Due Monday, 1999 December 06.

1 Overview

The project is to write a program collecting and displaying computer and operating system statistics. This is a good review of several topics we have covered in this course.

2 Specifications

The project is to write a program that collects and display the following categories of computer and operating system statistics: cpu specifications, interrupt queue statistics, systemwide memory use statistics, kernel statistics, and process information.

More specifically, your program should display

cpu: for at least one of the processors, display

- processor number
- processor name
- cpu MHz
- cache size

interrupt queue: for each interrupt queue, display

- the number and name of queue
- number of interrupts

memory use: (systemwide)

physical memory: total available, used, free, size of shared, size of cached memory in bytes xor kBs

swap memory: total available, used, free in bytes xor kBs

page movements: number of pages the virtual memory system has paged in and out and the number of swap pages in and out since system boot

kernel statistics:

- kernel version
- boot time and time since booting
- time spent in user mode, user mode with low priority, system mode, and idle task
- load averages for last one, five, and fifteen minutes (Note `/proc/loadavg` also returns the number of tasks currently running, the total number of tasks, and the last pid. Ignore these.)

process information: for the user-specified process, display

name and IDs: process ID (pid), executable name, user ID, user's name, process state (running, sleeping but interruptible, sleeping uninterruptible, zombie, stopped), parent's pid, entire process command line

running times:

- starting time of process
- total cpu running time of process
- total user-mode cpu running time of process
- total kernel-mode cpu running time of process

virtual memory use: virtual memory size, number of pages in physical memory, maximum number of bytes in physical memory

virtual memory faults: numbers of minor and major page faults (A minor page fault does not require loading a page from disk.)

Your program should also have the capability to display a list of processes with enough information displayed that the user could determine the process ID of an interesting process.

Your program need only be concerned with real user IDs, not effective user IDs. To convert a uid to a name, you can use the `getpwuid(3)` command.

Your program need only work for Linux 2.2 or higher on a computer science machine with one processor. Please format times in understandable formats, e.g., *hh*:*mm*:*ss*; consider using library functions. Feel free to ignore fractions of seconds when appropriate.

Implement the user interface as you see fit. For example, you can use FLTK or some other toolkit to wrap the program in a window with menus. Alternately, you can write a text-based program using command-line arguments to specify behavior or even directly query the user. It's your choice.

The Fast Light Tool Kit is a GNU Library General Public License C++ tool kit to "wrap windows" around programs running under X, OpenGL, and Win32. Parsing command-line arguments can be simplified using the Gnu Libc argp interface. See the info pages for Gnu's libc, then choose Process Startup, Program Arguments, Parsing Program Arguments, Argp.

3 Obtaining the Information

There are several different ways to obtain the information to display:

- Use system calls to obtain the information.

- Use Linux's `/proc` directory. The pseudofiles in this directory contain computer and operating system statistics. Since many of these files contain timely information, they are created when needed. For example, when process 273 starts, a directory `/proc/273` is created. Thus, obtaining the information can be as simple as reading and parsing the appropriate file in the appropriate directory.

  Although the `/proc` directory is one of Linux's contributions to the art of operating system implementation, it is not well documented. Information can be found at /usr/src/linux-2.2.5/Documentation/proc.txt (also available here) and in the proc(5) manual page. When features are not documented, it can be useful to resort to just `cat`ing the desired `/proc` files.

- Use the more portable LibGTop library. This library is designed to be a portable way to extract system specific data about a computer and its operating system(s). Version 1.0.1 is available on the Janus computers; version 1.0.6 is available via the WWW. (Version 1.1.12 right now is only for the Solaris operating system.) Since the library is new, it is likely to have bugs, but, if it works, your code will be portable to other operating systems. (I could not figure out the necessary loader flags to use the library.)

3.1 Ethics

Reading through the source code of Linux (or other operating systems') and of application programs with similar functionality for examples how to write the program is considered unethical and will probably increase the amount of time necessary to complete the assignment.

4 Programming Tips

Use any programming language you desire. In fact, perl may be the best language for this assignment.

When programming, be sure to turn on all compiler warnings possible to illuminate as many issues as possible. This will greatly increase the possibility your program is correct and portable. When using `gcc` and `g++`, use the `-Wall` option. I do not know what the Java options are.

To minimize your workload, use as many library functions as possible. Depending on the language you use, information is available on UNIX manual pages, info pages (use the UNIX command `info`), WWW pages, and books. For more specific information, come see me, telling me for what you desire, and what language you are using.

Program and test the various pieces of your code before adding new features. This helps narrow down the region where errors can be occurring.

To check your program's correctness, compare your program's output to that of `free(1)`, `top(1)`, and `gtop(1)`.

5 Grading and Logistics

You can work in groups of up to three people. Grades will be assigned primarily according to correctness and functionality. Adequate documentation of the project and the code will assist in assigning partial credit (but, again, grades will primarily be assigned according to correctness and functionality).

Here is a possible point distribution:

compilation, no memory leaks, etc.	5–15 points
easily understood user interface	10–20 points
functionality and correctness	65–85 points
total	100 points

We do not have time for grading to be interactive. Just make sure your program works.

5.1 Submission

Use anonymous ftp to atlas03 to submit your program's source code. Submit to ftp://atlas03.cs.trinity.edu/pub/joldham/4320/pp3/. If you need to package together several files, use `tar(1)`.

Here is one way to submit your program:

1. ftp the material.

   ```
   ftp atlas03.cs.trinity.edu
   ```

   Use "anonymous" as your password.

   ```
   cd pub/joldham/4320/pp3
   put <source-code>
   bye
   ```