CARMA Developers Tutorial

- Basics (CVS, CARMA, CARMA_PKG, CARMA_TOOLS)
- Building (install_all, configure, make)
- Programming (carma::util:Program, make)
- Debugging (OPT=1, cppunit)
- Exception handling

Details in: `$CARMA/doc/SEDesign.tex`

Basics

- CVS
  - $CVSROOT
  - $CVS_RSH=ssh and ssh authentication
- $CARMA (our CVS-based sources)
- $CARMA_PKG (tar ball repository)
- $CARMA_TOOLS (compiled, via $CARMA)

Building

- One simple script with reasonable defaults:
- `conf/install_all`
  - `carma=$CARMA`
  - `carma_tools=$CARMA_TOOLS`
  - `do_tools=0`
  - `do_carma=1`
  - `do_tbox=0`
- Easy to wrap build scripts

Building

- $CARMA_PKG now also includes carma_cvs.tar.gz, build at 35,000 ft
  - Useful to have :pserver:anonymous@cvs access
- Not everything in $CARMA_PKG is *essential* yet
- Developer disk space needed: (status January 2004)
  - CARMA: 200MB
  - CARMA_PKG: 200MB
  - CARMA_TOOLS: 240MB

Building: conversion to new

- Re-install using install_all
- Re-build:
  - `cd $CARMA`
  - `cvs update`
  - `./configure`
  - `make config clean scripts carma tests`
- No java check yet !

Hierarchical Makefile's

- Automatic dependancy building (.d files) vs. Makefile.rules
- Top level:
  - *make clean incs libs bins tests docs*
- `carma/module/:`
- `carma/correlator/xxx/yyy:`
  - `Deeply nested libraries?`

Command Line User Interface (*class Program, SimpleProgram)*

- Uniform and simple to use
  - all programs understand "--help"
- Programs self-describe
  - `program –keywords`
- Keywords: **program** and **system** keywords
- `keyword=value`
- "`--`" to separate "key=val" from free form

CLUI (cont'd)

- “- -” to separate “key=val” from free form
  - How do we advertise the -options vs. key=val?

CLUI (cont'd)

- $CARMA/doc/Program.tex
  - Needs more essential doxy's in Program.cc

Debugging

- Different sandbox (or edit makedefs)
  - --with-debug
- Command: “make OPT=1”

Exception Handling

- Exception handling
  - dynamic and verbose messages!

“not enough memory” “file could not be opened”

Software Infrastructure needs

- Build system
  - *autoconf* based hierarchical makefile's
  - Tinderbox w/ extensive run-time tests (+dox)
  - CARMA_PKG (/sw/carma_pkg)
  - CARMA_TOOLS
- Linux distributions
  - **Redhat9** vs. RHEL/3 (UML: for experimentalists)
    - Mdk82@UMD and FC1 (2.4.22) appear to work fine too
  - Kernel **2.4.20** vs. 2.4.23 (multi-threading)
  - Compiler **3.2.2** vs. 3.3.2 (ANSI standards)

Todo's

- Proper CST webpage @ mmarray.org
- No global `<config.h>` or `<carma.h>`
- No "make docs" for local doxygen testing
- Does tinderbox look at "make tests" at all?
  - Needs more smarts in tinderbox
- Watch out for long compile times
- Configuration system (e.g. for default keys)
- Daisy-chaining carma's (`carma_{start,end}`)

Todo's (cont'd)

- Make -j
- Make OPT=1 COVERAGE=1... (flavor building)
- Unified carma_ctl start|stop|restart|status.\...
  - Can we live without .csh, .sh, .pl, .py .... versions?
- Binary developer distributions? (carma_tools)
- Other compilers ? (intel8.0 @linux, gcc @sol)