Assignment 3 Tutorial

System Calls

Papadogiannakis Manos
papamano@csd.uoc.gr

CS-345: Operating Systems
Computer Science Department
University of Crete

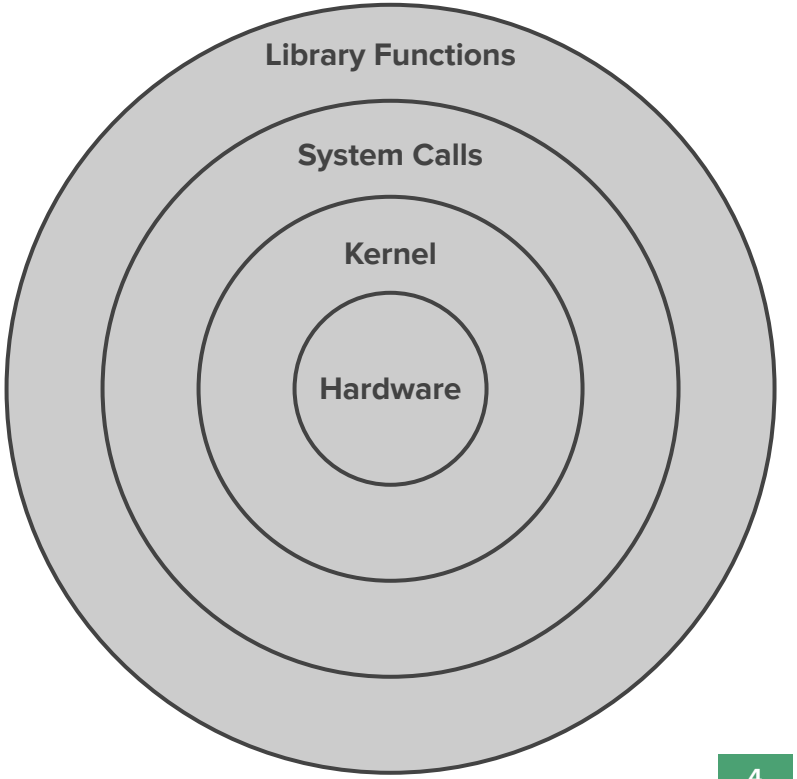
Outline

- Linux Kernel
- System Calls
- Emulator
- Implementing a new system call

Linux Kernel - System Calls

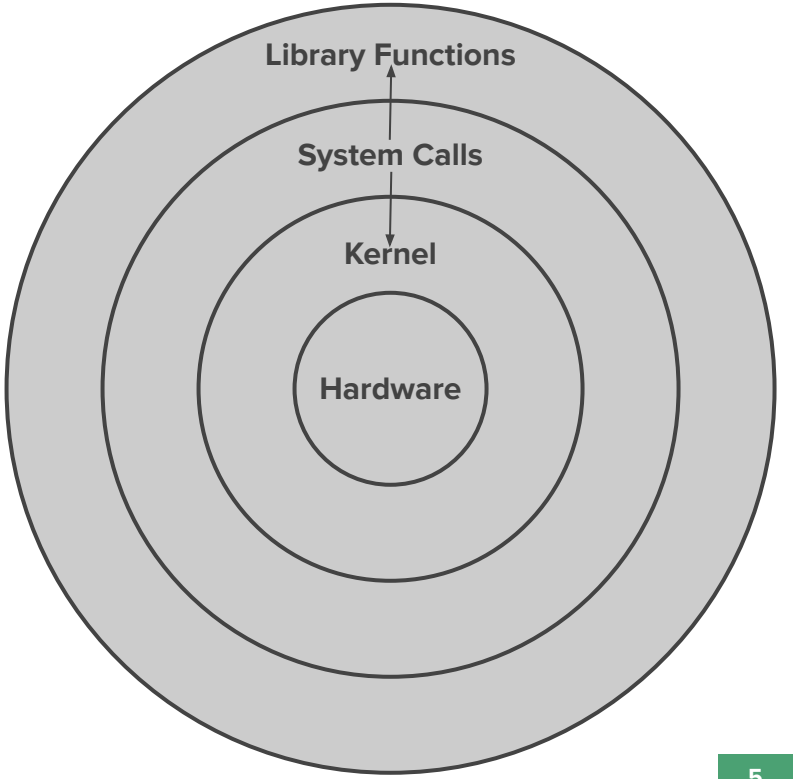
Linux Kernel

- **Heart of the Operating System**
- **Interface between resources and user processes**
- **What the Kernel Does**
  - Memory Management
  - Process Management
  - Device Drivers
  - **Systems Calls**



System Calls

- The interface between a process and the Operating System
- Method for a program to request a service from the kernel

