NAME truss – trace system calls and signals

SYNOPSIS **truss** [−**fcaeil**] [−[**tvx**] [**!**]*syscall*...] [−**s** [**!**]*signal*...] [−**m** [**!**]*fault*...] [−[**rw**] [**!**]*fd*...]
[−**o** *outfile*] *command* | −**p** *pid*

DESCRIPTION **truss** executes the specified command and produces a trace of the system calls it performs, the signals it receives, and the machine faults it incurs. Each line of the trace output reports either the fault or signal name or the system call name with its arguments and return value(s). System call arguments are displayed symbolically when possible using defines from relevant system headers; for any pathname pointer argument, the pointed-to string is displayed. Error returns are reported using the error code names described in **intro**(2).

OPTIONS The following options are recognized. For those options that take a list argument, the name **all** can be used as a shorthand to specify all possible members of the list. If the list begins with a **!**, the meaning of the option is negated (for example, exclude rather than trace). Multiple occurrences of the same option may be specified. For the same name in a list, subsequent options (those to the right) override previous ones (those to the left).

−**p** Interpret the *command* arguments to **truss** as a list of process-ids for existing processes (see **ps**(1)) rather than as a command to be executed. **truss** takes control of each process and begins tracing it provided that the userid and groupid of the process match those of the user or that the user is a privileged user. Processes may also be specified by their names in the **/proc** directory, for example, **/proc/12345**.

−**f** Follow all children created by **fork**() or **vfork**() and include their signals, faults, and system calls in the trace output. Normally, only the first-level command or process is traced. When −**f** is specified, the process-id is included with each line of trace output to indicate which process executed the system call or received the signal.

−**c** Count traced system calls, faults, and signals rather than displaying the trace line-by-line. A summary report is produced after the traced command terminates or when **truss** is interrupted. If −**f** is also specified, the counts include all traced system calls, faults, and signals for child processes.

−**a** Show the argument strings that are passed in each **exec**() system call.

−**e** Show the environment strings that are passed in each **exec**() system call.

−**i** Do not display interruptible sleeping system calls. Certain system calls, such as **open**() and **read**() on terminal devices or pipes can sleep for indefinite periods and are interruptible. Normally, **truss** reports such sleeping system calls if they remain asleep for more than one second. The system call is reported again a second time when it completes. The −**i** option causes such system calls to be reported only once, when they complete.

–l	Include the id of the responsible lightweight process with each line of trace output. If **–f** is also specified, both the process-id and the lightweight process id are included.
–t [**!**]*syscall,...*	System calls to trace or exclude. Those system calls specified in the comma-separated list are traced. If the list begins with a **!**, the specified system calls are excluded from the trace output. Default is **–tall**.
–v [**!**]*syscall,...*	Verbose. Display the contents of any structures passed by address to the specified system calls (if traced). Input values as well as values returned by the operating system are shown. For any field used as both input and output, only the output value is shown. Default is **–v!all**.
–x [**!**]*syscall,...*	Display the arguments to the specified system calls (if traced) in raw form, usually hexadecimal, rather than symbolically. This is for unredeemed hackers who must see the raw bits to be happy. Default is **–x!all**.
–s [**!**]*signal,...*	Signals to trace or exclude. Those signals specified in the comma-separated list are traced. The trace output reports the receipt of each specified signal, even if the signal is being ignored (not blocked). (Blocked signals are not received until they are unblocked.) Signals may be specified by name or number (see **<sys/signal.h>**). If the list begins with a **!**, the specified signals are excluded from the trace output. Default is **–sall**.
–m [**!**]*fault,...*	Machine faults to trace or exclude. Those machine faults specified in the comma-separated list are traced. Faults may be specified by name or number (see **<sys/fault.h>**). If the list begins with a **!**, the specified faults are excluded from the trace output. Default is **–mall –m!fltpage**.
–r [**!**]*fd,...*	Show the full contents of the I/O buffer for each **read**() on any of the specified file descriptors. The output is formatted 32 bytes per line and shows each byte as an ascii character (preceded by one blank) or as a 2-character C language escape sequence for control characters such as horizontal tab (\ t) and newline (\ n). If ascii interpretation is not possible, the byte is shown in 2-character hexadecimal representation. (The first 12 bytes of the I/O buffer for each traced **read**() are shown even in the absence of **–r**.) Default is **–r!all**.
–w [**!**]*fd,...*	Show the contents of the I/O buffer for each **write**() on any of the specified file descriptors (see **–r**). Default is **–w!all**.
–o *outfile*	File to be used for the trace output. By default, the output goes to standard error.

