Final Project (Homework/Final Exam) Instructions

Contents:

DUE DATE:

Thursday, June 9, 11:59 p.m. NO LATE SUBMISSIONS!

Project description:

Your project title will be "Introduction to OpenACC," with contents as implied by that title. OpenACC is a system with syntax similar to OpenMP, which has multiple possible back ends to compile to, similar to Thrust. Here are the details:

- The user is assumed to: (a) be a good programmer, (b) understand computer systems, e.g. environment variables, caches, etc., and (c) have background in OpenMP and CUDA.
- A few open source compilers exist for OpenACC, but at the present time the best is probably Omni. It is possible to use gcc, with proper configuration, but the latter condition may be onerous. I've placed Omni in **~matloff/Pub** on CSIF; it compiled "right out of the box" for me, no tweaking needed at all. See example below.
- Your paper must include at least two full examples of OpenACC code, drawn from our homework programming assignments. In lieu of one of them you might choose an example from our textbook, but only with prior approval by me.
- *The quality of your project will depend largely on how adventurous and curious you are*. If you set out to write something really useful, then it will be so.
- Unless you object, I will display the best project papers on our Web page.

Some general issues:

- As you know, this Project counts both as a Homework assignment and as a substitute for an in-class Final Exam. All in all, it will require time similar to approximately 1.5 Homework assignments. **START EARLY!** A lot of things you think will be easy, postponable to the last minute, may prove to be much harder than you think.

  A typical good-quality report will run, say, 4-5 pages, excluding code listings, graphs and figures. But there is no magic formula; a really excellent report might be shorter than this, and a mediocre one might be much longer.

- **Please note the open-ended nature of the Project**. As you can see, these Project specs don't take the form of "Step 1, do this, Step 2, do that..."

Writeup:

- You are required to use LaTeX to write up your report, with the output being a PDF file. I have a quick tutorial. Also, all the **.tex** files for our own ECS 158 textbook are in **http://heather.cs.ucdavis.edu/~matloff/158/PLN** ; by comparing output to input, you can see how to do lots of things in LaTeX .

  Make sure that your graphs are in either **.png**, **.jpg** or **.pdf** format. You may find the following code useful:

```
# prints the currently displayed graph to the
# file filename; suffix can be "pdf", "png" or "jpg"
pr2file <- function (filename)
{
    origdev <- dev.cur()
    parts <- strsplit(filename,".",fixed=TRUE)
    nparts <- length(parts[[1]])
    suff <- parts[[1]][nparts]
    if (suff == "pdf") {
        pdf(filename)
    }
    else if (suff == "png") {
        png(filename)
    }
    else jpeg(filename)
    devnum <- dev.cur()
    dev.set(origdev)
    dev.copy(which = devnum)
    dev.set(devnum)
    dev.off()
    dev.set(origdev)
}
```

  I recommend that you use R's **ggplot2** package to generate your graphs; see my tutorial at the end of my ECS 132 book. In any case, all graphs and figures must be input by your **.tex** file, and appear within your report itself.

- It is of the utmost importance that your report be of **PROFESSIONAL QUALITY**. This means:

  - Clarity! This is very difficult, but quite achievable if you work at it. Ask friends not in the class to read it and tell you if it makes sense. Keep in mind that if you do the writeup at the last minute, the

biggest casualty will be clarity.

- Correct spelling, grammar and usage!
- Professional-quality presentation! It doesn't necessarily have to be fancy, but should not look sloppy.
- To be avoided like the plague:
    - Avoid vague use of the word "it." Example: "It is embarrassingly parallel"--exactly WHAT is embarrassingly parallel?
    - "If you wouldn't eat it, don't serve it." This used to be a sign to employees in fast food restaurants, admonishing them not to serve sloppily cooked food. In our case here, "If you wouldn't find the premise of an analysis credible, don't present that analysis."
- You must include an appendix (use the LaTeX **\appendix** command), with the following contents.
    - Your code, clearly commented, using the $\LaTeX$ **listings** package. Also include in a text portion (i.e. NOT in your code) an explanation of the highlights of your code.
    - Include a brief account of "who did what" in this Project among the various group members. Not everybody need have done exactly equal work, but something of the nature "Transported a USB key to campus" doesn't count as a contribution. :-) (A group actually said this in their report one time about the contribution of a certain member.)

- Adhere to the UCD Code of Ethics. Having persons outside your team do any of the work, including the writing, or engaging in paid help of any kind, is unacceptable.

IMPORTANT SUBMISSION DETAILS:

- Your group submits just ONE copy of the report.
- Submit your report, including all files (**.tex**, **.pdf**, **.c**,**.R** etc.) to my **handin** site on CSIF, directory **158project** (see Syllabus). The name of your file must be of the form **email1.email2....tar**, where each **emaili** is the UCD e-mail address of group member i, e.g. **bclinton.gbush.bobama.tar**. Note the periods separating fields. Your **.tar** file must contain only regular files, NO SUBDIRECTORIES.
- Make sure that all partners' names are on the report, and that the e-mail addresses in the file name are EXACTLY the official UCD e-mail addresses for the students. These are the addresses at which you have been receiving your Quiz results. DON'T RISK HAVING A TEAM MEMBER FAILING TO GET CREDIT FOR THE PROJECT.

Grading criteria:

- Technical content of the work.
- Adherence to instructions.

- Professional quality of the work.

Omni example:

```
#include
#define N 2000000000
#define vl 1024

int main(void) {
  double pi = 0.0f;
  long long i;
  #pragma acc parallel vector_length(vl)
  #pragma acc loop reduction(+:pi)
  for (i=0; i
```

This function, a test from one of the OpenACC tutorials, calculates π. Compile and run:

```
pc45:~% ompcc -o picpu pi.c
pc45:~% ompcc -o pigpu -acc pi.c
pc45:~% time picpu
pi=3.1415926536
12.875u 0.001s 0:12.93 99.5%    0+0k 1056+0io 4pf+0w
pc45:~% time pigpu
pi=3.1415926536
0.873u 0.228s 0:01.71 63.7%     0+0k 656+56io 2pf+0w
pc45:~%
```