THE CRYSTAL NUGGETMASTER
Part II of the First Report on
THE CRYSTAL PROJECT

by

Robert Cook
Raphael Finkel
Bob Gerber
David DeWitt
Lawrence Landweber

Computer Sciences Technical Report # 500

April 1983

The Crystal Nuggetmaster

Part II of the First Report on The Crystal Project

Robert Cook

Raphael Finkel

Bob Gerber

David DeWitt

Lawrence Landweber

CONTENTS

1. Introduction to Crystal

The University of Wisconsin Crystal project was funded starting in 1981 by the National Science Foundation Experimental Computer Science Program to construct a multicomputer with a large number of substantial processing nodes. The original proposal called for the nodes to be interconnected using broadband, frequency-agile local network interfaces. Each node was to be a high performance 32 bit computer with a approximately 1 megabyte of memory and floating-point hardware. The total communications bandwidth was expected to be approximately 100 Mbits/second.

During the first year of the project, these specifications have been refined. We have decided to buy approximately 40 node machines, each a VAX-11/750. The interconnection hardware will be the Proteon ProNet. Currently, the ProNet is available in a 10 Mbits/second version. We have contracted with Proteon to increase the effective bandwidth to 80 MBits/second.

1.1. Software Overview

The purpose of this hardware is to promote research in distributed algorithms for a wide variety of applications. In order to provide different applications simultaneous access to the network hardware, we have designed a software package called the *nugget* that resides on each node. In brief, the nugget provides the following facilities:

1. The nugget enforces allocation of the network among different applications by virtualizing communications within partitions of the network. These partitions are established interactively through a host machine.
2. Backing store is shared among the nodes by nugget facilities to virtualize disks.
3. Interaction between the user and individual machines is provided by the nugget facility of virtual terminals.

4. Initial loading, control, and debugging of programs on node machines is controlled by nugget software.

The Charlotte operating system is designed to provide standard interactive operating system support within a Crystal partition. The Charlotte *kernel* provides

1. processes
2. multiprocessing
3. inter-process communication that hides node boundaries
4. mechanisms for scheduling, store allocation, and migration.

All policies in Charlotte are concentrated in *utility processes*. They are designed so that each such process controls a policy on its own set of machines. The set may range in size from one machine to the entire partition. The processes that control the same resource on different machine sets communicate with each other to achieve global policy decisions. The utilities that have been designed so far include a switchboard, a program starter, and the file server. In addition, there are non-policy utilities for command interpretation and program connection.

We expect that Crystal will be used for a wide range of applications. Currently research is underway in distributed operating systems, programming languages for distributed systems, tools for debugging distributed systems, multiprocessor database machines, parallel algorithms for math programming, numerical analysis and computer vision, and evaluating alternative protocols for high performance local network communications.

All Crystal software is being written in a local extension to Modula. Our compiler, which runs on a VAX running Berkeley 4.1 Unix, employs syntactic error correction through the FMQ algorithm and is quite fast. The code it generates compares well with that produced by the C compiler.

1.2. Phases of the project

The first phase of the project was dedicated to defining both the hardware and the software. This phase ended in December 1982. Decisions were reached concerning both the node machines and the interconnection devices. The node machine decision was difficult. We had to balance our concerns for reliability, availability, speed, and cost. The machine we chose, the VAX-11/750, although not as fast as others we investigated, had the advantage of being a known architecture for which our Modula compiler already generates code. The Proteon network is currently available. We have been using this network to interconnect our Unix VAX machines and have found it to be extremely reliable.

During the first phase, the nugget was specified and a prototype implementation was completed on a network of eight Digital Equipment PDP-11/23 computers connected by the Megalink CSMA broadband network manufactured by Computrol. Charlotte was also specified and the kernel debugged on this network.

The second phase of the project has just gotten underway. We are finalizing the nugget specifications, which changed in minor ways when we decided that the node machines would be VAXen. The nuggetmaster, which controls the partitions, has also been specified. Charlotte is undergoing debugging of the utility processes. During this phase, which lasts until July 1984, we will transfer the nugget and Charlotte to the node machines and modify them as necessary for the ProNet. Charlotte will be modified to fit with the nugget. (Until now, they have been developed independently.) The utility processes will be supplemented with login and authentication processes, and the file system will be converted to use Crystal disks instead of a file system on the host machine. We plan to have a production, stable operating system by the end of this phase.

