

Software Development with Antelope

Daniel Quinlan

Boulder Real Time Technologies, Inc.



Outline

Antelope Software Development

Simple example

Languages: A comparison

References

Hints for tcl, perl, and c

Mini-applications

Extending Antelope

Less well known interfaces (C)

Predict example



Antelope S/W Development

follows Unix example: simple tools, which can be combined in a complex fashion.

- framework
- collection of subroutines in libraries



Development Framework

- Where are binaries installed?
\$ANTELOPE/{bin,lib}
- Where can you find data files?
\$ANTELOPE/{data}/
- How do you make and install programs?
make(1), \$ANTELOPEMAKE



- How do you configure programs?
command line => parameter files
- How do you handle errors?
elog routines (rt logging, sys errors)
- How do you write documentation?
man pages



Starting an application in C

1. Make a directory to work in.
2. `template dbsimple.c > my.c`
3. `edit my.c`
4. `lint my.c`
5. repeat 3, 4
6. `make my.o`
7. `ldlibs my.o => $(DBLIBS)`
8. Makefile:

```
BIN=my
ldlibs=$(DBLIBS)
include $(ANTELOPEMAKE)
```



9. parameter file, PF=my.pf

10. manpage - > my.1, MAN1=my.1

Final Makefile:

```
BIN=my  
MAN1=my.1  
PF=my.pf  
ldlibs=$(DBLIBS)  
include $(ANTELOPEMAKE)
```



Don't want to install into \$ANTELOPE?

- SUBDIR=/local
- DEST=/home/you/development

before

```
include $(ANTELOPEMAKE)
```



Antelope Interfaces

database manipulation

parameter files

epoch time calculations

path manipulation

waveform access

orb access/packet unstuffing



Languages

There are 6 possible languages from which to choose:

- sh
- Tcl/Tk
- Perl
- Matlab
- C
- Fortran



Shell

easy

quick

limited by existing applications

limited strings

limited calculations

limited variables

use `/bin/sh`, not `/bin/tcsh` or `/bin/csh`



Tcl/Tk

hands down easiest for making GUI
easy to extend with your c functions
simple language with confusing quoting
rules (think c-shell)

actually interactive

future cloudy: seems to have a smaller
group of users; BRTT will continue to
support it indefinitely, because various
fundamental programs are in tcl/tk.



Tcl Programs in Antelope

dbe

dbevents

dbhelp

dbloc2 GUI

dbnoise

ecrontab

orbmonrtd

q330mon, k2mon

rtm



Perl

faster

more complete, better designed language

best tool for string manipulation

widespread throughout Unix, Mac and PC world, especially system administration and web cgi scripts.

looks at first like someone was typing on wrong row of keyboard.



Antelope programs in Perl

antelope_update	cronrun	dbloc2
dbmerge	deposit	getid
grepsrc	pktmon	rtbackup
rtdbclean	rtdemo	rtexec
rtincident	rtinit	rtmail
rtreport	rtrun	truncate_log



Matlab

numerical work, especially prototypes
not for real time operations
very little string manipulation
not meant for scripts



C

when nothing else will do

require speed

used a lot

need interfaces not available in other
languages



Other interfaces available from C

associative arrays and lists

waveforms

response functions

niceplot graphics

travel time calculations

vogle (3d) graphics

map transformations

waveform displays

square earth tiling



Fortran

the one you know

compatibility with existing routines

fast numerical operations

complex arithmetic

incomplete, not used as much.

in many cases, you can create a fortran
interface to c routine



References

Software Development tutorial (new here)

<http://www.brnt.com/workshop.tar.gz>

Antelope Toolbox for Matlab, Kent
Lindquist

Datascope tutorial (on cd)

Reference Booklets:

User, Programmer, Scripting



Man pages; look at the html version

file:/opt/antelope/4.4/antelope.html

FAQ ***http://www.brnt.com/faq***

Antelope-users mailing list

antelope-users@brnt.com

Antelope User's Group web site

http://www.indiana.edu/~aug



Tcl hints

package require Tclx ; cmdtrace on
(like set -x or set -v in shell script)

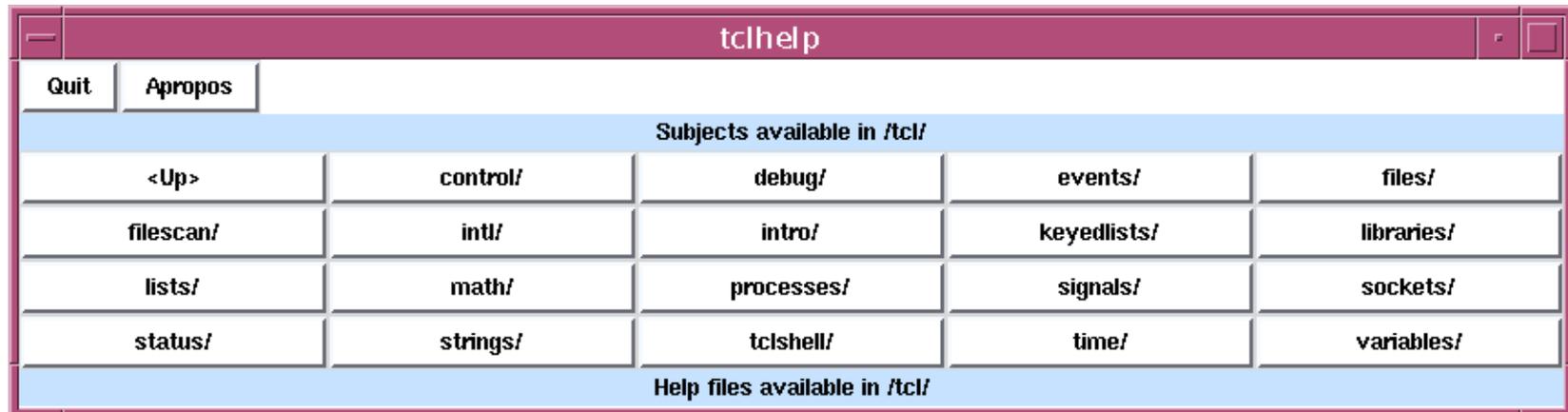
http://www.tcl.tk

Tcl/Tk Pocket reference O'Reilly

tcltk-widget



tclhelp: lookup tcl functions



The screenshot shows a window titled "tclhelp" with a menu bar containing "Quit" and "Apropos". Below the menu bar is a light blue header bar with the text "Subjects available in /tcl/". Underneath is a table with 5 columns and 4 rows of subjects. Below the table is another light blue footer bar with the text "Help files available in /tcl/".

Subjects available in /tcl/				
<Up>	control/	debug/	events/	files/
filescan/	intl/	intro/	keyedlists/	libraries/
lists/	math/	processes/	signals/	sockets/
status/	strings/	tclshell/	time/	variables/

