CSCI-3753: Operating Systems Fall 2019

Abigail Fernandes

Department of Computer Science

University of Colorado Boulder

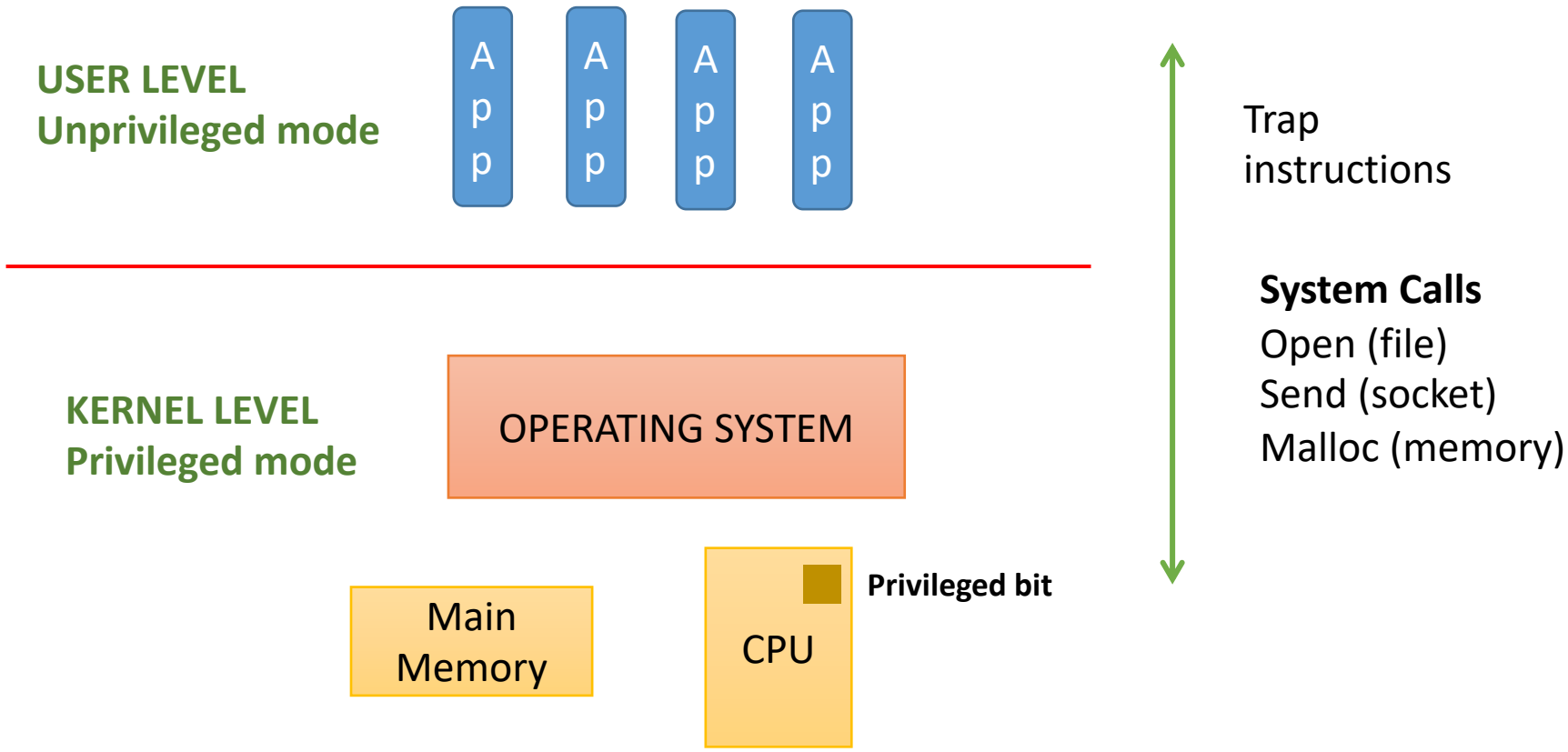
Logistics

- Office Hours on Monday, 10:15 – 12:15 (**ECAE 133**)
- Discussion forum - **Moodle**
- First Recitation Quiz!
- Special accommodations
- Problem Set 1 Questions

