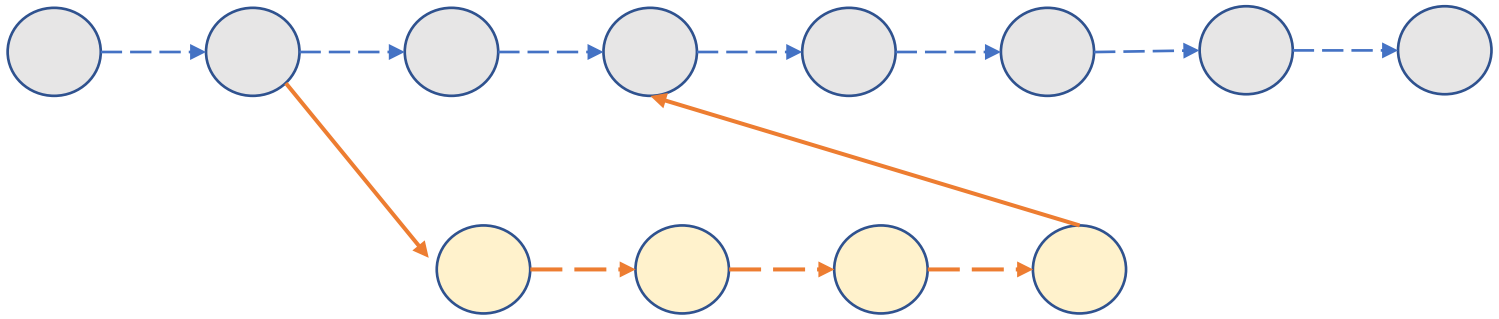
管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
345 [pid 17480] mmap(0x7fc30ecdf000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FI
    XED|MAP_DENYWRITE, 4, 0x1dc000) = 0x7fc30ecdf000
346 [pid 17480] mmap(0x7fc30ece5000, 33688, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FI
    XED|MAP_ANONYMOUS, -1, 0) = 0x7fc30ece5000
347 [pid 17480] close(4)                          = 0
348 [pid 17480] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
    1, 0) = 0x7fc30eaff000
349 [pid 17480] arch_prctl(ARCH_SET_FS, 0x7fc30eaffac0) = 0
350 [pid 17480] mprotect(0x7fc30ecdf000, 12288, PROT_READ) = 0
351 [pid 17480] mprotect(0x7fc30edb9000, 4096, PROT_READ) = 0
352 [pid 17480] mprotect(0x7fc30edd7000, 4096, PROT_READ) = 0
353 [pid 17480] mprotect(0x7fc30edde000, 4096, PROT_READ) = 0
354 [pid 17480] mprotect(0x7fc30ee61000, 4096, PROT_READ) = 0
355 [pid 17480] mprotect(0x7fc30f108000, 8192, PROT_READ) = 0
356 [pid 17480] mprotect(0x7fc30f12e000, 4096, PROT_READ) = 0
357 [pid 17480] mprotect(0x7fc30f311000, 4096, PROT_READ) = 0
358 [pid 17480] mprotect(0x1c75000, 8192, PROT_READ) = 0
359 [pid 17480] mprotect(0x7fc30f359000, 8192, PROT_READ) = 0
360 [pid 17480] munmap(0x7fc30f316000, 69006) = 0
361 [pid 17480] brk(NULL)                           = 0x3313000
362 [pid 17480] brk(0x3334000)                      = 0x3334000
363 [pid 17480] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=65536*1024, rlim_max=RLIM
    64_INFINITY}) = 0
364 [pid 17480] openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXE
    C) = 4
[No Name][+1]                                  [strace] unix utf-8 Ln 345, Col 1/2923
```

Right Shift + Right Alt

.c到.s



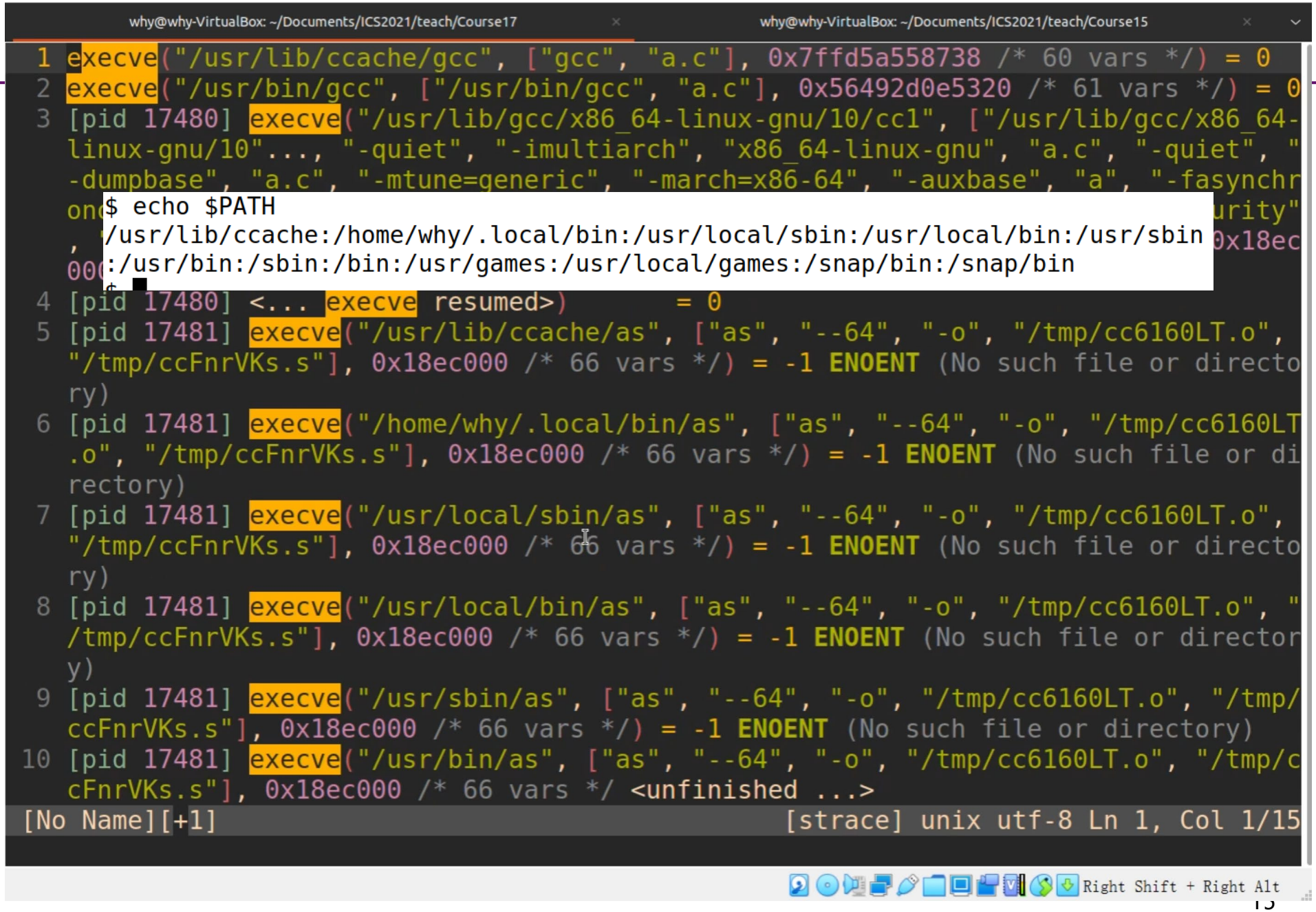
执行cc1
.c → .s

管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × | why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
248 access("/usr/lib/gcc/x86_64-linux-gnu/10/", X_OK) = 0
249 newfstatat(AT_FDCWD, "/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper", {st_mode=S_
    IFREG|0755, st_size=655432, ...}, 0) = 0
250 access("/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper", X_OK) = 0
251 access("/tmp", R_OK|W_OK|X_OK)           = 0
252 openat(AT_FDCWD, "/tmp/ccFnrVKs.s", O_RDWR|O_CREAT|O_EXCL, 0600) = 4
253 close(4)                                 = 0
254 newfstatat(AT_FDCWD, "/usr/lib/gcc/x86_64-linux-gnu/10/cc1", {st_mode=S_IFREG|07
    55, st_size=25697696, ...}, 0) = 0
255 access("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", X_OK) = 0
256 pipe2([4, 5], O_CLOEXEC)                 = 0
257 vfork( <unfinished ...>
258 [pid 17480] close(4)                     = 0
259 [pid 17480] execve("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", ["/usr/lib/gcc/x86_64
    -linux-gnu/10"..., "-quiet", "-imultiarch", "x86_64-linux-gnu", "a.c", "-quiet",
     "-dumpbase", "a.c", "-mtune=generic", "-march=x86-64", "-auxbase", "a", "-fasyn
    chronous-unwind-tables", "-fstack-protector-strong", "-Wformat", "-Wformat-secur
    ity", "-fstack-clash-protection", "-fcf-protection", "-o", "/tmp/ccFnrVKs.s"], 0
    x18ec000 /* 66 vars */ <unfinished ...>
260 [pid 17479] <... vfork resumed>)          = 17480
261 [pid 17479] close(5)                     = 0
262 [pid 17479] read(4, "", 16)              = 0
263 [pid 17480] <... execve resumed>)        = 0
264 [pid 17479] close(4)                     = 0
265 [pid 17480] brk(NULL <unfinished ...>
[No Name][+1]                          [strace] unix utf-8 Ln 264, Col 1/2923
```

