

truss(1)

НАЗВАНИЕ

truss - трассировка системных вызовов и сигналов

СИНТАКСИС

```
truss [ -fcaeildD ] [ - [ tTxv ] [ ! ] системный_вызов , ... ]
      [ - [ ss ] [ ! ] сигнал , ... ]
      [ - [ mM ] [ ! ] сбой , ... ]
      [ - [ rw ] [ ! ] fd , ... ]
      [ - [ uU ] [ ! ] библиотека , ...
      : [ : ] [ ! ] функция , ... ]
      [ -o файл_результатов ] команда | -p pid ...
```

ОПИСАНИЕ

Утилита **truss** выполняет указанную команду и выдает трассировку выполняемых ею системных вызовов, получаемых сигналов, и вызванных ею машинных сбоев. Каждая строка результатов трассировки либо сообщает о сбое, либо содержит имя сигнала или системного вызова, вместе с его аргументами и возвращаемыми значениями. Аргументы системного вызова выдаются, по возможности, в символьном виде, с использованием макросов, определенных в соответствующих заголовочных файлах системы. Для указателей на имена файлов выдается соответствующая строка. Информация об ошибках выдается с использованием имен, описанных на странице справочного руководства **intro(3)**.

При указании соответствующей опции (см. описание опции **-u**), **truss** также выдает информацию о входе в пользовательские функции, вызываемые трассируемым процессом, и выходе из них, причем для отражения вложенности вызовов используются отступы.

ОПЦИИ

Утилита распознает представленные ниже опции. Для опций, принимающих аргумент-список, имя **all** можно использовать как сокращение, задающее все возможные элементы списка. Если список начинается с символа **!**, значение опции меняется на противоположное (например, исключить вызовы, а не трассировать). Одну и ту же опцию можно задавать несколько раз. Для одного и того же имени в списке, последующие опции (те, что правее) имеют приоритет над предыдущими (теми, что левее).

-p

Интерпретировать аргументы команды **truss** как список идентификаторов существующих процессов (см. **ps(1)**), а не как команду для выполнения. **truss** берет на себя управление каждым из процессов и начинает трассировать его, если только идентификатор пользователя- и группы-владельца процесса совпадают с идентификаторами текущего пользователя, или утилита вызвана привилегированным пользователем. Процессы можно также указывать по соответствующим именам в каталоге **/proc**, например, **/proc/12345**.

-f

Отслеживать все процессы-потомки, созданные системным вызовом **fork()** или **vfork()** и включать в результаты трассировки информацию об их сигналах, сбоях и системных вызовах. Обычно трассируются только команды или процессы верхнего уровня. Если указана опция **-f**, в каждую

строку трассировки добавляется идентификатор процесса, показывающий, какой процесс выполнил системный вызов или получил сигнал.

-c

Считать протрассированные системные вызовы, сбои и сигналы, а не выдавать информацию о них построчно. После завершения работы трассируемой команды или прерывания работы утилиты **truss** выдается итоговый отчет. Если указана также опция **-f**, итоговые значения учитывают также все протрассированные системные вызовы, сбои и сигналы порожденных процессов.

-a

Выдавать строки аргументов, переданные в каждом системном вызове **exec()**.

-e

Выдавать строки среды, передаваемые при каждом системном вызове **exec()**.

-i

Не выдавать информацию о прерываемых спящих системных вызовах (interruptible sleeping system calls). Некоторые системные вызовы, такие как **open()** и **read()** для терминальных устройств или программных каналов, могут "заснуть" не определенное время и являются прерываемыми. Обычно утилита **truss** сообщает об таких спящих системных вызовах, если они остаются спящими более одной секунды. Еще одна строка сообщения о системном вызове выдается при его завершении. Опция **-i** вызывает однократное информирование о таких вызовах, при их завершении.

-l

Включать идентификатор соответствующего *легковесного процесса* (lightweight process - LWP) в каждую строку результатов трассировки. Если указана также опция **-f**, выдаются идентификаторы процесса и легковесного процесса.

-d

Включать временную отметку в каждую строку результатов трассировки. Временная отметка выдается в начале строки в виде поля **секунд.сотых**. Она показывает, сколько секунд прошло с начала трассировки. Первая строка результатов трассировки содержит базовое время, от которого отсчитываются отдельные временные отметки, в виде количества секунд с начала эры UNIX (см. **time(2)**) и строки даты (см. **ctime(3C)** и **date(1)**). Выдаваемое время соответствует времени наступления соответствующего события. Для всех системных вызовов событием является завершение системного вызова, а не начало.

-D

Включать дельту по времени в каждую строку результатов трассировки. Значение выдается в виде поля **секунд.сотых** и представляет время, прошедшее с момента обработки последнего события тем же легковесным процессом (LWP). В частности, для системных вызовов это время не совпадает с временем работы системного вызова.

