University of Hamburg
Department of Informatics
Group of Scientific Computing
K. Chasapis, Dr. J. Kunkel, Dr. M. Dolz, M. Wiedemann

Praktikum „Kernel Programming“
in WiSe 2014/2015
Assignment 1
Deadline: 13.01.2015

1 Exercise: Environment Setup (15 pt.)

Kernel programming requires a special system configuration. For this exercise you have to set up a Linux image that you will use to solve the kernel programming assignments. Details on the packages that you have to install can be found at the "Building, running and monitoring the Linux kernel" lecture slides.

Although, we do not impose any restriction in the Linux distribution that you will use you are oblige to provide us access to your machine unless you choose `CentOS 7.0`. Since it is very likely that you will crash the operating system it is highly recommended to use a virtual machine. You can use any virtualization you prefer.

Once you finish with the machine's setup, execute the following shell commands `time`, `uname -a`, `lsmod`, `/proc/modules`, `/proc/cpuinfo`, `/proc/mem`, `free -m` and submit their output as the solution of this exercise.

2 Exercise: Makefile (20 pt.)

In the code examples' section of the software lab's website you will find the `hello_world` kernel module. In this is exercise you will have to write and deliver the Makefile that can be used to build this module.

Once you have build the module, try to experiment with it by returning different values in the `init` function and check the results in the kernel's log file.

3 Exercise: Module Parameters (20 pt.)

This exercise is intended to familiarize yourself with the kernel module parameters. You have to extend the `hello_world` module by passing more parameters. More specific you will pass an array of any type you wish.

During the initialization process the kernel module should print these parameters. Submit the modified hello_world C file and a text file that contains of the **modinfo** tool. (you can use the same Makefile with the 2nd exercise).

4 Exercise: Character Device Driver Documentation (20 pt.)

The Linux kernel implements character device drivers for various reasons. Two of the commonly used are `/dev/zero` and `/dev/null`.

In this exercise you will have to locate in which files these modules are implemented and understand how they work. Submit a text file that will describe the implementation of these modules.

5 Exercise: Implementation of a Circular Buffer (25 pt.)

In this exercise you have to extend the functionality of the `char_module` that can be found in the "code examples" section of the software lab's website. This module already supports the `open` and `close` function. Your task is to will implement the `read`, `write` and `lseek` functions.

The functionality of the program is very simple. Every `write` call fills up a `char` buffer inside the kernel module. Once the buffer is full, the `write` starts from the beginning of the buffer. Following `read` calls start from the beginning and return the values stored in the buffer. Similar to `write`, once they reach the end of the buffer they start again from the beginning. The `lseek` function can be used to move the read cursor in the desire position inside the buffer. Initially the size of the buffer is set by a pre-processor parameter or as a command line parameter when the module is loaded. Moreover, you should modify the module so it can have many instances. Each instance of the module will have a private circular buffer. Using `IOCTLs` you should be able to: *i)* modify the circular buffer size, *ii)* get the size of the buffer, *iii)* get the number of instances that are currently loaded.

Test your kernel module using a program written in C and also using the linux command tools `echo` and `cat`. Submit *i)* the Makefile that you will use to build the module, *ii)* the updated source code of the char_module and *iii)* the C program that you used to test the kernel module.

Submission

You should submit the solution of above exercises in a single tar file named after your last name and the assignment number (e.g chasapis_1.tar). You should include main functions and also test programs. Submit the solution of your exercise by email to your corresponding supervisor. More details of this assignment will be given during the lecture time. You are encouraged to send questions to our mailing list.