Right Shift + Right Alt


why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17
why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15
1 execve("/usr/lib/ccache/gcc", ["gcc", "a.c"], 0x7ffd5a558738 /* 60 vars */) = 0
2 execve("/usr/bin/gcc", ["/usr/bin/gcc", "a.c"], 0x56492d0e5320 /* 61 vars */) = 0
3 [pid 17480] execve("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", ["/usr/lib/gcc/x86_64-
linux-gnu/10"..., "-quiet", "-imultiarch", "x86_64-linux-gnu", "a.c", "-quiet", "
-dumpbase", "a.c", "-mtune=generic", "-march=x86-64", "-auxbase", "a", "-fasynchr
$ echo $PATH
/usr/lib/ccache:/home/why/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin
:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
4 [pid 17480] <... execve resumed>) = 0
5 [pid 17481] execve("/usr/lib/ccache/as", ["as", "--64", "-o", "/tmp/cc6160LT.o",
"/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directo
ry)
6 [pid 17481] execve("/home/why/.local/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT
.o", "/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or di
rectory)
7 [pid 17481] execve("/usr/local/sbin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o",
"/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directo
ry)
8 [pid 17481] execve("/usr/local/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "
/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or director
y)
9 [pid 17481] execve("/usr/sbin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "/tmp/
ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directory)
10 [pid 17481] execve("/usr/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "/tmp/c
cFnrVKs.s"], 0x18ec000 /* 66 vars */ <unfinished ...>
[No Name][+1]
[strace] unix utf-8 Ln 1, Col 1/15
Right Shift + Right Alt

