1 Purpose

The purpose of this assignment is to learn some details about how the Linux kernel stores information about its running state. Since we don't know anything about the kernel organization at this point, we will use the `/proc` file system to provide this information.

2 Exercises

This assignment will be broken down into four parts. The first part will be to produce information about a Linux system manually. The remaining three parts will be to write a program that does this for you.

2.1 Info by hand

From a machine that is running the 2.6 kernel, use the `/proc` file system to determine answers to the following questions. With the answers, include the location (file and line) that provided the information for your answer.

1. CPU model number and name, and family
2. Exact Linux version
3. How long (in seconds and converted to days, hours, and minutes) has the machine been running since it was last rebooted?
4. How much time (in USER_HZ) has the CPU spent in user mode? user mode with low priority? system mode? idle task?
5. What are the load averages over the last 1 minute? 5 minutes? 15 minutes?
6. How much RAM does the OS belive the hardware has? swap space?
7. How much RAM is available for use? swap space?
8. How many disk reads have been issued? writes?
9. How much time (in milliseconds) has been spent reading? writing?
10. How many context switches has the kernel performed?
11. How many processes has the kernel created since boot?
12. How many processes are currently available to run?

2.2 Standard output

Write a simple program that outputs the following information when it is run. You should get the information from the same location as you did in the previous section.

1. CPU model number and name, and family
2. Exact Linux version
3. How long (converted to days, hours, and minutes) has the machine been running since it was last rebooted?

2.3 Stats

Modify the program from the previous section. Normally it should output the same information, but if the command line option `-s` is passed to it; then it should add the following information. Again, you should get the information from the same location as you did in the first section.

1. How many disk reads have been issued? writes?
2. How much time (in milliseconds) has been spent reading? writing?
3. How many context switches has the kernel performed?
4. How many processes has the kernel created since boot?
5. How many processes are currently available to run?

2.4 Load stats

Modify the program from the previous section. It should output the same information unless the command line option `-l` is passed to it. The `-l` option takes two additional parameters. The first is a sampling interval (s) and the second is a total time interval (t). Starting at time 0, the program should output the following information. After every s seconds, the program should output the new values for the same information. This should repeat until time t.

1. What are the load averages over the last 1 minute? 5 minutes? 15 minutes?
2. How much RAM does the OS belive the hardware has? swap space?
3. How much RAM is available for use? swap space?

3 Helpful hints

The `/proc` filesystem has a ton of files. Most of the files in the top directory are readable by anyone. Each process has a subdirectory that contains specific information to that process (hence the name of the filesystem). You should be able to find all the information you need for this assignment in those top-level files.

The filesystem is heavily documented (like much of the Linux kernel) in the form of man pages. The man page you're interested in is `man proc`. If you want more information about man pages, you can look at the `man` man page.

4 Constraints

There are some programs that interpret the `/proc` filesystem for you (e.g., `procinfo`). Don't use them. The point of this assignment is to learn about the filesystem on your own.

You may use any language available to you on the CSS network. If you have time, I would suggest trying C since you might like to get practice using it before the next assignment which must be done in C. Otherwise, languages like Perl that work well with manipulating text files will probably get the job done most quickly.

I will test the program on my machine running the 2.6.14 or 2.6.15 kernel. It might be good to test if you program works on multiple computers with different setups so that you are sure it will work on mine.