Antelope Quick Start
Antelope Quick Start - Overview

• What is Antelope?
  – Roadmap
  – Distribution, support and development cycle
  – Contributed software

• Installation and Administrative
  – Installing and registering
  – Basic directory structure
  – Installing license and patches
  – Managing patches
  – Licenses
  – User setup and requesting support

• Documentation
  – Usage line
  – UNIX man pages
  – Web
  – Tutorials and guides
  – Source-code examples

• Basic concepts
  – UNIX paradigm
  – Antelope parameter files
  – SEED vs. CSS name conventions
  – Specification of time
  – Using UNIX regular expressions

• Quick sampling of Antelope
  – Initial setup
  – Running rtdemo
  – A look at an Antelope database
  – Where to go from here

April 2004
What is *Antelope*?
What is **Antelope**?

- **Antelope** is a **software platform** which provides a standard substrate (middleware) for interconnecting diverse data sources with diverse data processing
  - A product with both commercial and public-domain (contributed) components
  - A product with both closed and open source components
  - A product with an open system architecture
  - A product that embraces the principles of modularity and interoperability
  - Encapsulates diverse specific data formats within standard packaging and uses standard protocols for communication
  - Can accommodate **any** packetized data
  - Uses relational database technology
  - High performance and reliability
  - Easy integration of user developed modules
What Is *Antelope*?: functions

- **Relational Database System: Datascope**
  - Generic information system functions
  - Seismology-specific processing functions using derivatives of the CSS3.0 schema (supports both interactive and automated batch-mode processing, usually not in real-time)
    - Data and metadata archiving (waveforms, site char., instr. char., etc.)
    - Information archiving (picks, hypocenters, magnitudes, etc.)
    - Import/export (SEED, SAC, SEGY, AutoDRM, PASSCAL, etc.)
    - “routine” network processing (detection, association, location, etc.)
    - Ground motion estimation (similar in function to USGS Shakemap)
    - Interactive analyst review
    - Archive management functions (backup, restore, copy, excerpt, etc.)
    - Interface with other research tools (SAC, MATLAB, etc.)
What Is **Antelope**?: functions

- **Antelope** Real-Time System (ARTS): ORB
  - Generic automated real-time functions based upon an Object Ring Buffer mechanism
  - Seismology-specific processing functions (supports automated real-time processing)
    - Seismic datalogger acquisition (Quanterra, Kinemetrics, Reftek, etc.)
    - Seismic dataserver import/export (LISS, CD-1, COMSERV, etc.)
    - Data, metadata and information archiving into Datascope
    - “routine” network processing (detection, association, location, etc.)
    - Inter-network data and information sharing
    - Real-time network management (hub management, datalogger monitoring and control, etc.)
    - Interface with other research tools (MATLAB)
What Is *Antelope*?: functions

- **Antelope** toolbox
  - Provides a rapid software development environment that exploits reuse of *Antelope* middleware
  - Interfaces for both Datascope and ARTS
  - Well documented c-language libraries
  - tcl/tk (tool command language) extensions
  - perl (practical extraction and report language) extensions
  - MATLAB interfaces
  - Various high level CASE tools, such as dbdesign and the packets compiler
Antelope distribution

- Distributed on a single CD
- Developed and runs in a “standard” UNIX environment
  - SUN SPARC/Solaris 2.8
  - X86/Linux 2.4.19
  - ARM/Linux 2.4.9 (under development)
  - MacOSX/Darwin (under development)
  - What about PC/Windows? – very unlikely
- ~380 programs and scripts
- ~60 software libraries
- generated from ~800,000 lines of source code and documentation
- 1/3 contributed from research community
How is *Antelope* developed and supported?

- **One year development cycle:**
  - New release comes out in March of each calendar year
  - Current release is maintained through a set of patches that can be downloaded and applied through a GUI
  - Patches are supplied through the BRTT web site, [http://www.brtt.com](http://www.brtt.com)
  - Note that multiple releases of *Antelope* can co-exist on the same host computer

- **Upgrades and maintenance:**
  - Upgrades to new releases are purchased separately through a maintenance contract
Contributed software in *Antelope*

- The *Antelope* user community contributes software to the standard *Antelope* distribution on a regular basis.
- The *Antelope* contributed software is maintained through the *Antelope* User Group (AUG) and their web site at [http://www.indiana.edu/~aug](http://www.indiana.edu/~aug)
  - All contributed software must be open source. A contributed software source-code CVS repository is maintained by the AUG host, Dr. Gary Pavlis, University of Indiana, and is accessible through the AUG web site.
  - The contributed source-code repository is readable by everybody put can only be updated by those with local login accounts. New contributors are encouraged and they can obtain login accounts from Gary Pavlis.
Contributed software in *Antelope*

- BRTT routinely builds executable versions of all contributed software and includes these executable modules in the final *Antelope* distribution CDs
- BRTT is also one of the open-source contributors
- BRTT does not distribute the contributed software source-code – this must be obtained directly from the AUG host
- BRTT refers any bug reports or support requests related to contributed software back to the original authors
Contributed software in Antelope

contrib/bin/magnitude/orbampmag/

Click on a directory to enter that directory. Click on a file to display its revision history and to get a chance to display diffs between revisions.