Help files available in /tcl/

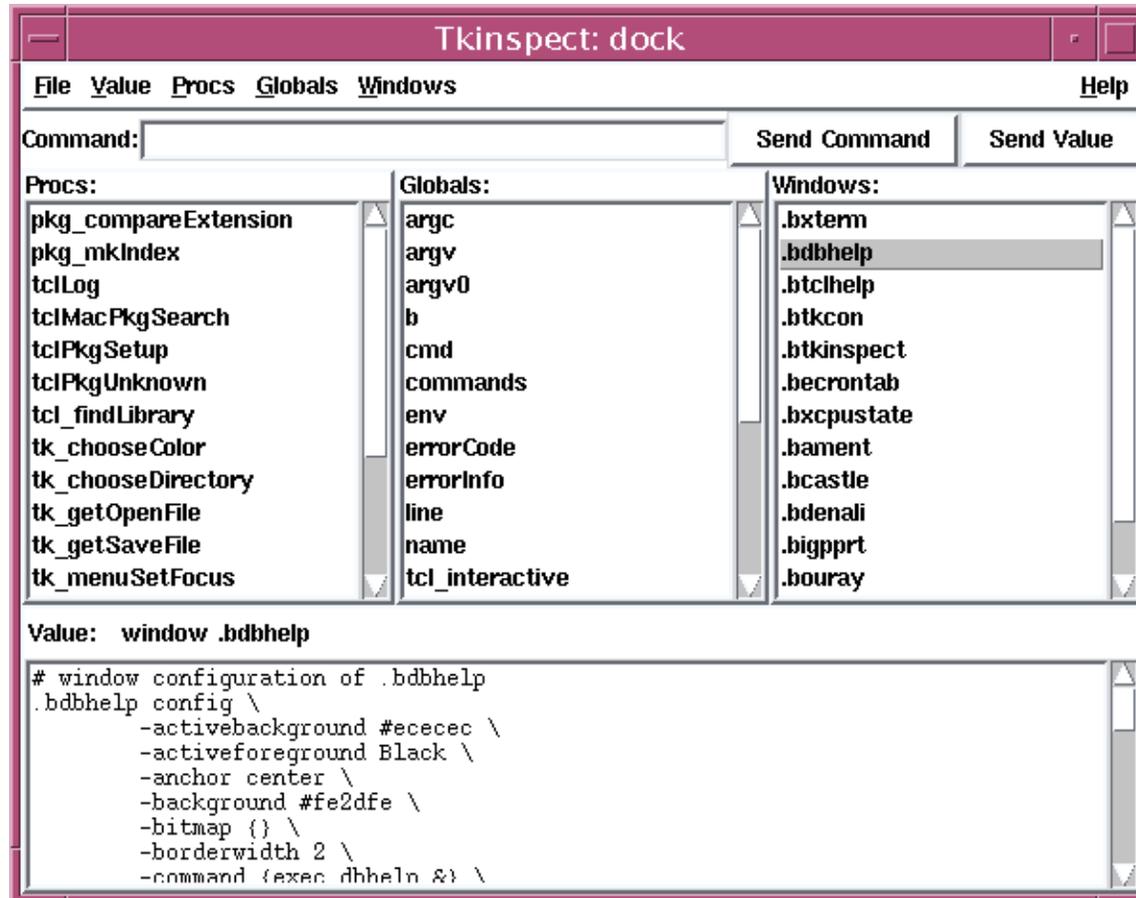


tkcon: talk to other tk applications

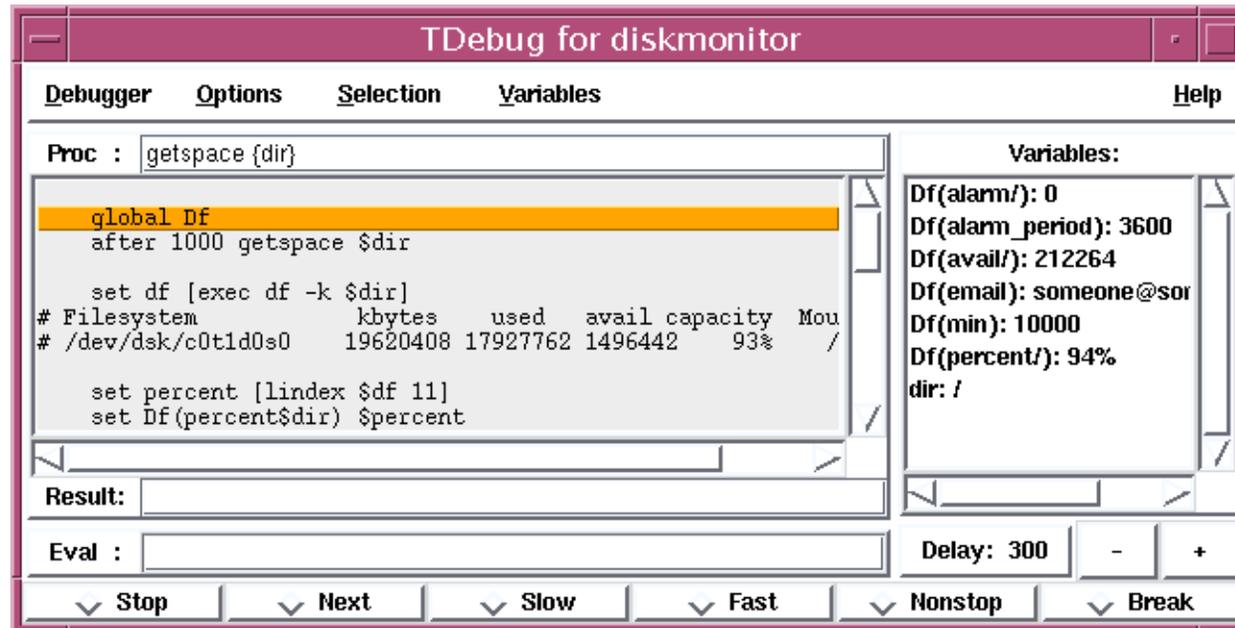
```
tkcon 2.1 Main
File Console Edit Interp Prefs History Help
>diskmonitor< (danq) 53 % array names Df
avail/ percent/ alarm/ min email alarm_period
>diskmonitor< (danq) 54 % set Df(min)
10000
>diskmonitor< (danq) 55 % set Df(avail/)
212264
>diskmonitor< (danq) 56 %
```



tkinspect: configure tk apps



tdb: graphical debugger



Global variables

One problem in tcl is local static variables. A routine does not have variables which preserve their value from one call to the next. Instead, you must save such values into global variables, much like Fortran common.

Suggestion: Choose a few (one) global variables, make them arrays and index into the array with the appropriate name.



E.g., for a program example, you might have an array Example, with elements like:

Example(gain)

Example(t0)

If you need further qualification, try putting two names together as the index:

Example(w0.t0)

Example(w1.t0)

Fewer global statements, easier to find and organize all the various global variables.