The third phase of the project will see large-scale applications actively pursued. Some of this work will start during the second phase. We also expect to re-evaluate the hardware decisions at some point during this phase. There is some reason to expect that frequency-agile modems will be available that will make communication within each partition truly independent of communication within other partitions. Each partition will be able to use its own set of frequencies. Work with optical fiber technology for computer interconnection is also underway at various laboratories around the country. Within five years, impressive bandwidths should be available, reaching into the gigabit/second range. We will continue to monitor progress in this area.

1.3. This report

The purpose of this report is to describe the current state of the design and implementation of the Crystal project. It is intended for readers who have no familiarity with Crystal and wish to see the design decisions that have been made. It is also intended for implementers who need a coherent and reasonably complete specification in order to interface various parts of the project. This dual readership requires us to repeat ideas, first presenting them in an overview fashion, and then diving into tedious details. We urge the reader to skip over those parts of the document that are not at the right level of detail. This report is divided into several documents.

This document discusses the design of the nuggetmaster. The nuggetmaster is a program employed by users for building and destroying virtual disks, allocating and deallocating partitions, and controlling the computation within a partition. It is composed of two logical parts, a *command interpreter* program and a *resource monitor*. The command interpreter defines the user interface to the Crystal multicomputer. The command interpreter can be invoked from any host machine as /usr/local/nci. Current host machines include two VAX-11/780 computers called "crystal" and

"uwvax". Node machines are partitioned under the control of a resource monitor process running on a single host. The command interpreter acts identically whether or not the resource monitor is available locally on the host. The resource monitor process also controls the allocation of disk resources. These two processes communicate through the communications protocol process that is replicated on each host.

2. The command interpreter

The command interpreter parses commands in much the same way as the Unix shell. The command set available to the user of the command interpreter is divided into five groups. Each command can be invoked with the "help option" to display online documentation for syntax and usage. The help option for all commands is specified by a lower case letter h, as in "command_name -h".

2.1. Resource allocation

Resource control commands are used to create and destroy virtual disks and partitions.

Virtual Disk Allocation

The user invokes the "newdisk" command (syntactic details later) to create a virtual disk. The "freedisk" command releases a virtual disk. Two kinds of virtual disk may be created. The first type, a *hardware-backed disk*, is allocated in units of cylinders on a physical disk attached to a node. The second type, a *software-backed disk*, is associated with a file on the host computer. During allocation, the nuggetmaster gives each virtual disk a *virtual disk identifier* in the range 0 .. MaxVirtualDisk by which the user may refer to this virtual disk in the future. This identifier is unique across partitions. We only describe the facilities for allocating virtual disks in general terms here; details are postponed to the end of this section.

The following allocation process is followed for hardware-backed disks: A user specifies the size in megabytes that will be required by the new virtual disk. This size request is converted by the nuggetmaster into a request for an appropriate number of cylinders. The nuggetmaster then allocates the required number of cylinders on a physical disk having enough available contiguous cylinders. The user may directly select the physical disk to support the requested virtual disk by specifying the identifier of a desired physical disk. The user may constrain the choice of physical disk by forcing it to be the same as that used for some other virtual disk or by forcing it to be different. In this manner a user may either ensure that distinct virtual disks are clustered on the same physical disk or prevent clustered allocation of virtual disks on physical disks.

Software-backed disks are intended for transferring data between files on a host computer and hardware-backed disks. Transfers can be performed by a client routine; it alternately reads from one virtual disk (say the software-backed one) and writes to the other one (the hardware-backed one). Alternately, the nuggetmaster "copy" command can be used to transfer data. This command provides a means both to initialize and dump the data on hardware-backed disks.

Allocating a partition that contains software-backed disks creates a background process owned by the user. This process runs on the user's current host machine as an extension of the nuggetmaster. This process provides the nuggetmaster interface to software-backed virtual disks. This host virtual disk server process is terminated when a partition is deallocated. A client specifies access to software-backed disks in an identical manner to that used for hardware-backed disks. For software-backed virtual disk accesses, the nuggetmaster will map the specified access into a comparable operation on a sequential file on the host machine associated with the virtual disk. Such accesses must be within the range of operations permitted by the protection

rights for the user associated with a host file.

