实验1：补充1

目录

[Arm64架构下增加系统调用 1](#_Toc163552115)

[基于QEMU实现动态调试内核 2](#_Toc163552116)

[TatakOS的安装和增加系统调用 4](#_Toc163552117)

## Arm64架构下增加系统调用

1. 修改文件 “include/uapi/asm-generic/unistd.h”, 参考如下添加（注意系统调用号不能随意加，只能一次加1）:

#define \_\_NR\_hello\_euler 294

\_\_SYSCALL(\_\_NR\_hello\_euler, sys\_hello\_euler)

最后的总数相应修改增加1：

#undef \_\_NR\_syscalls

#define \_\_NR\_syscalls 295

2. 在 include/linux/syscalls.h 添加代码，加在在/\*kernel/sys.c\*/后面

/\* hello euler \*/

asmlinkage long sys\_hello\_euler(void);

3. kernel/sys.c 添加代码

SYSCALL\_DEFINE0(hello\_euler)

{

printk(KERN\_INFO "Hello, OpenEuler!");

return 0;

}

4. 编写测试代码testhello.c：

/\* gcc testhello.c -o testhello \*/

#include <stdio.h>

#include <unistd.h>

#include <sys/syscall.h>

int main()

{

int ret = 0;

printf("Test sys\_hello\_euler:\n");

ret = syscall(294); // \_\_NR\_hello\_euler is not ready

printf("ret = %d!\n", ret);

return 0;

}

可用dmesg | tail -n 10 查看：

[ 4.190557] pciehp 0000:00:04.0:pcie004: Failed to check link status

[ 4.191642] pciehp 0000:00:04.1:pcie004: Failed to check link status

[ 4.192573] pciehp 0000:00:04.2:pcie004: Failed to check link status

[ 4.211387] pciehp 0000:00:04.3:pcie004: Failed to check link status

[ 4.212357] pciehp 0000:00:04.4:pcie004: Failed to check link status

[ 4.885905] random: crng init done

[ 4.886778] random: 7 urandom warning(s) missed due to ratelimiting

[ 4.989937] IPv6: ADDRCONF(NETDEV\_UP): eth0: link is not ready

[ 6.167465] VFS: Open an exclusive opened block device for write vda2 [3463 resize2fs].

[ 295.066720] Hello, OpenEuler!

5. 注意：

（1）可以临时把云主机配置提高，比如提高到8核、16核，相应编译内核时设置-j8 -j16，来提高编译速度。（记得后续及时关机，否则代金券会被很快用掉）

（2）实现复杂的代码逻辑，可以先按照内核模块编译，测试好其核心逻辑（内核模块编译参考实验二），这样编译速度更快很多，且不用每次重启系统。

（3）还可以利用其它调试方式，比如参考一些书，如《Linux内核设计与实现》中第18章。

（4）如果增加系统调用到其它文件（即非sys.c中），可参考网上贴子，修改编译配置 https://biscuitos.github.io/blog/SYSCALL\_ADD\_NEW\_ARM64/

其它可能有用的参考https://www.linuxbnb.net/home/adding-a-system-call-to-linux-arm-architecture/

## 基于QEMU实现动态调试内核

好处一：可以更了解内核，好处二：可以重编译内核后不用重启电脑。

可以在自己更熟悉的系统中，如x86-64架构下， Ubuntu系统中进行。

参考：

<https://ctf-wiki.org/pwn/linux/kernel-mode/environment/qemu-emulate/>

<https://www.kernel.org/doc/html/latest/dev-tools/gdb-kernel-debugging.html>

甚至可以不用busybox：

<https://zhuanlan.zhihu.com/p/105069730?utm_id=0>

遇到问题“Remote 'g' packet reply is too long: ...”可参考

<https://wiki.osdev.org/QEMU_and_GDB_in_long_mode>

initrd解释

<https://developer.ibm.com/articles/l-initrd/>

参考qemu启动参数

qemu-system-x86\_64 -kernel ../kernal/linux-5.4.11/arch/x86/boot/bzImage -initrd ./rootfs.img -append "root=/dev/ram nokaslr" -m 512 -smp 2 -s -S

另一端GDB启动后，

可以用apropos lx 可以看到所有加了脚本后（即如加了add-auto-load-safe-path /home/zengyingpei/Documents/kernal/linux-5.4.11/scripts/gdb/vmlinux-gdb.py 到~/.gdbinit中）额外的内核命令

可以做比如，打印pid为10的进程的task struct

p $lx\_task\_by\_pid(10)

启动过程中随时可以ctrl+C打断，查看目前执行栈：

电脑软件截图

描述已自动生成

## TatakOS的安装和增加系统调用

TatakOS是基于xv6-riscv操作系统更改后的，可以在K210开发板上执行的操作系统，是计算机学院章复嘉老师指导的2022操作系统大赛获奖作品。而是xv6-riscv（https://pdos.csail.mit.edu/6.828/2012/xv6.html）是MIT的Russ Cox, Frans Kaashoek, Robert Morris三位教授为其讲授6.828 Operating System Engineering（https://pdos.csail.mit.edu/6.828/2023/）课程研制的小型操作系统，它具有防Unix风格、代码量小、功能简单、易于理解等优点，适合用于对操作系统进行全方位理解。xv6-riscv的原本代码在1w行左右，tatakOS对其做了一些功能扩展并提供了中文文档。

tatakOS

代码<http://gitlab.ji-sang.com/yztz/tatakos/-/tree/master/>

文档<http://gitlab.ji-sang.com/yztz/tatakos/-/tree/master/doc>

可根据其文档进行安装以及增加系统调用

Xv6

代码 <https://github.com/mit-pdos/xv6-riscv/tree/riscv>

文档 <https://pdos.csail.mit.edu/6.828/2012/xv6.html>

课程 <https://pdos.csail.mit.edu/6.828/2023/>