# XZNU

2020.12.9

胡浦云

2018202133

# 目录

# 实验目的

实现一个“非常”简化的设备驱动程序。

# 实验环境

实验所用系统为 Ubuntu 20.04
QEMU 版本为 QEMU emulator version 5.1.0
系统为实体机

```
ceerrep@ceerrep-PHA14:~/pr/Xinu/compile$ screenfetch
                          ./+o+-       ceerrep@ceerrep-PHA14
                  yyyyy- -yyyyyy+      OS: Ubuntu 20.04 focal
               ://+//////-yyyyyyo      Kernel: x86_64 Linux 5.4.0-48-generic
           .++ .:/++++++/-.+sss/`       Uptime: 1d 22h 45m
         .:++o:  /++++++++/:--:/-       Packages: 2107
        o:+o+:++.`..```.-/oo+++++/      Shell: bash
       .:+o:+o/.          `+sssoo+/     Resolution: 1366x768
  .++/+:+oo+o:`             /sssooo.    DE: GNOME 3.36.4
 /+++//+:`oo+o               /::--:.    WM: Mutter
 \+/+o+++`o++o               ++////.    WM Theme: Adwaita
  .++.o+++oo+:`             /dddhhh.    GTK Theme: Yaru [GTK2/3]
       .+.o+oo:.           `oddhhhh+     Icon Theme: Yaru
        \+.++o+o``-````.:ohdhhhhh+      Font: Ubuntu 11
         `:o+++ `ohhhhhhhhyo++os:       Disk: 20G / 103G (20%)
           .o:`.syhhhhhhh/.oo++o`       CPU: Intel Core i7 M 640 @ 4x 2.8GHz
               /osyyyyyyo++ooo+++/      GPU: GeForce GT 325M
                   ````` +oo+++o\:      RAM: 4849MiB / 6437MiB
                          `oo++.
```

# 设计思路

1. 屏幕显示部分，若只是单纯显示字符和回滚，则第一次实验的 bootloader 就已实现，故考虑加入图形支持。
   - 在保护模式直接切换显示模式需要编写驱动，因而考虑切换至虚拟 8086 模式调用 BIOS 中断进行切换
   - 图形模式下显示缓冲区从 0xA0000 开始，文本模式下从 0xB8000 开始。出于测试目的，图形模式下输出直接写显存，文本模式下则是调用 BIOS 中断。
   - 这样换行，滚屏和退格会由 BIOS 处理，人工处理代码在 bootloader 中。
2. 键盘输入部分， 注册一中断处理程序，若缓冲区不满则将字符写入缓冲区，阻止调度的同时令信号量加一。

# 设计细节

### 虚拟 8086 模式

1. 根据 intel 文档，启用虚拟 8086 模式需要令 EFLAGS 的 VM 位置一，该位只能使用 iret 及任务门调用等方式修改，不能使用 popf 直接修改。
```

2. 进入 v8086 模式以后，CPU 只能寻址虚拟地址的低 1M，因而需要将实模式代码拷贝至低端内存，同时划分出实模式栈。
3. 参数可以直接写入 TSS 中，从寄存器传递。

代码

```
void v8086_call(void *func,
      uint16 ax,
      uint16 bx,
      uint16 cx,
      uint16 dx,
      uint16 si,
      uint16 di,
      uint16 ds,
      uint16 es)
{
  memset(&v8086_tss, 0, sizeof v8086_tss);
  v8086_tss.io[sizeof(v8086_tss.io) - 1] = 0xFF;
  v8086_tss.tss.link = 0;
  asm("movl %%cr3, %%eax\n\t"
    : "=a"(v8086_tss.tss.cr3)
    :
    :);

  // disable interrupt, enable v8086, set IOPL=3
  asm("pushfl\n\t"
    "pop %%eax\n\t"
    "andl $0xffffdff, %%eax\n\t"
    "orl  $0x23000, %%eax"
    : "=a"(v8086_tss.tss.eflags)
    :
    :);

  v8086_tss.tss.cs = 0;
  v8086_tss.tss.eip = (uintptr)((char *)func - code16_source_start + CODE16);

  v8086_tss.tss.ss = 0;
  v8086_tss.tss.esp = v8086_tss.tss.esp0 = (uintptr)CODE16STACKTOP;

  v8086_tss.tss.ds = ds;
  v8086_tss.tss.es = es;

  v8086_tss.tss.eax = ax;
  v8086_tss.tss.ebx = bx;
  v8086_tss.tss.ecx = cx;
  v8086_tss.tss.edx = dx;
  v8086_tss.tss.esi = si;
  v8086_tss.tss.edi = di;

  v8086_tss.tss.link = 0;
  v8086_tss.tss.ldtr = 0;
  v8086_tss.tss.iopb_offset = sizeof(tss_t);