Partition Allocation

Partitions are allocated as sets of nodes and are associated with virtual disks. The user invokes the "new" command (syntactic details later) to create a partition. This command includes the name of the *partition request file* that specifies the partition. If all the specifications in the partition request file can be satisfied, an appropriate partition is allocated. The allocated partition is designated by a unique integer identifier. Otherwise the request is denied and the user informed why the request could not be met.

A partition request file is a sequence of descriptor records surrounded by parentheses:

request file ::= ['(' descriptor record ')'] ...

The partition must satisfy each descriptor record in the file. The first descriptor record tells the version number of the request file, that is, it identifies the version of format specifications that were used to create the request file. Another descriptor record declares the virtual disks for the partition. The remaining descriptor records identify the attributes of the nodes. Each of these functions will be described shortly. The help documentation for the "new" command displays the range of values appropriate to each of the option selectors.

All descriptor records are lists of option names and selections for those options, separated by semicolons:

descriptor record ::= option_name = selector
[; option_name = selector] ...

Blanks and tabs are optional in descriptor records between tokens.

Version Number Descriptors

The first descriptor record indicates which version of the command interpreter specifications is assumed for the rest of the file. Out-of-date request files cause warning messages or are refused entirely. As request-file formats change, test versions of the specifications will use version numbers that have not yet been released; while these specifications are under development, production versions still work. An example of a version descriptor is:

(version = 1)

Currently active versions are described by the manual documentation for the "new" command.

Virtual Disk Descriptors

A request file may have a descriptor record to specify the virtual disks shared by the nodes of the partition. This record binds virtual disk identifiers, which are assigned during virtual disk allocation, to logical disk numbers, which are assigned during partition creation. Logical disk numbers are unique only within partitions and serve as local names that have a common meaning within a partition. All references to the same logical disk by different nodes within the same partition are translated into references to the same virtual disk.

The selector values for the "vdisk" option specify virtual disk identifiers. Logical disk numbers are assigned on the basis of position within a descriptor record, counting from 0. Here is how a descriptor record might specify the virtual disks for a partition:

(vdisk=23; vdisk=7; vdisk=31)

This descriptor has the effect of associating virtual disk 23 with logical disk 0, virtual disk 7 with logical disk 1, and virtual disk 31 with logical disk 2.

Node Descriptors

After the version and the virtual disks have been described, the remaining descriptor records specify the individual nodes of a partition. Some options select nodes on the basis of certain attributes. For example, the "mem" option specifies memory requirements. Other options associate attributes with a node once it has been selected. For example, the "vterm" option specifies the number of virtual terminals to be associated with a node. Other options both select nodes and associate attributes with them. For example, the "pterm" option specifies a number of physical terminals.

Generally, one descriptor record is used for each node. However, a number of nodes having identical attributes may be covered by one descriptor record. Here is such a "joint" descriptor record:

```
( count = 3; mem = 500; pterm = 2; bind = test.o )
```

This descriptor record requests any three nodes. Each must have at least 500KB of memory and two physical terminals. The file test.o is associated with each of the three nodes, that is, test.o forms the basis for a load module to be downloaded onto all three machines.

Partition Request File Options

Here is a complete list of the options recognized in descriptor records of a partition request file. Named constants are used to characterize the appropriate range of values for the option selector values. The defined values for these constants can be displayed by using the "show -n" command.

bind: The selector specifies an object module name. This option associates that object module with the node(s) described by this descriptor record. This module will eventually be downloaded into that node, after it is linked with

the nugget program and a logical-to-physical resource map.

count: This option specifies a repetition count. It indicates that this descriptor record specifies characteristics of a number of nodes that have identical attributes. Selector values should be in range 1..Max_node. It is possible to specify a range of values for the count option. For example, count = 3 .. 10 indicates a request for at least 3 and no more than 10 node machines. This is the only option where an explicit range of selector values is allowed.

dev: This option selects an appropriate node by specifying devices that must be attached to it. Selector values must be legal device names. A current list of such names can be found by invoking "new" with the help option.

mem: This option selects an appropriate node by specifying a minimum main store requirement in kilobytes. Selector values should be in the range 1..Max_mem.

node: This option selects a particular node by naming it. The selector value should specify a physical node identifier in the range 1..Max_node. This option is meaningless with the count option.

