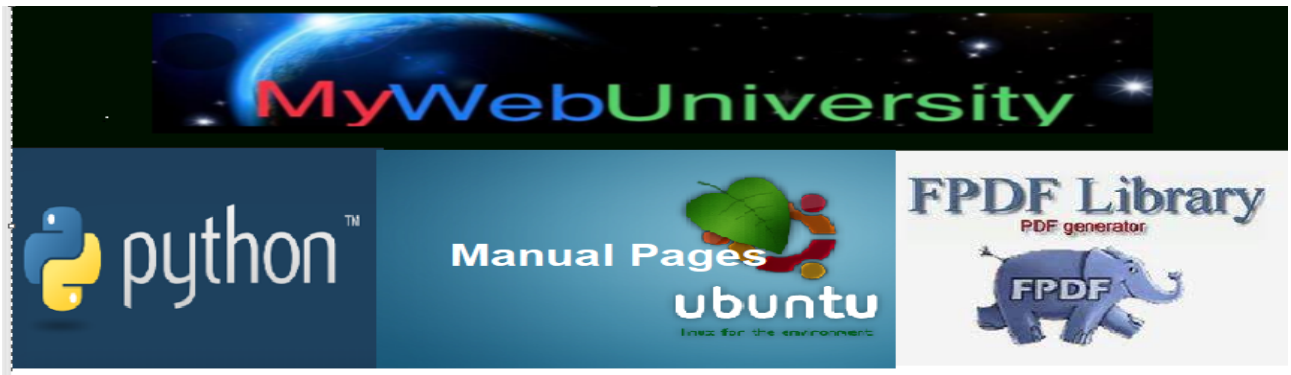

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'perf-script.1'

$ man perf-script.1

PERF-SCRIPT(1) perf Manual PERF-SCRIPT(1)

NAME

perf-script - Read perf.data (created by perf record) and display trace output

SYNOPSIS

perf script [<options>]

perf script [<options>] record <script> [<record-options>] <command>

perf script [<options>] report <script> [script-args]

perf script [<options>] <script> <required-script-args> [<record-options>] <command>

perf script [<options>] <top-script> [script-args]

DESCRIPTION

This command reads the input file and displays the trace recorded.

There are several variants of perf script:

'perf script' to see a detailed trace of the workload that was recorded.

You can also run a set of pre-canned scripts that aggregate and summarize the raw trace data in various ways (the list of scripts is available via 'perf script -l'). The following variants allow you to record and run those scripts:

'perf script record <script> <command>' to record the events required for 'perf script report'. <script> is the name displayed in the output of 'perf script --list' i.e. the actual script name minus any language extension. If <command> is not specified, the events are recorded using the -a (system-wide) 'perf record' option.

'perf script report <script> [args]' to run and display the results of <script>. <script> is the name displayed in the output of 'perf script --list' i.e. the actual script name minus any language extension. The perf.data output from a previous run of 'perf script record <script>' is used and should be present for this command to succeed. [args] refers to the (mainly optional) args expected by the script.

'perf script <script> <required-script-args> <command>' to both record the events required for <script> and to run the <script> using 'live-mode' i.e. without writing anything to disk. <script> is the name displayed in the output of 'perf script --list' i.e. the actual script name minus any language extension. If <command> is not specified, the events are recorded using the -a (system-wide) 'perf record' option. If <script> has any required args, they should be specified before <command>. This mode doesn't allow for optional script args to be specified; if optional script args are desired, they can be specified using separate 'perf script record' and 'perf script report' commands, with the stdout of the record step piped to the stdin of the report script, using the '-o -' and '-i -' options of the corresponding commands.

'perf script <top-script>' to both record the events required for <top-script> and to run the <top-script> using 'live-mode' i.e. without writing anything to disk. <top-script> is the name displayed in the output of 'perf script --list' i.e. the actual script name minus any language extension; a <top-script> is defined as any script name ending with the string 'top'.

[<record-options>] can be passed to the record steps of 'perf script record' and 'live-mode' variants; this isn't possible however for <top-script> 'live-mode' or 'perf script report' variants.

See the 'SEE ALSO' section for links to language-specific information on how to write and run your own trace scripts.

OPTIONS

<command>...

Any command you can specify in a shell.

