Appendix A: Background of Perl

In this appendix, we introduce a brief historical, sociological and psychological reasons for why Perl works the way it does. We also include a brief history of Perl.

When learning a new language, it is often helpful to learn the history, motivations, and origins of that language. In natural languages such as English, this helps us understand the culture and heritage of the language. Such understanding leads to insight into the minds of those who speak the language. This newly found insight, obtained through learning culture and heritage, assists us in learning the new language.

This philosophy of language instruction can often be applied to programming languages as well. Although programming languages grow from a logical or mathematical basis, they are rarely purely mathematical. Often, the people who design, implement and use the language influence the language, based on their own backgrounds. Because of the influence the community has upon programming languages, it is useful, before learning a programming language, to understand its history, motivations, and culture. To that end, this chapter examines the history, culture, and heritage of the Perl language.

A.1 A Brief History of Perl

Larry Wall, the creator of Perl, first posted Perl to the `comp.sources' Usenet newsgroup in late 1987. Larry had created Perl as a text processing language for Unix-like operating systems. Before Perl, almost all text processing on Unix-like systems was done with a conglomeration of tools that included AWK, `sed', the various shell programming languages, and C programs. Larry wanted to fill the void between “manipulexity” (the ability of languages like C to “get into the innards of things”) and “whiptitude” (the property of programming languages like AWK or ‘sh’ that allows programmers to quickly write useful programs).

Thus, Perl, the Practical Extraction and Report Language\(^1\), was born. Perl filled a niche that no other tool before that date had. For this reason, users flocked to Perl.

Over the next four years or so, Perl began to evolve. By 1992, Perl version 4 had become very stable and was a “standard” Unix programming language. However, Perl was beginning to show its limitations. Various aspects of the language were confusing at best, and problematic at worst. Perl worked well for writing small programs, but writing large software applications in Perl was unwieldy.

The designers of the Perl language, now a group, but still under Larry’s guidance, took a look around at the other languages that people were using. They seemed to ask themselves: “Why are people choosing other languages over Perl?” The outcome of this self-inspection was Perl, version 5.

The first release of version 5 came in late 1994. Many believed that version 5 made Perl “complete”. Gone were the impediments and much of the confusion that were prevalent in version 4. With version 5, Perl was truly a viable, general purpose programming language and no longer just a convenient tool for system administrators.

\(^1\) Perl has also been given the name, Pathologically Eclectic Rubbish Lister. Choose the one with which you feel more comfortable. Some even insist that Perl is no longer an acronym for anything. This argument has merit, since Perl is never written PERL.