Current directory: antelope_contrib / contrib / bin / magnitude / orbampmag

<table>
<thead>
<tr>
<th>File</th>
<th>Rev</th>
<th>Age</th>
<th>Author</th>
<th>Last tag entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Makefile</td>
<td>1.4</td>
<td>3 weeks</td>
<td>horn</td>
<td>several files run now under linux. Works with stations having loc-codes. Some p...</td>
</tr>
<tr>
<td>lab.c</td>
<td>1.1</td>
<td>3 weeks</td>
<td>horn</td>
<td>forgot to add template-parameters for orbamps</td>
</tr>
<tr>
<td>mlab.c</td>
<td>1.1</td>
<td>3 weeks</td>
<td>horn</td>
<td>Forgot to add template-parameters for orbamps</td>
</tr>
<tr>
<td>mlab.h</td>
<td>1.1</td>
<td>3 weeks</td>
<td>horn</td>
<td>Forgot to add template-parameters for orbamps</td>
</tr>
<tr>
<td>mlab.of</td>
<td>1.1</td>
<td>3 weeks</td>
<td>horn</td>
<td>Forgot to add template-parameters for orbamps</td>
</tr>
<tr>
<td>orbampmag.c</td>
<td>1.4</td>
<td>3 weeks</td>
<td>horn</td>
<td>several files run now under linux. Works with stations having loc-codes. Some p...</td>
</tr>
<tr>
<td>orbampmag.h</td>
<td>1.5</td>
<td>3 weeks</td>
<td>horn</td>
<td>several files run now under linux. Works with stations having loc-codes. Some p...</td>
</tr>
<tr>
<td>orbampmag.of</td>
<td>1.2</td>
<td>3 weeks</td>
<td>horn</td>
<td>several files run now under linux. Works with stations having loc-codes. Some p...</td>
</tr>
</tbody>
</table>

Show only files with tag: All tags / default branch: Selected module path or alias: contrib/bin/magnitude/orbampmag

BRTT

April 2004
Installation and Administrative
Installation and Administrative: Installing and Registering

• Prior to installation:
  – You will need roughly 800 Mbytes of space
  – You will need your e-mail running properly (in order to send the license request)
  – You will need to be connected to the internet (in order to obtain software patches)
  – You will need a graphical X-windows interface (in order to run the installation scripts)
  – Install as root or some other user?
    • Probably root for multiple user access
  – **Antelope** MUST install at `/opt/antelope`
    • Either 1) space at `/opt/antelope`, or 2) a symbolic link at `/opt/antelope` to some other place, or 3) auto/remote mounting at `/opt/antelope` from some other place
      • Probably need root permission for creating `/opt/antelope`
  – Look at **README** on the root directory of the **Antelope** distribution CD for more information
• Installation:
  – Mount the **Antelope** distribution CD
  – `cd` to the CD root directory and run `./start`

• After splash/title, license window and then
• **Changes** button will show a detailed log of software changes since last release. It is **REAL IMPORTANT** to read the Changes log carefully – especially if you are upgrading to a new release.
• Click on **Next>>** and you will eventually get to the install screen.

• Click on **Install Everything** and on **Next>>** again.

• If you already have a valid **license.pf** file, then click on **Yes**, otherwise click on **No - Register Software**

• Please fill out the registration form completely. Pay careful attention to:
  – e-mail address
  – hostname
  – ip-address
  – license type

• Click on **Register** when you are done
• If your e-mail is running properly, the request has been already mailed
• If your e-mail is not working, then a file with the license request, $HOME/BRTT-license-request, which was automatically created, can be copied to a machine with e-mail and sent to register@brtt.com
• Once you have received an e-mail reply from register@brtt.com, save as a text file and run install_license
Installation and Administrative: Basic directory structure
## Installation and Administrative: Basic directory structure

