Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

CFH12K: Control Software and Data Handling

S. Isani, J.-C. Cuillandre, B. Starr

Canada-France-Hawaii Telescope, 65-1238 Mamalahoa Hwy, Kamuela HI 96743

G. Luppino

Institute for Astronomy, 2680 Woodlawn Drive, Honolulu HI 96822

Abstract. The CFHT 12K by 8K CCD mosaic wide-field imager is the largest of its kind in the world (Cuillandre et al. 2000). Acquiring data with this camera puts a strain on CFHT's computer systems due to the amount of data generated, but the task itself is relatively simple: process 200 MB images as quickly as possible during the 58 second readout, and maximize shutter-open time. We felt the solution to this problem should be equally simple, yet powerful. The latest and greatest so-called innovations are not always the best choice for meeting these goals. The computers and software used for astronomy must be as efficient and reliable as possible. When problems do occur, the system should be easy to understand and diagnose quickly. Based on experience gained from CFH12K's predecessor, the UH8K, we built a Unix/C command-line interpreter to operate the camera and telescope. An optional, and completely separate GUI layer is provided through a Web server.

1. Development Philosophy

Standard software practices were employed, like modular structure, code re-use, and utilization of the latest tools, but with care to avoid the temptation of only showing off new tools. The final system was built using many components that existed even when your first author was still in a baby crib. Core components include a Unix/X system, a VT100 color-capable terminal (emulator), the "curses" text library, GNU readline, bourne shell scripts, and SGML to describe the graphical interface (SGML is an old standard similar to XML but with some added conveniences.) Some of our solutions may not seem spectacular, but the bottom line is they work well and reliably. A single night that is affected by a problem with a license manager, Java interpreter, or complex custom software can quickly cancel anything gained from a more glamorous implementation.

2. The Command Line

Experience showed us that an ideal system would have both command line and graphical interfaces. A clean solution was to have everything driven by commands with a consistent and fool-proof syntax, and have the graphical interface trigger commands as a completely separate layer. Our observers and staff engineers alike have noted the advantages of a command line:

- Most efficient for repetitive engineering tasks;
- Most efficient for remote access. For example, an engineer wants to make a quick check of the dewar temperature over a slow modem link;
- Most efficient for experienced observers and "power users";
- A good solution for automation (exposure sequences, queue mode observing, and sky surveys).

From the developers' point of view, a command or menu driven program is also one of the fastest and easiest to implement.

2.1. DetCom

The command-line interpreter (CLI) that controls the CFH12K is a C program called "DetCom." DetCom runs on the machine closest to the camera hardware, takes commands on "standard input" and writes FITS files (or Multi-Extension FITS files) directly to disk (either local disk or an NFS mounted disk.) As each command is completed, a new prompt is returned with a token indicating if the last command passed or failed. All other diagnostics are printed in human-readable feedback messages to the terminal. DetCom has very few dependencies. It needs access to the camera hardware, which means it typically runs on a different computer than the one used by the observer. It also needs assembled DSP code files, which it transfers to the SDSU II CCD controllers (Leach et al. 1998). With only these two software components, it is possible to take an exposure and write a FITS file. This can be a useful to isolate problems or check if the camera is functioning in a lab setting. DetCom reads basic mosaic and detector parameters from the DSP file and does not require any other configuration files or databases.

DetCom can process pixel data from the camera in one of three ways:

1. Generate a single Multi-Extension FITS (MEF) file per exposure that contains an extension for each amplifier (12 in the case of CFH12K);
2. Generate separate FITS files for each amplifier. These files are collected in a subdirectory per exposure. This mode is useful for observers who have their own data reduction software that does not handle MEF;
3. Send scrambled or descrambled pixels to NOAO's Data Capture Agent. In this case, the Data Capture Agent creates the MEF file. This could be used to implement realtime displays or on the fly data processing in the future. We do not use this option currently.

At the same time, a binned-by-8 composite of the entire mosaic is generated in standard FITS format. This, and the complete FITS file(s) are ready within a few seconds of the end of a readout.

