NAME

eserver – EMTS transputer server

SYNOPSIS

eserver [*options*]

DESCRIPTION

eserver is a transputer server program that is compatible with the Inmos iserver. **eserver** also provides a number of server extensions (see SERVER EXTENSIONS below) depending on the capabilities of your host operating system. If no options are specified, **eserver** displays brief usage instructions and configuration information.

OPTIONS

–si Information mode. By default, the server generates no information or warning messages. Supplying the **–si** option puts the server in **verbose** mode causing it to print a small number of information messages and warning messages (if any) to the standard error stream. Note that this option is implicit in the **–sb** option (see below).

–sd Debug mode. By default, the server generates no debugging messages. Supplying the **–sd** option puts the server in **debug** mode causing it to print copious debugging messages to the standard error stream. This option should not normally be required.

–ss Causes the server to enter its serving loop in which it receives **SP** request packets from the stub, processes each request, and transmits reply packets back to the stub. Packets are either processed sequentially (the default) or concurrently (if the client transputer process has elected to exploit the multi-threaded functionality of the server and if the **multi-threading** server extension is available (see SERVER EXTENSIONS below). Note that this option is implicit in the **–sb** option (see below).

–se Causes the server to instruct the stub to monitor the transputer error flag of the selected target transputer system. If the stub detects that the error flag has been set, the server is informed causing it to terminate with a suitable error message. The user may then wish to invoke a debugger.

–sm Causes the server to instruct the stub to use the **MT** protocol (see SERVER EXTENSIONS below) instead of the **SP protocol** (the default) on the link to the target transputer system. This option is only required for transputer programs that have been developed to exploit the multi-threaded functionality of the server (see SERVER EXTENSIONS below).

–sr Causes the server to instruct the stub to reset the sub-system of the selected target transputer system. Note that this option is implicit in the **–sb** option (see below).

–sa Causes the server to instruct the stub to assert the analyse signal on the sub-system of the selected target transputer system.

–sp *n* Causes the server to allocate a peek buffer of *n* kbytes.

–sl *linkname*

Causes the server to use the specified *linkname* which defines the name of the EMTS stub host to which a connection is requested, together with the numeric identifier (optional) of the target transputer system to select. The syntax of *linkname* is *stubname*[:*targetid*]; if no *targetid* is specified, the server attempts to select an arbitrary free target. This option can be used to override the value of the TRANSPUTER environment variable (see ENVIRONMENT below). This option is provided for compatibility with the Inmos iserver.

–st *targetid*

Causes the server to use the specified *targetid* which defines the numeric identifier of the target transputer system to select. This option can be used to override the value of the EMTS_TARGET environment variable (see ENVIRONMENT below), the *targetid* part of the TRANSPUTER environment variable (see below ENVIRONMENT), and the *targetid* part of the *linkname* as defined by the **–sl** option.

–sn *stubname*

Causes the server to use the specified *stubname* which defines either the name of the stub host as registered in the **hosts(5)** data base or the IP address of the stub host (in 'dot' format). This option can be used to override the value of the EMTS_STUBNAME environment variable (see ENVIRONMENT below), the *stubname* part of the TRANSPUTER environment variable (see ENVIRONMENT below), and the *stubname* part of the *linkname* as defined by the **–sl** option.

–sc *filename*

Causes the server to copy the specified *filename* to the selected target transputer system. Note that this option is implicit in the **–sb** option (see below).

–sb *filename*

Causes the server to reset the target transputer system, to boot the selected target transputer system with the specified *filename*, to enter **verbose** mode and to start handling requests from the client transputer system. This is equivalent to specifying the **–sr –ss –si –sc** *filename* options.

ENVIRONMENT

TRANSPUTER

The *linkname* to use when attempting to start a session with a target transputer system. This environment variable is supported for compatibility with the Inmos iserver.

EMTS_STUBNAME

The *stubname* to use when attempting to start a session with a target transputer system. The value of this environment variable overrides that of the *stubname* part of the TRANSPUTER environment variable.

EMTS_TARGET

The *targetid* to use when attempting to start a session with a target transputer system. The value of this environment variable overrides that of the *targetid* part of the TRANSPUTER environment variable.

SERVER EXTENSIONS

The availability of the server extensions depends on the capabilities of your host operating system. Invoke the **eserver** without any options in order to see which extensions were enabled at compile time. If *server extensions* are **enabled,** this means that you have access to the **xterm** and **system** extensions described below. If *multi-threading* is **enabled,** this means that you have access to the **multi-threading** extension described below.

MULTI-THREADING

The **multi-threading** server extension is available if your server host is running SunOS 4.1.1 or higher.

The server has been programmed to use multiple threads and non-blocking system calls when support for this is provided by the host operating system. Each request packet from a client transputer process is handled by a separate server thread. If the request cannot be processed immediately (eg it might be a blocking i/o request), the server suspends the thread until it becomes runnable. Parallel threads servicing requests from other client processes can continue to be handled by the server. This improves the utilisation of the server and removes the single-threaded i/o bottleneck that is familiar to many transputer programmers. The multi-threaded nature of the **eserver** is consistent with the multi-tasking potential of the transputer and the multi-processing potential of transputer networks.

