UI6. Command Line Interfaces and Scripting

Date: 19 Aug 2002 (draft v0.1)
Contributors: J. Chiang (GSFC-UMBC)

Purpose

This document specifies the requirements for the command line and scripting interfaces of the Science Tools that will have those capabilities, such as the Likelihood Tool (A1).

Commands

Each command shall perform a single, well-defined analysis task. In order to ensure "easy" compatibility with the APIs of several different scripting language, command arguments and output shall consist only of integers, floating point numbers (single or double precision), strings, or globally defined C/C++ pointers. Depending on the command, the termination of user input will be signified either by a newline or an end-of-file. Other data constructs such as structures, classes, or objects shall not be passed to or returned by any command. Output from the commands can given as the return value of the function or directed as text to standard output or standard error.

Scripting

The scripting language shall provide a means of combining common sequences of commands. However, its use should not be required to analyze the LAT data. The user should be able to perform any analysis "by hand", via a GUI for example, without having to learn the scripting language.

Command Line Interface Requirements

The capabilities of the command line interface shall include:

1. Command recall and completion (e.g., GNU readline)
2. Session and error logging
3. System call access
4. Online help (via a `man`-like facility or launchable GUI)
5. Interrupt handling
6. Session recovery
7. Configurability, via environment variables and a user-supplied startup script or as specified in a predefined default configuration. The environment variables and script will allow the user to specify such things as preferred coordinate systems, directory paths, log file names, etc..

Scripting Language Requirements

The scripting language must have the following qualities and/or capabilities:

1. Flow control (e.g., loops, conditional constructs)
2. Math operations (trignometric and special functions, etc.)
3. Allows dynamic loading of modules
4. Extendable/Embeddable
5. Allows GUI programming
6. Supports object oriented programming
7. Programmable exception handling
8. A native command line interpreter
9. Supported by SWIG (Simplified Wrapper and Interface Generator, `http://www.swig.org/`)

The candidate languages that have so far been considered are Perl, Python, Tcl, and Ruby. All of these languages satisfy requirements 1–5 and 9, and the most current versions of these languages support object-oriented programming. As far as I know, Perl does not have an interactive command line interpreter. Python and Tcl have programmable exception handling; the other two may have it as well. Perl and Tcl are probably the most widely known, although Tcl is disliked by many.