CS446 Assignment 1

Operating Systems, Winter 2022

Question 1 [10 points]: Linux Kernel Module for Listing Tasks:

In this project, you will write a kernel module in C that lists all current tasks in a Linux system. Review the Lab 2 and Lab 3 material (solutions for the two labs were posted on Canvas), which deals with creating Linux kernel modules and `task_struct` data structures in Linux. This question should be completed using the Linux virtual machine you installed as part of Lab1.

Hints: The swapper/idle process, as stated in Lab2b is called the `init_task`. Use this as the root (starting task) to traverse through the list of tasks in the system.

Deliverables:

1. **q1.c** - You are to provide your solution as a single C program named q1.c that contains the entire solution for question 1.
2. **q1output.txt** - You are also to provide the output of the dmesg command in a text file called q1output.txt. The output should show that the kernel module was loaded into the kernel. It should also show the list of all tasks in your Linux virtual machine.

Note: To run the code for question 1, please follow the following steps:

1- Clear the kernel log buffer: sudo dmesg -c
2- Compile the new q1.c: make
3- Load the kernel module: sudo insmod q1.ko
4- Check the kernal log buffer and save: dmesg >> q1output.txt
5- Remove the kernel module: sudo rmmod q1
6- Print the output to file: dmesg >> q1output.txt

Question 2 [10 points]: Multithreaded Sorting Application:

Write a multithreaded sorting program that works as follows:
A list of integers is divided into two smaller lists of equal size. Two separate threads (which we will term ***sorting threads***) sort each sublist using a sorting algorithm of your choice. The two sublists are then merged by a third thread—a ***merging thread***—which merges the two sublists into a single sorted list. Because global data are shared across all threads, perhaps the easiest way to set up the data is to create a global array. Each sorting thread will work on one half of this array. A second global array of the same size as the unsorted integer array will also be established. The merging thread will then merge the two sublists into this second array. Graphically, this program is structured according to the below figure:

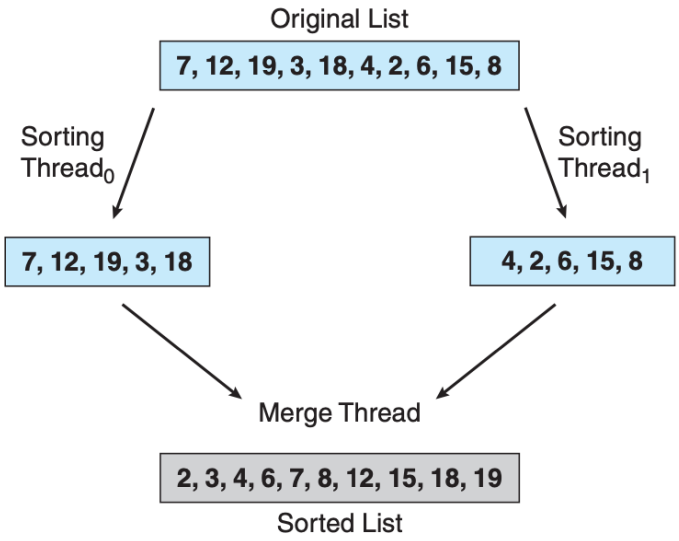

Figure: Multithreaded sorting

This assignment will require passing parameters to each of the sorting threads. In particular, it will be necessary to identify the starting index from which each thread is to begin sorting. The parent thread will output the sorted array once all sorting threads have exited.

Note:

- **You are required to pass the start and end index for the sublist as parameters when you create a thread. Look at lab test 2 solution (posted on Canvas), OR you can use your solution of lab test 2 to pass parameters to each thread.**
- **To sort, you can use any algorithm of your choice.**
- **Uses simple merge sort for merging the two sorted sublists.**

Deliverables: :

1. **q2.c** - You are to provide a C file named q2.c that contains the entire solution for question 2.
2. **output2.txt** - You are also to provide a text file called output2.txt that contains three test cases (of your choice) and your solution's output for these test cases.