Perl Hints

perldoc -f stat

perldoc -q tk

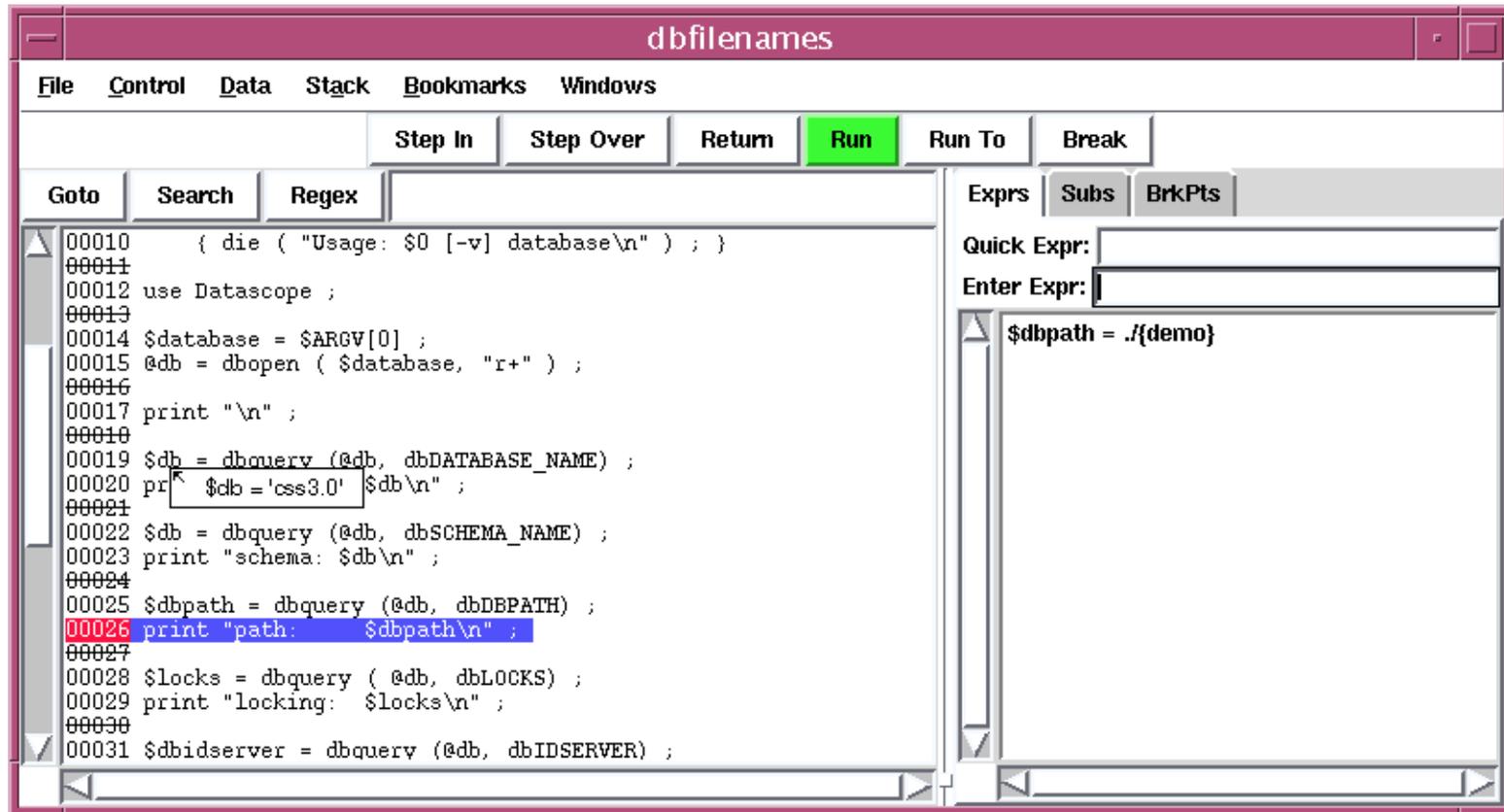
<http://www.perl.org>

Perl and Perl/Tk Pocket references
O'Reilly

perl-widget



Perl debugger: *perl -d:ptkdb*



The screenshot shows the Perl debugger ptkdb interface. The window title is "dbfilenames". The menu bar includes File, Control, Data, Stack, Bookmarks, and Windows. The toolbar contains buttons for Step In, Step Over, Return, Run (highlighted in green), Run To, and Break. Below the toolbar are tabs for Goto, Search, and Regex. The main window displays a Perl script with line numbers 00010 through 00031. Line 00026, `print "path: $dbpath\n";`, is highlighted in blue. To the right of the script is a panel with tabs for Exprs, Subs, and BrkPts. The Exprs tab is active, showing a "Quick Expr:" field and an "Enter Expr:" field containing the expression `$dbpath = ./{demo}`.

```
00010 { die ( "Usage: $0 [-v] database\n" ) ; }
00011
00012 use Datascope ;
00013
00014 $database = $ARGV[0] ;
00015 @db = dbopen ( $database, "r+" ) ;
00016
00017 print "\n" ;
00018
00019 $db = dbquery ( @db, dbDATABASE_NAME ) ;
00020 print "db: $db\n" ;
00021
00022 $db = dbquery ( @db, dbSCHEMA_NAME ) ;
00023 print "schema: $db\n" ;
00024
00025 $dbpath = dbquery ( @db, dbDBPATH ) ;
00026 print "path: $dbpath\n" ;
00027
00028 $locks = dbquery ( @db, dbLOCKS ) ;
00029 print "locking: $locks\n" ;
00030
00031 $dbidserver = dbquery ( @db, dbIDSERVER ) ;
```