pterm: This option selects a particular node by specifying a minimum number of of physical terminals. This option also associates an attribute with a successfully selected node: the number of accessible physical terminals. Once a node is selected for a partition, one of its attributes indicates the number of physical terminals accessible. This number may be less than the actual number of physical terminals attached to a node. Selector values should be in the range 1..Max_pterm.

vdisk: This option specifies the virtual disks that will be accessible to a partition. Only one descriptor record should have vdisk options, and it should contain no other options. Selector values should be virtual disk identifiers in the range 1..Max_vdisk. The virtual disks are given logical names in the order

that they appear in the descriptor record.

version: This option specifies the version of command interpreter specifications that the request file is following. This option should be the sole element in the first descriptor record. Selector values are in the range Oldest_Valid_Version_Id .. Newest_Valid_Version_Id.

vterm: This option specifies the number of virtual terminals that will be associated with a node. By default, every node has a virtual terminal that serves as a console. This option can be used to associate additional virtual terminals with a node. No physical terminals need be attached to a processor for virtual terminals to exist. Virtual terminals are described further in the section on virtual terminal commands. Selector values should be in the range 1..Max_vterm.

xdisk: This option specifies a node by naming a hardware-backed virtual disk implemented on a physical disk attached to that node. The selector value for this option must be the identifier of a hardware-backed virtual disk owned by the user. Not only is the specified node added to the partition, but the partition will have exclusive access to that physical disk. Exclusive control does not imply unlimited access to a physical disk; client programs in the partition are only allowed to access those virtual disks named in the "xdisk" option. Protection for the virtual disks of other users is ensured by the nugget. During the period that a user has exclusive access to a physical disk, other users cannot include any virtual disks on that physical disk in new partitions.

The effect of the "xdisk" option is only to assert control over a particular node to which a physical disk is attached. The vdisk command must also be used to make a particular virtual disk accessible to the partition. The "xdisk" option prevents other partitions from seeing virtual disks residing on the exclusively controlled physical

disk. For this reason, a time limit is imposed on the life of partitions exerting such exclusive control. Any partition that does not voluntarily release exclusive control within the specified period of time will be automatically deallocated, thereby releasing the physical disk for public use. This time limit is currently defined to be that time of day when files systems on the host machines are archived.

Several "xdisk" requests may be made for a partition. If more than one of the specified virtual disks resides on the same physical disk, only the first use has an effect. Exclusive disk acquisition is provided so that performance studies can be made in partitions without interference from other partitions. A request specifying the "xdisk" option will fail if any other partition is allocated with a virtual disk on the indicated physical disk. The failure notice will name the user holding exclusive access to prevent abuse of exclusive allocation.

2.2. Resource Control Commands

The following commands are presented interactively by the user to the command interpreter:

free partition_id [-h]

The specified partition is released. The nugget master will only release partitions that are owned by the user. All nodes of a released partition become eligible for reallocation to other partitions and are reset to the boot state. Virtual disks are disassociated from the partition, but they are not deallocated.

freedisk [virtual_disk_id] [-h]

The specified virtual disk is deallocated. If the virtual disk is backed by hardware, then the cylinders it is occupying are released. This command fails if the virtual disk is accessible in any currently allocated partition. In this case, the partition must be released before the virtual disk can be

destroyed. If the released disk is a software-backed virtual disk, then the nugget master will remove the virtual disk from its tables, but the file associated with the virtual disk will be left intact.

new [filename] [option]

A new partition is allocated. The partition is described in the partition request file "filename". The specifications in that file are compared against the characteristics of available nodes. The number of allocated machines and a partition_id are returned to the user if the request is successful. If a partition request cannot be satisfied, the reasons for failure will be returned to the user. The recognized options are:

-i A request file is built interactively. The nugget master prompts the user for the information necessary to build a partition request. The resulting partition request can be written out to a file for later use. This request file will be automatically formatted according to the specifications for a partition request file and can be used for subsequent non-interactive partition requests.

-p The request is kept pending if it cannot be satisfied immediately, but might be satisfied eventually. The nuggetmaster will inform the user when the pending request has been granted, either directly if the user is still talking to the command interpreter or via the mail facility otherwise.

-h Documentation associated with the "new" command is displayed.

Specifics about a newly allocated partition can be obtained be using the "show" command, described below.

newdisk [options] ...