<table>
<thead>
<tr>
<th>Directory Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/opt/antelope/4.6/</code></td>
<td>Main Antelope release sub-directory. The ANTELOPE environment variable must be set to this. Note that multiple releases can co-exist in <code>/opt/antelope</code>.</td>
</tr>
<tr>
<td><code>/opt/antelope/data/</code></td>
<td>Example and demo data for Datascope and ARTS.</td>
</tr>
<tr>
<td><code>/opt/antelope/perl5.8/</code></td>
<td>Off-the-shelf perl distribution.</td>
</tr>
<tr>
<td><code>/opt/antelope/tcltk8.4/</code></td>
<td>Off-the-shelf tcl/tk distribution.</td>
</tr>
<tr>
<td><code>$ANTELOPE/bin/</code></td>
<td>Antelope executable binaries and scripts</td>
</tr>
<tr>
<td><code>$ANTELOPE/data/</code></td>
<td>“Data” needed by Antelope software modules. Includes parameter file defaults, schema descriptions, travel time tables, mapping data, instrument stage responses, perl and tcl script fragments, etc.</td>
</tr>
<tr>
<td><code>$ANTELOPE/data/pf/</code></td>
<td>Antelope default parameter files for programs and scripts. Also home of license.pf file.</td>
</tr>
<tr>
<td><code>$ANTELOPE/demo/</code></td>
<td>Scripts and parameter files for rtdemo</td>
</tr>
<tr>
<td><code>$ANTELOPE/doc/</code></td>
<td>Tutorials and guides.</td>
</tr>
<tr>
<td><code>$ANTELOPE/example/</code></td>
<td>Source code examples for Antelope toolbox.</td>
</tr>
<tr>
<td><code>$ANTELOPE/include/</code></td>
<td>Include files for Antelope toolbox.</td>
</tr>
<tr>
<td><code>$ANTELOPE/lib/</code></td>
<td>Dynamic link libraries for Antelope toolbox.</td>
</tr>
<tr>
<td><code>$ANTELOPE/man/</code></td>
<td>UNIX-style man pages and HTML documents that cover all aspects of Antelope.</td>
</tr>
<tr>
<td><code>$ANTELOPE/patched/</code></td>
<td>Describes the patch state of the release.</td>
</tr>
<tr>
<td><code>$ANTELOPE/patches/</code></td>
<td></td>
</tr>
<tr>
<td><code>$ANTELOPE/static/</code></td>
<td>Static link versions for Antelope toolbox libraries.</td>
</tr>
</tbody>
</table>
# Installation and Administrative: Important files

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ANTELOPE/setup.csh</td>
<td>These are shell script fragments that set all of the necessary environment variables and search paths to run Antelope. Both C-shell and Bourne shell versions are included. Users source one of these files, usually from their shell startup scripts.</td>
</tr>
<tr>
<td>$ANTELOPE/setup.sh</td>
<td></td>
</tr>
<tr>
<td>$ANTELOPE/README</td>
<td>These are copied from the original Antelope distribution CD.</td>
</tr>
<tr>
<td>$ANTELOPE/Changes</td>
<td></td>
</tr>
<tr>
<td>$ANTELOPE/Release</td>
<td>Master Antelope web index for all documentation. You can run netscape $ANTELOPE/antelope.html to access this web page.</td>
</tr>
<tr>
<td>$ANTELOPE/license.txt</td>
<td>A valid Antelope license file must be here in order to run Antelope.</td>
</tr>
<tr>
<td>$ANTELOPE/antelope.html</td>
<td></td>
</tr>
<tr>
<td>$ANTELOPE/data/pf/license.pf</td>
<td></td>
</tr>
</tbody>
</table>
Installation and Administrative : Installing licenses and patches

We just got back our e-mail response from BRTT with the new *Antelope* license. We save the e-mail in a text file. This is what we do next…

Setup *Antelope* environment variables and search paths

Check to see if everything is set properly

Fire up the *install_license* script
• Browse to the saved e-mail and hit **Next>>**. Note that you can enter the raw e-mail text file.

• Click on **Copy license file?**. The license will be copied to `$ANTELOPE/data/pf/license.pf`

• This window means your license is OK, it is properly installed and you are ready to go…almost. The final step in the installation is to apply the most recent software patches. You do this by clicking on **Run antelope_update**
• It is VERY VERY important to apply ALL of the patches for **Antelope**. Keep clicking on the Install buttons, starting with **Install antelope_update.0** until all of the patches have been applied.

• When all of the patches have been applied you can exit from this window and you are ready to go.

---

**Antelope 4.5 Patch Updater**

Found 14 new patches

Stop Antelope Systems before applying patches

- **Install antelope_update.0**
  This patch is only needed for Linux systems.

