Linux操作系统分析

陈香兰（xlanchen@ustc.edu.cn）

计算机应用教研室@计算机学院
嵌入式系统实验室@苏州研究院
中国科学技术大学

Spring 2011

Outline

编译Linux在QEMU模拟器上运行

制作带grub启动的磁盘映像

利用qemu+gdb来调试linux

编译Linux在QEMU模拟器上运行

- qemu+linux-2.6.26
  1. 准备模拟器
  2. 编译Linux内核
  3. 准备根文件系统

1、准备模拟器qemu

1. sudo apt-get install qemu
2. 有的源中不带qemu，则需要自己编译
   - 获得qemu源代码qemu-0.12.3.tar.gz，并解压缩
   - 配置、编译，并安装到指定的目录下
     - ./configure –prefix=PATH_TO_YOUR_QEMU_INSTALL_DIR –target-list=i386-softmmu
     - make
     - su -c “make install”
   - 编译安装完成之后，如何使用qemu？
     - 可以通过指定路径的方式使用qemu，此时qemu在安装目录下的bin目录中
     - 可以将安装目录/bin加入到PATH环境变量中，此时可以在任何目录下直接使用qemu

编译Linux内核 I

- 获得linux-2.6.26.tar.gz，解压缩得到目录linux-2.6.26，不妨称之为Linux源代码根目录（以下简称源码根目录）
  - tar -zvxf linux-2.6.26.tar.gz
- 进入源代码根目录
- 可以使用make help得到一些编译内核的帮助信息
- 我们采用i386的缺省编译
  - make i386_defconfig
  - make
- 可以观察一下编译过程中的输出信息，特别是编译最后阶段的输出信息。

准备好一个简单的根目录和应用程序 I

- 准备一个应用程序
  例如一个helloworld输出循环，使用静态链接的方法编译成一个可执行文件，命名为init
  - gcc -static -o init xxx.c
- 建立目标根目录映像
  - dd if=/dev/zero of=myinitrd4M.img bs=4096 count=1024
  - mke2fs myinitrd4M.img
  - mkdir rootfs
  - sudo mount -o loop myinitrd4M.img rootfs
- 将init拷贝到目标根目录下
  （linux启动后期会在根目录中寻找一个应用程序来运行，在根目录下提供init是一种可选方案）

准备好一个简单的根目录和应用程序 II

- cp init rootfs/
- 准备dev目录
  - sudo mkdir rootfs/dev
  - linux启动过程中会启用console设备
    - sudo mknod rootfs/dev/console c 5 1
  - 另外需要提供一个linux根设备，我们使用ram
    - sudo mknod rootfs/dev/ram b 1 0
- sudo umount rootfs
- 至此，一个包含简单应用程序的根目录映像myinitrd4M.img就准备好了

运行

- qemu -kernel PATH_TO_linux-2.6.26/arch/x86/boot/bzImage -initrd myinitrd4M.img -append "root=/dev/ram init=/init"
- 可以看到系统能够启动，并且在启动后看到init的输出结果

也可以利用busybox建立根文件系统 I

- 下载busybox的源代码，解压缩
- make help可以得到一些编译busybox的帮助信息
- 我们在缺省编译的基础上，稍作修改
  - make defconfig
  - make menuconfig修改如下配置：
    - enable：busybox settings –> build options –> build busybox as a static binary（no share libs）
    - enable：busybox settings –> installation options –> dont use /usr
  - make
- 准备根目录映像，并安装busybox到根目录映像中
  - 我们使用前面生成的myinitrd4M.img
  - sudo mount -o loop myinitrd4M.img rootfs

也可以利用busybox建立根文件系统 II

- 在busybox目录下
  sudo make CONFIG_PREFIX=PATH_TO_rootfs/ install
- sudo umount rootfs

- 运行
  - qemu -kernel PATH_TO_linux-2.6.26/arch/x86/boot/bzImage -initrd myinitrd4M.img -append “root=/dev/ram init=/bin/ash”
  - 此时可以进入busybox提供的shell环境

在busybox的shell中运行helloworld

- 提供一个helloworld程序，静态编译成hello
  - gcc -static -o hello xxx.c
- sudo mount -o loop myinitrd4M.img rootfs
- sudo cp hello rootfs
- sudo umount rootfs
- 运行
  - qemu -kernel PATH_TO_linux-2.6.26/arch/x86/boot/bzImage -initrd myinitrd4M.img -append “root=/dev/ram init=/bin/ash”
  - 进入shell后，运行
    - ./hello

制作根目录的另一种方法 I

- 前面是手工建立映像，
  还可以利用linux源代码直接编译成映像
  1. 准备将被作成映像的根目录内容
     - mkdir rootfs
     - mkdir rootfs/dev
     - cp init rootfs/
     - sudo mknod rootfs/dev/console c 5 1
     - sudo mknod rootfs/dev/ram b 1 0
  2. 配置Linux-2.6.26，使之能生成根目录映像
     - **General setup —>**
       [*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
       (../rootfs) Initramfs source file(s)