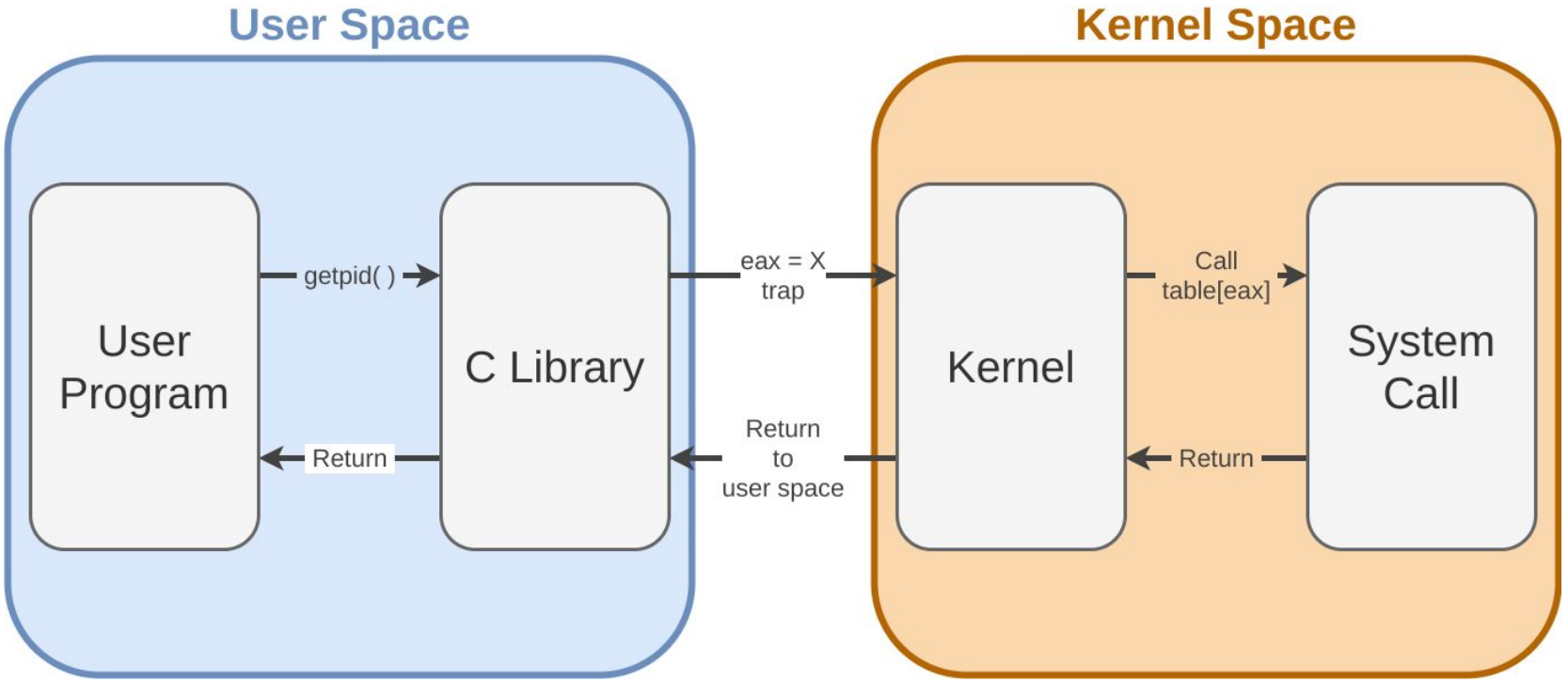


System Calls

- **Process Control**
  - fork, exit, wait
- **File Manipulation**
  - open, read, close
- **Device Manipulation**
  - ioctl, release
- **Information**
  - getpid, gettid
- **Communication**
  - pipe, socket
- **Security**
  - chmod, chown

System Calls

```
printf( "The process ID is %d\n", getpid() );
```



System Calls

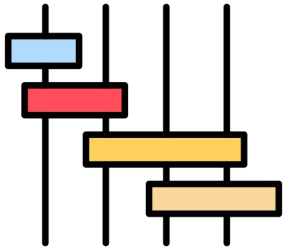
- System calls without wrapper functions:

```
syscall(long number, ...);
```

- A library function that invokes the system call with the specified number
- System call numbers can be found in `<sys/sycall.h>`

Assignment 3

- **Support for a new Scheduling Policy**
  - Shortest Period First



- **Implement two new system calls**
  - `set_period_params(...)`
  - `get_period_params(...)`

Linux Kernel

- Getting the source code...

```
$ cd /spare
$ mkdir <username>
$ chmod 700 <username>
$ cd <username>
$ cp ~hy345/qemu-linux/linux-2.6.38.1.tar.bz2
$ tar -jxvf linux-2.6.38.1.tar.bz2
```

Linux Kernel

- Compiling it...

```
$ cd linux-2.6.38.1
$ cp ~hy345/qemu-linux/.config .

<Implement additional functionality>

$ make ARCH=i386 bzImage
```

Emulator

- Load the image and start the guest OS

```
$ cp ~hy345/qemu-linux/hy345-linux.img .
$ qemu-system-i386 -hda hy345-linux.img
```

