深圳大学
Shenzhen University

操作系统

xv6实验题目解答-2

计算机与软件学院

大作业Part I-3（代码理解问题(bootasm.S)）

- 为什么主引导记录（硬盘第一个扇区）要存放在0x7C00开始的内存地址？（提示：这是历史遗留问题）
- 0x7C00=0111,1100,0000,0000=$(2^5-1)*2^{10}$=32KB-1KB
- 这篇文档里面给出了详细的解释：
  - https://www.glamenv-septzen.net/en/view/6
  - 1981年推出的IBM PC 5150上运行的DOS1.0需要最少32KB的内存。
  - 8086/8088 used 0x0 - 0x3FF for interrupts vector, and BIOS data area was after it.（所以主引导记录不方便放在内存开头的位置）
  - 主引导扇区512字节，它所使用的栈和数据区需要额外的512字节。
  - 所以，32KB的从0x7C00开始的1K字节用于读入主引导记录。

大作业Part I-3（代码理解问题(bootasm.S)）

- L21，“# Physical address line A20 is tied to zero…”这是著名的Gate-A20，请介绍一下为什么要设定Gate-A20。
  - Nobody wants A20, but it continues to haunt us.
  - The 8088 in the original PC had only 20 address lines, good for 1 MB.
  - 越界的地址会自动抹除。比如：
  - 0x10ffef=1'0000'1111'1111'1110'1111
  - 在8088中会变成‘0000’1111‘1111’1110‘1111，首位的1比特消失了。

大作业Part I-3（代码理解问题(bootasm.S)）

- 当286推出时，它有24条地址线，也即24位的地址。
- 286CPU应该对8088 100%兼容（通过real mode）但是由于CPU的BUG，它竟然没有在real mode里面对地址进行高位截断！
- 这导致很多依赖于高位截断的原本运行于8088的程序无法在286CPU里面运行。
- IBM决定在主板总线加入一个开关，用于开启/禁用0x100000地址位。这称为Gate-A20。

大作业Part I-3（代码理解问题(bootasm.S)）

- L21-L38，这是一段让人一头雾水的代码，请查找资料，解释一下这段代码为何和enable A20有关。（参考https://www.win.tue.nl/~aeb/linux/kbd/A20.html）
  - Since the 8042 keyboard controller(PS/2) happened to have a spare pin, that was used to control the AND gate that disables this address bit. The signal is called A20, and if it is zero, bit 20 of all addresses is cleared.

大作业Part I-3（代码理解问题(bootasm.S)）

- Why do we have to worry about this nonsense?
  - By default the A20 address line is disabled at boot time, so the operating system has to find out how to enable it.
  - (看bootasm.S中的代码)
  - 0x64 (*Command Port*)是PS/2控制器的命令端口。
  - 0x60 (*Data Port*)是PS/2控制器的数据端口。

大作业Part I-3（代码理解问题(bootasm.S)）

- inb $0x64,%al # Get status
- 控制端口把状态送到寄存器al中。1号比特代表输入缓冲区的状态（0为空，1为满）；
- 当缓冲区为空时，送0xd1到0x64端口，再送0xdf到0x60端口enable A20 (送0xdd到0x60端口disaable A20)。

Lab3实验题目解答

- 试解释一下yield函数、scheduler函数和sched函数的用途。（三个函数都在proc.c，看注释）
- yield: Give up the CPU for one scheduling round.
- sched: 保存当前进程上下文，切换到scheduler。
- scheduler: 调度器本身，选择一个进程，切换到那个进程的上下文，让它获得CPU，但那个进程让出CPU时，又会回到scheduler函数。

Lab3实验题目解答

- 结合书本，确定xv6使用的是哪种调度算法。给出你的理由（通过分析代码证明你的观点）。
- trap.c L103-L106，xv6采用的是时间片轮转法。

Lab3实验题目解答

- (proc.c L268) 有一个疑问，似乎每次xv6都是从进程表开头开始查找Runnable的进程。如果刚从CPU切换下来的进程恰好是进程表的第一个PCB，会不会调度器永远都选择它进行调度？
- 非常简单，scheduler进行上下文切换后，把CPU让给用户进程，再次上下文切换回到CPU的时候，它的指令指针还指向内层循环里面呢，scheduler会一直遍历整个进程表，然后在外层循环再从头开始。

Lab3实验题目解答

- 在xv6界面上按Ctrl+P，你会看到终端上会显示当前系统中的进程。
- 仿照相关代码，实现以下功能：
  - 在xv6界面上按Ctrl+R，打印出当前系统中sleeping的进程，并确定它们对应的等待队列是什么（chan/waiting channel）。

Lab3实验题目解答

- 按Ctrl+P执行的函数是procdump（proc.c）。
- 调用procdump的代码位于console.c（L196）。
- 如何确定SLEEPING进程对应的等待队列？
  - 设置进程状态为SLEEPING的唯一代码在proc.c L364，位于sleep函数里面。
  - wakeupl函数（proc.c），唤醒所有挂在chan上的进程。
  - 如果你查看wakeup被调用的地方，你就能看到，chan只不过是内存地址而已。
  - 因此把chan所对应的内存地址打印出来即可。