Writing Concrete Softwares

K. P. Vo
AT&T Labs

Abstract

A key to effective construction of large data and information systems is the availability of software libraries dealing with various aspects of programming, I/O, memory allocation, etc. Many software libraries are available. The successful ones tend to exhibit the below common attributes:

1. Each provides functionality not available elsewhere; 2. Each performs at a level not easily duplicated; 3. Each is general, yet easy to program and use; and last but not least, 5. Each is widely portable.

Further, libraries are more effective as software construction tools if they share common interface characteristics and can work with one another well. This means that, to the extent possible, libraries should be designed based on a small set of standard architectural features.

Over the past several years, with help from colleagues, I have been writing a number of high performance C libraries dealing with buffered I/O, memory allocation, sorting, data compression, data differencing, and container data types. These libraries embed effective algorithms and heuristics, some of which are not found elsewhere. A few of the libraries are publically available and used by programmers world-wide. Locally, they have been used to build applications that deal with gigabytes of data on a daily basis.

In this talk, I shall discuss a library architecture useful for designing libraries of the types mentioned. Examples shall be drawn from the design and usage of popular libraries such as Sfio (buffered I/O), Vmalloc (memory allocation), Cdt (container data types), Recsort (string sorting), and Vdelta (data compression and differencing). These libraries are examples of what I call "concrete softwares", i.e., highly reliable and reusable software libraries that allow application-specific extensions without any code modification or code generation.