-t [!]системный_вызов,...

Какие системные вызовы трассировать или не трассировать. Системные вызовы, указанные в списке через запятую, трассируются. Если список начинается символом **!**, соответствующие системные вызовы исключаются из списка трассируемых. Стандартное значение опции - **-tall**.

-T [!]системный_вызов,...

Какие системные вызовы должны приводить к остановке процесса. Указанные системные вызовы добавляются к списку, заданному в опции **-t**. Если произойдет один из указанных здесь вызовов, утилита **truss** останавливает процесс и оставляет его в таком состоянии. Т.е. перестает трассировать процесс и завершает работу, но оставляет процесс остановленным по завершении соответствующего системного вызова. Затем остановленный процесс можно обработать отладчиком или другим средством изучения состояния процессов (см. **proc(1)**). К остановленному процессу снова можно применить утилиту **truss** с теми же или другими опциями для продолжения трассировки. Стандартное значение опции - **-T!all**.

Остановленный таким образом процесс нельзя перезапустить из приложения или сигналом **kill -CONT**, поскольку остановлен он при возникновении соответствующего события с помощью средств **/proc**, а не стандартной посылкой сигнала остановки (см. **signal(3HEAD)**). Для запуска остановленного так процесса можно использовать команду **prun(1)**, описанную на странице справочного руководства **proc(1)**.

-v [!]системный_вызов,...

Информировать детально. Выдает содержание любых структур, передаваемых указанным системным вызовам по адресу (если эти вызовы трассируются опцией [-t](#)). Показываются как входные значения, так и значения, возвращаемые операционной системой. Для каждого поля, используемого как входное и выходное, выдается только выходное значение. Стандартное значение опции - **-v!all**.

-x [!]системный_вызов,...

Выдавать аргументы указанных системных вызовов (если они трассируются опцией [-t](#)) в "сыром", обычно, шестнадцатеричном виде, а не в символьическом. Эта возможность предназначена для безнадежных хакеров, которым для полного счастья надо видеть исходные значения байтов. Стандартное значение опции - **-x!all**.

-s [!]сигнал,...

Какие сигналы трассировать или не трассировать. Указанные в списке через запятую сигналы трассируются. В результатах выдается информация о принятии каждого сигнала, даже если он проигнорирован (но не заблокирован). (Заблокированные сигналы не принимаются, пока не будут разблокированы.) Сигналы можно задавать по имени или по номеру (см. заголовочный файл [`<sys/signal.h>`](#)). Если список начинается с символа **!**, указанные сигналы не включаются в результаты трассировки. Стандартное значение опции - **-sall**.

-S [!]сигнал,...

Какие сигналы должны приводить к остановке процесса. Указанные сигналы добавляются к множеству, заданному в опции **-s**. При получении одного из перечисленных сигналов, утилита **truss** останавливает процесс и оставляет его в таком состоянии. (см. описание опции [-T](#)). Стандартное значение опции - **-S!all**.

-m [!]сбой,...

Какие машинные сбои трассировать или не трассировать. Трассируются все сбои, перечисленные в списке через запятую. Сбои можно указывать по имени или по номеру (см. заголовочный файл [`<sys/fault.h>`](#)). Если список начинается с символом **!**, указанные сбои не включаются в результаты трассировки. Стандартное значение опции - **-mall -m!fltpage**.

-M [!]сбой,...

Какие машинные сбои должны приводить к остановке процесса. Указанные сбои добавляются к множеству, заданному в опции **-m**. Если происходит один из указанных сбоев, утилита **truss** останавливает процесс и оставляет его в таком состоянии. (см. описание опции [-T](#)). Стандартное значение опции - **-M!all**.

-r [!]fd,...

Выдавать полностью содержимое буфера ввода/вывода для каждого вызова **read()** с любым из перечисленных *файловых дескрипторов* (**fd**). Результат форматируется по 32 байта в строке и каждый байт выдается как ASCII-символ (предваренный одним пробелом) или как двухбайтовая управляющая последовательность языка C для управляющих символов вроде табуляции (**t**) и новой строки (**n**). Если интерпретация в виде ASCII-символа невозможна, байт представляется в шестнадцатеричном виде, как два символа. (Первые 12 байтов буфера ввода/вывода для каждого протрассированного вызова **read()** показываются даже при отсутствии опции **-r**.) Стандартное значение опции - **-r!all**.

-w [!]fd,...

Выдавать содержимое буфера ввода/вывода для каждого вызова **write()** с любым из перечисленных *файловых дескрипторов* (см. описание опции **-r**). Стандартное значение опции - **-w!all**.

-и [!]библиотека,...[:][!]функция,...