- **Install quanterra_and_misc.0**
  The major change in this patch is to the program

- **Install tkdbpick.0**
  This fixes a problem related to using tkdbpick on

- **Install rtxexec.0**
  This should eliminate some warning messages from rtxexec

- **Install stack_ds_leaks.0**

Re-Start Antelope Systems

Update log

**Not Installed: antelope_update.0**
This patch is only needed for Linux systems.

It adds the -P option to GNU tar command line, so that GNU tar doesn’t strip out the leading slash, and deposit the patch below the current directory.

*** CAUTION ***

1) Install this patch before installing other Linux patches.

---

April 2004
Installation and Administrative: Managing patches

BRTT Antelope Release 4.5, Patch S99_antelope.0 7/25/03

The released version of S99_antelope did not work for some situations, depending on the shell of root or the user. This new version fixes that problem, and introduces a new Makefile for installing S99_antelope correctly.

If you have installed S99_antelope to cause your systems to restart automatically after a reboot, we recommend you...
Installation and Administrative : Managing patches

- Patches for **Antelope** can be downloaded and applied at any time by running **antelope_update**
- If your current **Antelope** is up to date, then this window will appear
- Notification of the availability of new patches is sent by e-mail from BRTT to all registered users
How does licensing work?

- **Antelope** software modules automatically read the `$ANTELOPE/data/pf/license.pf` file to determine the license type and to read the license keys.
- Based upon the license type, the **Antelope** software module then looks at the system time, queries the host computer and/or its internet connections and computes another set of license keys.
- If the computed keys match the keys in the `license.pf` file, then the license is OK and the module will run properly, otherwise an error exit will occur.
- A set of different license keys can be listed in the `license.pf` file, in which case the license checking software will check the computed keys against all of the keys in the `license.pf` file. If any of the keys match, then the module will run properly.
- The current state of licensing for a particular **Antelope** installation can be checked by running `check_license [-v[v]]`
- New licenses **MUST** be requested by running `register_antelope`
Installation and Administrative : Licenses

- License types
  - Subnet licenses
    - These licenses will enable multiple host computers that have ip-addresses within a single class-C subnet (first three numbers of numerical ip-address).
    - BRTT will not give out subnet licenses for “private” networks (e.g. 192.168.xxx.xxx) or for any subnets that are not name served and do not belong to qualified users. Usually this means that we will not provide subnet licenses for host computers that get their ip-addresses via DHCP or private non-name served addresses behind a firewall.
  - Node licenses
    - These licenses will enable a single host computer regardless of its internet connections.
    - This is accomplished through the use of a “host fingerprint” – a hashed version of a complex hardware-based identification profile that is intended to uniquely identify a particular computer system.
  - Other license restrictions
    - Independent of the subnet and node licensing, all Antelope licenses can be time-limited, architecture/OS limited, and/or capability limited.
Installation and Administrative : Licenses

- The current state of licensing for a particular Antelope installation can be checked by running `check_license -vv`
- The node fingerprint, `hostinfo`, and the list of currently configured ip-addresses is determined.
- Each license key is compared against the node fingerprint and the ip-addresses to determine if any are OK
Installation and Administrative : User setup

• Include Antelope setup file in shell sessions:
  – For C-shell based shells (e.g. csh, tcsh):
    source /opt/antelope/4.6/setup.csh
  – For Bourne based shells (e.g. sh, ksh, bash):
    source /opt/antelope/4.6/setup.sh
  – Normally this is done in the user shell startup script (e.g. .cshrc, .tcshrc, .shrc)

• Avoid setting the following environment variables:
  – LD_LIBRARY_PATH – this will wreak havoc with Antelope in unpleasant and unpredictable ways. There is a section in the Antelope FAQ that discusses this.
  – Generally, it is best to avoid any of the standard perl or tcl environment variables (e.g. PERLLIB, etc.), although we have tried to make Antelope impervious to these settings
Installation and Administrative : User setup

```
kor%     
kor%     
kor%     
kor% more ~/.cshrc
#

set prompt = "%! `hostname`% "

umask 002
limit coredumpsize 0
unlimit descriptors

set history=50
set filec

set ANTELOPE=/opt/antelope/4.6
source $ANTELOPE/setup.csh

unsetenv LD_LIBRARY_PATH
kor%   
```
Installation and Administrative : Requesting support

• **ALL SUPPORT REQUESTS MUST BE E-MAILED TO support@brtt.com**
  – Support requests include bug reports, questions or any other problems with the software or understanding the software
  – **DO NOT SEND SUPPORT REQUESTS TO OUR PERSONAL E-MAIL ADDRESSES** – this can drastically reduce the effectiveness of the response, if any.

