First thoughts: SCD programming talent assessment

Marc Paterno

2019-10-10 SCD Projects meeting

Assessing SCD programming skills

- I do not think the “skills survey” data collected some while ago contains much useful information.
- Levels of expertise: how to decide?
  - expert: specific skills for each language (C++, Python) that are critical [see later slides]
  - familiar: “Have you have used the language in question for real work, not just for homework or toy problems.”
- Propose to send email to all SCD members, and ask department heads to collect and affirm responses. Department heads should delegate to group leaders as they see fit.
- Will assess the results: were these the right questions to ask? May need follow-up.

Python: what indicates expertise?

Please feel free to suggest additional or alternate indicators of expertise.

- Have you been lead, or co-lead, designer of a deployed application?
- Have you worked on a python project that is deployable with one of the standard Python deployment tools?
- Which of the common Python testing systems do you regularly use?
- Have you made significant use of the Python metaclass facilities?
- Have you made significant use of any Python profilers?
- Have you made significant use of `numpy`?
- Have you made significant use of `cython` or `numba`?
- Have you made significant use of `virtualenv`?
- Have you worked on concurrent applications?
- Have you worked on high-performance networking applications?
- Have you written a Python module in C or C++?
- Have you mentored others in Python?

C++: what indicates expertise?

Please feel free to suggest additional or alternate indicators of expertise.

- Have you contributed class or function templates to a library used by others?
- Do you know how virtual functions are commonly implemented by compilers?
- Have you made significant use of variadic templates?
- Have you made significant use of template metaprogramming?
- What C++ testing systems do you regularly use?
- Do you have experience with distributed programming?
- Do you have experience with multithreaded programming?
- Have you made significant use of RAII?
- Have you made significant use of any profilers in C++ code?
- Have you mentored others in C++?

Other things to assess (1 of 2)

- These are considerations for some later date
- Other languages are, or have been, in use at the lab. Do we care about:
  - Java
  - Go (used in at least one project)
  - shell scripting languages
  - Perl
  - Ruby (note that *Redmine* is a Ruby *rails* application)

Other things to assess (2 of 2)

- There are many critical tools in wide use
  - `git`
  - `cmake`
  - Linux tools (*e.g.* linker, debugger)
- Not a programming language, but an important skill: database use and schema design.
- For `git` and `cmake` in particular, sometimes 15 minutes of help from a real expert can avoid many hours of wasted effort. Having a list of *go-to* experts in various tools might be useful. Is there a good technology for allowing people to ask questions to the right group of experts, without getting them overwhelmed? Mailing lists have not been great for this. Would slack channels be any better?
- Do we care about web programming?

What comes after?

- Under previous leadership, it was made clear that Fermilab does not do training in programming. However, Liz organized the first beginner's C++ class we have had in years. What, if any, additional training might we consider?
- My personal view:
  - The course given by Glenn Downing was good-to-excellent for people just getting started. It is good to repeat this periodically, maybe yearly.
  - Finding trainers for more advanced level material is harder, and likely to be less effective except when very targeted (*e.g.* a class for learning Kokkos).
  - Is it feasible to send junior people to C++ or Python programming conferences? Such conferences contain much excellent material, and can generate much enthusiasm for learning.
  - There are many useful hour-long talks from such conferences available online. I would like to try arranging some sessions in which a group of people view, and discuss, such a session. I think we would need to try a few to gauge both interest and value.
- What about instruction in related technologies specific to our environment:
  - What about SpackDev?
  - What about future framework tutorial material?