Divide Proprietary Dependencies and Perhaps Conquer- The Physical Acoustic Lab Integrated Distributed Lab System

By Justin Hedley
Physical Acoustics Lab at the Colorado School of Mines

The Physical Acoustic Lab (PAL) Integrated Distributed Lab System is an Open Source, perhaps proprietary, platform-*dependent* system for acquiring and viewing data in the PAL lab.

Should an Open Source software system be required to run under an Open Source operating system, such as Linux? I would argue that this is not necessary. Linux may be Open Source, but Open Source software is not necessarily required to always be Linux compatible. Requirements may force *parts* of an Open Source system to run only on proprietary platforms such as Windows using proprietary interfaces such as commercial C libraries; however, using Java and/or Jython, these proprietary parts of the system may be isolated so multiple platforms, including Linux, can share the system.

The Java platform is a programming language and Application Programming Interface (API, i.e., libraries), which runs on a number of operating systems including Linux, Windows, and Mac OS X. Java Native Interface (JNI) allows C/C++ platform-dependent libraries to be accessed directly from Java programs. Now, such platform-dependent parts of the system can be isolated from other platforms such as Linux using Java distributed computing tools such as Remote Method Invocation (RMI).

Jython, a Java implementation of Python, seamlessly integrates a Java API using the simpler language syntax of Python. Extension to a Java system may be implemented in Jython.

The API and development of the PAL lab system will be used to describe how and why such an Open Source system was built. For four types of lab devices, a Polytec™ Vibrometer, a Polytec™ Scanning Controller, a Gage™ CompuScope Oscilloscope card, and a Newport™ Universal Motion Controller, abstractions were developed for the device and its parameters. The abstractions were implemented for four actual lab devices. Communication with the CompuScope was achieved via JNI wrapping of the commercial Windows C interface since no (commercial) Linux drivers are available. Communications with the other devices were achieved via serial port communications using Java Communications (javax.comm) API. Java RMI was used to distribute the devices across the net. Jython was used so others could implement their experiments by extending a Java class. A Java GUI was developed to manage device access and run experiments implemented in Jython. Viewers and data processing tools were developed using the Mines Java Tool Kit, which are incorporated the Jython experiment implementation if desired. An effort will be made to see if this system might help motivate Gage™ to Open Source their Beta Linux drivers, from an apparently abandoned project.