CPSC 424/624 HW #1
Spring 2016
You must work on your own.
You can make use of freely available bash script examples you find on the Internet.
Last revision: 1/31/2016

Submission instructions: Please submit a PDF with all your answers including a listing of the bash scripts from problem #3 to 'turnitin' on blackboard. Also, please submit using 'handin' a tar.gz file that contains your updated ex2 code and your two bash script files.

Grade weights (Q1/2/3)

- Undergrad: 20/40/40
- Grad: Same but the two parts to Question 1 are graded 10/10

Question 1 Read the paper "The Reincarnation of Virtual Machines" by Rosenblum. In one brief paragraph, summarize this paper.

Question 1-A In addition and only for CPSC624 students. Please read the SMP VM Scheduling paper. Please provide a very brief writeup that address the following:

- What is the problem addressed by the paper ?
- They mention the term workloads, what does this mean in the context of the study?

Question 2. Download the ex2.tar.gz package. Last year the class obtained data sets produced by a version of the program that timed how long a sleep system call actually slept. The data sets were then analyzed/visualized. Some students used excel and others used the matlab program provided (plotDataPDF.m).

Instead of focusing on the variability with the sleep time, you are to obtain timings for how long it takes to compute a checksum on a block of data. The checksum code is provided (and it's already linked in the Makefile). In particular, you will be conducting experiments designed to explore the potential impacts of scheduling as the size of the block of data varies from small to large (i.e., which allows us to control aspects of the memory footprint used by the program).

The inner loop might look as follows:

```
time = getTime();
csumResult = csum(bufferPtr, bufferSize);
time = getTime() - time;
printf("%3.7f 0X%xd  \n",time,csumResult);
```

Your analysis should include a comparison of the probability distributions associated with data sets. Additional analysis might be required. The following will get you started:

- Copy the exampleUnixTime.c file to hw1.c
- Modify the makefile to build target hw1which produces an executable hw1.
- Add the following program parameters:
    - iterationCount (unsigned int) : specifies number of iterations
    - dataBlockSize (unsigned int) : specifies the size of the data block that is involved.
- On entry, the program should malloc a buffer of size dataBlockSize and initialize all bytes to a random number (between value 0 and 255).

The executable must be called hw1 and it must accept the 2 parameters described. By redirecting standard out to a file (hw1 1000000 1000000 > hw1.dat) your program will produce a dataset in a specified file.

A reasonable setting for iterationCount is 1000000 although you might need to decrease this for very large dataBlockSize values (otherwise it takes too long to run). Run your program three times using the following parameters:

- 1000000 4
- 1000000 256
- 1000000 65535

Plot the PDF of each of these data sets (using three separate graphs). In addition to showing the PDF, indicate statistical summary results for each data set including the mean, median, and standard deviation. One would expect the statistics to scale with the scaling factor. Show evidence that this appears to be true or false.

In prep for Question 3, consider the impact of running multiple instances of the hw1 program concurrently, for cases including when your VM is configured to run on a single CPU/core or multiple CPUs/cores. First, determine if your VM can run on multiple CPUs/cores. Please run 'apt-get install sysstat' and get familiar with the mpstat utility program. Try 'mpstat –u –P ALL 2'. This shows the CPU utilization for each CPU/core every 2 seconds. The final component of question 2 is to run two instances of the hw1 program concurrently. Use the hw1 parameter settings (100000, 256). Open three shell sessions.

- Shell 1, issue './hw1 1000000 256 > hw1Q2-shell1.out
- Shell 2, issue './hw1 1000000 256 > hw1Q2-shell2.out
- Shell 3, issue 'mpstat –u –P ALL'.

Record the output from the mpstat command. And list the statistical summary for the two data sets. You do not have to plot the PDF of these results.

Question 3. You will be using your program from the previous questions to explore in more detail possible behaviors or impacts of the Linux OS scheduler. You must develop and several bash scripts to automate the analysis. I would expect at least 2 are required. One script (please call it runHW1.sh) automates running the following experiments (and creates the datasets). The second script (please call it analysisHW1.sh) analyzes all datasets associated with the experiment and creates a single results file. The challenge is designing a script that does 'everything' you need for your analysis. You are allowed to develop a top level script, perhaps one for each experiment if necessary. This script could call runHW1.sh any number of times to carry out the experiment. You can also develop and use a top level script for analyzing results.

We identify 3 experiments. Each experiment involves generating 20 data sets. We define an 'iteration' to mean running the program with a particular set of parameters creating a data set. In each experiment, you are to vary the dataBlockSize as follows: $2^i : i \in \{1,2, \dots ,20\}$ for each run. The i variable represents the 'scaling parameter' for each iteration. This technique allows us to easily vary the dataBlockSize over orders of magnitude.

