Chapter 14: Scripting Languages

July 13, 2016

Why scripting?

- Scripting languages are strongest at asking other *programs* to do something; call it “glue” if you like.

Batch and interactive mode common

- REPL-type batch quite common

Brevity is often the case

- print "Hello world!"; is likely to work just fine in a scripting language, with perhaps a bit of syntactic sugar.
- Reification of quite a bit of user-land experience as syntax for pre-existing objects inside the language environment.

Flexible, dynamic typing is the usual case

- Coercion is quite common; also, types can become quite baroque (look at the perlio manual page, for instance)

Primary Access to system-level objects.

- io, file/directory manipulation, process management, database access, ipc, networking. . .

Pattern-matching and string manipulation

- PCRE

High-level data types

- Sets, bags, dictionaries, lists, tuples, . . .

Shells

- Truest type of scripting; consists literally of the commands to a given shell
- Globs and regular expressions
- Pipes and redirection
- Here documents
- Quoting with interpolation and without
- Functions

The crunch-bang

- Dire but neat hack

Text processing and report general

- PERL = "Practical Extraction and Reporting Language"
- Sed, stream editing
- Awk, the moribund

Python

- Unusually, it actually cares about indentation and end-of-line characters
- Structurally, well, some people really like it and some don't; fundamentally, it's not a lot different than most scripting languages, but certainly is willing to take on (and sometimes drop) new features.

Ruby

- A purely object-oriented language; the author wanted a language more expressive than Perl and more object-oriented than Python.

Extension languages

- If a software package exposes an API, then one can build around that API; while this might be external, the scripting could also incorporated into that environment. For instance, GIMP, Inkscape, and emacs all have significant extension capabilities. Some software packages have their own extension languages; others are more oriented to an API-type environment.

CGI scripting

- CGI scripting is quite simple; a process receives a http request (often moderated in some fashion, though not necessarily) and then replies with a similarily formatted response. Text -> text is a modality ideally suited to scripting languages.

CGI scripting

- PHP: a lamentable language; of all the languages we will talk about this semester, is probably the most troubled. Widely deployed, it has awkward constructions such as $==$ versus $===$.

CGI scripting

- Server-side javascript is now competing strongly with PHP; see this article.

Scripting languages and innovation

- Names and Scopes
  - Wide variety of ideas here: Perl, in its usual accommodating fashion allows about any type of scoping you like; Python and Javascript use classical nesting and lexical scoping
  - TCL has unusua methods for “external” scoping (see fig 13.18)

Scripting languages and innovation

- TCL, like Pleasant, uses strings for everything

Scripting languages and innovation

- Perl has the very unusual idea of letting the program have rather direct access to subroutine parameters as an anonymous array

Scripting languages and innovation

- Scripting languages generally have excellent string manipulation tools as built-in features; i.e., there exists language syntax for the tool rather than expressing its use as calls to library routines.

Scripting languages and innovation

- One outstanding example is Perl's support for regular expressions as part of the syntax of the language.