制作根目录的另一种方法 II

3. 编译，可以看到在usr目录下生成了一个initramfs_data.cpio.gz文件
4. 运行
   - qemu -kernel arch/x86/boot/bzImage -initrd usr/initramfs_data.cpio.gz -append “root=/dev/ram init=/init”

制作带grub启动的磁盘映像

1. 获得grub
2. 制作grub启动软盘
3. 准备磁盘映像
4. 将磁盘映像升级为带grub启动的

获得grub，并制作grub启动软盘

- 下载grub-0.97-i386-pc.tar.gz，解压缩
- 查看解压缩得到的目录
- 建立软盘映像
  - dd if=/dev/zero of=a.img bs=512 count=2880
- 添加grub启动功能
  - sudo losetup /dev/loop3 a.img
  - sudo dd if=./grub-0.97-i386-pc/boot/grub/stage1 of=/dev/loop3 bs=512 count=1
  - sudo dd if=./grub-0.97-i386-pc/boot/grub/stage2 of=/dev/loop3 bs=512 seek=1
  - sudo losetup -d /dev/loop3
- 测试是否能进入grub界面
  - qemu -fda a.img

准备磁盘映像，并制作带grub启动的磁盘映像 I

- 准备磁盘映像
  - dd if=/dev/zero of=32M.img bs=4096 count=8192
  - sudo losetup /dev/loop3 32M.img
  - 在磁盘映像上建立一个活动分区
    - sudo fdisk /dev/loop3
  - sudo losetup -d /dev/loop3
  - 将活动分区格式化成ext2fs，并mount到rootfs目录上
    - sudo losetup -o 32256 /dev/loop3 32M.img
      其中，32256是分区的起始位置，为63×512
      其中，63是通过file 32M.img得到的startsector信息
    - sudo mke2fs /dev/loop3
    - sudo mount /dev/loop3 rootfs
  - 将前面制作的bzImage和myinitrd4M.img拷贝到rootfs中

准备磁盘映像，并制作带grub启动的磁盘映像 II

- 增加grub功能
  - 准备相关目录，并拷贝一些必要的文件
    - sudo mkdir rootfs/boot
    - sudo mkdir rootfs/boot/grub
    - sudo cp ./grub-0.97-i386-pc/boot/grub/* rootfs/boot/grub
  - 在rootfs/boot/grub中编写menu.lst，具有如下内容

```
default 0
timeout 30
title linux on 32M.img
root (hd0,0)
kernel (hd0,0)/bzImage root=/dev/ram init=/bin/ash
initrd (hd0,0)/myinitrd4M.img
```

- 利用grub启动软盘，在硬盘映像上添加grub功能
  - qemu -boot a -fda a.img -hda 32M.img

准备磁盘映像，并制作带grub启动的磁盘映像 III

- 进入grub界面后
  root (hd0,0)
  setup (hd0)

- 测试从磁盘启动进入grub界面
  - qemu -hda 32M.img

在qemu中启动gdb server

- 运行
  - qemu -kernel arch/x86/boot/bzImage -s -S
  - 可以看到在新打开的qemu虚拟机上，整个是一个黑屏，此时qemu在等待gdb的链接
  - 关于-s和-S选项的说明
    - -S freeze CPU at startup (use 'c' to start execution)
    - -s shorthand for -gdb tcp::1234
      若不想使用1234端口，则可以使用-gdb tcp:xxxx来取代-s选项

建立gdb与gdbserver之间的链接

- 在另外一个终端运行gdb，然后在gdb界面中运行如下命令
  - target remote：1234
    则可以建立gdb和gdbserver之间的连接
  - c
    让qemu上的Linux继续运行
- 假如在前面使用-gdb tcp::xxxx，则这里的1234也要修改为对应的xxxx
- 问题：此时没有加载符号表，无法根据符号设置断点

加载vmlinux中的符号表，设置断点

- 在gdb界面中targe remote之前加载符号表
  - file vmlinux
- 在gdb界面中设置断点
  - break start_kernel
    断点的设置可以在target remote之前，也可以在之后
- 在设置好start_kernel处断点并且target remote之后可以继续运行，则在运行到start_kernel的时候会停下来，等待gdb调试命令的输入
- 此后可以继续设置新的断点，...
- 问题：此时尽管有符号表，但是无法显示源代码

重新配置Linux，使之携带调试信息

- 在原来配置的基础上，重新配置Linux，使之携带调试信息
  - **kernel hacking—>**
    [*] compile the kernel with debug info
- 重新编译（时间较长）
  - make
- 此时，若按照前面相同的方法来运行，则在start_kernel停下来后，可以使用list来显示断点处相关的源代码

Thanks !

The end.