管理 控制 视图 热键 设备 帮助

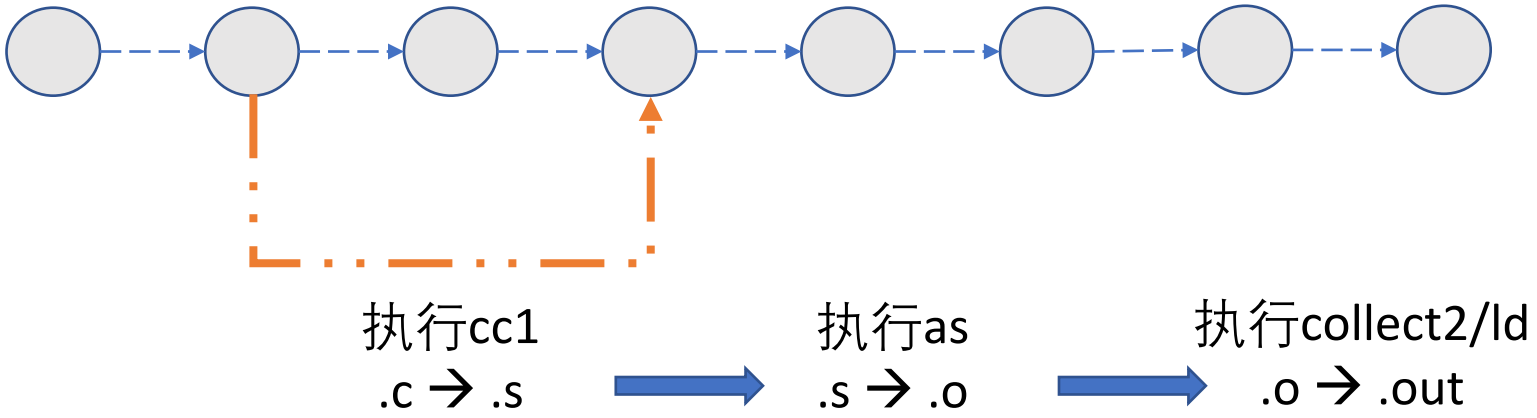
why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × | why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
 1 execve("/usr/lib/ccache/gcc", ["gcc", "a.c"], 0x7ffd5a558738 /* 60 vars */) = 0
 2 execve("/usr/bin/gcc", ["/usr/bin/gcc", "a.c"], 0x56492d0e5320 /* 61 vars */) = 0
 3 [pid 17480] execve("/usr/lib/gcc/x86_64-linux-gnu/10/cc1", ["/usr/lib/gcc/x86_64-
   linux-gnu/10"..., "-quiet", "-imultiarch", "x86_64-linux-gnu", "a.c", "-quiet", "
   -dumpbase", "a.c", "-mtune=generic", "-march=x86-64", "-auxbase", "a", "-fasynchr
   onous-unwind-tables", "-fstack-protector-strong", "-Wformat", "-Wformat-security"
   , "-fstack-clash-protection", "-fcf-protection", "-o", "/tmp/ccFnrVKs.s"], 0x18ec
   000 /* 66 vars */ <unfinished ...>
 4 [pid 17480] <... execve resumed>)       = 0
 5 [pid 17481] execve("/usr/lib/ccache/as", ["as", "--64", "-o", "/tmp/cc6160LT.o",
   "/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directo
   ry)
 6 [pid 17481] execve("/home/why/.local/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT
   .o", "/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or di
   rectory)
 7 [pid 17481] execve("/usr/local/sbin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o",
   "/tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directo
   ry)
 8 [pid 17481] execve("/usr/local/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "
   /tmp/ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or director
   y)
 9 [pid 17481] execve("/usr/sbin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "/tmp/
   ccFnrVKs.s"], 0x18ec000 /* 66 vars */) = -1 ENOENT (No such file or directory)
10 [pid 17481] execve("/usr/bin/as", ["as", "--64", "-o", "/tmp/cc6160LT.o", "/tmp/c
   cFnrVKs.s"], 0x18ec000 /* 66 vars */ <unfinished ...>
[No Name][+1]                                  [strace] unix utf-8 Ln 1, Col 1/15
```

Right Shift + Right Alt

回到最初的问题

- 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`
- 现在，一位同学对这个过程提出了质疑
  - 如何说服这位同学?



答案：用工具!

- 观察 trace: 查看 gcc 是否调用了其他命令
  - `strace -f -qq gcc a.c 2>&1 | vim -`
    - 可以使用 `grep` (shell) 或 `:g!` (vim) 筛选感兴趣的系统调用
- 调试 gcc: 查看每一个阶段的中间结果
  - 在哪里打断点?
    - gdb-internals.txt

管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

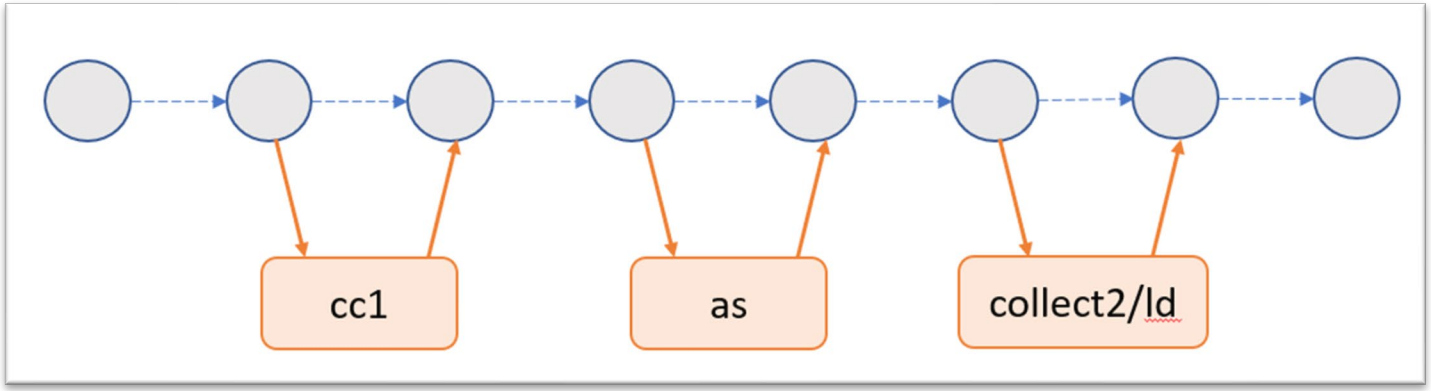
```
$ ls
a.c  a.out  gdb-internals.txt
$
```

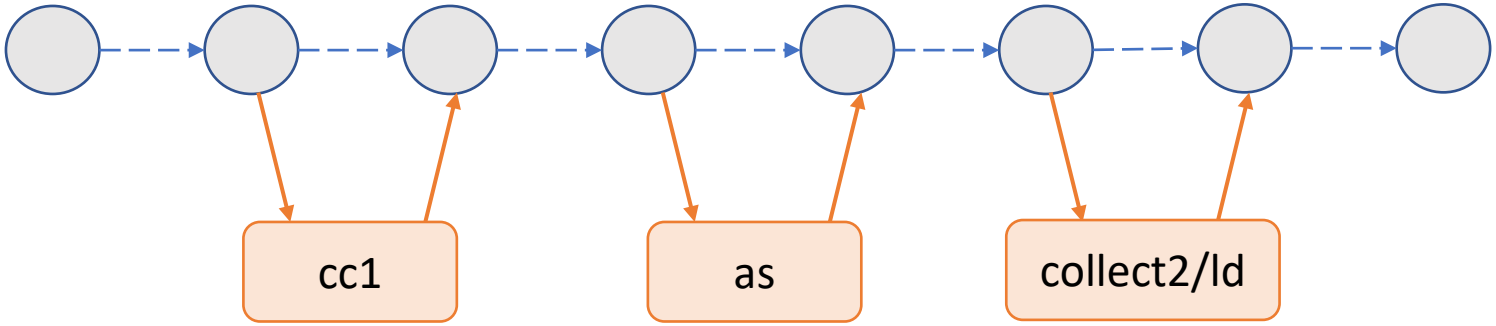
Right Shift + Right Alt

回到第一次C语言课

- 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`

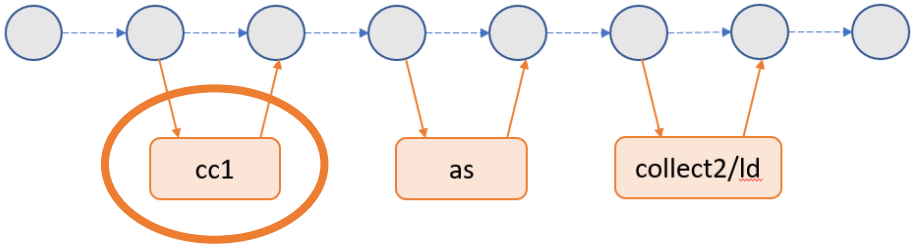
- 现在，一位同学对这个过程提出了质疑
  - 我不信！我就觉得是 gcc 一个程序直接搞定的
    - 道理上完全可以这么实现
  - 如何说服这位同学?





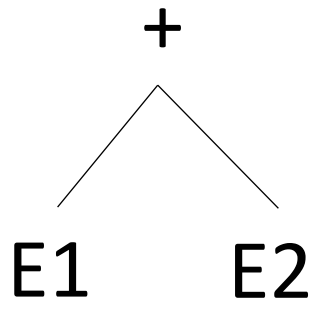
课堂上见过的轮子们

编译器 (.c → .s)

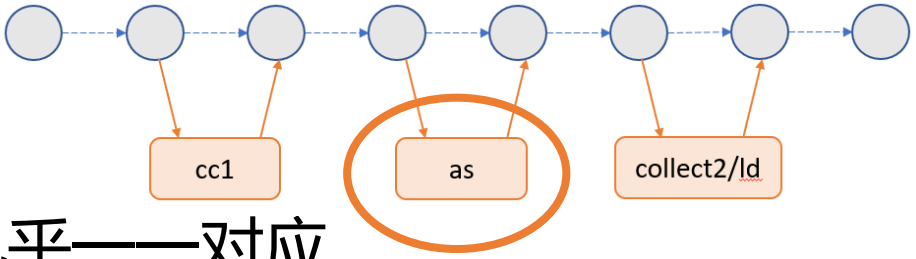


- 同 “表达式求值问题” (PA1)
  - 编译器 (OJ 题)：输入一个字符串，输出计算它的汇编代码
    - 表达式 `(a + b) * (c + d)`
    - → 指令序列 (stack machine)
      - `push $a; push $b; add; push $c; push $d; add; mul`

- 现代编译器
  - 预编译 → 词法分析 → 语法树 → IR 中间代码 → 多趟优化 (内联、传播/折叠、删除冗余) → 指令生成/寄存器分配



汇编器（.s → .o）



- 除去预编译，汇编代码和指令几乎一一对应
  - 根据指令集手册规定翻译
  - 生成符合 ELF 规范的二进制目标文件
    - `printf("\x7f\x45\x4c\x46");...`

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000
00000010: 0100 3e00 0100 0000 0000 0000 0000 0000
00000020: 0000 0000 0000 0000 b81e 0000 0000 0000
00000030: 0000 0000 4000 0000 0000 4000 0f00 0e00
00000040: 5548 89e5 4883 ec10 4889 7df8 4889 75f0
00000050: 488b 55f0 488b 45f8 4801 d048 8905 0000
00000060: 0000 488b 0500 0000 00ba 0d00 0000 4889
00000070: c648 8d3d 0000 0000 b800 0000 00e8 0000
00000080: 0000 488b 0500 0000 0048 8b10 4889 1500
00000090: 0000 008b 5008 8915 0000 0000 0fb6 400c
```