-D, --dump-raw-trace=

Display verbose dump of the trace data.

--dump-unsorted-raw-trace=

Same as --dump-raw-trace but not sorted in time order.

-L, --Latency=

Show latency attributes (irqs/preemption disabled, etc).

-l, --list=

Display a list of available trace scripts.

-s [lang], --script=

Process trace data with the given script ([lang]:script[.ext]). If the string lang is specified in place of a script name, a list of supported languages will be displayed instead.

-g, --gen-script=

Generate a starter script. If a language is given then the script is named perf-script.[ext] according to the language. If a file path is given then python is used for files ending .py and perl used for files ending .pl.

--dlfilter=<file>

Filter sample events using the given shared object file. Refer perf-dlfilter(1)

--dlarg=<arg>

Pass arg as an argument to the dlfilter. --dlarg may be repeated to add more arguments.

--list-dlfilters

Display a list of available dlfilters. Use with option -v (must come before option --list-dlfilters) to show long descriptions.

-a

Force system-wide collection. Scripts run without a <command> normally use -a by default, while scripts run with a <command> normally don?t - this option allows the latter to be run in system-wide mode.

-i, --input=

Input file name. (default: perf.data unless stdin is a fifo)

-d, --debug-mode

Do various checks like samples ordering and lost events.

-F, --fields

Comma separated list of fields to print. Options are: comm, tid, pid, time, cpu, event, trace, ip, sym, dso, dsoff, addr, symoff, srcline, period, iregs, uregs, brstack, brstacksym, flags, bpf-output, brstackinsn, brstackinsnlen, brstackdisasm, brstackoff, callindent, insn, disasm, insnlen, synth, phys_addr, metric, misc, srccode, ipc, data_page_size, code_page_size, ins_lat, machine_pid, vcpu, cgroup, retire_lat, brcntr,

Field list can be prepended with the type, trace, sw or hw,
to indicate to which event type the field list applies.
e.g., -F sw:comm,tid,time,ip,sym and -F trace:time,cpu,trace

perf script -F <fields>
is equivalent to:
perf script -F trace:<fields> -F sw:<fields> -F hw:<fields>

i.e., the specified fields apply to all event types if the type string
is not given.

In addition to overriding fields, it is also possible to add or remove
fields from the defaults. For example
-F -cpu,+insn

removes the cpu field and adds the insn field. Adding/removing fields
cannot be mixed with normal overriding.

The arguments are processed in the order received. A later usage can
reset a prior request. e.g.:
-F trace: -F comm,tid,time,ip,sym

The first -F suppresses trace events (field list is ""), but then the
second invocation sets the fields to comm,tid,time,ip,sym. In this case a
warning is given to the user:
"Overriding previous field request for all events."

Alternatively, consider the order:
-F comm,tid,time,ip,sym -F trace:

The first -F sets the fields for all events and the second -F
suppresses trace events. The user is given a warning message about
the override, and the result of the above is that only S/W and H/W
events are displayed with the given fields.

It's possible tp add/remove fields only for specific event type:
-Fsw:-cpu,-period

removes cpu and period from software events.

For the 'wildcard' option if a user selected field is invalid for an
event type, a message is displayed to the user that the option is
ignored for that type. For example:

$ perf script -F comm,tid,trace
'trace' not valid for hardware events. Ignoring.
'trace' not valid for software events. Ignoring.

Alternatively, if the type is given an invalid field is specified it
is an error. For example:

perf script -v -F sw:comm,tid,trace
'trace' not valid for software events.

At this point usage is displayed, and perf-script exits.

The flags field is synthesized and may have a value when Instruction
Trace decoding. The flags are "bcrosyiABExghDt" which stand for branch,
call, return, conditional, system, asynchronous, interrupt,
transaction abort, trace begin, trace end, in transaction, VM-Entry,
VM-Exit, interrupt disabled and interrupt disable toggle respectively.
Known combinations of flags are printed more nicely e.g.
"call" for "bc", "return" for "br", "jcc" for "bo", "jmp" for "b",
"int" for "bci", "iret" for "bri", "syscall" for "bcs", "sysret" for "brs",
"async" for "by", "hw int" for "bcyi", "tx abrt" for "bA", "tr strt" for "bB",
"tr end" for "bE", "vmentry" for "bcg", "vmexit" for "bch".
However the "x", "D" and "t" flags will be displayed separately in those
cases e.g. "jcc (xD)" for a condition branch within a transaction
with interrupts disabled. Note, interrupts becoming disabled is "t",
whereas interrupts becoming enabled is "Dt".