Трассировка вызова функции пользовательского уровня. **библиотека,...** - это список, через запятую, имен динамически компонуемых библиотек без суффикса **".so.n"**. **функция,...** - это список имен функций через запятую. В обоих случаях, имена могут включать метасимволы сопоставления с образцом *****, **?**, **[]** с таким же значением, что и в командном интерпретаторе **sh(1)**. Но применяются эти метасимволы к именам библиотек/функций, а не файлов. Пустой список имен библиотек или функций интерпретируется как ***** и приводит к трассировке всех библиотек или функций в библиотеке. Начальный символ **!** в любом из списков означает, что список задает имена библиотек или функций, обращения к которым трассировать не надо. При исключении библиотеки из трассируемых, исключаются все функции этой библиотеки; любой список функций, идущий после

списка не трассируемых библиотек, игнорируется.

Если список библиотек отделяется от списка функций одним двоеточием (:), то трассироваться будут вызовы библиотек извне, но не вызовы одних функций библиотеки из других функций той же библиотеки. Два двоеточия (::) означают, что трассировать надо все вызовы, независимо от того, откуда они сделаны.

Шаблоны имен библиотек не соответствуют выполняемому файлу или динамически компонуемой библиотеке, если нет точного совпадения (**I*** не соответствует **ld.so.1**). Для трассировки вызовов функций этих объектов необходимо задавать имена явно, например:

```
truss -u a.out -u ld ...
```

Имя **a.out** для трассировки вызовов выполняемого файла надо использовать буквально; оно не задает имя конкретного выполняемого файла. При трассировке вызовов функций **a.out** предполагается трассировка всех вызовов (по умолчанию используется ::).

Можно задавать несколько опций **-u** - они обрабатываются слева направо. Если процесс скомпонован с библиотекой **-lthread**, в результаты трассировки включается идентификатор нити, выполнившей соответствующий вызов. Утилита **truss** просматривает динамическую таблицу символов в каждой библиотеке для поиска имен функций, и просматривает также стандартную таблицу символов, если она не была удалена (с помощью **stripe**).

-U [!]библиотека,...:[:]![!]функция,...

Какие вызовы функций пользовательского уровня должны останавливать процесс. Указанные функции добавляются к множеству, заданному в опции **-u**. Если вызвана одна из этих функций, утилита **truss** останавливает процесс и оставляет его в таком состоянии. (см. описание опции **-T**).

-o файл_результатов

Задает файл, в который будут выдаваться результаты трассировки. По умолчанию результаты выдаются в стандартный поток ошибок.

Имена системных вызовов, принимаемые опциями **-t**, **-T**, **-v** и **-x** см. в разделе 2 справочного руководства man: System Calls.

Если утилита **truss** используется для запуска и трассировки указанной команды, и использована опция **-o** или стандартный поток ошибок перенаправлен в файл, не связанный с терминалом, то процесс **truss** по ходу работы игнорирует сигналы **hangup**, **interrupt** и **quit**. Это помогает трассировать интерактивные программы, перехватывающие сигналы **interrupt** и **quit**, поступающие с терминала.

Если результаты трассировки направляются на терминал или трассируются уже запущенные процессы (с помощью опции **-p**), утилита **truss** при получении сигналов **hangup**, **interrupt** и **quit** прекращает трассировать все процессы и завершает работу. Это позволяет пользователю прекратить выдачу избыточной трассировочной информации и прекратить трассировку ранее работавших процессов. Процессы, трассировка которых прекращена таким образом, продолжают работать нормально, как будто их никогда не трассировали.

ПРИМЕРЫ

Пример 1: Трассировка команды

Следующий пример выдает на терминал трассировочную информацию для команды **find(1)**:

```
example$ truss find . -print >find.out
```

Пример 2: трассировка типичных системных вызовов

Чтобы трассировать только системные вызовы **open**, **close**, **read** и **write**, выполните:

```
example$ truss -t open,close,read,write find . -print >find.out
```

Пример 3: Трассировка сценария командного интерпретатора

Следующая команда выдает трассировку команды **spell(1)** в файл **truss.out**:

```
example$ truss -f -o truss.out spell document
```

spell - это сценарий командного интерпретатора, так что, необходимо указать флаг **-fB**, чтобы трассировался не только командный интерпретатор, но и порожденные им процессы. (Сценарий **spell** запускает конвейер из восьми процессов.)