管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

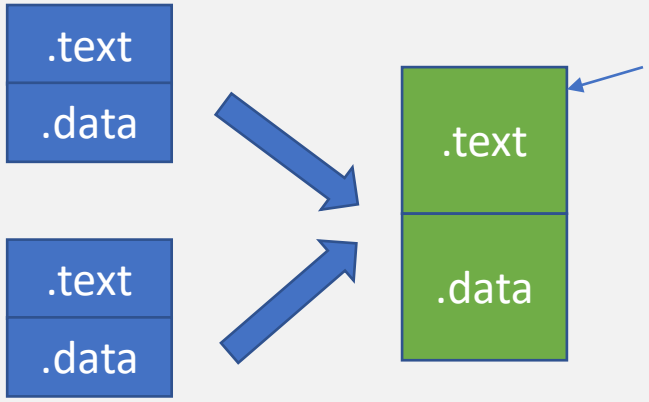
```
$ ls
a.c  a.out  gdb-internals.txt
$
```

Right Shift + Right Alt

链接器（.o → a.out）

- 按照 ld script 指示完成二进制文件构造、符号解析和重定位
  - gcc a.c -Wl,--verbose
    - “.”代表当前位置；基本功能：粘贴；定义符号

```
SECTIONS {
    . = 0x100000;
    .text : { *(entry) *(.text*) }
    etext = .; _etext = .;
    .rodata : { *(.rodata*) } .data : { *(.data*) }
    edata = .; _edata = .;
    .bss : { *(.bss*) }
    _start_start = ALIGN(4096);
    . = _start_start + 0x8000;
    _stack_pointer = .;
    end = .; _end = .;
    _heap_start = ALIGN(4096);
    _heap_end = 0x8000000;
}
```



管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × | why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