c hints

lint (lint wrapper)

indent (*look for .indent.pro*)

```
cextract -D__STDC__ -I/opt/antelope/dev/include -P +p
```

```
protoize -c "-I$ANTELOPE/include -I/opt/antelope/  
tcltk8.3/include"
```

ldlibs



C Debugging

```
ldd [-s] `which trexcerpt`
```

truss

```
truss -a -vall -u\* -f -o /tmp/truss ptree $$
```

stack trace on fault

```
elog_init(argc, argv) ;
```

collector in workshop



Memory problems

setenv LD_PRELOAD watchmalloc.so.1

(grepsrc -l LD_PRELOAD /usr/share/man/)

workshop

works sometimes

purify



ups: the best debugger

The screenshot shows the ups debugger window titled "ups cpredict". The interface includes a menu bar with options like Help, Search, Windows, and Quit. Below the menu is a toolbar with icons for Expand, Collapse, Dup, Del, Format, and Decl. The main area is divided into several sections: Target (./cpredict), Signals, Environment, Untyped variables, Source files, and Functions. The Functions section shows a list of functions, with "main" selected. The Breakpoints section shows a breakpoint at "function:main" in "cpredict.c:75" which is active. Below the main area is a control bar with buttons for Restart, Next, Step, Cont, Stop, Attach, Detach, and Kill. The bottom section displays the source code for "cpredict.c:81" with a breakpoint set at the "sprintf" line. A context menu is open over the "sprintf" line, showing options: Add breakpoint, Execute to here, Jump to here, and Edit source.

```
Target ./cpredict
cpredict 32.867 117.256 0 2000-001
Signals
Environment
Untyped variables
Source files
Functions
  main                                     cpredict.c:75
    char <lat{0}>                            "32.867"
    char <depth{0}>                          "0"
Breakpoints
  function:main                             cpredict.c:75   active

Restart  Next  Step  Cont  Stop  Attach  Detach  Kill
cpredict.c:81  Back  Search  Up  Down

table = pfget_string (pf, "table" );
if ( dbopen_table ( table, "r", &db ) < 0 ) {
  die ( 0, "Can't open table %s\n", table );
}
#stop;
sprintf ( expr, "distance(lat,lon,%s,%s)", lat, lon );
printf ( "distance: %s\n", expr );
printf ( "time: %s\n", ptime(distance(lat,lon,%s,%s),%s)", lat, lon, depth );
printf ( "pgm: %s\n", expr, &pgm, 0 );
```



Mini Applications

a command line window to an Antelope library function:

Datascope	<i>dbhelp, dbe, dbcalc, ..</i>
epoch time	<i>epoch</i>
parameter files	<i>pfecho</i>
path	<i>abspath, relpath, cleanpath</i>
orb/pkts	<i>orbstat -i</i>
travel times	<i>ttcalc</i>



Extending Antelope

your waveform format

your packet formats

your travel time calculator

your location program (in dbloc2)

your processing/editing/graphics program
in dbe

eelog ELOG_CALLBACK routines



Little known C interfaces

stock

strmatches, strcontains

hexdump

abspath, relpath, cleanpath

ask, askyn, askynaq

db

dbprocess

dbggroup (bundle, bundletype)

dbcompile



dbstrtype
dbuntangle
dbsever
dbseparate

tt: ttcalc, ucalc

proj: pj_init, pf_fwd, pj_inv, pj_free

grx: fortran callable 2d graphics

vogle: c 3d graphics

pixaddress:



Contrived Problem:

print the arrival time for an event (at a certain time and coordinates), at a list of stations (from a database)

<http://www.brtt.com/workshop.tar.gz>

example code:

```
% ls $ANTELOPE/examples  
c/          perl/      shell/     vogle/  
fortran/    predict/  tcltk/
```



