Creating and Intercepting System Calls

Note: I've used Linux Kernel 3.18.1 here, 64-bit

1 Adding a new system call

1.1 syscall_64.tbl

In the linux source, modify the file `arch/x86/syscalls/syscall_64.tbl`
Browse to the section with system-calls of common architecture
Add a new line to add the system call:

```
<a-free-number>  common  <sys-call-name>  <sys-call-function-name>
```

Typically, the sys-call-function name is '`sys_sys-call-name`'.

Here in my example, I'm adding the syscall **psyscall**, which is denoted by the function name **sys_psyscall**. I found that the number 322 wasn't used. So, the line I added would be:

```
322   common   psyscall   sys_psyscall
```

1.2 syscalls.h

Modify the file `include/linux/syscalls.h`.
To the end of the file, add

```
asmlinkage long <sys-call-function-name>(args);
```

For me, it is

```
asmlinkage long sys_psyscall(int x);
```

1.3 code directory

In the linux source code root, create a directory <sys-call-name>. Create a program `syscall.c` and a `Makefile` inside it.

```
mkdir psyscall
cd psyscall
touch syscall.c
touch Makefile
```

1.4 psyscall/syscall.c

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

// asmlinkage long <sys-call-function-name>(args)
asmlinkage long sys_psyscall(int x)
{
    //functionality
    printk(KERN_ALERT "Called with value %d.\n", x);
    return 0;
}
```

1.5 psyscall/Makefile

```
obj-y := syscall.o
```

1.6 Kernel Makefile

Modify the kernel `Makefile`. Find the line starting with '**core-y**'. By default, it looks like:

```
core-y      := usr/
```

Modify it to

```
core-y      := usr/ <name-of-created-directory>/
```

For me, it was

```
core-y      := usr/ psyscall/
```

Now compile and install the kernel. The system call will be available on booting to the new kernel.

2 Intercepting the system call

In a loadable kernel module,

2.1 Starting interception

1. Acquire system call

   ```
   sct = PAGE_OFFSET
   while(sct[__NR_close] != sys_close)
       sct += PAGE_OFFSET;
   ```

   When that condition is satisfied, `sct` is having the system call table address.

2. Change permission of that location to obtain write-access Get the pte of the `sct` address using lookup_address and OR it with `_PAGE_RW`

   ```
   pte->pte = pte->pte |= _PAGE_RW;
   ```

3. System call interception can be started by:

   ```
   backup = (void *)sct[__NR_psyscall];
   sct[__NR_psyscall] = (unsigned long *)new_psyscall;
   ```

2.2 Stopping interception

1. System call interception can be stopped by:

   ```
   sct[__NR_psyscall] = (unsigned long *)backup;
   ```

2. Change the `sct` address permission back to read-only by AND-ing the pte with ~~`_PAGE_RW`.

   ```
   pte->pte = pte->pte &= ~_PAGE_RW;
   ```