Пример 4: Сокращение объема результатов

Следующий пример дает огромный объем бесполезной трассировочной информации:

```
example$ truss nroff -mm document >nroff.out
```

поскольку 97% строк сообщают о системных вызовах **lseek()**, **read()** и **write()**. Для сокращения объема получаемых результатов можно трассировать так:

```
example$ truss -t !lseek,read,write nroff -mm document >nroff.out
```

Пример 5: Трассировка библиотечных вызовов извне библиотеки **libc**

В этом примере трассируются все вызовы пользовательского уровня, обращающиеся извне к стандартной библиотеке **libc**:

```
example$ truss -u libc ...
```

Пример 6: Трассировка библиотечных вызовов из самой библиотеки **libc**

В этом примере трассируются все вызовы стандартной библиотеки **libc**, выполненные из самой библиотеки:

```
example$ truss -u libc:: ...
```

Пример 7: Трассировка вызовов любой библиотеки, кроме **libc**

Вот как можно трассировать все пользовательские обращения ко всем библиотекам, кроме библиотеки **libc**:

```
example$ truss -u '*' -u !libc ...
```

Пример 8: Трассировка вызовов функций **printf** и **scanf**

В этом примере трассируются все обращения пользовательского уровня к функциям семейства **printf** и **scanf**, содержащимся в стандартной библиотеке **libc**:

```
example$ truss -u 'libc:*printf,*scanf' ...
```

Пример 9: Трассировка пользовательских вызовов любой функции

В этом примере трассируются все вызовы функций пользовательского уровня, откуда они бы ни делались:

```
example$ truss -u a.out -u ld:: -u :: ...
```

Пример 10: Детальная трассировка системного вызова

Следующая команда обеспечивает детальную трассировку системных вызовов процесса с идентификатором 1, **init(1M)** (выполнять ее может только привилегированный пользователь):

```
example# truss -p -v all 1
```

Если прервать работу **truss**, процесс **init** продолжит работать как обычно.

ФАЙЛЫ

/proc/*
файлы процессов

АТРИБУТЫ

Описание следующих атрибутов см. на странице справочного руководства **attributes(5)**:

ТИП АТРИБУТА	ЗНАЧЕНИЕ АТРИБУТА
Доступен в пакете	SUNWtoo (32-битовая платформа)
	SUNWtoox (64-битовая платформа)

ССЫЛКИ

date(1), find(1), proc(1), ps(1), sh(1), spell(1), init(1M), intro(3), exec(2), fork(2), lseek(2), open(2), read(2), time(2), vfork(2), write(2), ctime(3C), threads(3THR), proc(4), attributes(5), signal(3HEAD)
Справочное руководство **man**, раздел 2: системные вызовы

ПРИМЕЧАНИЯ

Некоторые из системных вызовов, описанных в разделе 2 справочного руководства, посвященном системным вызовам, отличаются от фамильских интерфейсов системы. Не удивляйтесь, если обнаружите небольшие отличия результатов трассировки от описаний, представленных в этом разделе справочного руководства.

Каждый сбой машины (кроме **page fault**, сбоя при обращении к странице) приводит к посылке сигнала легковесному процессу (LWP), вызвавшему сбой. Информация о принятом сигнале будет идти сразу же за каждой строкой информации о машинном сбое (кроме **page fault**), если только соответствующий сигнал не заблокирован.

Операционная система устанавливает, из соображений защиты, ряд ограничений на трассировку процессов. В частности, любая команда, объектный файл которой (**a.out**) не может быть прочитан пользователем, не может трассироваться этим пользователем; программы с установленными битами **suid** и **sgid** может трассировать только привилегированный пользователь. Если только не запущена привилегированным пользователем, утилита **truss** теряет контроль над любым процессом, выполнившим (**exec()**) объектный файл, недоступный для чтения или с установленными битами **suid/sgid**; такие процессы продолжают работать normally, но уже не трассируются утилитой **truss**, с момента выполнения соответствующего системного вызова **exec()**.

Во избежание конфликтов с другими управляющими процессами, утилита **truss** не будет трассировать процесс, управляемый другим процессом через интерфейс **/proc**. Это позволяет утилите **truss** применяться к отладчикам на базе средств **proc(4)** или к другим экземплярам самой себя.

В результатах трассировки символы табуляции используются, исходя из предположения о стандартном шаге табуляции (восемь символов).

Результаты трассировки для нескольких процессов или для *многонитевого* (multithreaded) процесса (содержащего более одного легковесного процесса, LWP) выдаются не строго упорядоченными по времени. Например, информация о вызове **read()** для программного канала может быть выдана раньше, чем о соответствующем вызове **write()**. Для любого одного LWP (обычный процесс содержит ровно один легковесный процесс), выдаваемые результаты строго упорядочены по времени.

При трассировке нескольких процессов, запускается отдельный управляющий процесс **truss** для каждого трассируемого процесса. В представленном выше примере трассировки команды **spell**, сама команда **spell** требует 9 слов в таблице процессов, один - для командного интерпретатора и 8 для конвейера из 8 процессов, а **truss** добавляет еще 9 процессов, что суммарно дает 18.

Не все возможные структуры, передаваемые в различные системные вызовы, выдаются при установке опции **-v**.

Последнее изменение: 15 июля 1998 года

Copyleft (no c) 2003 [B. Кравчук](#), [OpenXS Initiative](#), перевод на русский язык