• Please make your support requests as detailed, clear and concise as possible
  – Often we will need to reproduce the problem in Boulder in order to fix it.
  – Include the following information:
    • Computer hardware (SPARC, x86, workstation, laptop, etc.)
    • Operating System (Solaris 8, SuSE 8.2/Linux 2.4.19, etc.)
    • Version of *Antelope*
    • Have you applied all of the patches?
    • The *exact* error messages you saw
    • The *exact* command line usage
    • We may need some of your data (look at `dbsnapshot` and `rtsnapshot`)

April 2004
Documentation
Antelope Documentation

- There are ~2000 pages of Antelope documentation in the forms of:
  1. UNIX “man” pages
  2. HTML versions of the man pages
  3. Tutorials
  4. Reference guides
  5. Source-code examples

- Most of the documentation can be accessed through the master Antelope documentation web page with

  `netscape $ANTELOPE/antelope.html`
Documentation: Usage line

• One of the simplest ways to see a terse description of the command line arguments for a program is to run the program with no arguments.
Documentation: UNIX man pages

- The UNIX manual (man) page documentation paradigm is at the heart of all *Antelope* documentation. *Antelope* man pages should be used as the definitive source of information about the *Antelope* software.
- Become familiar with man section numbers (i.e. 1=commands, 2=system calls, 3=functions and libraries, etc.)
- Learn how to display man pages from a shell type-in prompt using the UNIX *man* command:
  
  ```
  man -k <key words>  to match key words against the key words for each man page
  man -s <section#> <mantitle>  to display the man page with title <mantitle> from section number <section#>
  ```

- Note that the UNIX *man* command has slightly different usage across the various implementations (Solaris, GNU, etc.). Look at your particular man usage with *man man*
Documentation: master web page

• The master Antelope documentation web page at file:///opt/antelope/4.6/antelope.html is highly indexed and cross-referenced. This is a logical place to start for new users. Spend some time navigating through the links to see how it is organized and what is available.

• Visit the Antelope FAQ web site by clicking on the link.
  – This site is maintained at our offices in Boulder and users can contribute questions and answers.
  – A convenient utility at this web site is a search engine that will search both the FAQ contents and the body of all of the Antelope man pages for the most recent release.

• Visit the Antelope User Group web site by clicking on the link.
  – This site is currently maintained at the University of Indiana and contains all of the source code for the contributed software.
Documentation: tutorials

- **Datascope: A Tutorial** – This is relatively up-to-date (2002) and is a very well written and highly recommended overall description of Datascope, both for users and developers. PDF version at $ANTELOPE/doc/asis/datascope.pdf. Data used in this tutorial can be found in /opt/antelope/data/db/demo.

- **Software Development with Antelope** – This is also relatively up-to-date (2002) and is targeted mainly for software developers. Highly recommended for all developers. However, the first several chapters are instructive for users as well. PDF version at $ANTELOPE/doc/swd/swd.pdf.

- **Antelope Toolbox for Matlab** – This defines the Antelope ↔ MATLAB interfaces, both Datascope and ARTS, as developed by Dr. Kent Lindquist. Kept up-to-date. Must reading for those wanting to access Antelope data objects from MATLAB. PDF version at $ANTELOPE/doc/matlab/Antelope_Toolbox_for_Matlab.pdf.

- **Analysis of Seismograms and Event Location (dbloc2)** – This is a relatively up-to-date (2002) HTML-only tutorial for using the Antelope analyst interactive review program for seismic events, dbloc2. Written by Dr. Gary Pavlis and used in a seismology lab class. Note that this is a URL link back to the Antelope User Group web site at Indiana University.

- **Adding New Velocity Models to dbgenloc** – This is a HTML-only tutorial for creating new velocity models for the contributed event location software, dbgenloc, written by Dr. Gary Pavlis. Note that this is a URL link back to the Antelope User Group web site at Indiana University.
Documentation: guides

• There are three quick reference guides that are usually distributed with the Antelope CDs:
  – They are patterned loosely after the O’Reilly “In a nutshell” series.
  – They organize software modules into logical groupings and they are meant to provide users and developers with quick and terse memory “joggers” for
    • Program and script usage syntax
    • Directory structures
    • Expression syntax
    • Environment variables
    • Database schemas
    • Time formats and time format codes
    • Script language extensions
    • Datascope constants
    • Antelope toolbox functions and subroutines
    • Antelope makefiles

• These guides are also available in PDF form
  – Antelope 4.6 User Reference Guide – This documents Antelope at a user level. PDF version at $ANTELOPE/doc/refguides/user.pdf
  – Antelope 4.6 Scripting Reference Guide – This documents Antelope for those who want to write their own scripts. PDF version at $ANTELOPE/doc/refguides/scripting.pdf
  – Antelope 4.6 Programmer Reference Guide – This documents Antelope for C and FORTRAN-level software developers. PDF version at $ANTELOPE/doc/refguides/c.pdf
Documentation: source-code examples

- There are a number of source-code examples that can be used by developers as templates. These examples are complete with standard Antelope Makefiles and it should be possible to “build” these source code examples into actual runable software modules (assuming you have the proper compilers and load linkers installed on your system).