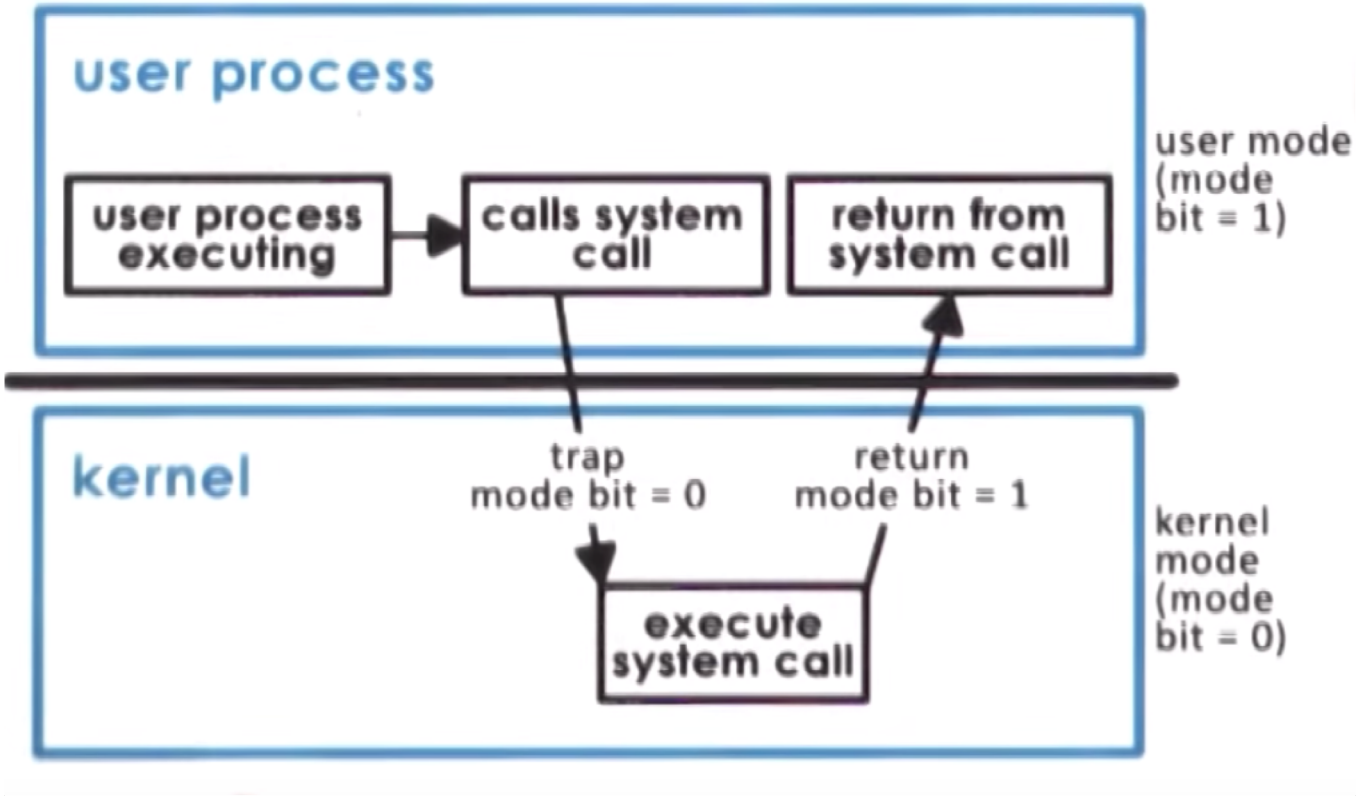
Week 2: PA1 & LKM

PA 1 – Questions & Answers

User/Kernel Protection Boundary



System call Flowchart



System call

TO make a system call, an application must

- Write arguments
- Save relevant data to a well defined location
- Make system call (using the specific system call number)