2.2. Director

A simple CLI such as DetCom has many limitations. But these can be overcome without any modifications to DetCom itself. Instead, a generic command line wrapper (similar to a Unix command shell) was created to make the command line more user-friendly, and more powerful. The basic features are:

- Merges several CLI's like DetCom and a telescope control CLI into one command set. Director allows commands to be routed to different places from a single point of control. This feature facilitates building complex observing scripts;
- CLI's can be running on different computers from each other, or on the same computer as Director. Standard Unix remote-shell daemons connect the CLI's to Director. Efficiency is not a problem, because CLI's are persistent so a remote-shell session is held open and always ready for new commands;
- Director listens for input from GUI's and scripts. CLI's receive GUI input on "standard input" as if it was typed;
- Director looks for the pass/fail tokens in the prompt to detect status of each command and passes it on to the caller;
- Director adds colors to feedback messages to highlight warning strings (yellow text) and failures (red text). (See Starr et al. 2000 for the CFH12K camera design and error handling description);
- Director hides some feedback messages (debugging messages, for example) but is able to reveal them after the fact;
- Director adds non-scrolling regions and a progress-bar which the CLI can update using simple text messages;
- Director provides a user friendly interface with features like command history and line editing.

One final feature of Director which has proven quite useful to our staff is the capability to run multiple instances of the user interface from *any internet location* with telnet and Web access. Each copy of the interface shows the same buffer although different views can have the debugging messages hidden or revealed. Think of it as a chat room, where everybody can see what is going on at the same time. After appropriate authentication, it is even possible to send commands from the copies or broadcast messages to the other windows.

More information on Director can be found at:

`http://rpm.cfht.hawaii.edu/director/`

2.3. Parallelization

A single threaded command line is not the easiest environment to parallelize actions. There are two solutions to this problem, which have served our needs well:

1. When actions go to different subsystems, it is possible to tell Director to run commands on different CLI's under its control simultaneously;
2. When the actions are controlled by the same CLI, such as moving a filter wheel and reading out the camera (both handled by DetCom), the command which we consider one that can proceed in the "background" is split into two commands: one which starts the action, and a second which waits for its completion. The allows another action to be started simultaneously, followed by a "wait" command when the background command must be complete.

3. Graphical Interfaces

Most users demand more than a command line to operate the system. Graphical interfaces are the most effective way to display current instrument status, and are also the best way to illustrate all the options to an new or altitude-impaired observer (the summit of Mauna Kea is at 14 000 ft., or 4200 meters.)

We were determined not to allow our software design to be driven by a choice of graphical interface. The graphical interface for the CFH12K uses the normal command set to trigger actions, and listens for feedback through a status database. This separation keeps DetCom simple, and also makes it easy to test the graphical interface separately. We just click on various buttons, checkboxes, and pulldown menus to verify that the appropriate commands are echoed.

This approach has the added benefit of teaching a user the command syntax. A text terminal with Director/DetCom shows the observer the commands that their button presses have caused. Users often start observing using the graphical interface and migrate to typing commands as they learn the syntax.

Currently, user interfaces are accessed locally and from around the world using HTTP to our custom intra-web server (RPM). Demonstration versions of the actual CFH12K interface are available at:

`http://rpm.cfht.hawaii.edu/~cfh12k/`

The interfaces use HTML forms and JavaScript (if available). The Web server generates the forms on the fly using a custom SGML DTD. The document structure and syntax is modeled after HTML, and is very easy to learn and maintain. Creating a new interface form that talks to our databases and sends commands is as simple as creating a new SGML text document.

The first incarnation of RPM was a CGI script that could parse these SGML files. It became apparent that the Web server (apache) was doing very little. Modifying the CGI script to listen on a socket directly, instead of relying on the Web server to invoke it improved performance and simplified the system, and essentially turned RPM into its own Web server. RPM is written in C++ and makes use of data structures provided by GNU libg++. Details on RPM and the SGML files that it converts into HTML forms can be found at:

`http://rpm.cfht.hawaii.edu/RPM/`

References

Cuillandre J.-C., Luppino, G., Starr B., Isani S. 2000, PASP, submitted

Leach, R., Beale, F., Eriksen, J. 1998, SPIE, 3355, 512

Starr B., Luppino, G., Cuillandre, J.-C., Isani S. 2000, SPIE, in print