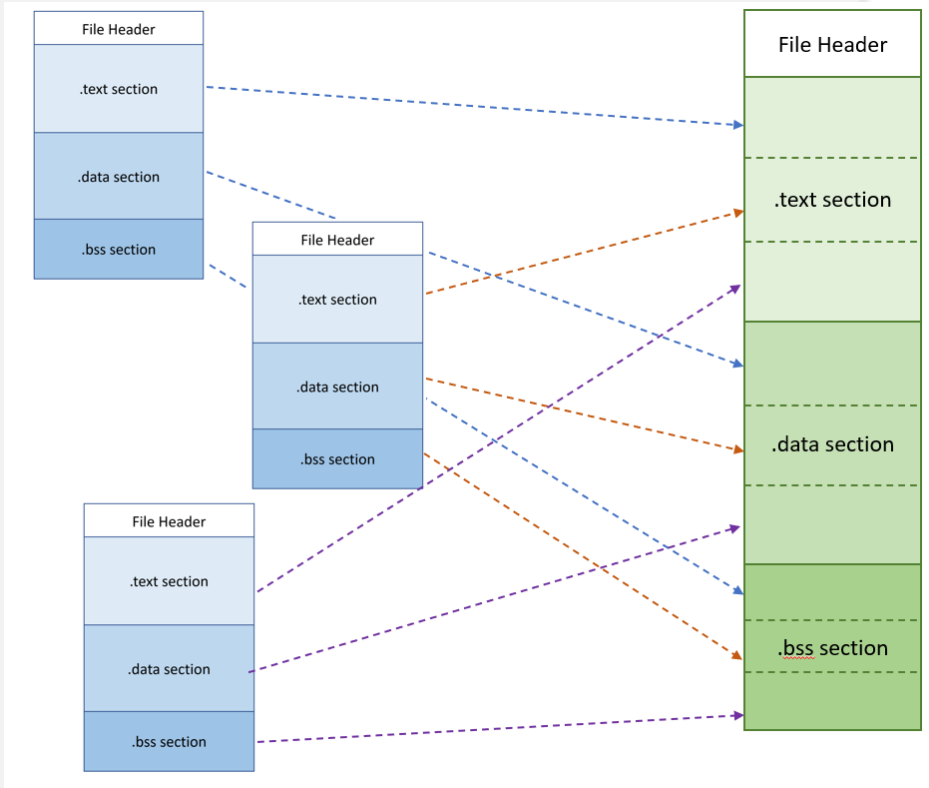
-$

Right Shift + Right Alt

链接器（.o → a.out）

- 按照 ld script 指示完成二进制文件构造、符号解析和重定位
  - gcc a.c -Wl,--verbose
    - “.”代表当前位置；基本功能：粘贴；定义符号

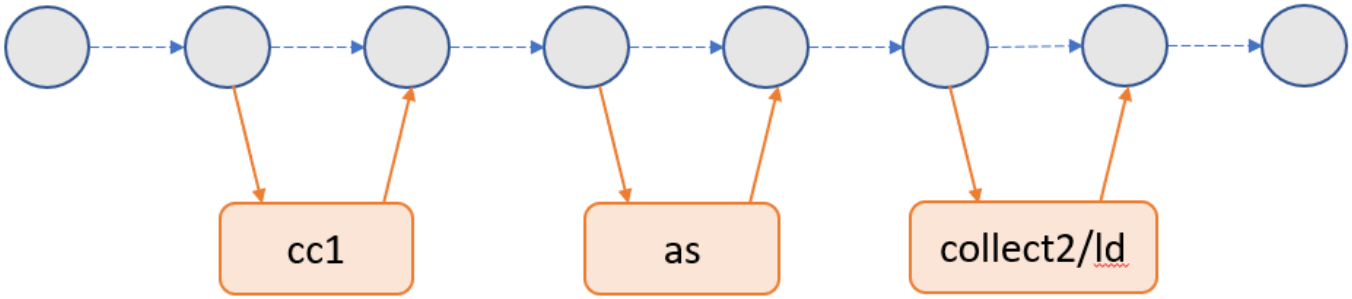
```
SECTIONS {
    . = 0x100000;
    .text : { *(entry) *(.text*) }
    etext = .; _etext = .;
    .rodata : { *(.rodata*) }
    .data : { *(.data*) }
    edata = .; _edata = .;
    .bss : { *(.bss*) }
    _start_start = ALIGN(4096);
    . = _start_start + 0x8000;
    _stack_pointer = .;
    end = .; _end = .;
    _heap_start = ALIGN(4096);
    _heap_end = 0x8000000;
}
```



加载器

- PA里做过了

- 编译、链接、加载
  - 好像没那么难啊（都是表达式求值和翻译）



单元测试框架

- YEMU 中的神奇代码

```
#define TESTCASES(_)
    \ _(1, 0b11100111, random_rm, ASSERT_EQ(newR[0], oldM[7]) )
    \ _(2, 0b00000100, random_rm, ASSERT_EQ(newR[1], oldR[0]) )
    \ _(3, 0b11100101, random_rm, ASSERT_EQ(newR[0], oldM[5]) )
    \ _(4, 0b00010001, random_rm, ASSERT_EQ(newR[0], oldR[0] + oldR[1]) )
    \ _(5, 0b11110111, random_rm, ASSERT_EQ(newM[7], oldR[0]) )
```

