BANDO 24664 ALL 3

Bando n.ro 24664/2022 – Prova orale – Traccia n.ro 1

1. Il candidato inquadri sinteticamente il tema della sicurezza informatica e il suo impatto sulle procedure di amministrazione dei calcolatori e delle reti informatiche.
2. Il candidato descriva brevemente la pila ISO/OSI, illustrando sinteticamente i livelli da cui è composta e le loro funzioni.

BANDO 24664/ALL 2

Bando n.ro 24664/2022 – Prova orale – Traccia n.ro 2

1. Il candidato descriva brevemente i principali componenti di una rete locale e le loro funzioni, integrando il discorso con eventuali esperienze professionali di installazione e gestione di reti locali.
2. Il candidato descriva brevemente i principali componenti di un calcolatore e le loro funzioni, integrando il discorso con eventuali esperienze professionali di utilizzo, installazione e gestione di calcolatori.

BANDO 24664 ALL 4

Bando n.ro 24664/2022 – Prova orale – Traccia n.ro 3

1. Il candidato illustri a grandi linee caratteristiche e funzioni fondamentali di un sistema operativo, integrando il discorso con eventuali esperienze professionali di utilizzo, installazione e gestione di calcolatori.
2. Il candidato descriva a grandi linee funzionamento, scopi ed applicazioni dei sistemi crittografici.

A Conversation with Larry Wall

The creator of Perl talks about language design and Perl

Eugene Eric Kim

In both language size and popularity, the Practical Extraction and Reporting Language (Perl) is probably the biggest of the little languages. While its text- and file-processing capabilities have made it the language of choice for system administrators and web developers, Perl's virtues as a rapid prototyping tool and full-fledged application language have become apparent as well.

Given Perl's idiomatic nature, it isn't surprising that Larry Wall, Perl's creator, has a background in linguistics. Larry is no stranger to useful, free software, having written and given away programs like metaconfig, rn, and patch. His active involvement and ever-present sense of humor—which permeates his newsgroup posts, source code, and *Programming Perl* (the book he coauthored with Randal Schwartz)—is largely responsible for the community that has grown and flourished around the language.

A recipient of *Dr. Dobb's* 1996 Excellence in Programming Award and currently a developer at O'Reilly & Associates, Larry recently sat down with *DDJ* technical editor Eugene Eric Kim to discuss the state of Perl, scripting language design, and the importance of culture to a language.

DDJ: What's new in the upcoming version 5.005 of Perl?
LW: Well, other than the usual round of bug fixes, there will be support for threading and compilation down to C code. Those are the main things.

DDJ: Is Perl 5.005 what you envisioned Perl to be when you set out to do it?
LW: That assumes that I'm smart enough to envision something as complicated as Perl. I knew that Perl would be good at some things, and would be good at more things as time went on. So, in a sense, I'm sort of blessed with natural stupidity—as opposed to artificial intelligence—in the sense that I know what my intellectual limits are.

Eugene is a technical editor at DDJ. *He can be contacted at eekim@ddj.com.*

I'm not one of these people who can sit down and design an entire system from scratch and figure out how everything relates to everything else, so I knew from the start that I had to take the bear-of-very-little-brain approach, and design the thing to evolve. But that fit in with my background in linguistics, because natural languages evolve over time.

You can apply biological metaphors to languages. They move into niches, and as new needs arise, languages change over time. It's actually a practical way to design a computer language. Not all computer programs can be designed that way, but I think more can be designed that way than have been. A lot of the majestic failures that have occurred in computer science have been because people thought they could design the whole thing in advance.

DDJ: How do you design a language to evolve?
LW: There are several aspects to that, depending on whether you are talking about syntax or semantics. On a syntactic level, in the particular case of Perl, I placed variable names in a separate namespace from reserved words. That's one of the reasons there are funny characters on the front of variable names—dollar signs and so forth. That allowed me to add new reserved words without breaking old programs.

DDJ: What is a scripting language? Does Perl fall into the category of a scripting language?
LW: Well, being a linguist, I tend to go back to the etymological meanings of "script" and "program," though, of course, that's fallacious in terms of what they mean nowadays. A script is what you hand to the actors, and a program is what you hand to the audience. Now hopefully, the program is already locked in by the time you hand that out, whereas the script is something you can tinker with. I think of phrases like "following the script," or "breaking from the script." The notion that you can evolve your script ties into the notion of rapid prototyping.

A script is something that is easy to tweak, and a program is something that is locked in. There are all sorts of metaphorical tie-ins that tend to make programs static and scripts dynamic, but of course, it's a continuum. You can write Perl programs, and you can write C scripts. People do talk more about Perl programs than C scripts. Maybe that just means Perl is more versatile.

DDJ: So, you don't think it's necessarily a language restriction? A language isn't necessarily a scripting language.
LW: Well, the typical usage is that you call something a scripting language if it's interpreted.