TCL/TK in DART And SDSS - a high level overview.

DART - the new Data Acquisition System being developed as a collaboration between the Fermilab Computing Division and 6 Experiments at Fermilab for 1995/6 time frame.

It provides coordination of VxWorks, Unix and some VMS processors to perform the overall experiment data acquisition tasks.

We use TCL and TK as the command line and graphical user interface paradigms throughout.

SDSS - the Sloan Digital Sky Survey - will image Pi steradians about the north galactic cap in five filters, and acquire on million spectra using a dedicated 2.5 meter telescope at the Apache Point Observatory.

We are developing the DA, Data Processing and Offline Data Processing infrastructure software and using TCL as the user interface throughout all the data systems.

Other known users at Fermilab:

CDF - DAQ and Electronics groups; Accelerator Division:

What we are using:

	SDSS	DART
TCL	V7.3	V6
TK	V3.6	V2
TCLDP	V3.2	

Why TCL/TK:

- We Had to have something to replace our Very VMS Specific Solution
- Free
- Already included a lot of features one could imagine wanting, even if they are not in the upfront requirements.
- Port to VxWorks exists -with minor modifications by us; Fairly easy to port to other OS (even VMS if the users insisted - remember this was 2 years ago now).
- Can be integrated with Fortran, C and C++ applications.
- Many of our clients refuse to use buttons - they still have VT100s at home (but some of them use curses...)
- User Interfaces are Religious:
  - By taking an existing, widely used package can trade development and support resources at the expense of getting “the perfect solution”.
  - There is No perfect solution anyway, (we have plenty of users who don’t like Mosaic.)
- Can layer the GUI above the command line interface in a flexible, configurable, extensible fashion.

Applications using Tcl/Tk:

AstroDA:

Take Common DA infrastructure and add functionality for particular Camera and Telescope.

Integrate instruments controlled over different media eg RS232, ethernet.

Use TclDP

Gain:

Easy to define and test primitives and then layer up to more complex, compound scripts.

Encourages writing small routines do to atomic operations.

Pain:

Does not encourage documentation (what does).

We have not looked into TCL debugger - people put CmdTrace on and use that.

Error reporting restricted/minimal

Proliferate verbs with no context - no Language contructs.

Applications using Tcl/Tk:

DART RunControl (OCP):

TCL only layer, communicates to central server, database and clients.

Configurable TK Gui on top.

Gain:

Separate functionality from presentation/user interface layer.

Run Control another matter for religion - allow experiments to configure their own gui.

Coded early in the game and easy to change as project proceeds.

Pain:

No generic, general interace to Fortran.

Support load of having different run control interfaces at each experiment not yet addressed.

too early to tell...

FASTBUS Diagnostic Tool:

Interface to IEEE Standard Routines and added features for filling, dumping buffers.
Again, layer TK GUI on top of TCL.

Any Disadvantages?

- Slow when looping over 4 million pixels.
- Low level - lots of peripheral support coding needed for every application
- No architecture imposed on name space, option names and consistency - no Language Syntax.
- Using strings for support of math functions has given us problems with accuracy and rounding.
- Asynchronous “event loop” not supported without using TK or doing something special.

Fermilab TCL - added value:

- Packaged as an FTCL Extension.
- Can be used with TCL V7 and TK V6 “off the net”
- Available in standard Fermilab distribution area KITS or ftp.fnal.gov to registered users.
- Most available also in VxWorks.
- (not fed back to the main community - Legacy specific we feel)

Command Line Editing:

Available on VT100 compatible terminals (including X-terminals).Allows the user to edit current command lines and recall previous ones using the keypad keys and CTRL sequences shown below:

h	print history
right arrow	move cursor right
Ctrl H	beginning of line
up arrow	recall previous command
Ctrl R	redraw line
DEL	delete char before cursor
!string	recall command with string
!x	recall command number x

Command Line Parsing:

Very loosely modeled on the VMS CLI routines; that is, call one routine that parses the user entered command and then others to determine what was specified. When parsing a command, the programmer specifies the formal prototype command as well as the user entered command.

The parsing routine will parse the user command based on the prototype. Basic syntax checking is done and defaults supplied, if any.

- A prototype may consist of zero or more parameters and zero or more options.
- Parameters are positional; they must be specified unless a default is given.
- Options are not positional and may or may not be specified.
- Options specified by the user without a value will take the default if present.

"p1 p2=cat -opt1 -opt2=dog -unsigned -double -int=3 -list"

p1 = parameter 1 with no default value
p2 = parameter 2 with default value = cat
opt1 = flag with no default value
opt2 = flag with default value = dog
unsigned = flag with no default value
double = flag with no default value
int = flag with default value = 3
list = flag with no default value

TCL Verb Aliasing:

Command aliases are similar in function to UNIX style aliases. They are useful for commonly used command(s) that may be expressed in one line. In-line parameters may be subtituted as well.

tcl> alias goGetCoffee {echo “Waiting, go get coffee for ‘1 minutes”}

tcl> goGetCoffee 4

Read Simple Named Parameters:

C, Fortran and TCL interface to read named parameters from a file. Supports Integers, Strings and Floating Point.

```
#This Consumers Name - must be 4 letters long, and capitals
ConsumerName          DART

PawViewer             1
MaxEventLength        2500    #long words
NumberSources         1
HoistServer00         fndawe.fnal.gov
StopOnEndOfRun        1       # stop on receipt of EOR record
StopOnNEvents         100000  # stop after NEVENTS
CrashProtectInterval  0   # call UserCrashProtect after n events
CrashProtectFile      crash_file.hbook
HoistPortNumber       50009
```

Extended Help Facility:

help -facil=facil1,facil2

List commands with facility = facil1 and facil2

help -search=pattern

List all commands whose messages contain pattern

For example SDSS online/quick analyis framework shiva:

shErrStack	shFitsIo	shFsao
shErrStack	shFitsIo	shFsao
shMouse	shRegPrint	shRegSci
shRegSim	shRegUtils	pgPlot
shTbl	shTclRegion	shTree
shUtils	shVector	tclLink

shiva > help -facil=shDiskio

dumpClose dumpDateDel dumpDateGet
dumpHandleRead dumpHandleWrite dumpOpen
dumpPtrsResolve dumpReopen dumpTypeGet
dumpValueWrite handleDelFromType
handleNewFromType
memberSet membersGet schemaGet
schemaGetFromType schemaKindGet
schemaKindGetFromType

shiva > help shMalloc

shMalloc USAGE: shMalloc len
shMalloc some space, and return it in a handle