A virtual disk is allocated. If the request can be granted, the user's disk allocation is updated. The following information is reported for hardware-backed virtual disks: the virtual disk identifier, the number of cylinders granted, and the name of the physical machine on which the disk is attached. If a request cannot be satisfied, the reason for failure will be reported. The "show" command, described below, can be used to examine virtual disk allocations. The "newdisk" command recognizes two distinct groups of options. The first group of options deal with hardware-backed disks:

-a virtual_disk_id ...

The "a" (for "apart") option forces the new virtual disk allocation to a physical disk distinct from any physcial disk supporting a virtual disks in the distinction list. This option can be used to force a "nonclustered" allocation of hardware-backed virtual disks.

-c virtual_disk_id

The "c" (for "cluster") option forces the new virtual disk allocation to the same physical disk occupied by the virtual disk in the distinction list.

-p physical_disk_id

The "p" (for "physical") option directly selects the physical disk to support the requested virtual disk.

size

This option specifies the size in megabytes required by the virtual disk. Virtual disks are allocated in units of physical disk cylinders. The megabyte size request is converted to a request

for cylinders. This option must be specified for hardware-backed virtual disks.

The following option specifies a software-backed virtual disk.

-f filename

The given filename names a file on the host computer. This file will be associated with a virtual disk identifier.

show [option]

This command displays data maintained by the nuggetmaster. Lower case letter options specify data specific to the current user, that is, the invoker of the command. Upper case letter options specify global data.

-d The user's virtual disks are described, both hardware- and software-backed.

-D All allocated virtual disks are described.

-h A help page is displayed describing the "show" command.

-i The partition request specifications in the most recent "new" command are displayed.

-m The hardware characteristics of nodes currently owned by the user are displayed.

-M The hardware characteristics of all nodes are displayed.

-n The names and defined constants used by the nuggetmaster are displayed. These names include defined constants used to characterize selector values for partition-request files. The names also include character strings used to identify available devices.

-p The user's partition, if any, is described. The information displayed is mostly concerned with the resources visible to a partition. The "m" option can be used to display information

describing the physical specifications for the node machines owned by a partition.

-P All partitions are described.

2.3. Node Control

Nodes in a partition have both a logical and a physical identifier. Logical node identifiers range from zero to the size of the partition. Logical machine 0 is always the user's host machine. The following commands refer only to logical node identifiers. Physical node identifiers are not explicitly referenced by either a user or a client. The nuggetmaster creates a logical-to-physical resource mapping table that is used in referencing nodes and virtual disks. This table and a nugget program are linked with a user program to create a crystal load module. The following commands can be used to monitor and control the execution states of specific nodes. In all these commands, the "a" ("all") option makes the command apply to all nodes, and the "h" options causes documentation to be displayed. With the exception of the "bind" command, the following commands involve nuggetmaster-to-nugget communication. If for some reason such communication fails, then the following commands will time out and report their failure to communicate.

bind [option]

A load module is formed for each node in the user's partition. User object modules are linked with a nugget program and a mapping table. The mapping table is used to map logical resources to their physical counterparts. The modules to be bound into the load module are usually specified at partition-allocation time. The load modules created by this command are not directly accessible to a user. Only one of the following options should be specified.

logical_node

This option specifies that the load module be created for a particular node instead of the entire partition. The name of the object file bound with the indicated logical_node at partition building time is used. The intent of specifying only one node is to allow the user to modify the program on one node without rebinding the entire partition.

-a

Load modules are created for the entire partition using the bindings specified in the user request file.

-h

This option prints documentation for the "bind" command, i.e. the help option.

boot [logical_node ...] [-a] [-h]

The specified nodes are returned to the BOOT state, readying them to accept load modules. This command is implicitly executed by the "load" command before it downloads load modules.

copy source_virtual_disk dest_virtual_disk [option] [-h] This command will copy data from one virtual disk to another. At least one of the virtual disks must be a hardware-backed virtual disk. Virtual disk copies are made in units of blocks. Use the "show -n" command to display the size of a virtual disk block. There is a single defined size for software-backed disk blocks. The sizes of hardware-backed virtual disks are determined by the characteristics of the underlying physical disk. The recognized options are:

[-s This option specifies a block address where the copy should originate. If this option is not specified, the default starting block is the initial block of the virtual disk.

[-l This option specifies the length in blocks for the copy. This size is interpreted to be a multiple of the block length for the source_virtual disk. If this option is not specified, the length of the copy is the total length of the source virtual disk. This command will fail if the length of the copy exceeds the allocated length for the destination virtual disk. In such a case, no data will be copied and the user will be informed of the error.

