# E0-253: Operating Systems Programming Assignment

## Background

eBPF is a Linux kernel technology that enables users to run sandboxed programs in the kernel space, without modifying the kernel source code or loading kernel modules. eBPF programs can be written in a restricted C-like language and compiled into bytecode that is verified and executed by the kernel. eBPF can be used for various purposes, such as networking, observability, tracing, and security, by attaching eBPF programs to different types of hooks in the kernel. eBPF is a powerful and flexible tool that allows users to extend and customize the kernel functionality at runtime, with high performance and safety guarantees.

## Assignment-1

The assignment consists of multiple small tasks as discussed below. Students can share with each other commands used to perform the first task below (introductory). Sharing is not allowed for other tasks.

1. Introductory: Write a program that prints "Hello World" whenever an execve() syscall is called by any process in the system. Remember to use bpf helpers from the bpf C library. Compile the program using clang (no need to compile clang, you can use a packaged version of clang.).
   (a) Observe the objdump to see it getting compiled to BPF bytecode rather than x86 asm code (no need to compile LLVM, explore LLVM options to extract bpf bytecode).
   (b) Use bpftool to load the program in kernel (you can either compile bpftool in Linux tree or install a packaged version).
   (c) Attach the bpf program to the execve syscall.
   (d) Observe "Hello World" output in the bpf ring buffer (Know more about the ring buffer, you will know where to look for output in system).

   **Expected Deliverable:** None. This is for you to get familiar with eBPF.

2. Develop a bpftrace script to show the number of files currently opened (it should be equal to file descriptors held by process under `/proc` directory) by a particular process.

   **Expected Deliverable:** Mostly a single-line program. Must have standard sections present in bpftrace script. Learn more about each section of the single-line script.

3. Write a BCC program that attaches to the execve() system call and prints the pid and the number of times it has called the syscall. (note how BCC handles all the steps performed in Task 1). Note that you need to first install Python bindings for this task. Hence, import and use the python BCC package.

   **Expected Deliverable:** A single file with your code. Your code must contain an inlined PROG section and BPF( ) call. You can use the `perf` tool to verify your output.

4. Write a BCC program to measure the average time taken by processes in the system (context switch time = difference between the time when the second process starts and the first process stops).

   **Expected Deliverable:** A single file with your code that outputs min, max and mean of the context switch time (the output should appear when `Ctrl+C` is pressed after running your program).

5. Write an eBPF program to compute the amount of anonymous memory used by a given program. Assume huge pages are disabled for this task.
   **Test case:** The test case has been uploaded in the class Teams.
   **Expected Deliverable:** A single code file that outputs the anonymous memory usage of given program.

## Submission Timeline:

29-Feb-2024 (midnight IST).

## Submission Instructions:

Each student should do the assignment by herself/himself (**no** group).
We will create an assignment in Gradescope where you have to upload a zipped folder containing your answers. The folder should be named after your SR number and it should contain one file for every question, except question 1. The files should be named after the question number. We will provide further instructions on how to submit your homework at a later point in time.

**References:**

[1] eBPF - Rethinking the Linux Kernel.
`https://www.youtube.com/watch?v=f-oTe-dmfyI`
[2] A Beginner's Guide to eBPF Programming - Liz Rice - Full Keynote.
`https://www.youtube.com/watch?v=lrSExTfS-iQ`
[3] `LearningeBPF(isovalent.com)`
[4] `KernelPwningwitheBPF-aLoveStory-chompieatthebits`