1 Purpose

This is your first assignment that really does a "real" thing inside the kernel instead of a quick exercise. This project will create an infrastructure that you will use in the following assignment. You must create a data structure and corresponding `/proc` files that represent the group of processes owned by each user ID. This data structure should be always kept up to date so that when the `/proc` files are queried for the information they contain, they can immediate return an answer instead of having to go through the list of all the processes and determine on the fly whether they are owned by the desired user ID.

2 Exercises

This assignment is broken down into two parts: creating the data structure and relaying the information in the data structure via the `/proc` filesystem.

2.1 Creating the data structure

You must create a data structure that keeps track of all the processes for each user ID. This data structure should be updated whenever a new process is created, an existing process is destroyed, or an existing process changes owners.

2.2 Creating the `/proc` files

You must create a directory in `/proc` that holds a set of files. One file should be created for each user in the above data structure. The name of the file should be the uid of that user. Each file should be readable, but not writeable since we want to query the values in this data structure–not change them. When a file is read, it should report the process IDs of each process owned by the appropriate user. This report should separate the IDs by a single space.

3 Helpful hints

Use your code from your previous assignments. They were specifically designed to help you know how to approach this assignment. They gave you the information to create the `/proc` files, create new source in the kernel, and use the functionality from a userspace program.

Do as much development as you can in userspace since you have access to a much more robust set of tools. It is also easier to recover from an application crashing than from the kernel crashing. When you think you've got things working, then you can move on to putting the functionality in the kernel. You will probably have things working correctly and so you won't have to deal with random crashes because of faulty code.

Do incremental development within the kernel so that you don't depend on the whole thing working before you start testing. It is too difficult to determine

where problems exist in the kernel to do something like this. Instead, you should do one thing and see if the kernel did that correctly. Then add something else and test again. This will greatly increase you chances of success.

4 Constraints

There are no new constraints from this assignment from the previous assignment. You must write this in C and have it work on my laptop, but you won't be doing anything kernel version specific so things should just work.