[-d This option specifies the initial block address on the destination virtual disk. The default is 0.

halt [logical_node ...] [-a] [-h]

The specified nodes are brought to the HALT state. The affected nodes must be either in RUNNING or PAUSE state.

load [logical_node ...] [-a] [-h]

The appropriate load modules are downloaded into the specified nodes. These modules are prepared by the "bind" command. Load modules are maintained by the nuggetmaster and are not directly accessible by the user. This command implicitly places the affected nodes in BOOT state before downloading begins.

pause [logical_node ...] [-a] [-h]

The specified nodes are placed in the PAUSE state. The characteristics of the PAUSE state are documented in the nugget specifications. This command is recognized if the node is in the RUNNING state. When the nugget enters PAUSE state a status message is sent to the nuggetmaster. Either a client process running on a node or the nuggetmaster may request a PAUSE state.

peek address [logical_node ...] [-a] [-h]

A word of memory is fetched from the 32-bit physical address of the indicated node(s). This command is recognized by a node in a RUNNING or PAUSE state. The contents of the fetched word are displayed.

poke address [logical_node ...] [-a] [-h]

The specified value is stored into the 32-bit physical address of the indicated node. This command is recognized by a node in a RUNNING or PAUSE state.

restart [logical_node ...] [-a] [-h]

The client on the specified node(s) is restored to the RUNNING state at symbolic location _client. This command is recognized when a node is in RUNNING or PAUSE state.

run [logical_node ...] [-a] [-h]

The client on the specified node(s) is returned to a RUNNING state. If the client was already running, this command has no effect.

state [logical_node ...] [-a] [-h]

The current state of the client on the specified node(s) is reported. The report indicates the state of the node: PAUSE, RUNNING, BOOT, or HALT. If an option is not given for this command, then the current state of the nug-getmaster is displayed. This data will include the current partition and current virtual terminal environments.

2.4. Virtual Terminal Control

The following virtual terminal commands can be used to inspect and control the various virtual terminals. Virtual terminals are characterized by a pair of numbers: (logical_node, virtual_term_id). Every node has by default at least one virtual terminal with a virtual_term_id of 0. This terminal is the virtual console for the node. The nug-

getmaster provides each partition one virtual terminal on the user's host machine. Since logical_node 0 designates the host computer, this host terminal is designated as virtual terminal (0,0). Command interpreter commands are recognized and processed by this host terminal (0,0), which is therefore sometimes referred to as the user's command console.

The user is at any time within some *environment*. The environment determines which virtual terminal in the partition is connected to the physical terminal on the host computer. Upon entering the command interpreter program, a user is placed in the environment of the command console (0,0), which recognizes all nuggetmaster commands. A user may create additional virtual terminals at the time a partition is built by associating virtual terminals with nodes, as described above under the "new" command.

Only *active* virtual terminals have accessible environments. Virtual terminals other than (0,0) become active when load modules have been downloaded into their nodes. If a once active node has entered a nonactive state (PAUSE, BOOT, or HALT), then the nuggetmaster will not attempt to forward any input from any of the associated virtual terminals. Such input will be discarded by the nugget master and the user will be informed of the inactive state of the node. The user may, however, view any previously buffered output from the particular node. When a user is in the environment of a virtual terminal other than the command console, entering the escape character allows the user to enter command interpreter commands. Within the virtual terminal environment of (0,0) characters are not read until a newline is entered, i.e. a non-cbreak mode. All other virtual terminal environments are by default set to a mode where input characters are immediately read as they are input, i.e. a cbreak mode. This input mode setting for a virtual terminal can be modified with the "termset" command. Output directed by a client at a virtual terminal that is not in

the current environment is buffered by the command interpreter until the environment is changed to that virtual terminal. The following command interpreter commands deal with virtual terminal control.

view logical_node virtual_term_id [option] ...

Output that has been buffered for a virtual terminal is displayed. The logical_node and virtual_terminal_id arguments specify a particular virtual terminal. The environment is not changed by this command. Instead, it is suspended while the user views the output for the specified virtual terminal. This command terminates when all buffered output for a virtual terminal has been displayed. The suspended environment is then reinstated. A user may prematurely terminate the "view" command by hitting a break key, which will immediately reinstate the suspended environment. These options are recognized:

-f This option causes a copy of the specified buffered output to be written to a file. This option, in effect, takes a snapshot of the current buffer for a virtual terminal. The "termset" command can be used to redirect all buffered output for a virtual terminal to a file named by the user. Such named buffers are controlled and accessed in the same manner as the default system buffers.