  for (int i = 0; i < 32; i++)
    v8086_tss.tss.redirect[i] = 0xFF;

  asm("lcall $0x30, $0\n\t");
}
```

出于未知原因， qemu 在执行 v8086 代码时内部的 IOPL 是 2，小于 v8086 的 CPL 3，会触发 GP#。所以需要将 TSS 中 IO 权限位全置 0（首行 memset，第二行根据要求，将最后一字节置 1）。

同时，根据 intel 文档，v8086 模式下软中断的处理既可以转发给保护模式，也可以从实模式的 IVT 中读取调用。但后者需要 VME，qemu 没有实现。所以在调用 int 指令时，由于权限不足会触发 GP#，所以安装一 Gerneral Protection Handler 解释执行此类指令。

```
void GeneralProtectionHandler_(int error_code)
{
  uint32 link = gp_tss.link;
  tss_t *prev = tss_array[gp_tss.link / 8];
  uint32 cs = prev->cs;
  uint32 eip = prev->eip;
  if (prev == &v8086_tss.tss && prev->eflags & 0x20000) // v8086
  {
    uint8 *addr = (uint8 *)(cs * 16 + eip);
    uint16 *stack = (uint16 *)(prev->ss * 16 + prev->esp);

    switch (addr[0])
    {
    case 0xF4: // hlt
      // Exit v8086 mode
      gdt[gp_tss.link / 8].sd_access &= ~0x2;

      gp_tss.link = prev->link;
      prev->link = 0;

      return;

    case 0xCD: // int
    {
      uint32 intno = addr[1];
      uint32 int_cs = ((uint16 *)0)[2 * intno + 1];
      uint32 int_ip = ((uint16 *)0)[2 * intno];

      // Save regs
      stack[-1] = prev->eflags;
      stack[-2] = prev->cs;
      stack[-3] = prev->eip + 2;

      prev->esp -= 6;

      prev->cs = int_cs;
      prev->eip = int_ip;

