

搭建內核開發調試環境

APR 5TH, 2012 | [COMMENTS](#)

閒來無事，總結一下內核開發調試環境的搭建過程，希望能對和我一樣的內核新手們有所幫助。

方案

我的測試系統在QEMU中運行，Host和Guest的架構都是x86_64，用Busybox生成的initrd做為根文件系統，KGDB做為調試器。

生成內核

內核中需要打開的選項是CONFIG_EXPERIMENTAL，CONFIG_DEBUG_INFO，CONFIG_KGDB和CONFIG_KGDB_SERIAL_CONSOLE，同時需要關閉CONFIG_DEBUG_RODATA選項。然後make bzImage編譯生成內核。具體選項的意義可以去翻內核文檔，這裡就不囉嗦了。

生成根文件系統

打開Busybox的CONFIG_STATIC和CONFIG_INSTALL_NO_USR選項，執行make和make install編譯並生成，然後參照下面的步驟創建initrd根文件系統：

```
mkdir temp && cd temp
```

```
#創建系統目錄
```

```
mkdir -p dev etc/init.d mnt proc root sys tmp
```

```
chmod a+rwx tmp
```

```
cp -rf ../busybox/_install/* ./
```

```
#掛載系統目錄
```

```
cat << EOF > etc/fstab
```

```
proc /proc proc defaults 0 0
```

```
sysfs /sys sysfs defaults 0 0
```

```
tmpfs /tmp tmpfs defaults 0 0
```

```
EOF
```

```
cat << EOF > etc/inittab
```

```
::sysinit:/etc/init.d/rcS
```

```
::respawn:-/bin/sh
```

```
tty2::askfirst:-/bin/sh
```

```
::ctrlaltdel:/bin/umount -a -r
```

```
EOF
```

```
cat << EOF > etc/init.d/rcS
```

```
#!/bin/sh
```

```
/bin/mount -a
```

```
#用mdev生成設備文件
```

```
/sbin/mdev -s
```

```
EOF
```

```
chmod 755 etc/init.d/rcS
```

```
find ./ | cpio -o -H newc | gzip > ../rootfs.img
```

啟動QEMU

```
qemu-system-x86_64 -kernel kernel.img -append \  
"root=/dev/ram rdinit=/sbin/init" -initrd rootfs.img
```

或

```
qemu-system-x86_64 -kernel kernel.img -append \  
"root=/dev/ram rdinit=/sbin/init kgdboc=ttyS0,115200 kgdbwait" \  
-initrd rootfs.img -serial tcp::1234,server
```

第二個命令開啟了KGDB，將Guest系統的串口映射到了Host系統的1234端口，並在啟動過程中等待gdb的連接。

啟動gdb

內核開啟KGDB的情況下，執行gdb vmlinux，其中vmlinux是未壓縮的內核。然後target remote localhost:1234連接kgdb。

接下來就和普通的gdb沒什麼大的區別了，比如在sched_clock函數處設置斷點break sched_clock, continue繼續運行，到達斷點後打印jiffies_64變量print jiffies_64等等。

另外，運行過程中可以在測試系統裡執行echo g > /proc/sysrq-trigger讓gdb重新得到控制權。

For 懶人

順手在github上建了個項目, 可以自動搭建整個內核開發調試環境, 詳見README.

<http://github.com/adam8157/kernel-studio>

```
git clone git://github.com/adam8157/kernel-studio.git
```

```
gcc -static test.c -o test
```

--

```
// /etc/init.d/rcS
#!/bin/sh
MAC=08:90:90:59:62:21
IP=192.168.100.2
Mask=255.255.255.0
Gateway=192.168.100.1
```

```
/sbin/ifconfig lo 127.0.0.1
ifconfig eth0 down
ifconfig eth0 hw ether $MAC
ifconfig eth0 $IP netmask $Mask up
route add default gw $Gateway
```

```
/bin/mount -a
/bin/mount -t sysfs sysfs /sys
/bin/mount -t tmpfs tmpfs /dev
/sbin/mdev -s
```

```
mount -o remount,rw,noatime -n /dev/root /
```

```
# 產生 rootfs
find . / | cpio -o -H newc | gzip > ../rootfs.img
```

1.

```
sudo ./nettap.sh // 必須加上 sudo
# nettap.sh
tunctl -u jason-yao -t tap0
ifconfig tap0 192.168.100.1 up
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -I FORWARD 1 -i tap0 -j ACCEPT
iptables -I FORWARD 1 -o tap0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

如果成功下ifconfig 會出現下面 info

```
tap0    Link encap:Ethernet  HWaddr 0a:18:ee:c4:f0:d0
        inet addr:192.168.100.1  Bcast:192.168.100.255  Mask:255.255.255.0
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:500
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

2.

```
# 啟動 // 必須加上 sudo
sudo qemu-system-x86_64 -kernel bzImage -append "root=/dev/ram rdinit=/sbin/init"
-initrd rootfs.img -k en-us -net nic -net tap,ifname=tap0,script=no
```

PS: 必須執行 sudo ./nettap.sh , 在啟動qemu 順序要注意 也都必需用 sudo