-h

Print the associated help page.

-z

Flush the buffer associated with the virtual terminal. This option can only be used to flush the buffer for the particular virtual specified by the view command. The "termset" command allows more than one buffer to be flushed.

| filter The output from the view command is directed through the specified unix filter.

termset [option] ...

Set or report the characteristics of virtual terminals. Options choose which terminals are affected and whether to set or report the status. The characteristics of the command console are modified by only some of the following options. The recognized options are:

-t logical_node [virtual_term_id ...]

This option specifies the virtual terminals that will be the object of the termset command. This option allows the termset command to be applied to some subset of the virtual terminals associated with a logical node. The virtual terminal list specifies the identifiers of the virtual terminals whose settings should be modified by this command. If no virtual terminals are listed, then the termset command is applied to all the virtual terminals associated with a logical node.

-a

This option is used to specify that the termset command will be applied to all the virtual terminals associated with a partition.

-c Set the input mode for the virtual terminal such that characters are immediately read as they are input, i.e. cbreak mode.

-C Set the input mode for the virtual terminal such that characters are read only when a newline is received, i.e. non-cbreak mode.

-e escape_character

The default escape character is <control-A>. This command redefines the escape character for all virtual terminals. This

escape character can be used to temporarily switch to the environment of virtual terminal (0,0). All input following the escape character up to a carriage return is passed to the virtual terminal (0,0) for evaluation. The original virtual terminal environment is reinstated when the virtual terminal (0,0) has completed its evaluation and execution of the input line.

-f filename

Output from the chosen terminal is buffered in the file specified rather than in one of the nugget master defined buffers. Note that this option is only valid for a termset command specifying a single virtual terminal. The termset command will report failure if this option is used to map the buffers of multiple virtual terminals into a single file.

-h

Documentation is displayed describing the "termset" command.

-l size

This command can be used to set the length of the buffer associated with a particular virtual terminal. The buffer may be either one of the system defined buffers or a user defined buffer (via -f option). Initially, all virtual terminal buffers are assigned the default length MaxVtermBuff. The definition of this constant can be displayed with the "show -n" command. Note that a user can effectively turn buffering off for a virtual terminal by setting the length of the associated buffer to zero.

-s

The output mode is set to "scroll mode". If an output buffer for a virtual terminal in scroll mode is filled, the oldest data will be lost.

-S

This option sets the output mode to "non-scroll mode". If a buffer in non-scroll mode is filled, the nuggetmaster will not accept any further output for the particular virtual terminal. In that case, flow control mechanisms will inform the client on that node that the virtual terminal is full.

-z

Any buffered output for the chosen virtual terminal(s) is flushed.

switch [option]

This command is distinguished by the substantial power that is implicit in its function of changing and defining the environment of the user. This command can effect changes in the virtual terminal environment that is local to a partition. Alternately, this command can effect a global environment change by causing the user to be switched to the context of an entirely different partition owned by the user. This latter case is only possible when a user simultaneously owns multiple partitions. Few of the nugget master commands take an argument specifying a target partition. Instead, the nugget master commands are interpreted within the context of whichever partition is current. This command can change the nugget masters notion of which of a user's partition is current. Thus, the notion of a current partition is used to disambiguate the meaning of nugget master commands in those cases when multiple partitions are owned. Upon entering the nugget master, a user will by default be placed in the context of that owned partition which has the lowest valued partition_identifier of all partitions owned by the user. The "state" command can be used to determine which of a user's partitions is defined to be current.

-t [logical_node virtual_terminal_id] The virtual terminal environment is changed. If a virtual terminal is not specified, this command returns the user to the environment of the command console, virtual terminal (0,0). This option has an effect only in respect to the virtual terminals of the current partition. If a logical_node virtual_terminal_id is not specified, then the user is switched back to the environment of the most recently visited virtual terminal.

-p partition_id This option can be used to change to the environment of a different partition that is owned by the user. Any buffers associated with virtual terminals of the previous partition are saved. Upon entering the environment of a different partition, a user is placed within the local environment of the virtual terminal (0,0), the command console.