The callindent field is synthesized and may have a value when
Instruction Trace decoding. For calls and returns, it will display the
name of the symbol indented with spaces to reflect the stack depth.

When doing instruction trace decoding, insn, disasm and insnlen give the
instruction bytes, disassembled instructions (requires libcapstone support)
and the instruction length of the current instruction respectively.

The synth field is used by synthesized events which may be created when

Instruction Trace decoding.

The ipc (instructions per cycle) field is synthesized and may have a value when

Instruction Trace decoding.

The machine_pid and vcpu fields are derived from data resulting from using

perf inject to insert a perf.data file recorded inside a virtual machine into

a perf.data file recorded on the host at the same time.

The cgroup fields requires sample having the cgroup id which is saved

when "--all-cgroups" option is passed to 'perf record'.

Finally, a user may not set fields to none for all event types.

i.e., -F "" is not allowed.

The brstack output includes branch related information with raw addresses using the

FROM/TO/EVENT/INTX/ABORT/CYCLES/TYPE/SPEC syntax in the following order:

FROM : branch source instruction

TO : branch target instruction

EVENT : M=branch target or direction was mispredicted

P=branch target or direction was predicted

N=branch not-taken

-=no event or not supported

INTX : X=branch inside a transactional region

-=branch not in transaction region or not supported

ABORT : A=TSX abort entry

-=not aborted region or not supported

CYCLES: the number of cycles that have elapsed since the last branch was recorded

TYPE : branch type: COND/UNCOND/IND/CALL/IND_CALL/RET etc.

-=not supported

SPEC : branch speculation info: SPEC_WRONG_PATH/NON_SPEC_CORRECT_PATH/SPEC_CORRECT_PATH

-=not supported

The brstacksym is identical to brstack, except that the FROM and TO addresses are printed in a symbolic form if possible.

When brstackinsn is specified the full assembler sequences of branch sequences for each sample

is printed. This is the full execution path leading to the sample. This is only supported when the

sample was recorded with perf record -b or -j any.

Use brstackinsnlen to print the brstackinsn lenght. For example, you
can?t know the next sequential instruction after an unconditional branch unless
you calculate that based on its length.
brstackdisasm acts like brstackinsn, but will print disassembled instructions if
perf is built with the capstone library.
The brstackoff field will print an offset into a specific dso/binary.
With the metric option perf script can compute metrics for
sampling periods, similar to perf stat. This requires
specifying a group with multiple events defining metrics with the :S option
for perf record. perf will sample on the first event, and
print computed metrics for all the events in the group. Please note
that the metric computed is averaged over the whole sampling
period (since the last sample), not just for the sample point.
For sample events it's possible to display misc field with -F +misc option,
following letters are displayed for each bit:
PERF_RECORD_MISC_KERNEL K
PERF_RECORD_MISC_USER U
PERF_RECORD_MISC_HYPERVISOR H
PERF_RECORD_MISC_GUEST_KERNEL G
PERF_RECORD_MISC_GUEST_USER g
PERF_RECORD_MISC_MMAP_DATA* M
PERF_RECORD_MISC_COMM_EXEC E
PERF_RECORD_MISC_SWITCH_OUT S
PERF_RECORD_MISC_SWITCH_OUT_PREEMPT Sp
$ perf script -F +misc ...
sched-messaging 1414 K 28690.636582: 4590 cycles ...
sched-messaging 1407 U 28690.636600: 325620 cycles ...
sched-messaging 1414 K 28690.636608: 19473 cycles ...
misc field _____/

-k, --vmlinux=<file>

vmlinux pathname

--kallsyms=<file>

kallsyms pathname

--symfs=<directory>

Look for files with symbols relative to this directory.

-G, --hide-call-graph

When printing symbols do not display call chain.

--stop-bt

Stop display of callgraph at these symbols

-C, --cpu

Only report samples for the list of CPUs provided. Multiple CPUs can be provided as a

comma-separated list with no space: 0,1. Ranges of CPUs are specified with -: 0-2.