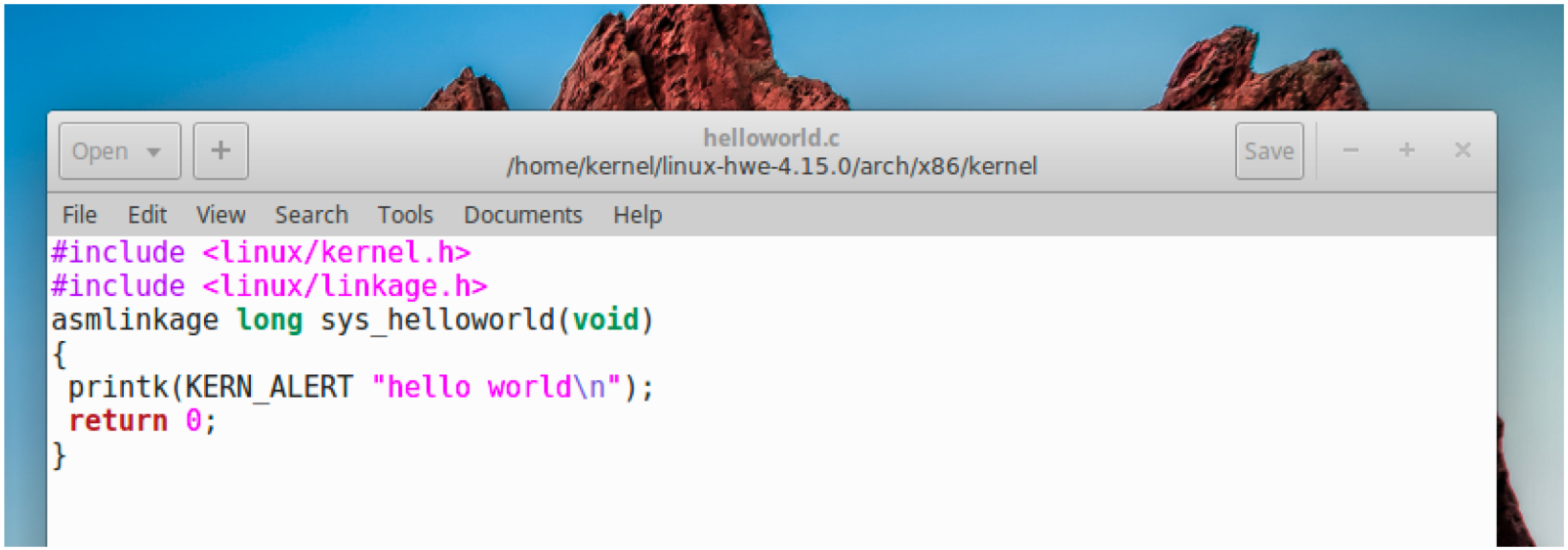
Executing a System call (Flowchart explained)

1. A user space program invokes the syscall
2. A (typically) software interrupt called a trap is triggered (INT)
3. Mode bit is flipped from user to kernel (1 to 0)
4. The interrupt tells the kernel which syscall was called
    1. Requisite data may be passed in
    2. The kernel verifies if all parameters are legal before executing the system call
5. After execution, mode bit flips and user program resumes

Adding a System Call

1. Write the system call source code

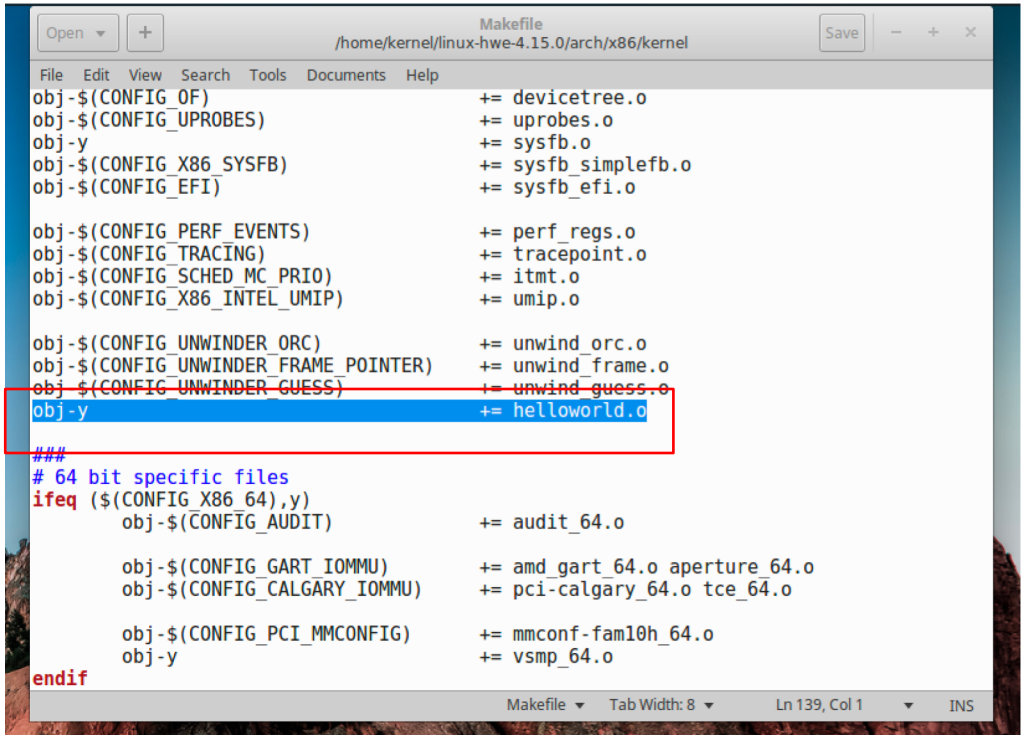
   sudo gedit arch/x86/kernel/helloworld.c