-h This is the help option.

2.5. Maintenance

The following privileged commands are intended for use only by the crystal support staff for maintenance and development.

fix [physical_node ...] [-g] [-h]

Update the resource monitor's data base to include new status for the listed nodes. By default, the nodes specified by this command are marked as broken machines and placed in a "broken machine partition" whose members are never allocated to users. The "g" option indicates that the specified nodes have been fixed; they are removed from the broken machine partition. If this command is invoked without arguments, then this command lists the members of the broken machine partition.

log [| filter] [-h]

Display the resource monitor's log of interesting events. If a filter is specified, then the output of this command will be piped through the specified Unix filter, e.g. "log | more" or "log | tail -f". This [-h] option is the help option.

gen [-h]

This command is used when the physical configuration of the multicomputer has been modified by the addition or movement of processors or devices. All partitions of node machines except the broken partition are deallocated by this option. The status of all physical disks is updated to reflect the data contained in the master disk specification file, disks.d. The status of all node machines is initialized to the state specified by the master node specification file, nodes.d. The format of the disks.d and nodes.d files are discussed in a later section that examines the nugget master library. The movement of physical disks to different node machines should be transparent to most users of virtual disks. Only in the case of an exclusive access to a virtual disk, will the corresponding physical address be important. User's will be informed via mail if they have been affected by the actions of this command, i.e. if a physical disk supporting any of their virtual disks has been moved, or if a owned partition has been deallocated.

2.6. Other

bye [-h]

Exit the nuggetmaster.

help

Print some introductory documentation.

3. The resource monitor

The resource monitor process maintains a database consisting of files that describe each partition, each virtual disk, each physical node, and each physical disk. Although a resource monitor process can reside on any host, only one resource monitor process should be running at any given time. The nugget master command interpreter communicates with the resource monitor through a communications process that is active on each host computer. The format of the messages exchanged by the command interpreter and the resource monitor vary depending on the type of resource that is controlled.

Most permanent nuggetmaster files are maintained in the directory /usr/lib/command interpreter and are replicated on every host. The following files are kept in this directory:

resource monitor

The executable file for the resource monitor process.

cns_mid

A text file that contains the name of the host computer that is currently the home of the resource monitor process. Currently, all cns_mid files must be manually updated if the resource monitor process is moved to a different machine.

disks.d

Physical disk specifications. This file has a format identical to a partition request file. Each descriptor record in this file describes the attributes of a particular physical disk. There are five option selectors that can be used to describe a disk:

disk The selector values for this option should specify an unique integer that will be used to identify the physical disk.

node The selector values for this option should specify the physical address of the node to which the disk is attached.

type This option can be used to specify the type of a disk. Selector values should be character strings that name a particular disk type.

cylsize This option is used to specify the megabyte size of a cylinder on the disk.

disksize This option specifes the total megabyte size of a disk.

Disks.d

Current allocations of virtual disks. The following information is maintained for each virtual disk:

node: the physical node attached to the physical disk

virtual_id: The virtual disk is described. The virtual and physical disk identifiers are given.

start: the beginning cylinder address for this allocation

length: the number of cylinders allocated

owner: Which user and which host the user is running on.

time: the time and date of creation of a virtual disk

nodes.d

Physical node specifications. The format of this text file is identical to that of partition request files. Each descriptor record in this file describes the attributes of a particular physical node machine. There are five options that can be used to describe a particular node machine.

node This option specifies the physical address of the node machine being described in the current descriptor record.

proc This option specifies which type of processor is begin described. The selector values for this option should selected from the available processor names given in the names.d file. The names.d file can be displayed from within the nuggetmaster by using the "show" command.

mem This option specifies the amount of memory available on a particular node machine. The selector values should specify a size in Kbytes.

pterm This option specifies the number of physical terminals that are attached to a node machine.

dev This option specifies the name of a device that is attached to a particular node machine. The selector values for this option should selected from the available device names given in the names.d file. The names.d file can be displayed from within the nuggetmaster by using the "show" command.

Nodes.d

Current allocations of partitions. The following information is maintained for each node:

owner: by whom and on which host

partition: partition identifier

time: time and date of allocation

binding: object module associated with the node

disks: identifiers of physically attached disks

memory: size of main store

peripherals: other attached devices

processor: type of processor

vterms: number of virtual terminals

pterms: number of physically attached terminals