U N I V E R S I T E I T S T E L L E N B O S C H U N I V E R S I T Y
j o u k e n n i s v e n n o o t k n o w l e d g e p a r t n e r

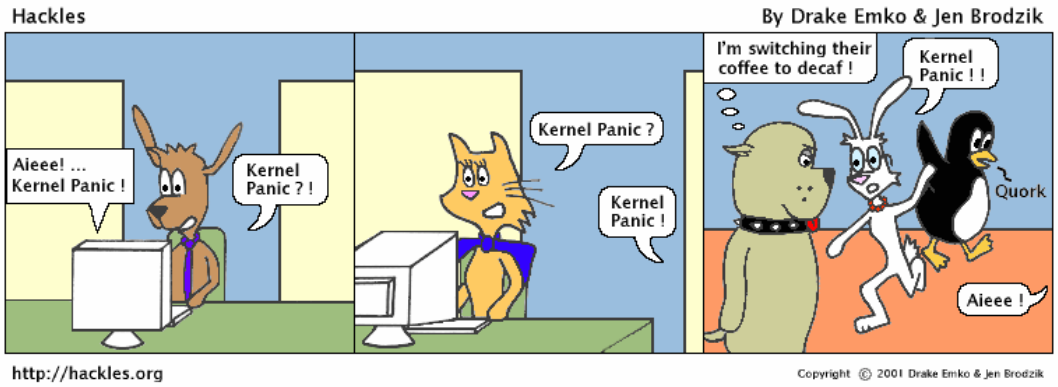
Computer Systems 414

Practical 4

2021

Aim of Practical 4:

1. To get practical experience on the Raspberry Pi and Beaglebone Black of platform debugging and performance, especially as related to compiler options.



Rules of Engagement

1. Please see the study guide for regulations regarding attendance. Part 4A of the practical to be done in the lab. Attendance is not required for part 4B.
2. Work in groups of two to three.
3. Obtain all relevant supporting documentation on learn.sun.ac.za
4. The report is to be submitted on learn.sun.ac.za. The deadline for the reports is (14:00) Wednesday, 21 April 2021. **No late submissions will be accepted.**
5. Do not forget to cite and give credit for any information reported which is not your property.
6. Google is your friend. Any information not given is left out on purpose. Search For your solution and on the internet or relevant manuals and documentation.

Assignment 4A

Refer to our previous practicals for the Beaglebone Black and Raspberry Pi setup. Use tools such as gcc as required.

Task (Use the uploaded files from SunLearn as required)

1. On the RaspberryPi, use the *primes.c* file and compile with *gcc -o <appname> <source-file>*
2. Measure the time the application takes to execute with: time ./app.
   a. All the output of the application writes to the screen. This takes long. You can shorten the time by writing to a file instead. We can do that in linux with the > (pipe) operator. Usage: ./app > app.txt ,you can see what the text file contains by copying it to Windows and viewing it.
   b. How long does the code take to execute? We can use the Linux time application to determine it. Usage: time ./app or in full combination: time ./app > app.txt.
   c. This is the standard compiler. Use optimisation of gcc: -O1, -O2 and see how the execution of the code changes. Explain your observations.
3. Use source code sprimes.c and repeat the speed tests. What changed to change the speed?
4. Use source code card.cpp to do further speed tests.
   a. This program in written in C++. You can compile it in the same way as the c-code by using g++ instead of gcc. Also use the -O3 (optimisation level 3) option, otherwise the code will take too long to execute.
   b. When you execute the program use: time ./card > card.ppm. Note the file name it is important. When you are done the file card.ppm should be 196 623 bytes. Copy it to Windows and double-click on it.
   c. Now use the following compiler flags for g++ (as options during compile) on the BBB: -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp
   d. Repeat BBB speed tests and explain your observations.
5. Write up your findings in a report and hand in on SUNLearn before the cutoff date.

Notes about execution time

The output of the time function can be interpreted as follows:

- real = total time of EVERYTHING (including overhead of printing to the terminal)
- user = time your code took to execute
- sys = time linux operating system took to do its stuff

Assignment 7B

Complete the following problems and hand in with your report on SUNLearn

1. Wolfe Q4-2, 4, 14
2. Wolfe Q4-28
3. Wolfe 3rd Edition Q8-4, 12