CS3401 : Operating Systems

Project 2:

Loading Kernel Modules and Listing Tasks

Mohamed Eliskandrani

900140998

Introduction:

There are two parts to this project. The aim of the first is to load kernel modules into the linux kernel dynamically and the aim of the second was to traverse and display all tasks in the system both linearly and using dfs. There are several states that the task can take which can be shown by the project. The first and most common of these is the running state which indicates that the process is active and serving its requests. Sleeping means the process is awaiting the resources to run. Interruptible sleep means the process is waiting for a particular time slot or event to occur. Uninterruptible sleep means that it will awake when waited-upon resources become available to it. Stopped means the process has ended or has been terminated but has not been removed from the process table to ensure that parent processes know that the child has terminated successfully. Zombie is a state where the parent dies before it releases its child in the process table.

Code:

The code uses four linux libraries init.h, kernel.h, module.h and sched.h. Init.h is the library that contains the macros needed for the project. Module.h is needed by all kernel modules. Kernel.h is needed to access the printk function under KERN_ALERT. Sched.h is needed to access the functions list_for_each, list_entry and for_each_process. For_each_process iterates over all task_struct variables defined in sched.h, list_for_each iterates over the linked list of tasks and list_entry gets the task_struct for the current entry. The dfs function works by using the list_for_each and list_entry functions to recursively iterate over the list by calling dfs(child) for each child which in turn gets called for child's children and only when the functions have exhausted all of the children of a parent task does it move on to the next task and so on. You can check that the dfs is working by checking the parent id of the tasks listed.

Difficulty and Complications:

Most of the time spent on the project was used towards researching how the aforementioned functions work and how to use them to traverse all tasks in a dfs manner and researching how the linux kernel is structured and how tasks can be accessed using the sched.h library. If I were to do this project again I would probably do a lot more research before attempting to write code and wasting time on code that is already implemented using functions and macros.