      return;
    }

    default:
      break;
    }
  }
  kprintf("GP# at 0x%x [%x:%x]\nCode: %x\n", link, cs, eip, error_code);
  panic("System halted");
}
```

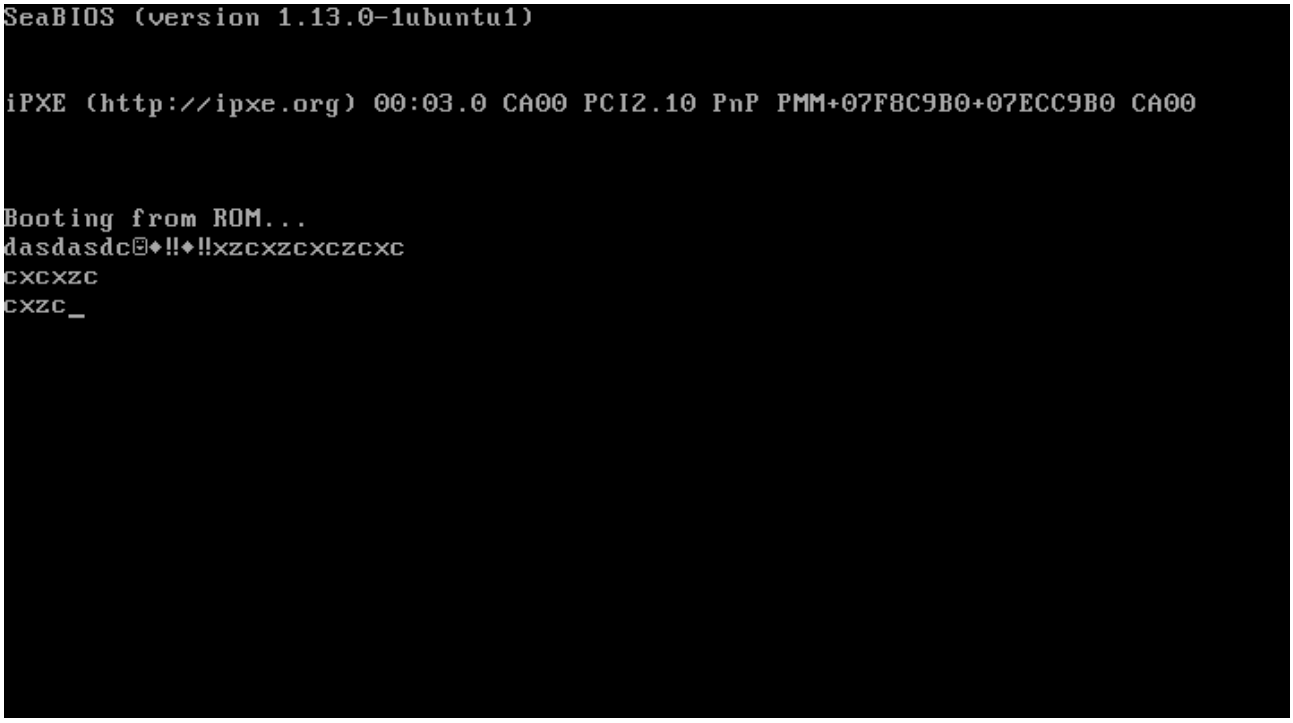
遇到 hlt 指令时会退出 v8086 模式，具体实现为令 gp_tss 直接返回 v8086_tss 的调用方，同时在 GDT 中释放 v8086_tss，避免 CPU 认为该任务重入。
同时，由于中断时 CPU 会保存 EFLAGS 并关中断，中断处理程序只需 pushad 即可。

之后使用其调用 0x10 号 BIOS 中断设置显示模式即可。

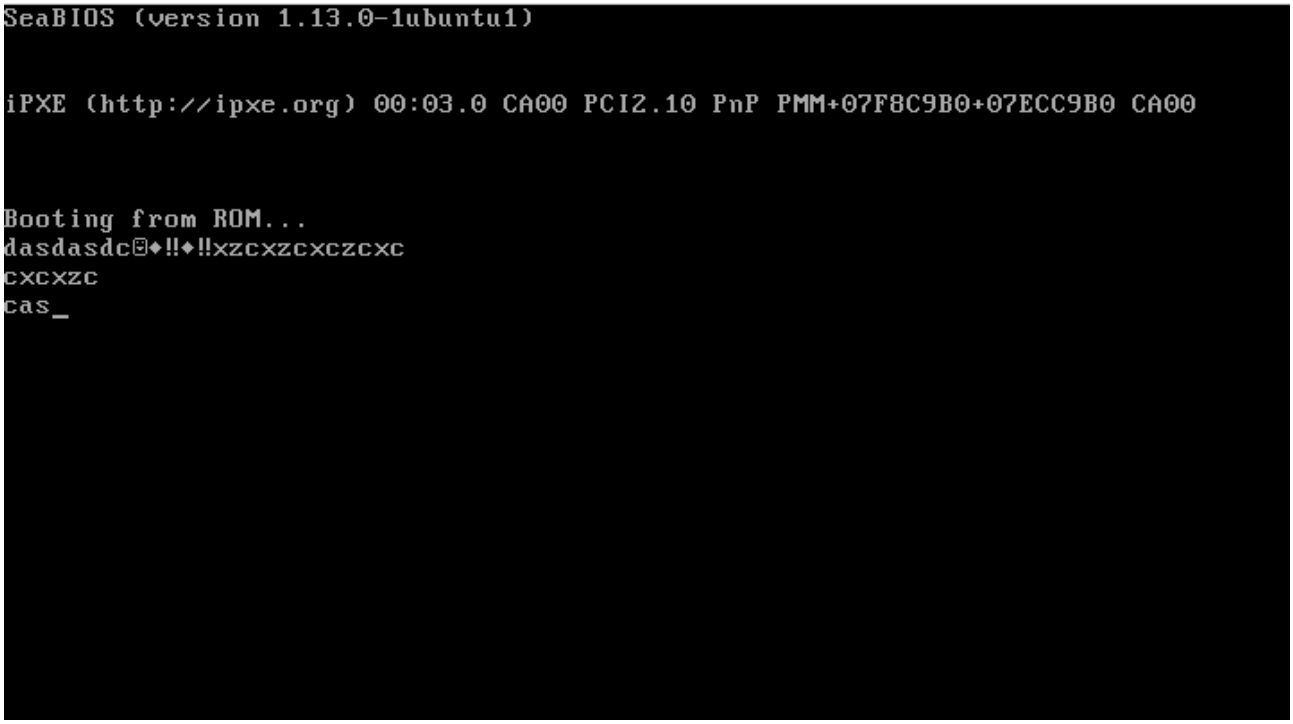
### 键盘中断

1. 中断触发时，如果可能，将字符放入缓冲区
2. 根据设置和 VGA 模式决定是否回显（图形模式下不能回显
3. 用户程序读时，如果可能，返回缓冲区中字符；否则根据设置决定是否阻塞

# 实验效果




回显和换行




退格

```
    30688 free pages of
    32736 total.
   135636 bytes of Xinu code.
          [0x00100000 to 0x001211D3]
   144204 bytes of data.
          [0x001248A0 to 0x00147BEB]



