Lecture 3. pc hardware, execution cycle, boot sequence, grub boot loader, Kernel Compiling

1. pc hardware

1) computer



Where are the files located? What’s in memory?

2) Intel x86 cpu



registers:

 ax, bx, cx, dx, cs, ds, ss, es, sp, bp, ip, si, di, flag register

 (for 32-bit: eax, ebx, ecx, edx, esp, ebp, eip, esi, edi)

 (for 64-bit: rax, rbx, rcx, rdx, rsp, rbp, rip, rsi, rdi)

ax, bx, cx, dx: General purpose. Can contain any data

cs, ds, ss, es : Segement registers. cs for code segment, ds for data segment, ss for stack segment, es for extra segment. A program loaded in memory has 3 segments: code, data, and stack. cs points to the base address of the code segment; ds points to the base of data segment; ss points to the base of stack segment.

sp : points to the current stack top

bp: points to the base location of the frame for the current function in the stack

ip : points to the current instruction

flag register: contains the system status

si,di : source index, destination index register

3. Execution cycle

CPU runs the execution cycle forever.

 Step 1: Fetch the next instruction pointed to by cs:eip.

 Step 2: Update eip.

 Step 3: Execute the fetched instruction.

 Step 4: Check if there was an interrupt.

 If interrupt (INT x)

 save flag register, cs, eip ( and ss, esp also in protected mode)

 replace cs, eip with the ISR location in IDT[x]

 go to Step 1

 Else

 go to Step 1

4. boot sequence

Refer to <http://duartes.org/gustavo/blog/post/how-computers-boot-up> or Appendix A of “Understanding Linux Kernel” book.

power on

 cs=0xF000, ip=0xFFF0

 with hidden base address, the first physical address is 0xFFFFFFF0.

bios starts:

 system test/initialization

 load/execute the 1st sector of the boot disk into memory at address 7c00h

 (boot disk is searched in pre-defined order, e.g. a, b, c, etc)

boot loader starts:

 find/load/execute os

os starts: setup()->startup\_32()->start\_kernel()

 set interrupt descriptor table

 initialize data structures for process/file/memory/io device

 generate system processes

 ..............

each system process runs (one of them is login process)

...............

login starts:

 prints "login:" and waits for user login

5. kernel compiling

Go to the Linux top directory and do

 # make bzImage

The above will compile the linux kernel and produce the compressed executable in arch/x86/boot/bzImage.

6. Replace boot image

When the power is on, the cpu loads boot loader program from the first sector of the booting disk. In our case it is Grub (GNU GRand Unified Bootloader) bootloader. The Grub bootloader will try to find the operating system (boot image) as specified in /boot/grub/grub.conf.

 ............

 **title=Gentoo Linux**

 root (hd0,0)

 kernel /kernel-genkernel-x86-2.6.23-gentoo-r8 root= ............

 initrd ..............

 **title=My Linux**

 root (hd0,0)

 kernel /boot/bzImage root=..........

 initrd .......

The above means there are two versions of operating system: Gentoo Linux and My Linux. The kernel image file of "Gentoo Linux" is in /kernel-genkernel-x86-2.6.23-gentoo-r8 and the kernel image file of "My Linux" is in /boot/bzImage. If we select "My Linux" during the booting process, Grub will load /boot/bzImage as the operating system. We replace this one with our new Linux.

 # cp arch/x86/boot/bzImage /boot/bzImage

Reboot the system with

 # reboot

And select "My Linux". Now the new Linux should be running.

7. Homework

0) Boot sequence.

0-1) When you boot the Linux system, the first program that runs is BIOS. Where is this program (the memory location)?

0-2) BIOS loads and runs the boot loader program (GRUB in Linux). Where is this GRUB program?

0-3) GRUB loads and runs the Linux executable file. Where is Linux executable file? How GRUB knows the location of Linux executable file?

1) Simple modification of the kernel.

Add

 printk("hello from me\n");

after

 printk(linux\_banner);

in start\_kernel(). Go to the Linux top directory and compile the kernel and replace the boot image. Reboot with this new kernel, and run dmesg to see if the kernel is printing "hello from me".

 # cd linux-2.6.25.10

 # cd init

 # vi main.c

 ....... modify start\_kernel()

 # cd .. – go back to the Linux top directory

 # make bzImage -- recompile the kernel

 # cp arch/x86/boot/bzImage /boot/bzImage

 --copy the new kernel image to boot location

 # reboot -- reboot the system with this new kernel and select "My Linux"

 ............

 (select “My Linux”)

 # dmesg > x

 # vi x -- now check if we can see our new message

2) start\_kernel() calls trap\_init(), and there are many trap\_init() functions defined in the kernel code. Make an intelligent guess about which trap\_init() would be called and insert some printk() in the beginning of this trap\_init() to see if it is really called by the kernel. Use grep in the top directory of the linux source tree to find out the locations of trap\_init():

 # grep -nr “trap\_init” \* | more

3) Find also the exact locations of init\_IRQ() and insert some printk() in the beginning of init\_IRQ() to confirm (actually you insert it in native\_init\_IRQ). Do the same thing for init\_timers() and time\_init().

4) Modify /boot/grub/grub.conf so that GRUB displays another Linux selection, My Linux2. Set the location of the kernel for this linux as /boot/bzImage2. Prepare two versions of My Linux such that when you select "My Linux" the kernel will display "hello from My Linux", and when you select "My Linux2", it displays "hello from My Linux2".

 # cd /boot/grub

 # vi grub.conf

 .......move the cursor to "title=My Linux" and copy 4 lines there (with 4yy)

 ........move the cursor to the last line and paste the 4 lines (with p)

 ........change this new kernel as below: Linux => Linux2, bzImage=>bzImage2, the rest same

 ……………….

 title=My Linux2

 root (hd0,0)

 kernel /boot/bzImage2 root=/dev/ram0 init=/linuxrc ramdisk=8192 real\_root=/dev/sda3 doscsi

 initrd /initramfs-genkernel-x86-2.6.23-gentoo-r8

 Go to start\_kernel() and add "hello from My Linux" and recompile the kernel

 and save it in /boot/bzImage. Now go back to start\_kernel() and change it to

 "hello from My Linux2", recompile the kernel and save it in /boot/bzImage2.

 Check if you have different boot message with different Linux.

5) Where is CPU at the end of the boot sequence when it prints "login" and waits for the user login? Explain your reasoning.