A transputer program that has been written to exploit the multi-threaded nature of **eserver** will be running the **MT** (multi-threaded) protocol on the link between target and stub instead of the standard **SP** protocol. The **MT** protocol is a simple extension to the **SP** protocol whereby a single-byte *client process identifier* is appended to every **SP** packet; a transputer link demultiplexor process uses this identifier to route reply packets back to their owners (see *EMTS User Guide*). In order to run such programs, the **–sm** option must be specified when invoking the

server.

XTERM SUPPORT

The **xterm** server extension is available if your server host is running Unix and the X11 (or OpenWindows) windowing systems.

The server can support multiple client-initiated **xterm(1)** terminal emulator windows. Each **xterm(1)** process gives its client a private pseudo-terminal allowing multiple transputer client processes to share the resources of a single physical terminal. If the **multi-threading** server extension is available, blocking i/o requests to one pseudo-terminal will not affect i/o operations to other pseudo-terminals.

SYSTEM PROCESS SUPPORT

The **system** server extension is available if your server host is running Unix.

The server can support multiple client-initiated Unix processes, the standard input and standard output streams of which are piped from and to the client transputer process. This would allow a client process to spawn a graphical windows-based front-end to a transputer application, for example. If the **multi-threading** server extension is available, blocking i/o requests on one system process will not affect i/o operations to other system processes.

EXAMPLES

example% *eserver*

Display brief configuration and help information.

example% *eserver -sb foo.btl*

Instruct the server to connect to the stub specified either by the *stubname* part of the TRANSPUTER environment variable or by the EMTS_STUBNAME environment variable, to select a target transputer system specified by the *targetid* part of the TRANSPUTER environment variable (if specified) or an arbitrary free target (if not specified), to reset the target, to boot the target with the the file *foo.btl*, to enter **verbose** mode, and to start servicing requests from the target.

example% *eserver -sa -sp 16 -se -si -ss -sc tds3load.b4 -f tds3.xsc*

Instruct the server to connect to the stub specified either by the *stubname* part of the TRANSPUTER environment variable or by the EMTS_STUBNAME environment variable, to select a target transputer system specified by the *targetid* part of the TRANSPUTER environment variable (if specified) or an arbitrary free target (if not specified), to analyse the target, to peek 16 kbytes of data from the transputer, to copy the file *tds3load.b4* to the target, to monitor the transputer error flag, to enter **verbose** mode, and to start servicing requests from the target. The additional arguments *-f tds3.xsc* are passed to *tds3load.b4* as command-line arguments.

example% *eserver -sm -sn mercury -st0 -sb foo.btl*

Instruct the server to connect to the stub host *mercury*, to select target *0*, to reset the target, to boot the target with the file *foo.btl*, to use the **MT** protocol on the link between stub and target, to enter **verbose** mode, and to start servicing requests from the target.

example% *eserver -se -sd -sl 143.65.43.8:2 -sb foo.btl*

Instruct the server to connect to the stub host at IP address *143.65.43.8*, to select target *2*, to reset the target, to boot the target with the file *foo.btl*, to enter **verbose** mode, to enter **debug** mode, to monitor the transputer error flag, and to start servicing requests from the target.

CONFIGURATION

eserver looks up the service name *emts* in an attempt to determine which port number to use when attempting to connect to a stub; if the service lookup fails, the default port number **1073** is used. See **services(5)** for more information.

DIAGNOSTICS

If the server detects a fatal error, it terminates returning an iserver-compatible exit code to the operating

system. It also prints an abort message to the standard error stream in the following format:

eserver: Abort *procid*:*abortid* - *abortmessage*

procid is a number identifying the program procedure from which the abort has originated. *abortid* is a unique number identifying the reason for the abort. *abortmessage* is a brief message indicating the reason for the abort.

The following exit codes are returned to the operating system:

0 - successful termination by a client transputer process.
1 - error termination by a client transputer process.
2 - server terminated by a user keyboard break.
3 - transputer error flag set.
4 - miscellaneous server error.
5 - non-standard termination by a client transputer process.
6 - server system error.
7 - Lightweight Process Library error.
8 - server terminated by the stub.

IMPLEMENTATION

All the EMTS programs use the BSD socket libraries and the TCP/TP protocol suite for handling communications over the network. They should be portable to a wide variety of platforms that support these libraries.

The current version of the server runs on MS-DOS (using either FTP's PC/TCP or Sun Microsystem's PC-NFS networking environments) as well as SunOS 4.* and Unix System V Release 4. The multithreading server extension relies on Sun's Lightweight Process threads library; the server may be ported to POSIX threads in a later release.

IDENTIFICATION

Author: Brendan Murphy.
Revision Number: 2.1; Release Date: 1993/11/01.
Copyright © Brendan Murphy 1991-1993. All Rights Reserved.
Copyright © INMOS Ltd, 1988. All Rights Reserved.

SEE ALSO

estub(1), ewho(1)
EMTS User Guide.