Adding a System Call

2. Add the new syscall to the Makefile

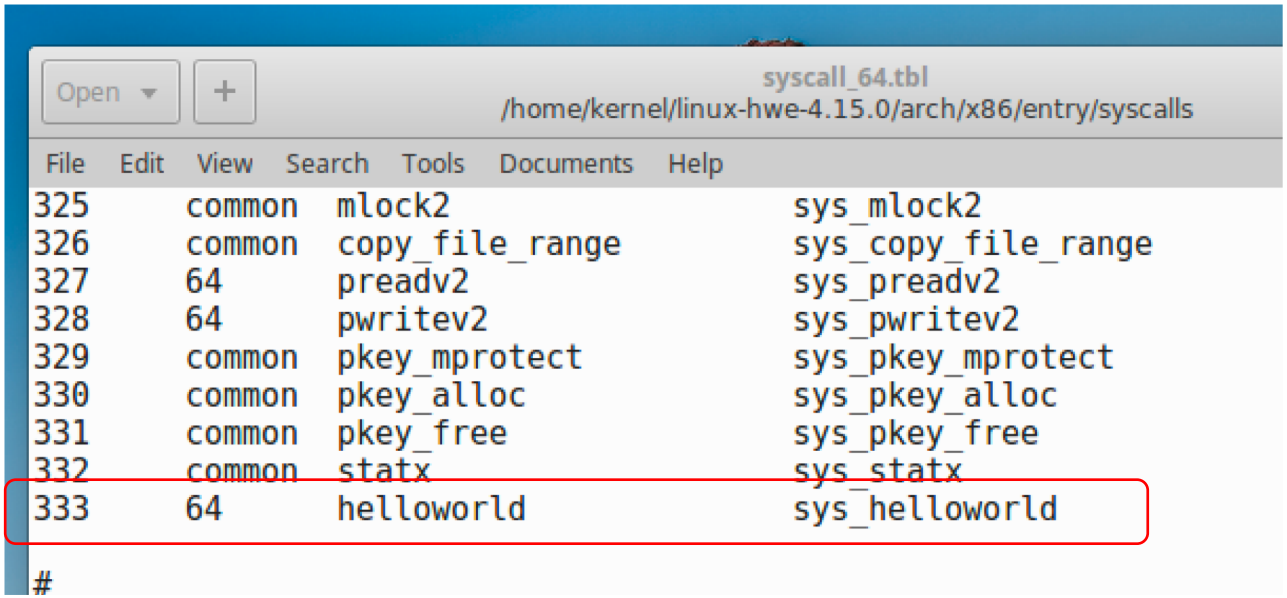
arch/x86/kernel/Makefile



Adding a System Call

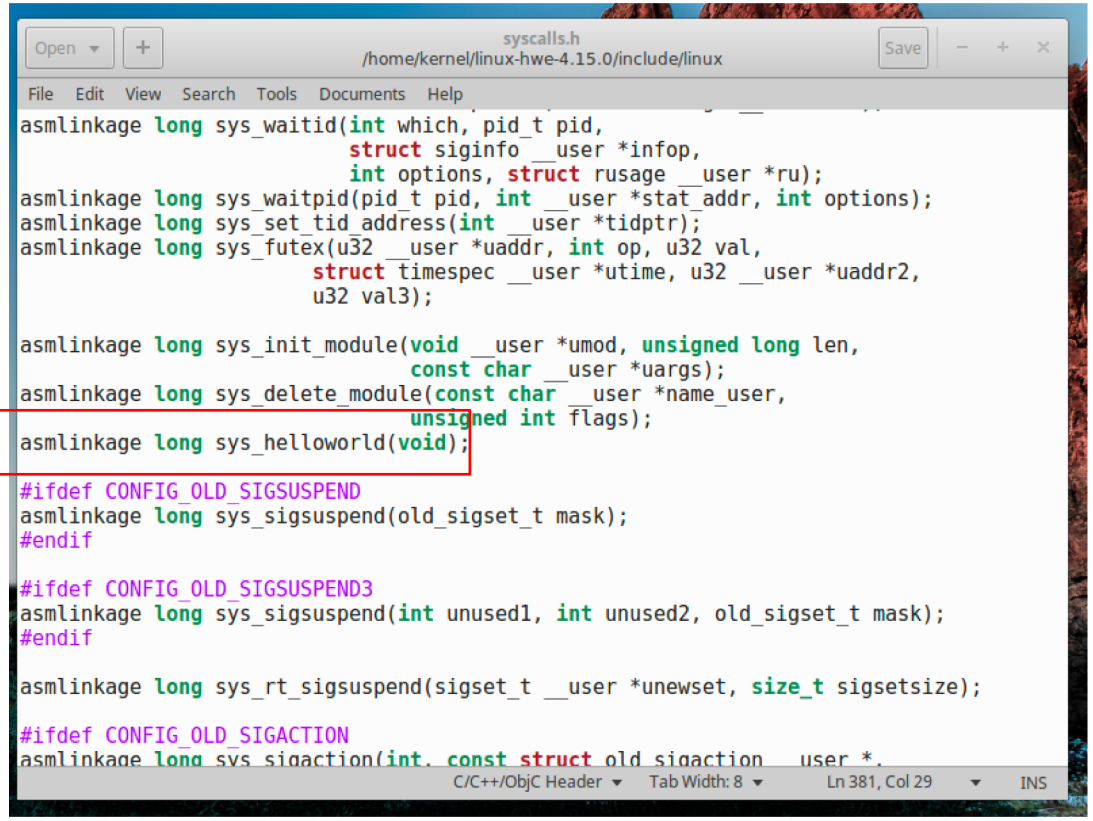
3. Add the syscall to the syscalls table

arch/x86/entry/syscalls/syscall_64.tbl



Adding a System Call

4. Add the syscall prototype to the syscalls header file include/linux/syscalls.h



Compiling the Kernel

1. Create a config file and setup a name to append to the release

```
sudo cp /boot/config-$(uname -r) .config
```

2. Make

```
sudo make -j2 CC="ccache gcc"
sudo make -j2 modules_install
sudo make -j2 install
```

3. Reboot

Passing values safely between user and kernel spaces

USER SPACE MEMORY ACCESS

```
unsigned long copy_from_user(void *to,
  const void __user *from,
    unsigned long n);
unsigned long copy_to_user(void __user *to,
  const void *from,
    unsigned long n);
```

```
get_user()
put_user()
```

```
printk
```

```
cat /var/log/syslog
dmesg
```

You have to write a new system call. This system call will be given two numbers and an address of where to store the results

Loadable Kernel Module (LKM)

How to add a code to a Linux Kernel

- Method 1:
  - Step 1: Add source files to the Linux kernel source tree
  - Step 2: Recompile the kernel

→PA1

→It takes a long time to compile the kernel source code (2-3 hours) !!!

- Method 2: Add code to the Linux kernel while it is running

Loadable Kernel Module (LKM)

Loadable Kernel Module (LKM)