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ANTELOPE/example/c/</td>
<td>Contains a number of example programs written in C. Addresses both Datascope and ARTS examples.</td>
</tr>
<tr>
<td>$ANTELOPE/example/fortran/</td>
<td>Contains a number of example programs written in FORTRAN.</td>
</tr>
<tr>
<td>$ANTELOPE/example/perl/</td>
<td>Contains a number of example perl scripts. In particular NEIC_qed2 will download the latest qed catalog from the USNEIC web site into a Datascope database.</td>
</tr>
<tr>
<td>$ANTELOPE/example/shell/</td>
<td>Contains a number of example shell scripts.</td>
</tr>
<tr>
<td>$ANTELOPE/example/tcltk/</td>
<td>Contains a number of example tcl scripts.</td>
</tr>
<tr>
<td>$ANTELOPE/example/vogle/</td>
<td>Contains a number of example tcl scripts that use the vogle extensions.</td>
</tr>
<tr>
<td>$ANTELOPE/example/predict/</td>
<td>Contains multiple implementations of the same function in various languages and scripts for examples in the Software Development with Antelope and Datascope: A Tutorial tutorials.</td>
</tr>
<tr>
<td>$ANTELOPE/example/datascope/</td>
<td></td>
</tr>
</tbody>
</table>
Basic Concepts
Basic Concepts: UNIX Paradigm

• **Antelope** embraces the UNIX “paradigm”. What does this mean?
  – **Antelope** is characterized by a large number of small, standalone, very modular programs and scripts, each doing a specific and usually narrow task *in a generalized fashion*.
  – The vast majority of **Antelope** software modules are “graphic-less” with no GUIs or interactive input.
  – Most **Antelope** software modules are configured at run-time through their UNIX command line arguments and through a set of ASCII text files, in a standard format, known as **Antelope** parameter files.
  – **Antelope** software modules support and make use of standard UNIX i/o mechanisms like i/o redirection, inter-process piping and TCP/IP sockets.
  – **Antelope** follows UNIX standards in documentation (UNIX man pages), file naming conventions, directory naming conventions, etc.
Basic Concepts: UNIX Paradigm

- All of the Antelope modules conform to input and output standards that promote a very high level of interoperability.
  - Standard output → standard input via UNIX pipes
  - Datascope database as an interface between processes
  - ARTS ORB as a real-time interface between processes
- This allows many Antelope modules to be run together in combinations that can effectively implement complex data processing tasks.
- This also eases the software development task.
- HOWEVER:
  - You have to be able to type from a shell prompt in a terminal window.
  - You have to be able to edit text files using your editor of choice.
  - Configuring Antelope to handle a particular complex data processing task requires certain skills and knowledge
    - Knowledge of Antelope modules
    - Knowledge of how to configure these modules
Basic Concepts: UNIX Paradigm

Example: Antelope processing network used in field computers currently by Servizio Sismico Nazionale in Italy
Basic Concepts: **Antelope** parameter files

- **Antelope** parameter files, called “pf” files, are pervasive and are a flexible but standard way for specifying program parameter and configuration information.

- A good and succinct description of pf files is contained in chapter 4 of the *Software Development with Antelope* tutorial.

- Also look at pf(5), pfecho(1), pfwhich(1)

- Let’s look at an example:
Basic Concepts: **Antelope** parameter files

- **Name-value specifications**: # signifies in-line comments
- **Associative array specification**
- **Nested associative array**

- **Associative arrays** are generalized string arrays with string indices. In a `pf` file they are defined by:
  ```
  arrayname &Arr{
  name1 value1
  ...
  }
  ```

- **Tables** are generalized and ordered string lists with integer indices. In a `pf` file they are defined by:
  ```
  tablename &Tbl{
  value1
  ...
  }
  ```

