lab5_debugfs 实验说明

本实验的目的:写一个内核模块,利用 debugfs 的 API 接口函数来创建新的节点,然后在节点里和用户空间进行数据交互。

基本实验步骤

1. 进入 rlk_lab/rlk_basic/chapter_10/lab1 目录。

export ARCH=arm

export CROSS_COMPILE=arm-linux-gnueabi-

make BASEINCLUDE=/home/figo/work/runninglinuxkernel/runninglinuxkernel_4.0

这里 BASEINCLUDE 指定你当前 runninglinuxkernel 4.0 的目录路径。

然后把*.ko 拷贝到 runninglinuxkernel_4.0/kmodules 目录下面。

运行如下脚本启动 Qemu。

#cd runninglinuxkernel 4.0

sh run.sh arm32 #启动虚拟机

在 Qemu 虚拟机:

#cd /mnt

insmod debugfs-test.ko

/mnt # insmod debugfs-test.ko I created benshushu on debugfs

创建的目录会在哪里呢? debugfs 一般会 mount 到/sys/kernel/debug/这个目录。 如果大家发现这个目录不存在,或者里面没有东西,说明 debugfs 没有挂载,或者输入"df" 命令来查看。那么可以手工挂载。

mount -t debugfs none /sys/kernel/debug/

进入到/sys/kernel/debug/目录后,你会发现有一个 benshushu 的目录。

```
/sys/kernel/debug # ls
bdi
                     kprobes
                                          suspend stats
benshushu
                     memblock
                                          tracing
clk
                     mmc0
                                          ubi
                                          ubifs
extfrag
                     pm qos
fault around bytes
                     regmap
                                          usb
gpio
                     regulator
                                          virtio-ports
                                          wakeup_sources
                     sleep time
hid
```

进入 benshushu 目录之后,你发发现有一个 my_debug 的节点。

```
/sys/kernel/debug/benshushu # ls
my_debug
/sys/kernel/debug/benshushu # ls -l
total 0
-r--r-- 1 0 0 0 Feb 3 13:56 my_debug
/sys/kernel/debug/benshushu #
```

接下来就可以对这个节点进行读写操作了。

```
/sys/kernel/debug/benshushu # cat my_debug
100
/sys/kernel/debug/benshushu # echo 200 > my_debug
param has been set to 200
/sys/kernel/debug/benshushu # cat my_debug
200
/sys/kernel/debug/benshushu #
```

进阶思考

debugfs 是内核专门为调试为设置的一个虚拟文件系统,在现有的内核代码中,有很多调试功能都集成到 debugfs 中,比如前面讲到的 动态打印功能。还有很多设备驱动子系统的调试功能等等。

有很多功能需要在 config 文件中打开 CONFIG_DEBUG_FS=y 选项。

```
Compile-time checks and compiler options

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing </>
includes, <M> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
</>
for Search. Legend: [*] built-in [] excluded <M> module <> module capable

[*] Compile the kernel with debug info
[] Reduce debugging information
[] Produce split debuginfo in .dwo files
[] Generate dwarf4 debuginfo
[] Provide GDB scripts for kernel debugging
[*] Enable _deprecated logic
[*] Enable _must_check logic
(1024) Warn for stack frames larger than (needs gcc 4.4)
[] Strip assembler-generated symbols during link
[] Generate readable assembler code
[] Enable unused/obsolete exported symbols
[] Track page owner

**Debug Filesystem
[] Run 'make headers_check' when building vmlinux
[] Enable full Section mismatch analysis
[] Force weak per-cpu definitions
```

以 regmap 这个子系统为例,打开 drivers/base/regmap/Makefile 文件:

```
1  obj-$(CONFIG_REGMAP) += regmap.o regcache.o
2  obj-$(CONFIG_REGMAP) += regcache-rbtree.o regcache-lzo.o regcache-flat.o
3  obj-$(CONFIG_DEBUG_FS) += regmap-debugfs.o
4  obj-$(CONFIG_REGMAP_AC97) += regmap-ac97.o
5  obj-$(CONFIG_REGMAP_IZC) += regmap-i2c.o
6  obj-$(CONFIG_REGMAP_SPI) += regmap-spi.o
7  obj-$(CONFIG_REGMAP_SPMI) += regmap-spmi.o
8  obj-$(CONFIG_REGMAP_MMIO) += regmap-mmio.o
9  obj-$(CONFIG_REGMAP_IRQ) += regmap-irq.o
```

我们发现只要系统中定义了 CONFIG_DEBUG_FS, 就会编译 regmap-debugfs.c 文件。系统运行之后,你会发现有一个 regmap 的目录。

```
/sys/kernel/debug # ls
bdi
                     kprobes
                                           suspend stats
                     memblock
                                           tracing
benshushu
clk
                                           ubi
                     mmc0
extfrag
                                           ubifs
                     pm qos
                                           usb
fault around bytes
                     regmap
                     regulator
                                           virtio-ports
gpio
hid
                     sleep_time
                                           wakeup_sources
/sys/kernel/debug #
```

进入 regmap 目录之后,你会发现里面有很多不同设备对应的调试 regmap 目录或者节点。

```
/sys/kernel/debug/regmap # ls
dcc:amp@0
                                dcc:volt@5
dcc:amp@1
                                smb:motherboard:mcc:osc@0
dcc:osc@0
                                smb:motherboard:mcc:osc@1
dcc:osc@1
                                smb:motherboard:mcc:osc@2
dcc:osc@2
                                smb:motherboard:mcc:reboot@0
dcc:power@0
                                smb:motherboard:mcc:reset@0
                                smb:motherboard:mcc:shutdown@0
dcc:power@1
dcc:volt@0
                                smb:motherboard:mcc:temp@0
dcc:volt@1
                                smb:motherboard:mcc:volt@0
dcc:volt@2
                                syscon.0.auto-sys id
dcc:volt@3
                                syscon.4.auto-sys_misc
dcc:volt@4
                                syscon.5.auto-sys procid
/sys/kernel/debug/regmap #
```

如果大家对这些问题感兴趣,可以关注笨叔的旗舰篇视频,笨叔会在视频中和大家详细解答。





微店:



微店 奔跑吧Linux内核



扫码识别

淘宝店: https://shop115683645.taobao.com/

微信公众号:

