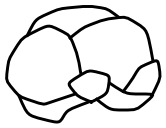


Contact

Research Computing
The Ohio State University
1224 Kinnear Road
Columbus, OH 43212

Consulting

Phone: (614) 292 8492
Internet: trollius@tbag.osc.edu

Copyright

This document is protected by copyright.
Author: GDB

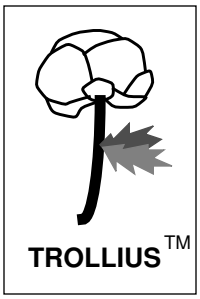
© Copyright 1992 The Ohio State University

Trademarks

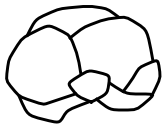
Trollius is a trademark of The Ohio State University and the Cornell Research Foundation.

UNIX is a trademark of AT&T.

X Window System is a trademark of the Massachusetts Institute of Technology.



Ohio Trollius 2.2 Release Notes



Synopsis

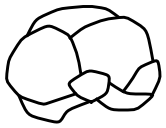
- TROLLIUSBNAIL environment variable
- simplified options to spread tool
- faster map tool, generates two routes

Topology Layer

It is typical for a user to be working on a single multicomputer size and topology for a long period of time. For convenience, all the topology tools check the environment variable TROLLIUSBNAIL for the filename of the boot schema after checking the -c option and before using the default boot schema, bnail.def.

By default, spread will employ frozen and recursive booting techniques. The -z and -e options are recognized, but redundant. Instead, the -Z and -E options direct spread not to use frozen and/or recursive techniques, respectively.

The rewritten map tool has less complexity and uses less memory but it also generates the second, shortest path, non-reconvergent route. See *Network Layer.*



Synopsis

- new vboot command with simplified options
- cast and boot monitoring commands
- trace control and monitoring commands
- new LED control and monitoring commands

Command Layer

As with previous releases, more control and monitoring commands have been added to increase the Trollius reputation of hands-on operation. Debugging capability is inherent in the system and does not rely on any one tool.

vboot

The rewritten vboot command drives the new remote boot daemon. It has less options and less complexity than the previous version. A user might rely on vboot to debug new or experimental hardware, or test a custom port of Trollius. To supplement vboot, a monitoring command for the remote boot daemon has been added.

```
btstate
```

cstate

A monitoring command has been added to look inside the cast daemon and debug multicast, multireel and cast registration problems.

```
cstate
```

trstate, ttrace

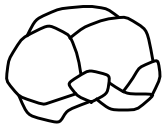
The trace daemon is controlled and monitored with two new commands. The ttrace command is necessary to unload traces from the nodes into files while the trstate command is useful for watching processes produce trace data and detecting trace data overflow problems.

```
trstate
ttrace
```

lestate

The tled command has been revised to suit the overhauled LED daemon and a monitoring command has been added.

```
lestate
```



rtvclr()
rtvrect()
rtvsnake()
rtvtext()
rtvstate()

LED Daemon

A popular demonstration and useful debugging tool has been revived for the 2.2 release. The LED daemon originally provided remote control over LEDs attached to an ITB node. The 2.2 implementation of ledd displays graphical LEDs in an X Window. The runtime library has been redesigned and added to the Buckeye remote library.

rledcc()
rledop()
rlestate()

Trace Daemon

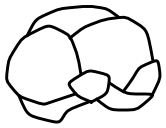
A new daemon, traced, is the central hub for trace collection and transport on each monitored node. It collects traces generated by local node processes and, on request from a remote client, transports them to any node in the multicomputer. Local trace generating processes attach and detach from traced. A monitoring service allows remote clients to follow the amount of trace data from each attached process. The trace daemon can also prompt its local attached processes to unload their current trace data by sending them the Trollius SIGTRACE signal. The trace daemon is completely independent of trace data format. The following functions have been added to the essential services (libt) and Buckeye remote libraries:

rtrattach()
rtrdetach()
rtrfforget()
rtrfget()
rtrsig()
rtrstate()
rtrsweep()

File Data Conversion

The file daemon and runtime library support representation conversion of homogeneous data. The following new flags are accepted by rfopen() and indicate that representation conversion is in effect:

O_DINT	data is all 4 byte integers
O_DFLT	data is all 4 byte reals
O_DDBL	data is all 8 byte reals



Synopsis

- new boot daemon featuring parallel, recursive booting
- pixel map daemon with X Window and G300 implementations
- new LED daemon with X Window implementation
- trace transport daemon
- file daemon data representation conversion capability

Remote Layer

Three new daemons and associated runtime libraries have been added for the 2.2 release. New daemons that are not central to the operating system will typically be found in the Buckeye dungeon. Two of the daemons have graphical user interfaces based on the X Window System and the Athena Widget Set.

Boot Daemon

The boot daemon has been completely replaced for the 2.2 release. It contains two new features to speed booting: the ability to recursively boot multiple nodes in parallel and the ability to boot using the local node's executing system image. The load module relocation service has been removed from bootd. Thus, bootd is not required to support program loading and execution. If an ITB node has no neighbours to boot, it need not run bootd.

The following bootd runtime functions have been removed from the remote library:

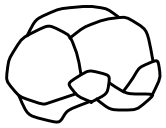
```
rboot()
rpatch()
```

The following bootd runtime functions have been added to the remote library:

```
rbtadd()
rbtboot()
rbtflush()
rbtspread()
rbtstage()
rbtstate()
```

MTV Daemon

The MTV daemon provides basic support for copying pixel maps to display devices. No drawing services are currently provided. There are two implementations of mtvd, one for an X Window display on an OTB node and one for a (G300, for example) graphics controller attached to an ITB node. The following functions have been added to the Buckeye remote library:



Second Route

The cast identifier registration procedure is internally simplified thanks to the availability of an alternative route for each destination in the local route table, maintained by the router (a network layer process). This second route is guaranteed to be the shortest number of hops in the node graph and is guaranteed not to reconverge with other paths originating from the local node.

Basic Message Passing

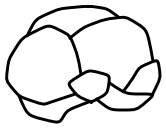
Trollius network message passing is designed to provide a comprehensive base for all communication paradigms. In the 2.2 release, it is recognized that the full needs of many would be served by a much simpler interface than nsend() and cousins. Also, a new user's introduction to parallel computing would be eased by a simpler interface. A new pair of message passing functions show a very basic, easy to understand interface.

```
bsend(struct nmsg *nhead;)
brecv(struct nmsg *nhead;)
```

These functions still take a single parameter, a pointer to a network message descriptor. The programmer uses only the node, buffer, length and flags fields of the descriptor and ignores all others. The flags field is only used to indicate the type of data being sent (integers, single or double precision reals). Unlike nrecv(), brecv() does specify the sender's node identifier.

```
sendb(int nodeid; void *buffer; int length;
     int datatype;)
recvb(int nodeid; void *buffer; int length;
     int datatype;)
```

For programmers who may be more comfortable with parameters versus a message descriptor, a pair of companion functions is provided.



Synopsis

- increased robustness in multi-packet nsend()
- uniform semantics for trecv()
- instrumented nsend()
- new cast daemon with fewer restrictions, more monitoring
- new basic message passing functions, bsend() and brecv()
- second route table with only shortest path, non-convergent routes

Network Layer

Trollius 2.2 includes a few key upgrades to the network message passing system, the lifeblood of everyday programing. Our general commitment in this crucial subsystem is to maintain stability while improving performance and robustness.

Multi-packet Messages

In the previous release, two or more multi-packet messages destined for the same node and event could intermix and result in two garbled messages. The problem has been fixed in 2.2 by placing restrictions on the range of node identifiers (the general range is a full 32 bits) only for nodes transmitting long messages with nsend(). See the nsend() manual page for precise details. In practice, this should be transparent to everyone. In the 2.3 release, the packet mix-up problem will be solved permanently without restricting the node identifiers.

trecv()

The flow control protocol behind tsend() and trecv() has been changed so that the receiver no longer must specify the sender's node identifier. This gives trecv() the same semantics as nrecv() and makes the shift between the transport and network sublayers less error prone.

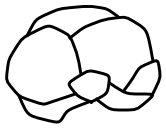
nsend()

Nsend() and nrecv() can generate communication traces in conjunction with the runtime trace system. See *Local Layer*. An important caveat is that trace generation uses the nh_data[7] field in the network message descriptor, making this field unavailable to the programmer if tracing is selected.

castd

The cast daemon has been completely rewritten to decrease complexity, remove caveats and add debugging capability. Most of the increased functionality concerns multireeling. It is now possible to effect multiple concurrent multireels on the same cast identifier but different event. A concatenate function has been added to the list of built-in combining functions. A remote layer function has been added to monitor and debug multireeling as well as cast identifier registration.

`rcstate()`



Synopsis

- trace collection and transport runtime system
- increased robustness in OTB kernel interface
- OTB kernel management of kill file for tkill tool

Local Layer

A trace collection and transport system, independent of trace format or contents, has been integrated with the local layer. Two new flags tell the runtime system whether or not traces are to be generated. The master switch, RTF_TRACE, causes memory to be allocated for runtime trace storage and enables the secondary switch, RTF_TRSWITCH, which allows traces to be stored.

Traces

A simple set of six traces has been designed and implemented for Trollius 2.2. Two of the traces are stored by the network layer and the remaining four are at the programmer's discretion through the following functions (see the manual pages for a complete description of each trace generator):

```
ktrstart()
ktrstop()
ktrcflow()
ktrdflow()
```

The master switch also installs a signal handler for the SIGTRACE signal[1] and a cleanup function. Both cause the total current contents of the trace buffers to be transported to a local trace daemon. See *Remote Layer* for details on the trace daemon.

OTB Kernel Interface

The OTB interface between the kernel and its clients has two UNIX implementations: UDP and message queues. The error reporting and robustness of both implementations has increased in 2.2. The user may see a somewhat cryptic error message as shown below.

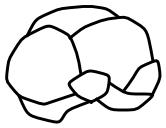
```
client[123]: error message
```

The word, client, is constant and simply means a Trollius process with the identifier given in brackets.

Kill File

When the OTB kernel starts, it creates a kill file with the filename equivalent to the user's login name. The process ID of each new kernel client is appended to the file. The file is used and deleted by tkill. See *Toolkit Layer.*

1. SIGTRACE replaces SIGROUTE for the 2.2 release. SIGROUTE may be added again in a later release.



The obsolete options are:

```
memstart
nodeid
lock
portable
```

The current options are:

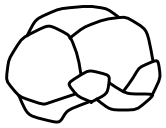
OPTION	ARG TYPE	DEFAULT
stack	integer	0x1400
priority	integer	1
share	0/1 flag	0
onchip	0/1 flag	0
delay	integer	0
topology	string	""

sboot The sboot tool has been updated to drive the new cboot tool. The communication pipe between the two tools no longer uses stderr, and is no longer a hazard to developers who may need to debug customized versions of cboot.[1]

hboot The hboot tool no longer creates a lock file with the identifier of each process it starts. A more comprehensive file of Trollius processes (now called the kill file) is maintained by the Trollius OTB kernel. See *Local Layer*. Hboot options relating to the lock file have been removed.

tkill While the responsibility for creating the kill file now rests with the local layer, the tkill tool still kills OTB Trollius processes and deletes the kill file. It no longer relies on the fact that the first process ID in the kill file is the kernel and that killing the kernel will normally kill all its client processes and release all OTB Trollius resources. Tkill now performs the same task for every process in the kill file, redundant with the kernel. If the kernel should abnormally fail, tkill will still make sure that no stray Trollius processes (and any associated resources) are left on the OTB node. The -d option of tkill will report on any process that passed through this double layer of death.

1. Since the 2.2 cboot tool is machine portable, the potential need for a customized version is greatly diminished.



Synopsis

- portable export, cboot and thaw
- fish becomes tfish
- cboot completely overhauled
- sboot and hboot overhauled
- minor changes in process schema grammar
- tkill rewritten, more effective

Toolkit Layer

The toolkit layer receives its second major overhaul in as many releases. The disadvantage of toolkit layer change is that all custom ports of Trollius must be reassessed and occasionally redesigned. The advantage is that more of the lowest layer is becoming portable and the porting and maintenance effort for all developers and administrators is steadily decreasing.

tfish

There is now only one machine dependent program, tfish, and one module, link.c, in the toolkit layer. All other programs are machine independent. The name of fish (a name with a history) has been changed to tfish in order not to clash with /usr/games/fish on some UNIX systems. An internal implementation policy change precipitated this move - Trollius no longer searches for OTB programs and executes their full pathnames. Instead, it relies on the search directory list in the environment to locate sub-tools. This policy also affects the topology layer.

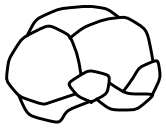
cboot

The complicated cboot tool has been completely overhauled, partially to simplify usage and partly to support the new 2.2 booting technique. Only the base address and stack size must be provided. All hope for fully capable UNIX device drivers in the multicomputer market has been lost and cboot now has an explicit option to switch from plain output file to hardware link device. All the cboot changes can be fully appreciated only by comparing old and new manual pages. See *Remote Layer* for a description of the 2.2 booting technique.

The patch tool is no longer supplied with Trollius. Its functionality has been incorporated into the cboot tool.

Process Schema

The process schema grammar has been altered slightly to remove obsolete capability and to standardize global and local option specification. A local option appears within the braces of a process entry and applies only to that entry. A global option applies to all processes in the schema, unless locally overridden. The list of current and obsolete options appear below. Note that the same syntax is used both locally and globally.



General Notes

Dungeon

The source release has undergone a major rehabilitation that affects installation and portability. The Trollius Project develops, maintains and delivers software in three separate dungeons (directory structures):

Trollius	the essential operating system
Buckeye	the supplementary operating system
LSTT	the Logical Systems Transputer Toolset

The big change from the previous release is that LSTT is no longer an integrated part of the Trollius dungeon. It is a separately installed software product. Of course, building the complete Trollius dungeon leaves no choice about installing an ITB compiler. See the *Trollius Porting Guide* and the *Trollius Installation Guide* for detailed information about the new software organization.

Environment

Trollius programs that search for files will consult the environment variable TROLLIUSHOME for the pathname of the installation directory. This pathname takes precedence after any command line option but before the original installation directory pathname. The advantage of TROLLIUSHOME is that the installation directory (which contains the Trollius binaries, libraries and configuration files) can be moved after it is built. A binary release of Trollius can be placed anywhere.

ParaGraph

A performance trace generation, collection and transport system has been added to Trollius 2.2. The raw data produced by this subsystem is converted by a tool called trade to a format acceptable to the ParaGraph[1] performance visualization system. ParaGraph was constructed at Oak Ridge National Laboratory by Michael Heath and Jennifer Etheridge. This Trollius performance monitoring project was accomplished in cooperation with Laboratoire Informatique du Parallelisme at Ecole Normale Superieure de Lyon.

`trade`

ParaGraph is freely available via Netlib. Send an electronic mail message to netlib@ornl.gov with the message body “send index from paragraph”.

1. Michael T. Heath, Jennifer Etheridge, “Visualizing the Performance of Parallel Programs”, IEEE Software, Sept. 1991, Vol. 8, No. 5, pp 29-39

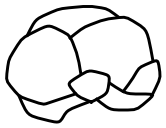


Table of Contents

How to Use This Document 2

General Notes 4

- Dungeon 4
- Environment 4
- ParaGraph 4

Toolkit Layer 5

- tfish 5
- cboot 5
- Process Schema 5
- sboot 6
- hboot 6
- tkill 6

Local Layer 7

- Traces 7
- OTB Kernel Interface 7
- Kill File 7

Network Layer 8

- Multi-packet Messages 8
- trecv() 8
- nsend() 8
- castd 8
- Second Route 9
- Basic Message Passing 9

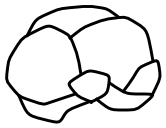
Remote Layer 10

- Boot Daemon 10
- MTV Daemon 10
- LED Daemon 11
- Trace Daemon 11
- File Data Conversion 11

Command Layer 12

- vboot 12
- cstate 12
- trstate, ttrace 12
- lestate 12

Topology Layer 13



How to Use This Document

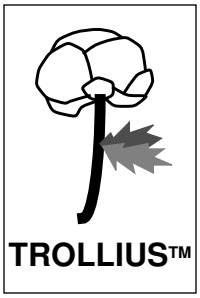
This document covers all aspects of programming and operation specific to the 2.2 release of Ohio Trollius. Changes are described with reference to the previous release. The first section describes companion software products and issues global to Trollius. The subsequent sections deal with changes at each layer of the operating system, from bottom to top.

This document is user oriented. It does not give much insight into internal changes. It does not detail every option and capability of every new command and function. The manual pages cover all the commands and functions in great detail and are meant to supplement this document.

Each section starts with a brief synopsis of the new or changed commands or functions covered by the section.

We have kept the font and syntax conventions to a minimum.

`code`	This font is used for things you type on the keyboard or see printed on the screen. It is used in code sections and tables but not in the main text.
<symbol>	This is a symbol used to abstract something you would type. This convention is used in commands and in the NaIL grammar but not for functions. Don't confuse this with the angle brackets that are actually part of the NaIL grammar or the angle brackets surrounding header files, part of the C preprocessor.
Section	Italics are used to cross reference another section in the document.



Ohio Trollius 2.2 Release Notes

Trollius is a programming environment for distributed memory MIMD parallel computers, aka multicomputers. This document covers all aspects of programming and operation specific to the 2.2 release of Ohio Trollius. Changes are described with reference to the previous release. The theme of 2.2 is visualization and robustness. For the first time Trollius contains higher level productivity tools with X Window based user interfaces. There have been extensive internal changes to the operating system to enhance robustness, but also performance, portability and maintainability.

For information on installing and administering Ohio Trollius 2.2, consult the *Trollius Installation Guide*.

Ohio Supercomputer Center

Research Computing
The Ohio State University

Document Series 2/7 **January 28, 1992**