Shell

The Shell module provides a simple command interpreter which can be used to interact with a ROME system. Its main purpose is to provide a command-line interface for filing system operations (for example copying files from floppy to hard disk, deleting and renaming files etc.) and recovery procedures. Depending on which other modules are present in the system, other commands may also work.

Process Information

Prototype Name	shell, note that this process uses stdio
Link Order	does not matter
Process Name	“shell”

Process Operation

The module has only a main process. It does not operate on message as such, but rather reads lines of input through the console interface and acts on the commands. Various commands are enabled by the appropriate *MODULE_XXX* definitions, to ensure that any shared library calls will be presenr in the final system. The commands are as follows.

>proc line sends *line* to *proc* as a *COMMAND* message. This can be used to configure processes manually.
Example: >dos drive floppy:0 id:a

cat *file* displays the contents of *file* onto the standard output stream. Unlike *UNIX* systems, the output cannot be redirected to another file.

cp *from to* copies the file *from* to the file *to.* The files can be on different filing systems.

cd *path* changes the current directory to the (relative) path. As not all filing systems may support the ‘..’ convention, the ‘^’ character moves up a directory level. It is not possible to move between devices using the cd command (see dev below).

console *pfx* changes prefix handling in the console. *pfx* is either *prefix* to turn on console prefixing, or *noprefix* to suppress console prefixing.

dev *path* sets the device root, which is added to the front of all files (followed by the current directory) to form a full path
Example: dev dos:/a/

diagnose [requires RFS] displays the RFS diagnostics for a filing system.

dns *name* [requires DNS] looks up *name* in the DNS as displays the resulting IP address.

edit *file* [required Screed] calls the full-screen editor to update the specified file.

format *dev* [requires RFS] formats specified device for use with the RFS filing system.

host *name* sets the (NIS) hostname for the machine.

logon *user* [requires NIS and NFS] sets the NFS authorisation for subsequent NFS transactions to the parameters for *user* in the NIS map.

ls [*l*] lists the current directory. The *l* option produces longer output. The exact fields that will be displayed depends on the underlying filing system being listed.

ln [*s*] *fr to* creates a link from *fr* to *to.* If the *s* parameter is given, the link is a symbolic link, otherwise a hard link is created (if the underlying filing system supports the operation).

load *file* [requires PISA] loads the specified file into the interpreter environment used by the shell.

mkdir *name* creates a (new) directory *name* in the current directory.

pwd displays the path to the current working directory on the standard output stream.

reboot [I386 only] causes the machine to reboot.

mv *from to* renames the *from* file as the *to* file. This will never work across different filing systems. Depending on the underlying filing system it may work across directories.

rm *file* deletes *file* from the current directory.

rmdir *name* deleted the (empty) directory *name* from the current directory.

sh *file* executes *file* as a set of shell commands.

shell *cmd* [requires PISA] passes *cmd* to the interpreter. This can be used to pass in commands that would otherwise be interpreted locally.

start *system* executes pre-compiled command sequences to configure sub-systems of a ROME system. The values in these commands will need editing for the specific environment in which the system is operating. For example, the *start net* command configures the ARP, IP, DNS and NIS layers for a particular machine. This is useful for system testing, but the SysManager module is the better way to handle such configurations.

ypmatch *id* [requires NIS] matches the *id* in the NIS tables.

else [requires PISA] all other commands are passed to the interpreter, if it is present in the system.