CPE453 Laboratory Assignment #1
The CPE453 Monitor

Michael Haungs, Spring 2011

1 Objective

In this assignment, you will create a program that can monitor various statistics on a process and the entire system. Performance monitoring programs are invalUble in assessing both operating system and user application changes. They can help you find bottlenecks in the system and help you maximize your effort to optimize code. In fact, you will find that this program will be useful in helping you complete Lab #4.

2 Resources

You should read the following Linux manual pages:

- proc: process information pseudo-filesystem.
- fork: peforms process creation.
- exec: execute a file.
- waitpid: wait for a process to change state.
- usleep: suspend execution for microsecond intervals.
- malloc,free: allocate and free dynamic memory
- time, ctime: date and time functions

You should look at the following files:

- /proc/stat
- /proc/meminfo
- /proc/diskstats
- /proc/loadavg
- /proc/pid/stat
- /proc/pid/statm
- http://lxr.linux.no/source/Documentation/iostats.txt

3 Assignment

You should implement your monitor in stages. Make sure that each stage is functioning properly before continuing on to the next stage.

3.1 Stage 1: Input Commands

Your monitor program takes up to four parameters. The first is an optional parameter that indicates whether or not the program will monitor systemwide statistics. The second parameter is the name of the executable you want monitored. The third parameter is the length of the interval, in micoseconds, the program will use to poll job and system statistics. The last parameter is the name of the log file your program will use (or possibly create) to send all of its output to. Here is the usage:

usage: mond [-s] <executable> <interval> <logFilename>

Your program must record process and conditionally systemwide statistics every "interval" microseconds until the executable program exits.

Here are some examples of valid commands:

```
mond -s ls 100 logFile
mond myBashScript 10 statData.txt
```

3.1.1 Timestamps

All of the messages you print from your *mond* program will be proceeded by a timestamp. In the rest of this document, you should realize that all timestamps shown are dynamically generated and need to be dynamically generated in your program as well.

3.2 Stage 2: Fork/Exec Executable

Your *mond* program will fork() and exec() the executable whose name you supply on the commandline. While your *mond* program is waiting for the executable to finish[1], it will continually collect statistics on the process while it's running (every interval microseconds).

Make sure your *mond* program properly forks, executes and waits for the given executable.

3.3 Stage 3: Collect and Report System Statistics

All statics you collect on the system are found in the files "/proc/stat", "/proc/meminfo", "/proc/loadavg", and "/proc/diskstats".

Type "cat /proc/stat" to see the contents of this file. The fields in this file are defined in the *proc* man page.

For this assignment, you must extract the following information from "/proc/stat":

- Time system spent in user mode,
- time system spent in system mode,
- time idle task has spent running,
- time system spent waiting for I/O to complete,
- time servicing interrupts,
- time servicing soft interrupts,
- number of interrupts serviced,

[1]Hint: Use waipid() with the WNOHANG option.

- number of context switches,
- number of forks,
- number of processes in a runnable state, and
- number of processes blocked waiting for I/O.

Type "cat /proc/meminfo". Read about the information in this file in the *proc* man page. You must extract the following fields from this file:

MemTotal:, MemFree, Cached:, SwapCached:, Active:, Inactive:

Type "cat /proc/loadavg". You must extract the following fields from this file:

1 minute job queue average, 5 minute job queue average, 15 minute job queue average

Finally, type "cat /proc/diskstats". You'll see a lot of information for different storage devices, both real and virtual. You are only interested in the line with "sda" in it. You must extract the following information from that line on sytem disk IO:

# of reads completed, # of disk sectors read, # of milliseconds spent reading, # of writes completed, # of disk sectors written, # of milliseconds spent writing