Shell example

```
#!/bin/sh
```

```
lat=$1
```

```
lon=$2
```

```
depth=$3
```

```
time=$4
```

```
epoch=`epoch +%E $time`
```

```
table=`pfecho -q predict table`
```

```
dbsort $table "distance(lat,lon,$lat,$lon)" |
```

```
dbselect - staname
```

```
  "strtime($epoch+ptime(distance(lat,lon,$lat,$lon),$d  
  epth))"
```



Tcl/Tk Example

```
#!/bin/sh
# \
exec $ANTELOPE/bin/atcl $0 "$@"
package require Datascope
package require Tclx

set lat    [lindex $argv 0]
set lon    [lindex $argv 1]
set depth  [lindex $argv 2]
set time   [lindex $argv 3]
set epoch  [str2epoch $time]

set table [pfget predict table]
set db [dbopen_table $table r]
set db [dbsort $db
        distance(lat,lon,$lat,$lon)]
```



```
set n [dbquery $db dbRECORD_COUNT]
loop i 0 $n {
    set db [lreplace $db 3 3 $i]
    set pred [dbeval $db
ptime(distance(lat,lon,$lat,$lon),$depth)]
    set pred [expr $pred+$epoch]
    set staname [lindex [dbgetv $db 0 $i staname] 0]
    puts [format "%-55s %s (%s)" $staname [strtime
$pred] [epoch2str $pred "%H:%M:%S.%s %z" ""]]
}
```



Perl Example

```
: # use perl
eval 'exec $ANTELOPE/../../perl/bin/perl -S $0 "$@"'
if 0;

use lib "$ENV{ANTELOPE}/data/perl" ;
use Datascope ;

($lat, $lon, $depth, $time) = @ARGV ;
$epoch = str2epoch ($time) ;
$table = pfget ("predict", "table") ;

@db = dbopen_table($table, "r") ;
@db = dbsort( @db, "distance(lat,lon,$lat,$lon)" ) ;

$n = dbquery (@db, "dbRECORD_COUNT");
for ($db[3] = 0 ; $db[3]<$n ; $db[3]++){
```



```
$pred = dbex_eval (@db,  
    "ptime(distance(lat,lon,$lat,$lon),  
        $depth)" ) ;  
$pred += $epoch ;  
($staname) = dbgetv(@db, "staname");  
printf "%-55s %s    (%s)\n",  
    $staname, strtime($pred),  
    epoch2str($pred,  
        "%H:%M:%S.%s %z", "") ;  
}
```



Matlab Example

```
#!/bin/sh
if [ $# != 4 ] ; then
    echo "Usage: $0 lat lon depth time"      exit 1
fi

exec /usr/local/bin/matlab -nodisplay -nojvm -nosplash
    << EOF > /dev/null
fid = fopen ( `/dev/stderr', 'w' ) ;

lat = $1 ;
lon = $2 ;
depth = $3 ;
time = '$4' ;
epoch = str2epoch (time) ;

pf = dbpf ( `predict' ) ;
```



```

dbtable = pfget ( pf, 'table' ) ;
[db, table] = strtok(dbtable, '.') ;
table = strrep(table, '.', '') ;

db = dbopen (db, 'r' ) ;
db = dblookup_table ( db, table) ;

expr = sprintf (
    'distance(lat,lon,%f,%f)', lat, lon);
db = dbsort( db, expr ) ;

expr = sprintf ('ptime(distance(lat,lon,%f,%f),%f)',
    lat, lon, depth ) ;
n = dbquery ( db, 'dbRECORD_COUNT' ) ;
for i = 0:1:n-1
    db.record = i ;
    pred = dbeval(db, expr) + epoch ;

```



```
staname = dbgetv(db, 'staname' ) ;  
s = strtime(pred) ;  
fprintf ( fid, '%-55s %s\n',  
        staname, s ) ;  
end  
exit ;  
EOF
```



