Programming Leftovers

By *Roy Schestowitz*
Created *13/02/2020 - 4:31pm*
Submitted by Roy Schestowitz on Thursday 13th of February 2020 04:31:04 PM Filed under Development [1]

- **Manage complex Git workspaces with Great Teeming Workspaces [2]**

  Somewhat like Python venv, but for languages other than Python, GTWS handles workspaces for multiple versions of multiple projects. You can create, update, enter, and leave workspaces easily, and each project or version combination has (at most) one local origin that syncs to and from the upstream?all other workspaces update from the local origin.

- **Why developers like to code at night [3]**

  If you ask most developers when they prefer to work, many will say their most productive hours are at night. This may be especially true for open source contributors who are contributing to projects outside of their day job (though hopefully within healthy limits to avoid burnout).

  Some like to start in the evening and work till the early hours while others get up super early?say, 4 a.m.?to get most of the programming work done before the daily grind kicks in.

  This work habit may make many developers seem like oddballs and misfits. However, there are quite a few reasons why so many programmers prefer to work during the odd hours:

- **Introducing our new Lua cheat sheet [4]**

  Not only is Lua simple in design, but it's also consistent in ways that many other languages are not. It has explicit scoping (so it's not dependent on indentation), it interfaces with C through simple wrappers, and it can accept raw C data as a data type. Lua's syntax is direct

and predictable, so once you learn a few structures, the rest is largely intuitive.

For example, the end keyword is used to close a clause, whether that clause is an if statement, a for or while loop, or a function. The table construct is the sole data-structuring mechanism in Lua, and it can be used to represent ordinary arrays, lists, symbol tables, sets, records, graphs, or trees, and it can even mimic object-oriented classes. Broad statements about Lua are plentiful, and they usually apply equally across the language. There aren't exceptions to the syntax: once you learn something, you can use that principle no matter what you're writing in Lua.

Lua is simple enough to fit on one side of a single-page cheat sheet, but we created a two-page cheat sheet for notes about syntax, data structures, important variables, and a few tricks and tips. Whether you're new to Lua or you've been using it for years, download our Lua cheat sheet and keep it handy. It'll make Lua (or at least writing it) even faster.

- **Excellent Free Tutorials to Learn Ada [5]**

  Ada is a structured, statically typed, imperative, wide-spectrum, multi-paradigm, object-oriented high-level, ALGOL-like programming language, extended from Pascal and other languages. The language was developed in the late 1970s and early 1980s. Ada is named after Augusta Ada Byron (often now known as Ada Lovelace), daughter of the poet Lord Byron.

  Ada has built-in language support for explicit concurrency, offering tasks, synchronous message passing, protected objects, and non-determinism. Ada incorporates the benefits of object-oriented languages without incurring the pervasive overheads.

  Other notable features of Ada include: strong typing, inherent reliability, modularity mechanisms (packages), run-time checking, parallel processing, exception handling, the ability to provide abstraction through the package and private type, and generics.

  Ada is particularly strong in areas such as real-time applications, low-level hardware access, and safety-critical software, as it has specialized design features, and high reliability. Most errors are detected at compile time and of those remaining many are detected by runtime constraints. While Ada was originally targeted at embedded and real time systems, the Ada 95 revision added support for object-oriented (including dynamic dispatch), numerical, financial, and systems programming. With its readability, scalability, and being designed for development of very large software systems, Ada is a good choice for open source development.

  Here?s our recommended tutorials to learn Ada. If you?re looking for free Ada programming books, check here.

- **digest 0.6.24: Some more refinements [6]**

Another new version of digest arrived on CRAN (and also on Debian) earlier today.

digest creates hash digests of arbitrary R objects (using the md5, sha-1, sha-256, sha-512, crc32, xxhash32, xxhash64, murmur32, and spookyhash algorithms) permitting easy comparison of R language objects. It is a fairly widely-used package (currently listed at 889k monthly downloads with 255 direct reverse dependencies and 7340 indirect reverse dependencies) as many tasks may involve caching of objects for which it provides convenient general-purpose hash key generation.