Experiment 1: On your VM, run your program 20 times as described above. Perform each iteration consecutively. Obtain 20 data sets.

Experiment 2: Repeat Experiment 1 but perform two iterations in parallel. In other words, your method will attempt to run the 20 iterations using 2 CPUS/cores. The challenge is how to do this! In Question 2, we achieved this by running two instances of the program concurrently (using two different shells). I leave the details up to you. One simplification is that you are not required to fully utilize the 2 CPUs throughout the entire time it takes to run all 20 iterations. Since each iteration requires different run times, it will be difficult to 'load balance' the work load perfectly over both CPUs. It is likely (and perfectly ok) for perhaps 8 iterations to run on one CPU/core and 12 iterations to run on the other. And it might turn out that the 8 iterations on CPU/core #1 finish before the 12 iterations on CPU/core #2. Our reasons for trying to load balance 20 iterations over 2 CPUs/cores is simply to give us further data to explore the behaviors and dynamics of the Linux scheduler.

Experiment 3: Repeat Experiment 2 however increase the priority for the runs on one core/CPU by running the program using the nice command (e.g., 'nice –n -10 ./hw1 1000000 8 > hw1.3') AND then decrease the priority for the other runs (e.g., 'nice –n 10 ./hw1 1000000 8 > hw1.3').

The requirements for runHW1.sh:

- The script should automate running of a particular experiment (or to the degree that makes sense).
- The required arguments:

- iterationCountP: Specifies the iterationCount parameter for the hw1 program.
- firstScalingP: Specifies the starting scaling parameter. For example, if this is 4, then the first iteration of the hw1 program would have the dataBlockSize set to 16 octets.
- numberRuns: This specifies how many data sets will be generated for the experiment.
- resultsDirectory: Specifies the directory that will hold the data files. If this directory does not exist it will be created.
- baseDataFileName: This specifies the base filename for all of the output data files. For example, if 'hw1Data' is specified, then all results files would be names 'hw1Data.x' where the x indicates the scale parameter. Any file that exists in the resultsDirectory that begins with the specified base filename should be deleted before the first iteration runs.

For example, issuing 'runHW1.sh 1000000 1 10 Results hw1' will perform 10 runs with the dataBlockSize ranging from 2 to 1024 octets. The results files are ./Results/hw1.1 through ./Results/hw1.10.

To run two iterations at a time, one approach is to have runHW1.sh call two other scripts, each running roughly half the work load, but to run in 'background' mode. As an example, to run hw1 in a background mode, 'hw1 1000000 2 > hw1.1 & ' This gives your script some control to 'fork' a child process to run a program. There are more sophisticated ways to do this (there are packages that extend bash with granular multicore support).

An alternative approach is to have a top level script run effectively two instances of runHW1.sh concurrently, with each runHW1.sh running a complete set of iterations. As an example, let's say you ran Experiment 1 varying the scale parameter from 2 25 (so the 24th iteration would be hw1(1000000, 33554432)). Assume the two instances run in different directories (Instance1 and Instance2). The top level script might include:

- ./runHW1.sh 1000000 2 25 ./Exp2/Instance1/Results hw1 &
- ./runHW1.sh 1000000 2 25 ./Exp2/Instance2/Results hw1 &
- And if you want the script to wait until both programs complete, the following should work:
    - while [1]; do fg 2> /dev/null; [$? == 1] && break; done

We leave it to you to figure out the details. The sophistication of the script is not that important. Instead, keep your focus on the main objective of the question: You will be using your program from the previous questions to explore in more detail possible behaviors or impacts of the Linux OS scheduler. By automating the procedures, you should be able to conduct a more thorough and in depth analysis than what was done just in question #2.

The requirements for analyzeHW1.sh:

- The script analyzes the results from one particular experiment.
- The required arguments
    - baseDataFileName: This specifies the base filename for all of the output data files. For example, if 'hw1Data' . The file names end in '.x' where the x indicates the scale parameter.
    - resultsDirectory : indicates the directory where all results files reside.
    - resultsFile: Specifies the name of the file that will contain final results generated by this script. Results for a data set should have the following:
        - First line indicates the name of the experiment (e.g., Experiment 1)
        - For each data set, a new line as follows:
          iterationCount scaling parameter mean median standard deviation.
- Your script must calculate the statistics using awk.
- Plot the pdf of several data sets from each experiment. The exact number and which data set is up to you- whatever you think is necessary to show interesting results.

In your homework writeup, please list the source code of all scripts as well as the summary statistics for each experiment. Include a summary and interpretation of the results. What can we say about the predictability of the scheduler based on the data? Identify any data that appears puzzling and provide your best guess to explain the result.