- Load the image and start the guest OS with new kernel

```
$ qemu-system-i386 -hda hy345-linux.img -append " root=/dev/hda" -kernel
  linux-2.6.38.1/arch/x86/boot/bzImage -curses
```

Implementing a new System Call

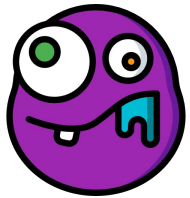
System Calls

Implementing a new System Call:

1. Define a system call number
2. Define a function pointer
3. Define a function
4. Implement the system call

Example

- Implement the infamous dummy_sys system call
- Takes one integer as a single argument
- Prints something and return the integer multiplied by 2

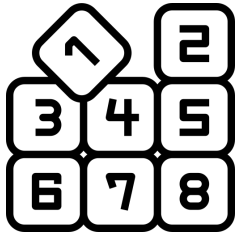


Define a System Call Number

- **Each system call has a number**

- **Edit** *linux-2.6.38.1/arch/x86/include/asm/unistd_32.h*
  - Define a new system call number
    #define __NR_dummy_sys 341

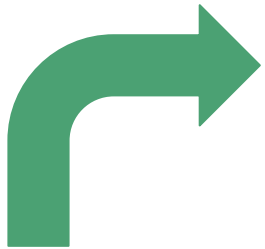
  - Increase the number of system calls by 1
    #define NR_syscalls 342



Define a function pointer

- **The kernel needs to have information pointing to the new system call**

- **Edit** *linux-2.6.38.1/arch/x86/kernel/syscall_table_32.S*
  - Add an entry at the bottom of the list
    .long sys_dummy_sys



Define a function

- **Edit** *linux-2.6.38.1/include/asm-generic/syscalls.h*

```
#ifndef sys_dummy_sys
        asmlinkage long sys_dummy_sys(int arg0);
#endif
```

Implement System Call

- **Create** *linux-2.6.38.1/kernel/dummy_sys.c*

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


asmlinkage long sys_dummy_sys(int arg0) {
    printk("Called dummy_sys\n");
    return ((long)arg0 * 2);
}
```

Compilation Process

- **Edit** *linux-2.6.38.1/kernel/Makefile*

```
obj-y += dummy_sys.o
```

- **Compile the kernel...**

Test new System Call

- **Start the VM with the new kernel**
  - $ qemu-system-i386 -hda hy345-linux.img -append "root=/dev/hda" -kernel linux-2.6.38.1/arch/x86/boot/bzImage -curses
- **Write a test application**
  - $ vi test.c
- **Compile the test application**
  - $ gcc -o demo.out test.c
- **Run the test**
  - $ ./demo.out
- **Check the kernel log**
  - $ dmesg | tail

Demo Application

```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

#define __NR_dummy_sys 341

int main(void) {
    printf("Trap to kernel level\n");
    syscall(__NR_dummy_sys, 42);
    printf("Back to user level\n");

    return 0;
}
```

Wrapper Function

- Macro

```
#define dummy_sys(arg1) syscall(341, arg1)
```

- Wrapper Function

```
long dummy_sys(int arg1) {
    return syscall(341, arg1);
}
```

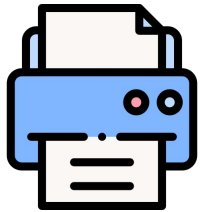
Notes

Process Data

- **You will need additional information stored for each process**
- **Edit** *linux-2.6.38.1/include/linux/sched.h*
  - Find the task_struct structure
  - Introduce new fields
- **Your system calls will interact with this fields**

Printk

- **Every time one of your system calls is executed you should print a message**
  - Your name and A.M.
- **You can view these messages from user level:**
  - dmesg
  - cat /var/log/messages
- **Very useful for debugging messages**



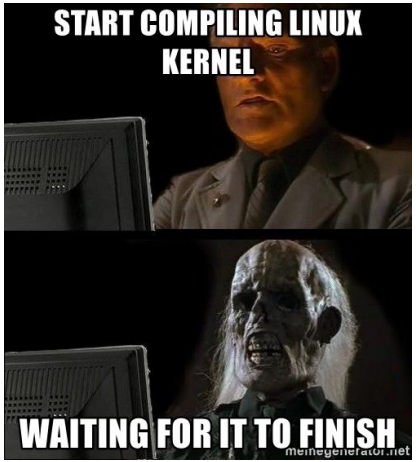
Hints

- **Useful kernel functions:**
  - for_each_process( )
  - get_current( )
  - access_ok( )
  - copy_from_user( )
  - copy_to_user( )

Turnin

What to submit:

1. bzImage
2. Modified or created source files
3. Test programs and headers in Guest OS
4. README



Credit

- Icons from FlatIcon, made by:
  - DinosoftLabs
  - surang
  - Freepik
  - Smashicons

Good luck!

papamano@csd.uoc.gr

Questions?