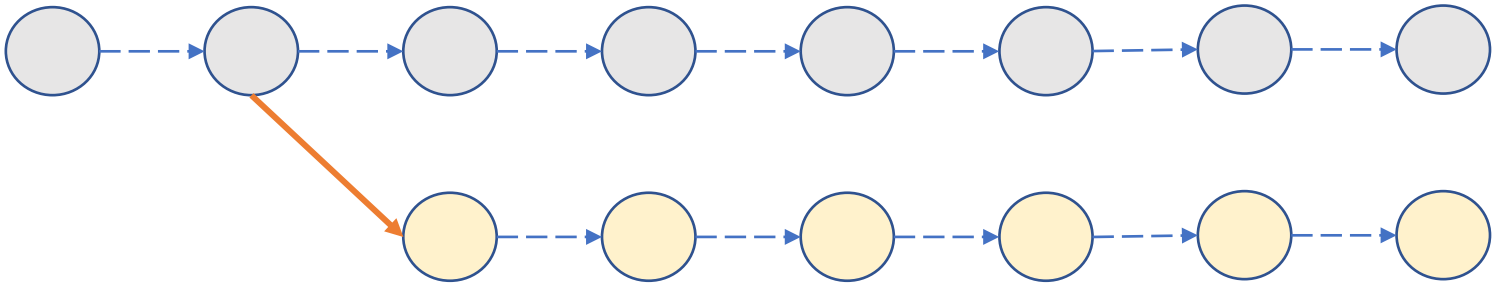
- 但是我相信大家都没有做好单元测试

Differential Testing

- 设置好状态；各走一步

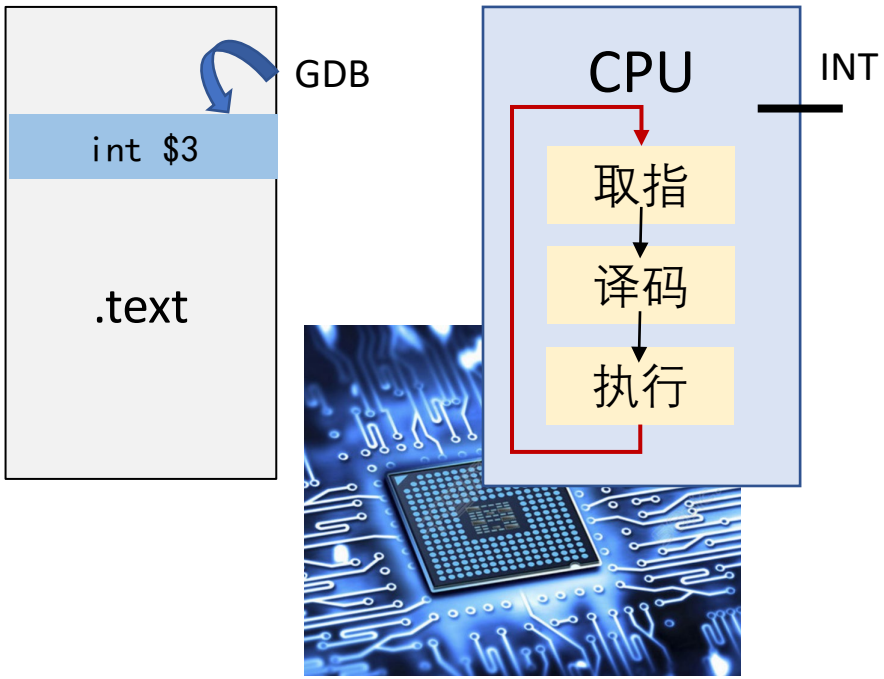
```
for i in range(int(sys.argv[1])):
    print('\n'.join(['si'] + [f'p ${r}' for r in ['eax', ...]]))
```

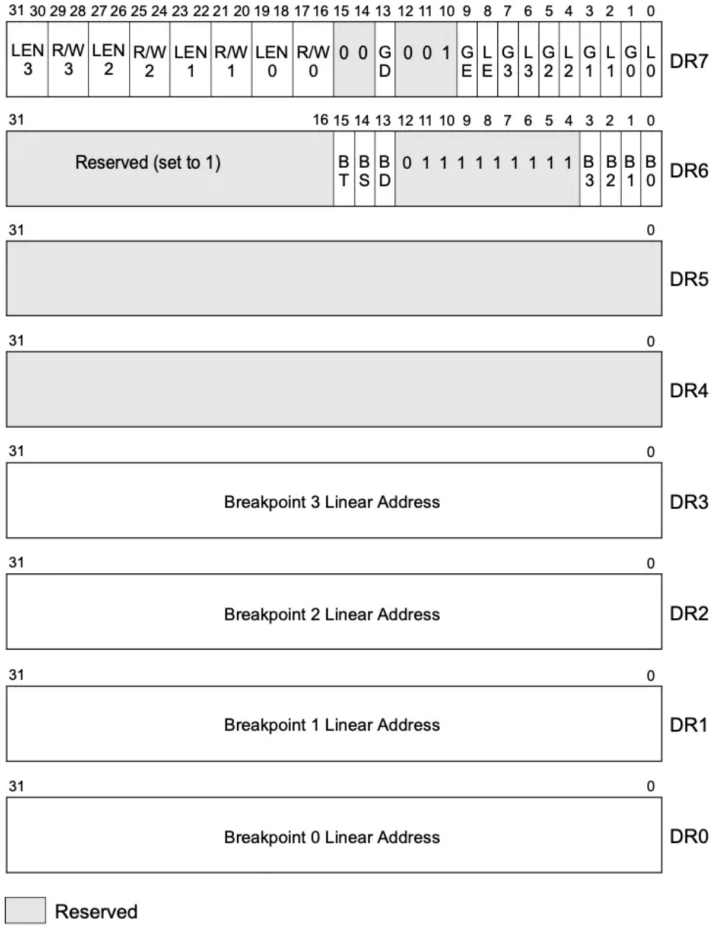
```
N=10000
    diff <(python3 cmdgen.py $N | ./x86-nemu-datui img) \
         <(python3 cmdgen.py $N | ./x86-nemu img)
```



调试器GDB

- 好奇它是怎么实现的?
- 考虑核心功能：在任意 PC 的断点
- 其他功能都可以基于断点实现
  - 单步调试：在下一条指令打断点
  - watch point: 单步调试 + 检查条件





管理 控制 视图 热键 设备 帮助

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × | why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

```
$
```