- LKM is a *chunk of code* that we add to the Linux kernel while it is running.

- Typically, one of the following modules:
  1. Device drivers
  2. Filesystem drivers
  3. System calls

Loadable Kernel Module (LKM)

- LKMs (when loaded) are very much part of the kernel.

→ Part of the kernel that is bound into the image that you boot is "*base kernel*".

- LKMs communicate with the base kernel.

Why LKM?

- DON'T have to rebuild the kernel
- Help diagnose system problems
  - A bug in a device driver which is bound into the kernel can stop the system from booting at all.
- Save memory
  - Have them loaded only when we are actually using them
- Much faster to maintain and debug

LKM Utilities

- ***insmod***: Insert an LKM into the kernel.
- ***rmmod***: Remove an LKM from the kernel.
- ***lsmod***: List currently loaded LKMs.
- ***kerneld***: Kernel daemon program
  - allows kernel modules to be loaded automatically rather than manually with insmod/modprobe
- ***modprobe***: Insert/remove an LKM or set of LKMs intelligently.
  - e.g., if you must load A before loading B, modprobe will automatically load A when you tell it to load B.

LKM Utilities

- ***depmod***: Determine interdependencies between LKMs.
- ***ksyms***: Display symbols that are exported by the kernel for use by new LKMs.
- ***modinfo***: Display contents of .modinfo section in an LKM object file.

Week 2 – Checklist

- ❑ Submit PS1 by 6PM today !!!
- ❑ Start and complete PA1
- ❑ Read more about LKM