Further explanation of the information contained in "/proc/diskstats" can be found in:

http://lxr.linux.no/#linux+v2.6.35.9/Documentation/iostats.txt

You must report these statistics every "interval" microseconds, when monitoring the system, and display them in the following way:

```
[Mon Feb 28 19:44:55 2011] System  [PROCESS] cpuusermode 130620 cpusystemmode 94358
idletaskrunning 23477368 iowaittime 62083 irqservicetime 10 softirqservicetime 1965
intr 15654894 ctxt 49559933 forks 32494 runnable 5 blocked 0 [MEMORY] memtotal 508484
memfree 46728 cached 206800 swapcached 14864 active 185928 inactive 225876 [LOADAVG]
1min 0.980000 5min 0.460000 15min 0.360000 [DISKSTATS(sda)] totalnoreads 159139
totalsectorsread 4906518 nomsread 1322808 totalnowrites 169973 nosectorswritten 5202736
nomswritten 594440
```

Note: the "forks" field corresponds to the "processes" field in "/proc/stat".

3.4 Stage 4: Collect and Report Job Statistics

All statistics you collect on individual processes are found in the file "/proc/pid/stat" and "/proc/pid/statm". Type "cat /proc/pid/stat" or "cat /proc/pid/statm", where you substitute "pid" with an actual number representing a running process, to see the contents of these files. The fields in these files are defined in the proc man page. This link shows you the actual Linux kernel code that generates the contents of the "/proc/pid/stat" file:

http://lxr.linux.no/linux+v2.6.38/fs/proc/array.c#L360

For this assignment, you must extract the following fields from "/proc/pid/stat":

comm, state, minflt, majflt , utime, stime, priority, nice, numthreads, vsize, rss

Also, you must extract the following fields from "/proc/pid/statm"

size, rss, share, text, data

The definitions of these fields (as well as there place in the file and type) are given in the proc man page.

You must report these statistics every "interval" seconds and display them in the following way:

```
[Mon Feb 28 19:44:55 2011] Process(32493)  [STAT] executable (ls-R) stat S minorfaults
382 majorfaults 0 usermodetime 0 kernelmodetime 0 priority 20 nice 0 nothreads 1 vsize
4964352 rss 269 [STATM] program 191 residentset 269 share 230 text 191 data 58
```

3.5 Stage 5: Test Test Test

First, scan your output and make sure all the values seem reasonable. Then, run a number of processes, like Firefox, and compare your log results with things you see from manuallying inspecting the /proc files. Third, write some simple C programs and make sure your log files report what you would expect to see.

4 Robustness

Ok, maybe this didn't deserve a whole section, but I just wanted to remind you to perform all necessary error handling. Your program should die gracefully in the case of an unexpected event (aka no segfaults). You may, however, assume the input will be formatted correctly.

5 Examples

For the following the command:

```
mhaungs@ubuntu:~/Courses/453/Lab1$ mond -s ls-R 10 logfile
```

Note: *ls-R* is a bash script which just runs "ls -R".

Here is the corresponding log file:

http://www.csc.calpoly.edu/~mhaungs/courses/CSC453/labs/1lab/logfile

Deliverables

Submit your work to me by April 7 before midnight. You need to submit the following:

1. All ".c" and ".h" files associated with your solution.
   (a) Each file should start with a comment formatted like this:
   http://users.csc.calpoly.edu/~mhaungs/courses/CSC453/comment_hdr.htm
2. A Makefile (must be spelled like that) that will compile your entire project when "make" is ran and produce an executable named *mond*.
3. A README (again, must be spelled exactly like that) file that describes (1) what you got working, (2) test cases that will work with your code, (3) Any bugs in your code, and (4) a bried description of each field you collected for individual jobs and the system. Please put both team member names (first and last) at the top of the README file.

Use the *Digital DropBox* tool in Blackboard to submit your files:

1. Create a zip archive of your work.
   (a) Rename the zipped folder in the following way:
   Student1Lastname_Student2LastName_Lab1.zip.
2. When using the Digital Dropbox tool, be sure to use the *Send File* button and not the *Add File* button to send me your compressed folder.
   (a) Only select me, Michael Haungs, as the recipient of your send.

GRADING NOTE 1: You must name all your files exactly as I specified above. Your output must exactly match the examples I have given above.

GRADING NOTE 2: Be sure to check the class forums regularly as that is where I'll post any changes/corrections to this assignment.