Exercise 25.2: Changing the Maximum Process ID

The normal behavior of a **Linux** system is that process IDs start out at `PID=1` for the **init** process, the first user process on the system, and then go up sequentially as new processes are constantly created (and die as well.)

However, when the PID reaches the value shown in `/proc/sys/kernel/pid_max`, which is conventionally 32768 (32K), they will wrap around to lower numbers. If nothing else, this means you can't have more than 32K processes on the system since there are only that many slots for PIDs.

1. Obtain the current maximum PID value.
2. Find out what current PIDs are being issued
3. Reset `pid_max` to a lower value than the ones currently being issued.
4. Start a new process and see what it gets as a PID.

Solution 25.2

In the below we are going to use two methods, one involving **sysctl**, the other directly echoing values to `/proc/sys/kernel/pid_max`. Note that the **echo** method requires you to be root; **sudo** won't work. We'll leave it to you to figure out why, if you don't already know!

1. ```
   $ sysctl kernel.pid_max
   $ cat /proc/sys/kernel/pid_max
   ```
2. Type:
   ```
   $ cat &
   [1] 29222
   $ kill -9 29222
   ```
3. ```
   $ sudo sysctl kernel.pid_max=24000
   $ echo 24000 >  /proc/sys/kernel/pid_max # This must be done as root
   $ cat /proc/sys/kernel/pid_max
   ```
4. ```
   $ cat &
   [2] 311
   $ kill -9 311
   ```

Please Note

Note that when starting over, the kernel begins at PID=300, not a lower value. You might notice that assigning PIDs to new processes is actually not trivial; since the system may have already turned over, the kernel always has to check when generating new PIDs that the PID is not already in use. The **Linux** kernel has a very efficient way of doing this that does not depend on the number of processes on the system.


   ```