This release comes a few month after the previous release. It contains a few contributed fixes, some of which prepare for R 4.0.0 in its current development. This includes a testing change to the matrix/array class, and corrects the registration for the PMurHash routine as pointed out by Tomas Kalibera and Kurt Hornik (who also kindly reminded me to finally upload this as I had made the fix already in December). Moreover, Will Landau sped up one operation affecting his popular drake pipeline toolkit. Lastly, Thierry Onkelinx corrected one more aspect related to sha1.

- **Do you CI? [7]**

  Continuous integration is often confused with build tooling & automation. CI is not something you have, it?s something you do.

  Continuous integration is about continually integrating. Regularly (several times a day) integrating your changes (in small & safe chunks) with the changes being made by everyone else working on the same system.

  Teams often think they are doing continuous integration, but are using feature branches that live for hours or even days to weeks.

  Code branches that live for much more than an hour are an indication you?re not continually integrating. You?re using branches to maintain some degree of isolation from the work done by the rest of the team.

- **Using Zuul CI with Pagure.io [8]**

  I attended Devconf.cz again this year ? I?ll try and post a full blog post on that soon. One of the most interesting talks, though, was CI/CD for Fedora packaging with Zuul, where Fabien Boucher and Matthieu Huin introduced the work they?ve done to integrate a specific Zuul instance (part of the Software Factory effort) with the Pagure instance Fedora uses for packages and also with Pagure.io, the general-purpose Pagure instance that many Fedora groups use to host projects, including us in QA.

  They?ve done a lot of work to make it as simple as possible to hook up a project in either

Pagure instance to run CI via Zuul, and it looked pretty cool, so I thought I?d try it on one of our projects and see how it compares to other options, like the Jenkins-based Pagure CI.

- **Declarative Systems Take Center Stage** **[9]**

  Declarative systems and languages are different from what we typically think of as computer code. Most computer languages are imperative, not declarative. In imperative systems, code defines a series of steps to be taken which the system then executes. This is the classic programming language paradigm.

  For example, a declarative system may be told to create a virtual machine by declaring the existence of the VM as the expected state of the system.

- **This Week in Rust 325** **[10]**

- **7 Best Programming Languages for Mobile Apps Development | HokuApps** **[11]**

  Java is probably one of the most popular programming languages out there for mobile app development and is used by a number of mobile app development services. Android OS, which is a widely used operating system, is written in Java; and so, if an app developer is well-versed with Java, then they?ll be able to create all kinds of Android apps.

  Besides Android apps, developers familiar with Java can develop games, embedded space, websites, server apps, and more. Moreover, Java can either be run in a browser window or in a virtual machine that does not require a browser. This is why it?s an extremely flexible programming language in terms of reusing code.

- **Working on these skills can get you high paying jobs** **[12]**

- **ML, data analytics most sought after skills for 2020: Study** **[13]**

- **Tech skills will dominate in 2020** **[14]**

Development

Source URL: http://www.tuxmachines.org/node/134041

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] https://opensource.com/article/20/2/git-great-teeming-workspaces
[3] https://opensource.com/article/20/2/why-developers-code-night
[4] https://opensource.com/article/20/2/lua-cheat-sheet
[5] https://www.linuxlinks.com/excellent-free-tutorials-learn-ada/
[6] http://dirk.eddelbuettel.com/blog/2020/02/12#digest_0.6.24
[7] https://benjiweber.co.uk/blog/2020/02/12/do-you-ci/
[8] https://www.happyassassin.net/2020/02/12/using-zuul-ci-with-pagure-io/
[9] https://www.cmswire.com/information-management/declarative-systems-take-center-stage/
[10] https://this-week-in-rust.org/blog/2020/02/11/this-week-in-rust-325/
[11] https://www.whatech.com/mobile-apps/blog/635075-7-best-programming-languages-for-mobile-apps-development-hokuapps
[12] https://www.orissapost.com/working-on-these-skills-can-get-you-high-paying-jobs/
[13] https://www.daijiworld.com/news/newsDisplay.aspx?newsID=673289
[14] https://www.deccanchronicle.com/technology/in-other-news/120220/tech-skills-will-dominate-in-2020.html