----------------------------------------
  __   __  _____   _   _   _   _
  \ \ / / |__ __|  | \ | |  | |  | |
   \ V /     | |    |  \| |  | |  | |
   / /\ \    _| |_   | \   |  | |  | |
  / /  \ \  |     |  | |\ |  \  --  /
  --    --  -----    -  -    ----
----------------------------------------


Welcome to Xinu!


xsh $ image
```

输入图形显示命令

```
   135636 bytes of Xinu code.
          [0x00100000 to 0x001211D3]
   144204 bytes of data.
          [0x001248A0 to 0x00147BEB]



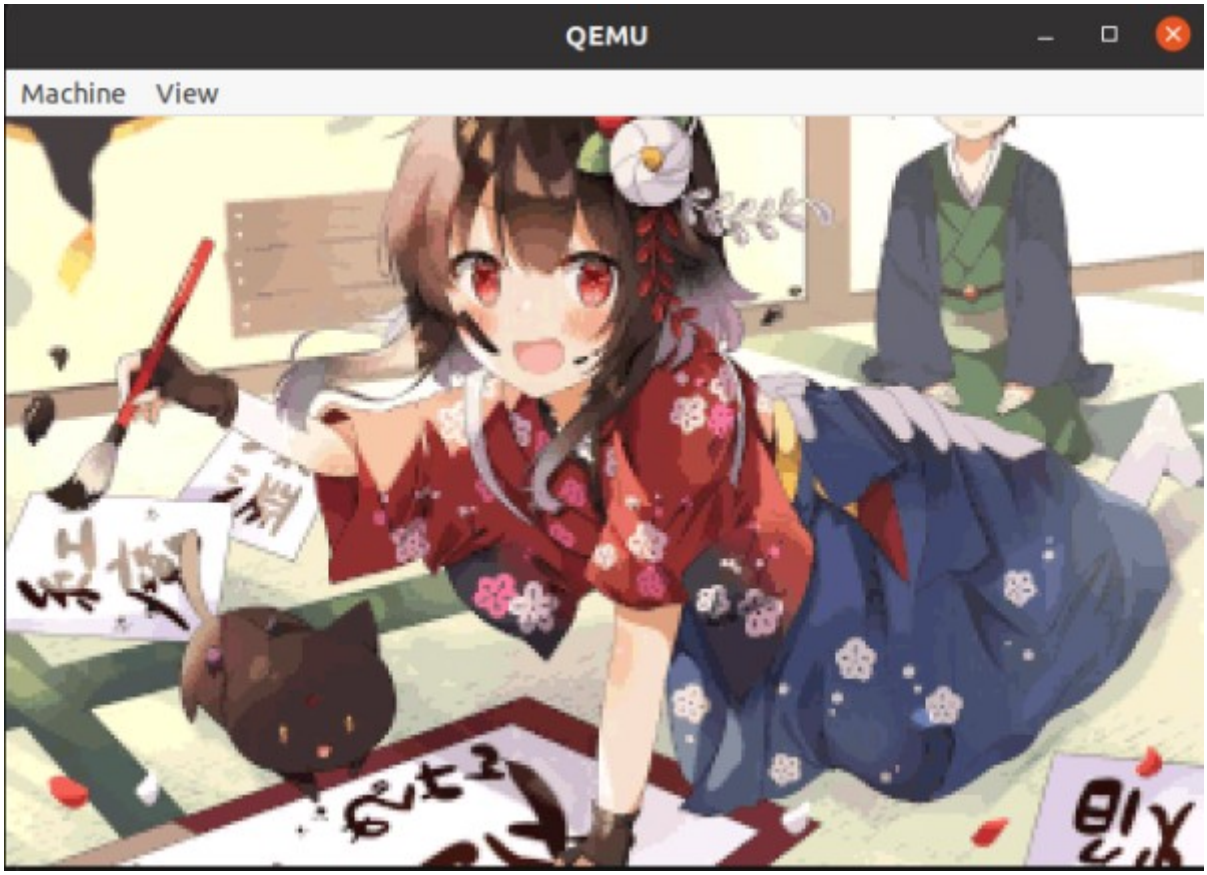
----------------------------------------
  __   __  _____   _   _   _   _
  \ \ / / |__ __|  | \ | |  | |  | |
   \ V /     | |    |  \| |  | |  | |
   / /\ \    _| |_   | \   |  | |  | |
  / /  \ \  |     |  | |\ |  \  --  /
  --    --  -----    -  -    ----
----------------------------------------


Welcome to Xinu!


xsh $ image
```

显示模式已经切换（VGA 变窄使 Qemu 窗口变窄）

先不阻塞读空缓冲区，之后阻塞并等待用户输入



图片显示

# 参考文献

[1] Intel. 2020. *Intel® 64 And IA-32 Architectures Software Developer Manuals*. [online] Available at: <https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html> [Accessed 9 December 2020].