C example

```
#include <stdio.h>
#include "coords.h"
#include "db.h"

static void
usage ()
{
    cbanner (" $Revision: 1.3 $",
            "[-v] lat lon depth time",
            "BRTT example",
            "BRTT",
            "support@brtt.com" ) ;
    exit (1);
}
int
main (int argc, char **argv)
```



```
{
    Dbptr    db;
    Pf       *pf;
    char     *table, *lat, *lon, *depth ;
    char     expr[256], staname[128],
            *s1, *s2 ;

    int      n ;
    double   epoch, pred ;
    Tbl      *keys ;
    Expression *pgm ;

    elog_init ( argc, argv ) ;

    if (pftread ("predict", &pf) != 0)
        die (0, "Can't read pf\n");

    if (argc-optind != 4)
        usage ();
}
```



```
lat    = argv[optind++];
lon    = argv[optind++];
depth = argv[optind++];
epoch = str2epoch(argv[optind++]);

table = pfget_string (pf, "table" ) ;

if(dbopen_table(table, "r", &db)<0)
    die(0,"Can't open %s", table ) ;
sprintf ( expr,
    "distance(lat,lon,%s,%s)",lat,lon);
keys = strtbl(expr, 0) ;
db = dbsort( db, keys, 0, 0 ) ;

    sprintf ( expr,
    "ptime(distance(lat,lon,%s,%s),%s),
    lat, lon, depth ) ;
```



```

dbex_compile(db, expr, &pgm, 0 ) ;
dbquery(db, dbRECORD_COUNT, &n ) ;
for ( db.record=0 ; db.record<n ;
      db.record++){
    dbex_eval (db, pgm, 0, &pred ) ;
    pred += epoch ;
    dbgetv(db,0,"staname",staname,0);
    s1=strtime(pred),
    s2=zepoch2str(pred,
                  "%H:%M:%S.%s %z", "") ) ;
    printf ("% -55s %s (%s)\n",
            staname, s1, s2 ) ;
    free(s1) ; free(s2) ;
}
dbex_free(pgm) ;
return 0;
}

```



Fortran example

```
program fpredict
  implicit none
#include "db.i"
  real*8 str2epoch, epoch, pred
  integer iargc, db(4), pf, pgm, i, result, n, keys
  character*256 table, expr, staname
  character*64 lat, lon, depth, time, stime

  call elog_init(0,0)

  call pfred("predict", pf, result )
  if ( result ) then
    call die ( "Can't read parameter file" )
  endif
```



```
if ( iargc() .ne. 4 ) then
    call die("Usage: fpredict lat lon depth time")
endif

call getarg(1, lat)
call getarg(2, lon)
call getarg(3, depth)
call getarg(4, time)
epoch = str2epoch(time)

call pfget_string(pf, "table", table )

if ( dbopen_table ( table, "r", db) .LT. 0 )
*   call die ( 0, "Can't open database" )
```



```
    write (expr, 100) lat, lon
100  format ('distance(lat,lon,', a, ', ', a, ')')

    call strtbl(keys, expr, 0)
    call dbsort ( db, db, keys, 0, "" )

    write ( expr, 101) lat, lon, depth
101  format ('ptime(distance(lat,lon,',
*      a, ', ', a, '), ', a, ')')
    result = dbex_compile ( db, expr, pgm, 0 )
```



```
call dbquery ( db, dbRECORD_COUNT, n )
do i = 0, n-1
  db(4) = i
  result = dbex_eval ( db, pgm, 0, pred )
  pred = pred + epoch
  result = dbgetv(db, "", "staname", staname, 0)
  call strtime(stime, pred)
  write (*, 110) staname, stime
end do

110 format ( a55, a )
end
```



Makefile

```
BIN=cpredict fpredict mlpredict predict \  
    tcldpredict perlpredict
```

```
ldlibs=$(DBLIBS)  
include $(ANTELOPEMAKE)
```

Missing MAN1=