See Section 2 of the *man Pages(2): System Calls* for system call names accepted by the **–t**, **–v**, and **–x** options. System call numbers are also accepted.

If **truss** is used to initiate and trace a specified command and if the −**o** option is used or if standard error is redirected to a non-terminal file, then **truss** runs with hangup, interrupt, and quit signals ignored. This facilitates tracing of interactive programs that catch interrupt and quit signals from the terminal.

If the trace output remains directed to the terminal, or if existing processes are traced (the −**p** option), then **truss** responds to hangup, interrupt, and quit signals by releasing all traced processes and exiting. This enables the user to terminate excessive trace output and to release previously-existing processes. Released processes continue normally, as though they had never been touched.

EXAMPLES

This example produces a trace of the **find**(1) command on the terminal:

example% truss find . −print >find.out

Or, to see only a trace of the open, close, read, and write system calls:

example% truss −t open,close,read,write find . −print >find.out

This produces a trace of the **spell**(1) command on the file **truss.out**:

example% truss −f −o truss.out spell *document*

spell is a shell script, so the −**f** flag is needed to trace not only the shell but also the processes created by the shell. (The spell script runs a pipeline of eight concurrent processes.)

A particularly boring example is:

example% truss nroff −mm *document* **>nroff.out**

because 97% of the output reports **lseek**(), **read**(), and **write**() system calls. To abbreviate it:

example% truss −t !lseek,read,write nroff −mm *document* **>nroff.out**

This example verbosely traces the activity of process #1, **init**(1M) (if you are a privileged user):

example% truss −p −v all 1

Interrupting **truss** returns **init** to normal operation.

FILES

/proc/*nnnnn* process files
/proc/process-id

SEE ALSO

intro(2), **proc**(4)

NOTES

Some of the system calls described in Section 2 differ from the actual operating system interfaces. Do not be surprised by minor deviations of the trace output from the descriptions in Section 2.

Every machine fault (except a page fault) results in the posting of a signal to the lightweight process that incurred the fault. A report of a received signal will immediately follow each report of a machine fault (except a page fault) unless that signal is being blocked.

The operating system enforces certain security restrictions on the tracing of processes. In particular, any command whose object file (**a.out**) cannot be read by a user cannot be traced by that user; set-uid and set-gid commands can be traced only by a privileged user. Unless it is run by a privileged user, **truss** loses control of any process that performs an **exec**() of a set-id or unreadable object file; such processes continue normally, though independently of **truss**, from the point of the **exec**().

To avoid collisions with other controlling processes, **truss** will not trace a process that it detects is being controlled by another process via the **/proc** interface. This allows **truss** to be applied to **proc**(4)-BASED debuggers as well as to another instance of itself.

The trace output contains tab characters under the assumption that standard tab stops are set (every eight positions).

The trace output for multiple processes or for a multithreaded process (one that contains more than one lightweight process) is not produced in strict time order. For example, a **read**() on a pipe may be reported before the corresponding **write**(). For any one lightweight process (a traditional process contains only one), the output is strictly time-ordered.

The system may run out of per-user process slots when tracing of children is requested. When tracing more than one process, **truss** runs as one controlling process for each process being traced. For the example of the **spell** command shown above, **spell** itself uses 9 process slots, one for the shell and 8 for the 8-member pipeline, while **truss** adds another 9 processes, for a total of 18. This is perilously close to the usual system-imposed limit of 25 processes per user.

Not all possible structures passed in all possible system calls are displayed under the **−v** option.