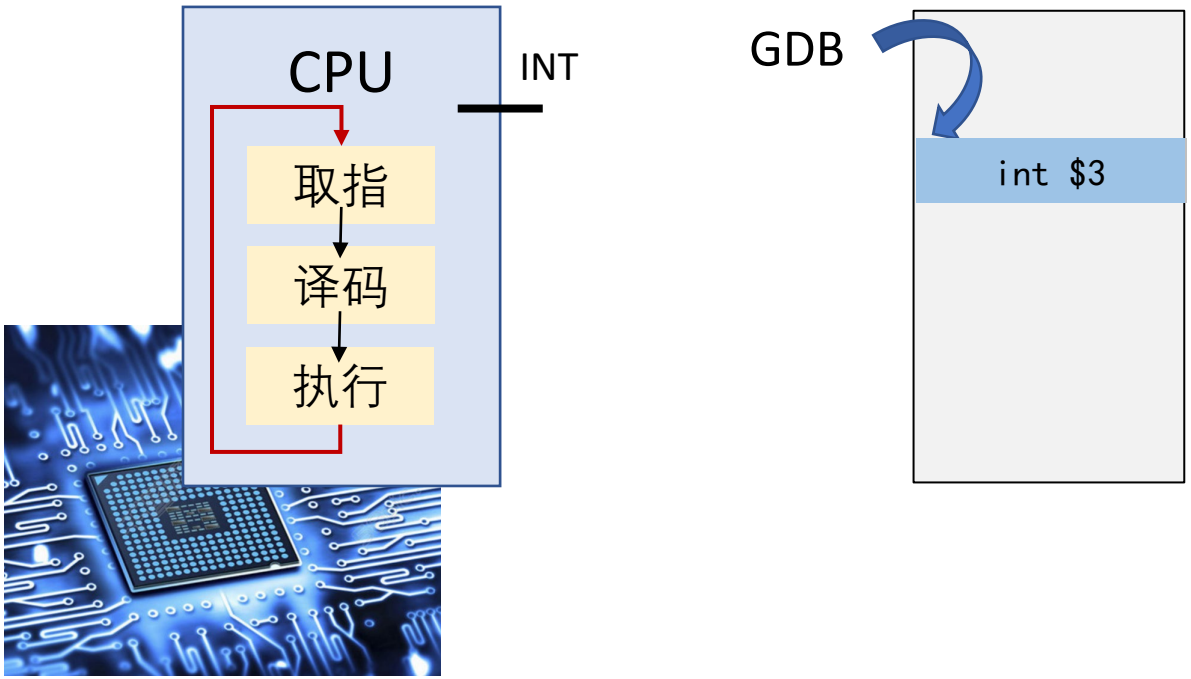

Right Shift + Right Alt

```
MPROTECT(2)                Linux Programmer's Manual                MPROTECT(2)

NAME
       mprotect, pkey_mprotect - set protection on a region of memory

SYNOPSIS
       #include <sys/mman.h>

       int mprotect(void *addr, size_t len, int prot);

       #define _GNU_SOURCE             /* See feature_test_macros(7) */
       #include <sys/mman.h>

       int pkey_mprotect(void *addr, size_t len, int prot, int pkey);

DESCRIPTION
       mprotect() changes the access protections for the calling process's memory
       pages  containing  any  part  of  the  address  range  in  the  interval
       [addr, addr+len-1].  addr must be aligned to a page boundary.

       If  the  calling  process tries to access memory in a manner that violates
       the protections, then the  kernel  generates  a  SIGSEGV  signal  for  the
       process.

       prot  is  a combination of the following access flags: PROT_NONE or a bit-
       wise-or of the other values in the following list:
```

调试器GDB

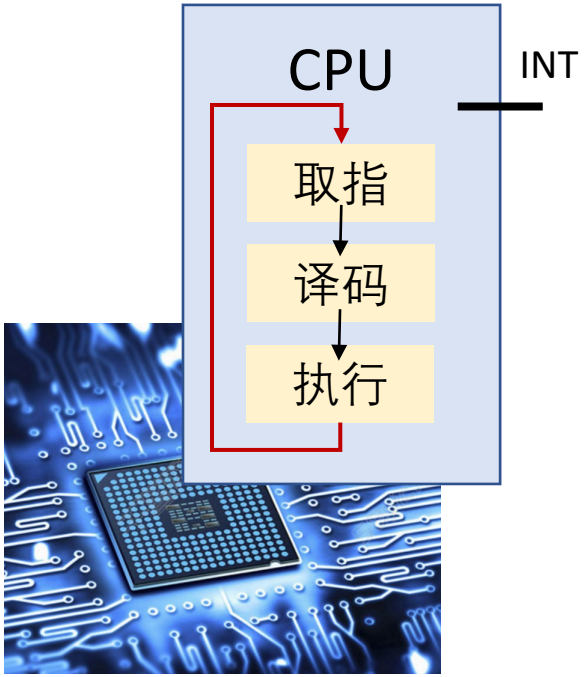
- 好奇它是怎么实现的?
- 考虑核心功能
- 其他功能都可
  - 单步调试:
  - watch point:

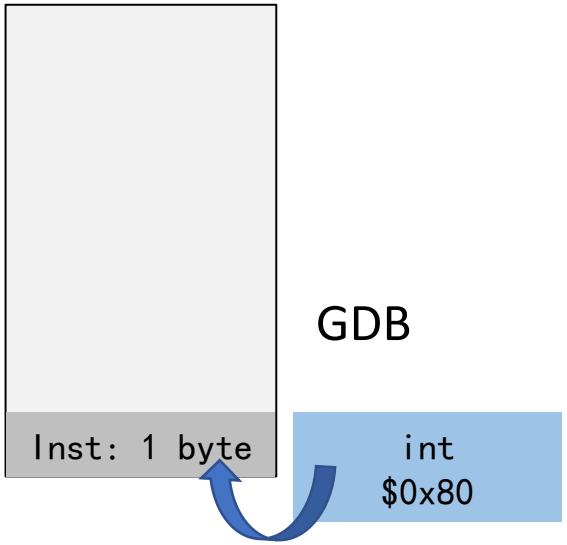
> The **ptrace**() system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers. It is primarily used to implement breakpoint debugging and system call tracing.



调试器GDB

- 好奇它是怎么实现的?
- 考虑核心功能：在任意 PC 的断点
- 其他功能都可以基于断点实现
  - 单步调试：在下一条指令打断点
  - watch point: 单步调试 + 检查条件





管理 控制 视图 热键 设备 帮助

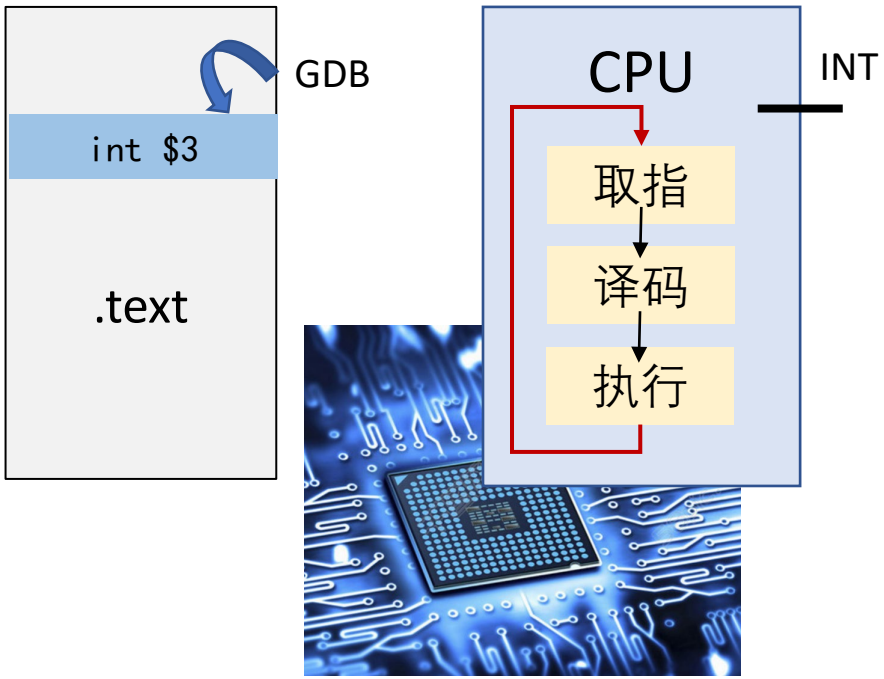
why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

