第 2 讲：操作系统与系统结构和程序设计语言

第三节：Rust 语言与系统编程

向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 2 月 16 日

提纲

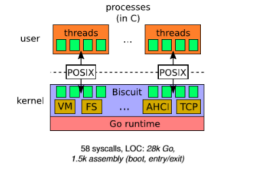
1. 第三节：Rust 语言与系统编程
   - 系统编程语言
   - 高级语言编译到机器指令

系统编程语言：对当前编程语言的理解

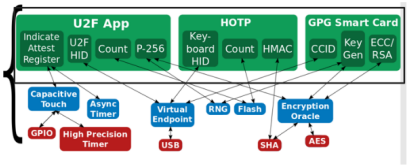
- 系统编程语言（system language）定义现在似乎没有特别严谨和一致的定义
- 系统编程语言用于构建控制底层计算机硬件的软件系统，并提供由用于构建应用程序和服务的更高级应用程序编程语言使用的软件平台。
- 开发操作系统的系统编程语言其实很多；还离不开汇编语言。

系统编程语言：对当前编程语言的理解

- 系统编程语言（system language）定义现在似乎没有特别严谨和一致的定义
- 系统编程语言用于构建控制底层计算机硬件的软件系统，并提供由用于构建应用程序和服务的更高级应用程序编程语言使用的软件平台。
- 开发操作系统的系统编程语言其实很多；还离不开汇编语言。



MIT 用 Go 语言开发了 Biscuit OS
缺少对实际应用的优化，OSDI'2018



Stanford 用 RUST 语言开发了 tock OS
缺少对面向通用系统应用的支持，SOSP'2017

系统编程语言：Why Rust

- CS140e：Stanford 实验性课程，Rust OS for Raspi3
- RVirt：MIT RISC-V Hypervisor
- Writing an OS in Rust ——BlogOS：详尽的 Rust OS 教程
- 产业界：蚂蚁金服：Occlum – Facebook：Libra

系统编程语言：Rust 的主要特性

- 内存 + 线程安全
- 高级语言特性
- 成熟的工具链
- 友好的助教 + 社区（OS 示例代码 + 文档）生态
- 学习 Rust 的入门门槛比较高 (认清你自己，量力而行)

高级语言编译到机器指令：C/Rust 代码

```c
//C CODE
int sum_to(int n) {
  int acc = 0;
  for (int i = 0; i < n; i++) {
    acc += i;
  }
  return acc;
}
```

```rust
//Rust CODE
fn sum_to(n:i32)->i32 {
  let mut acc = 0;
  for i in 0..n {
    acc += i;
  }
  return acc
}
```

高级语言编译到机器指令：汇编代码

```
# sum_to(n)
# expects argument in a0
# returns result in a0
sum_to:
mv t0, a0           # t0 <- a0
li a0, 0            # a0 <- 0
loop:
add a0, a0, t0      # a0 <- a0 + t0
addi t0, t0, -1     # t0 <- t0 - 1
bnez t0, loop       # if t0 != 0: pc <- loop
ret
```

- 有限的抽象
  - 无类型的位置参数 – 没有局部变量 – 仅寄存器

高级语言编译到机器指令：RISC-V 寄存器

表: RISC-V 寄存器描述

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	——
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	——
x4	tp	Thread pointer	——
x5-7	t0-2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10-11	a0-1	Function arguments/return values	Caller
x12-17	a2-7	Function arguments	Caller
x18-27	s2-11	Saved registers	Callee
x28-31	t3-6	Temporaries	Caller

高级语言编译到机器指令：机器指令

- 机器甚至看不到汇编代码
- 查看机器指令的二进制编码
  - 每条指令：16 位或 32 位
  - 例如 ‘mv t0，a0‘编码为 0x82aa
  - 从 asm 不太完全一对一编码 (但是很接近)
- 另一个函数将如何调用 sum_to?

```
1 main:
2   li  a0, 10          # a0 <- 10
3   call sum_to
```

高级语言编译到机器指令：函数调用

- 函数调用的语义是什么?

```
call label :=
  ## ra <- address of next instruction
  ra <- pc + 4        ;
  ## jump to label
  pc <- label         ;
```

- 机器不理解标签
- 换为相对于 PC 的跳转或绝对跳转

高级语言编译到机器指令：函数返回

- 函数返回（return）的语义是什么??

```
ret :=  pc <- ra
看看汇编代码: demo1.S
(gdb) file user/_demo1
(gdb) break main
(gdb) continue
(gdb) layout split
(gdb) stepi
(gdb) info registers
(gdb) p $a0
(gdb) advance 18
(gdb) si
(gdb) p $a0
```

高级语言编译到机器指令：调用约定

- 如何传递参数?
  - a0，a1，...，a7，放在堆栈上
- 值如何返回?
  - a0，a1
- 谁保存寄存器?
  - 指定为已保存的呼叫者或被呼叫者
  - ra 可以是保存被呼叫者的寄存器吗
- 汇编代码应遵循此约定
- GCC 生成的 C 代码遵循此约定
- RUST 代码遵循此约定
- 这意味着各种代码之间都可以互操作

高级语言编译到机器指令： 理解 stack