# AOS Assignment 3

## Linux Kernel Module

> 💡 Note: Use of Ubuntu 18.04 is required for this assignment.

The purpose of this assignment is to introduce Linux Kernel programming in the context of writing a Linux Kernel module. Kernel code executes in kernel (privileged mode) where program instructions can command the CPU to perform any operation allowed by the CPU's architecture. This allows any instruction to be executed, any I/O operation to be initiated, and any area of memory accessed directly.

**It is best to complete this tutorial on a virtual machine.**

Make sure all the necessary libraries are installed

```
sudo apt update
sudo apt install make binutils gcc
```

## Download kernel module starter code

- To begin, download the tar.gz archive containing a basic Hello World Linux kernel module.
- First, create a directory for this assignment, and then download the tar gz:

```
mkdir a3
cd a3
wget http://faculty.washington.edu/wlloyd/courses/tcss422/assignments/hello_module.tar.gz
tar xzf hello_module.tar.gz
cd hello_module
ls l
```

The **hello_module** contains **“helloModule.c”**, which includes source code for the most basic Linux kernel module. Inspect the code. The module consists of an initialization method and a cleanup method. The initialization method is triggered automatically as an “event” when the module is loaded. The cleanup method is triggered when the kernel module is unloaded. Kernel modules are loaded dynamically into the operating system.

## Compile, install, and test a Linux Kernel Module

1. Compile the module
2. Load the kernel object file
3. Display what has been written to the kernel log file, check the last 10 lines using
4. Unload the module and check the logs

## Count the number of running Linux tasks

> 💡 Only privileged kernel code can read kernel data structures.

For this activity, we make use of **task_struct** linux kernel data structure. It is used for the inspection of running threads and the processes.

**Activity:**

Add a function to the source code. Make use of the helper function **for_each_process** and the data structure **task_struct to** to count the number of running tasks.

Now, call this function from the init function which will be run when the module is loaded.

Note:

- Make sure **sched.h** and **mm_types.h** are present on the system.
- Recompile and reload the kernel module and check output to the syslog.