April 2004
Basic Concepts: *Antelope* parameter files

- **pf** files are always read and evaluated by *Antelope* programs, using a standard tool from the *Antelope* toolbox, in the following fashion:
  - **pf** filename syntax is *pfname*.pf (they always end in .pf)
  - **pf** names (e.g. *pfname*) can be specified by many *Antelope* programs in their command line arguments. The *pfname* for a particular program usually defaults to the program name. Note that **pf** names DO NOT contain the .pf part of the filename, nor should they ever contain file pathnames (i.e. anything with a /). They are NOT UNIX filenames.
  - **pf** objects are parsed from files by the *Antelope* toolbox in a standard way that involves evaluating a series of UNIX files in a set of specified file directory paths:
    - The pathnames used for evaluating the **pf** files are specified in the **PFPATH** environment variable, which defaults to $ANTELOPE/data/pf::
    - **pf** files, *pfname*.pf, are evaluated successively in each of the directories specified in **PFPATH**. Parameters in files later in the search path overwrite those from files earlier in the search path.
Basic Concepts: **Antelope** parameter files

You can explore parameter files with `pfecho` and `pfwhich`:

- **pfecho** – interactive version of a program for "echoing" parameters parsed from a set of parameter files. It takes a **pf** name as an argument which, for this case, is **dbgrassoc**

- **assoc_method** is one of the simple parameter names and **tttaup** is its value

- **grid_params** is the name of an associative array

  - references the **teleseismic** element of the **grid_params** associative array, itself another associative array

- **pfwhich** → lists the UNIX files that were processed to compose the **dbgrassoc pf** object
Basic Concepts: SEED vs. CSS

- **Antelope** embraces three major standards that have been in use within the seismological community – 1) SEED, 2) the database schema, known as *css3.0*, that has been used for many years by the international nuclear monitoring community and 3) and the CD-1 standard used by CTBTO.

- Our goal is to design **Antelope** so that these standards can co-exist (this concept is at the heart of **Antelope**).

- The biggest incompatibility between these standards is with the basic waveform channel naming conventions:
  - SEED uses four distinct network, station, channel and location ASCII codes
  - *css3.0* uses two distinct station and channel ASCII codes
  - The maximum allowed sizes of these codes are different between the two standards
  - There are some additional conventions imposed on specific codes that are different between the two standards (e.g. SEED channel codes)

- The **Antelope** toolbox provides conversion tools that transform back and forth from SEED naming conventions to *css3.0* naming conventions. See trmapnames(1), foreign(3).
Basic Concepts: Specification of time

• Absolute time is an essential variable within any environmental monitoring system.

• How is time represented internally within **Antelope**?
  – Time is represented internally as an 8-byte, double precision, floating point epoch time in seconds referenced to 1970/01/01 00:00:00 at Greenwich. It can be thought of as a floating point version of the standard UNIX epoch time.
  – This representation of time was adopted from the nuclear monitoring community, from the **css3.0** standard.
  – As with the UNIX and **css3.0** epochs, **Antelope** epoch times **DO NOT** include leap seconds (they do include leap days)
  – The current 8-byte binary representation of time has a resolution of ~1 microsecond.

• The **Antelope** toolbox provides a broad range of time conversion tools that transform back and forth from epoch time to “human” time (i.e. year-month-day-hour-minute-second). See chapter 3 in the **Software Development with Antelope** tutorial and epoch(1), epoch(3).
Basic Concepts: Using UNIX regular expressions

• Many Antelope software modules make use of “regular expression matching” as a powerful and very general method for specifying complex processing parameters in a compact manner.

• Antelope adheres to the UNIX regular expression standard for regular expression syntax.

• If you are not familiar with UNIX regular expressions, then it would be helpful to become familiar. Most UNIX handbooks will cover this. See regex(5) on a SUN system for a good summary.

• The Antelope toolbox provides many tools that that are based upon UNIX regular expressions. See greparr(3), morph(3) and any of the many programs that use UNIX regular expressions in parameter files, such as trdefaults(5) and $ANTELOPE/data/pf/trdefaults.pf
Quick Sampling of Antelope
• Initial setup using rtdemo
• Anatomy of the Southern California Real-Time Demo
• Running the Southern California Real-Time Demo
• A look at an Antelope database
• Where to go from here
Antelope Real-Time Demos

- These are easy-to-run, pre-configured, fully functional Antelope real-time executives that demonstrate many Antelope features including:
  - Acquisition of real waveform data packets into an ORB
  - Standard real-time network processing
  - Waveform and parameter archiving into a Datascope database
  - Command and control of real-time processing
  - Real-time system configuration
  - A simple perl example script using the Antelope toolbox extensions.