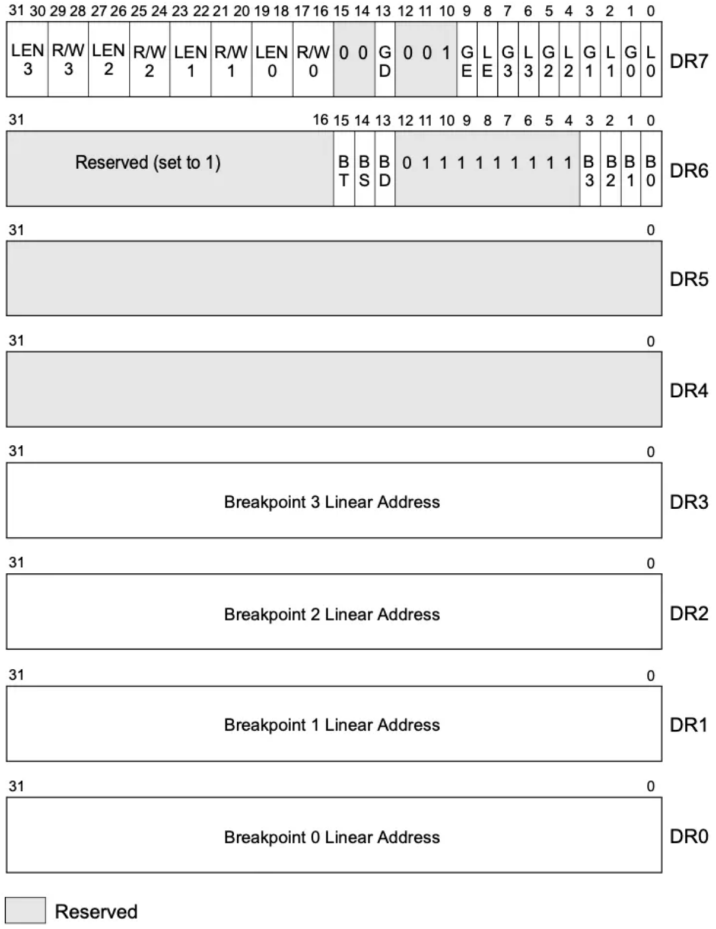
```
$ vim a.S
```


Right Shift + Right Alt

调试器GDB

- 好奇它是怎么实现的?
- 考虑核心功能：在任意 PC 的断点
- 其他功能都可以基于断点实现
  - 单步调试：在下一条指令打断点
  - watch point: 单步调试 + 检查条件





Trace/Profiler

- strace: 刚才已经展示过威力了
  - 如何实现?

```
$ strace ls
execve("/usr/bin/ls", ["ls"], 0x7fff683feed0 /* 60 vars */) = 0
brk(NULL)                               = 0x55d1f69ba000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff7ba12c80) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=69006, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 69006, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff23c092000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 p\0\0\0\0\0\0"..., 832) = 83
2
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=166272, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff23c0
90000
mmap(NULL, 177672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff23c064000
mmap(0x7ff23c06a000, 106496, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
E, 3, 0x6000) = 0x7ff23c06a000
mmap(0x7ff23c084000, 32768, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x200
00) = 0x7ff23c084000
mmap(0x7ff23c08c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
, 3, 0x27000) = 0x7ff23c08c000
mmap(0x7ff23c08e000, 5640, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
, -1, 0) = 0x7ff23c08e000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\206\2\0\0\0\0\0"..., 832
) = 832
```

why@why-VirtualBox: ~/Documents/ICS2021/teach/Course17 × why@why-VirtualBox: ~/Documents/ICS2021/teach/Course15 ×

PTRACE(2) Linux Programmer's Manual PTRACE(2)

NAME

ptrace - process trace

SYNOPSIS

```
#include <sys/ptrace.h>

long ptrace(enum __ptrace_request request, pid_t pid,
            void *addr, void *data);
```

DESCRIPTION

The **ptrace**() system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers. It is primarily used to implement breakpoint debugging and system call tracing.

A tracee first needs to be attached to the tracer. Attachment and subsequent commands are per thread: in a multithreaded process, every thread can be individually attached to a (potentially different) tracer, or left not attached and thus not debugged. Therefore, "tracee" always means "(one) thread", never "a (possibly multithreaded) process". Ptrace commands are always sent to a specific tracee using a call of the form

```
ptrace(PTRACE_foo, pid, ...)
```

Manual page ptrace(2) line 1 (press h for help or q to quit)

静态分析工具 Lint

- 大家见到的第一个 lint: `gcc -Wall -Werror`
  - cppcheck
  - Cert C Coding Standard
    - 自带 checker

Introduction

由 Robert Seacord创建, 最终由 Robert Schiela修改于 一月 03, 2017

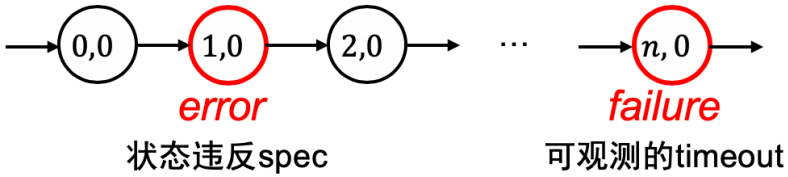
This standard provides rules for secure coding in the C programming language. The goal of these rules and recommendations is to develop safe, reliable, and secure systems, for example by eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. Conformance to the coding rules defined in this standard are necessary (but not sufficient) to ensure the safety, reliability, and security of software systems developed in the C programming language. It is also necessary, for example, to have a safe and secure design. Safety-critical systems typically have stricter requirements than are imposed by this coding standard, for example requiring that all memory be statically allocated. However, the application of this coding standard will result in high-quality systems that are reliable, robust, and resistant to attack.

动态分析工具 Sanitizer

- 本质：运行时额外增加的 assert()
  - 回顾调试理论

```
for (int i = 0; i < n; i++)
  for (int j = 0; j < n; i++)
    ...
```

fault 程序bug



- 一些非常实用的 assertions
  - assert(IS_GUESTPTR(ptr));
    - (PMEM_MAP_START <= (x) && (x) < PMEM_MAP_END)
  - assert(IS_SMALLINT(x));
    - (0 <= (x) && (x) <= 100)
  - assert(IS_BOOL(ptr));
    - ((x) == 0 || (x) == 1)

总结

我们学到了什么?

“程序是个状态机。”

“我们可以观测状态机的设计、实现和执行。”

- 能实现几乎任何工具 (的玩具版)
  - 并且能在需要的使用善用它们
    - 编译器 (gcc)
    - 汇编器 (as)
    - 链接器 (ld)
    - 调试器 (gdb)
    - 追踪器 (strace/ltrace)
    - profiler (perf)

Cheers!

(你们完成了了不起的事情！)