Linux 内核新增系统调用

以 Debian 12 为例

1 下载并解压 Linux Kernel 源码

在 https://www.kernel.org/ 处下载，以 6.6.2 为例。

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.6.2.tar.xz
tar -xvf linux-6.6.2.tar.xz
cd linux-6.6.2
```

2 添加系统调用函数

新增 `kernel/hello.c` 文件如下：

```
// kernel/hello.c

#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello){
    printk(KERN_INFO "Hello, Linux 6.6.2\n");
    return 0;
}
```

`SYSCALL_DEFINE0`是一个宏，用于定义一个不接受任何参数的系统调用。这个宏处理系统调用的名称和参数数量。在这个例子中，`hello`是系统调用的名称。数字`0`表示这个系统调用不接受任何参数。

`printk`是内核中的一个函数，用于输出日志信息。`KERN_INFO`是日志级别，表示这条信息是一条普通的信息性消息。`"Hello, Linux 6.6.2\n"`则是要输出到内核日志的实际消息。当这个系统调用被执行时，这条消息会被记录在内核日志中，可以用`dmesg`命令查看。

3 系统调用表中注册系统调用

对于 x86 架构，修改 `arch/x86/entry/syscalls/syscall_64.tbl` 文件

添加以下内容，注意代码序号，按序存放

```
# For x86_64
335     64      hello                   sys_hello
```

```
333     common  io_pgetevents           sys_io_pgetevents
334     common  rseq                    sys_rseq
# For x86_64
335     common      hello                   sys_hello

# don't use numbers 387 through 423, add new calls after the last
# 'common' entry
424     common  pidfd_send_signal       sys_pidfd_send_signal
425     common  io_uring_setup          sys_io_uring_setup
```

4 在系统调用头文件中声明

修改`include/linux/syscalls.h`，在最后一行`#endif`之前添加以下代码

```
int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
                int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
                int optlen);
// include/linux/syscalls.h

asmlinkage long sys_hello(void);

#endif
```

```
1  // include/linux/syscalls.h
2
3  asmlinkage long sys_hello(void);
```

`asmlinkage`是一个宏，它用于告诉编译器该函数的参数不是通过寄存器传递的（这是C调用惯例的常见方式），而是通过系统调用的堆栈传递的。

这是因为用户空间程序通常通过`syscall`指令来进行系统调用，该指令将参数放在寄存器中，而系统调用服务例程（即系统调用的实际实现）需要从堆栈中获取这些参数。

5 在 kernel 目录下的 Makefile 中添加引用

修改 `kernel/Makefile`，更新代码如下

在 `obj-y` 的最后添加 `hello.c`

```
# SPDX-License-Identifier: GPL-2.0
#
# Makefile for the linux kernel.
#

obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o capability.o ptrace.o user.o \
            signal.o sys.o umh.o workqueue.o pid.o task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o ucount.o regset.o ksyms_common.o hello.o

obj-$(CONFIG_USERMODE_DRIVER) += usermode_driver.o
```

6 重新编译并安装内核

安装依赖

```
apt install dwarves build-essential libncurses-dev bison flex libssl-dev libelf-dev bc
```

构建之前确保你在 linux-6.6.2 目录下

编译内核需要很长的时间

```
make defconfig        # 使用默认配置
# make menuconfig     # 或者自选配置
make -j$(nproc)       # 编译内核
make modules_install # 安装内核模块
make install          # 安装内核
```

在编译内核后出现`Kernel: arch/x86/boot/bzImage is ready  (#1)`表示成功

对于使用 grub 的机器，使用以下命令安装内核，然后重启系统

```
update-grub
reboot
```

重启后使用 `uname -a` 检查内核版本，如下表示成功

```
root@vps:~/linux-6.6.2# uname -a
Linux vps 6.6.2 #1 SMP PREEMPT_DYNAMIC Sat Nov 25 16:46:36 UTC 2023 x86_64 GNU/Linux
root@vps:~/linux-6.6.2#
```

7 测试系统调用

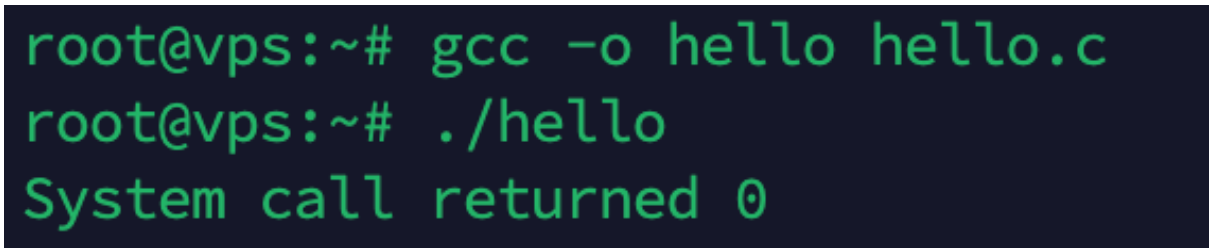
编写 `hello.c` 程序

```c
#include <stdio.h>
#include <sys/syscall.h>
#include <unistd.h>

#define SYS_hello 335 // 使用你在syscall_64.tbl中设置的系统调用号

int main(){
    long int ret = syscall(SYS_hello);
    printf("System call returned %ld\n", ret);
    return 0;
}
```

编译运行

```
gcc -o hello hello.c
./hello
```

出现 returned 0 表示成功，-1表示失败



在 dmesg 中查看消息

```
dmesg | tail
```

```
root@vps:~# dmesg | tail
[    3.020133] systemd[1]: Starting systemd-sysusers.service - Create System Users...
[    3.042373] systemd[1]: Started systemd-journald.service - Journal Service.
[    3.086802] random: crng init done
[    3.112697] EXT4-fs (sda1): resizing filesystem from 26181627 to 26181627 blocks
[    3.115948] systemd-journald[201]: Received client request to flush runtime journal.
[    3.561282] virtio_net virtio1 eth0: renamed from enp0s18
[    3.903181] FAT-fs (sda15): Volume was not properly unmounted. Some data may be corrupt. Pleas
e run fsck.
[    4.924050] ISO 9660 Extensions: RRIP_1991A
[   45.226091] systemd[524]: memfd_create() called without MFD_EXEC or MFD_NOEXEC_SEAL set
[   74.459061] Hello, Linux 6.6.2 - Haotian Chen.
root@vps:~#
```

可以看到输出了 `Hello, Linux 6.6.2`