- These are good tools for both new users and for experienced developers.
Initial Setup using \texttt{rtdemo}

- Setup the \texttt{Antelope} environment.
- Check the license.
- Run \texttt{rtdemo}

- Chose \texttt{socalif} from the initial “demos” window.
- Chose a directory for installing the \texttt{rtdemo\_socalif} subdirectory.
- The \texttt{rtm} window will pop up along with several message windows. Click \texttt{Dismiss} on the top-most message window to make it go away.

April 2004
Anatomy of rtdemo_socalif

- `<rtroot>/bin/` Home for “special” scripts and programs.
- `<rtroot>/bin/clean` Will clean out the entire `<rtroot>/` directory for re-run.
- `<rtroot>/bin/logmon` Simple example perl script for displaying program log messages.
- `<rtroot>/db/` Home for main archive Datascope database. This will contain all archive waveforms and network processing parameters.
- `<rtroot>/dbmaster/` Home for site and instrument meta-data Datascope database. Contains all site and instrument parameters.
- `<rtroot>/events/` Points to raw disk ORB packet event dumps.
- `<rtroot>/logs/` Home for all program and system log files.
- `<rtroot>/orb/` Home for main orbserver ORB files.
- `<rtroot>/pf/` Home for all program parameter files.
- `<rtroot>/rtsys/` Home for special rt-system Datascope database
- `<rtroot>/state/` Home for program state files used to synchronize stop-start cycling.
- `<rtroot>/rtexec.pf` Main rt-system configuration parameter file.
The `rtexec.pf` file is the master configuration parameter file that controls the program `rtexec`. This file specifies:

1. The programs, complete with command line arguments, that will be run and monitored.
2. The order of program startup and shutdown.
3. The main ORB and database names.
4. A complete set of environment variables.
5. A customized crontab that is independent of the UNIX system crontab.
6. A complete set of system limit parameters (e.g. coredumpsize).
7. E-mail addresses for automatic reports of errors and system statistics.
8. A list of maximum allowed disk resources.
9. A list of user-defined functions that map to the bottom row of `rtm` buttons.

`rtexec` is the master control executive that starts all of the other programs, monitors, re-starts (if necessary), etc. Note that `rtexec` is a graphic-less program. `rtexec` responds dynamically to changes in the `rtexec.pf` file. `rtm` is basically a simple GUI front-end for editing the `rtexec.pf` file.
**Data flow for rtdemo_socalif**

- **stream2orb**
  - dumps ORB wf packets
  - archives waveforms populates wfdisc table

- **orb2db**
  - ORB packets
  - waveform
    - AZ_PFO/CBBLS
  - detection
    - /db/detection
  - pick list
    - /pf/orbassoc
  - event pars
    - /pf/orb2dbt

- **orb2dbt**
  - mag pars
    - /pf/orbmag
  - netmag
    - /db/netmag
  - stamag
    - /db/stamag
  - origin
    - /db/origin

- **orbassoc**
  - travel time grid
  - ttgrid
  - crude event association (time coincidence based)

- **orbdetect**
  - single channel event detection and onset time estimation

- **orbtrigger**
  - travel time-back projection based associator (grid search for candidate hypocenters)

- **dbassoc_rt**
  - associates events from external catalogs

- **orbserver**
  - computes magnitude estimate

- **external catalog**
  -波形データをアーカイブするためのデータベース、foreign keys

- **id server**
  - populates all other tables, also does a “smart” association of event parameters

- **dumps ORB wf packets**
  - stream2orb

- **external catalog**

- **ORB packets**
  - waveform
    - AZ_PFO/CBBLS
  - detection
    - /db/detection
  - pick list
    - /pf/orbassoc
  - event pars
    - /pf/orb2dbt

- **orbassoc**
  - travel time grid
  - ttgrid
  - crude event association (time coincidence based)

- **orbdetect**
  - single channel event detection and onset time estimation

- **orbtrigger**
  - travel time-back projection based associator (grid search for candidate hypocenters)

- **dbassoc_rt**
  - associates events from external catalogs

- **orbserver**
  - computes magnitude estimate

- **external catalog**

- **ORB packets**
  - waveform
    - AZ_PFO/CBBLS
  - detection
    - /db/detection
  - pick list
    - /pf/orbassoc
  - event pars
    - /pf/orb2dbt

- **orbassoc**
  - travel time grid
  - ttgrid
  - crude event association (time coincidence based)

- **orbdetect**
  - single channel event detection and onset time estimation

- **orbtrigger**
  - travel time-back projection based associator (grid search for candidate hypocenters)

- **dbassoc_rt**
  - associates events from external catalogs

- **orbserver**
  - computes magnitude estimate

- **external catalog**