Default is to report samples on all CPUs.

-c, --comms=

Only display events for these comms. CSV that understands file://filename entries.

--pid=

Only show events for given process ID (comma separated list).

--tid=

Only show events for given thread ID (comma separated list).

-I, --show-info

Display extended information about the perf.data file. This adds information which may

be very large and thus may clutter the display. It currently includes: cpu and numa

topology of the host system. It can only be used with the perf script report mode.

--show-kernel-path

Try to resolve the path of [kernel.kallsyms]

--show-task-events Display task related events (e.g. FORK, COMM, EXIT).

--show-mmap-events Display mmap related events (e.g. MMAP, MMAP2).

--show-namespace-events Display namespace events i.e. events of type PERF_RECORD_NAMESPACES.

--show-switch-events Display context switch events i.e. events of type PERF_RECORD_SWITCH or

PERF_RECORD_SWITCH_CPU_WIDE.

--show-lost-events Display lost events i.e. events of type PERF_RECORD_LOST.

--show-round-events Display finished round events i.e. events of type

PERF_RECORD_FINISHED_ROUND.

--show-bpf-events Display bpf events i.e. events of type PERF_RECORD_KSYMBOL and

PERF_RECORD_BPF_EVENT.

--show-cgroup-events Display cgroup events i.e. events of type PERF_RECORD_CGROUP.

--show-text-poke-events Display text poke events i.e. events of type PERF_RECORD_TEXT_POKE and PERF_RECORD_KSYMBOL.

--demangle

Demangle symbol names to human readable form. It?s enabled by default, disable with --no-demangle.

--demangle-kernel

Demangle kernel symbol names to human readable form (for C++ kernels).

--addr2line=<path>

Path to addr2line binary.

--header Show perf.data header.

--header-only Show only perf.data header.

--itrace

Options for decoding instruction tracing data. The options are:

i synthesize instructions events

y synthesize cycles events

b synthesize branches events

c synthesize branches events (calls only)

r synthesize branches events (returns only)

x synthesize transactions events

w synthesize ptwrite events

p synthesize power events (incl. PSB events for Intel PT)

o synthesize other events recorded due to the use of aux-output (refer to perf record)

l synthesize interrupt or similar (asynchronous) events (e.g. Intel PT Event Trace)

e synthesize error events

d create a debug log

f synthesize first level cache events

m synthesize last level cache events

M synthesize memory events

t synthesize TLB events

a synthesize remote access events

g synthesize a call chain (use with i or x)

G synthesize a call chain on existing event records

l synthesize last branch entries (use with i or x)

L synthesize last branch entries on existing event records

s skip initial number of events

q quicker (less detailed) decoding

A approximate IPC

Z prefer to ignore timestamps (so-called "timeless" decoding)

T use the timestamp trace as kernel time

The default is all events i.e. the same as --itrace=iybxwpe,

except for perf script where it is --itrace=ce

In addition, the period (default 100000, except for perf script where it is 1)

for instructions events can be specified in units of:

i instructions

t ticks

ms milliseconds

us microseconds

ns nanoseconds (default)

Also the call chain size (default 16, max. 1024) for instructions or

transactions events can be specified.

Also the number of last branch entries (default 64, max. 1024) for

instructions or transactions events can be specified.

Similar to options g and l, size may also be specified for options G and L.

On x86, note that G and L work poorly when data has been recorded with

large PEBS. Refer linkperf:perf-intel-pt[1] man page for details.

It is also possible to skip events generated (instructions, branches, transactions,

ptwrite, power) at the beginning. This is useful to ignore initialization code.

--itrace=i0nss1000000

skips the first million instructions.

The 'e' option may be followed by flags which affect what errors will or

will not be reported. Each flag must be preceded by either '+' or '-'.

The flags are:

o overflow

l trace data lost

If supported, the 'd' option may be followed by flags which affect what debug messages will or will not be logged. Each flag must be preceded by either '+' or '-'. The flags are:

a all perf events

e output only on errors (size configurable - see linkperf:perf-config[1])

o output to stdout

If supported, the 'q' option may be repeated to increase the effect.

To disable decoding entirely, use --no-itrace.

--full-source-path

Show the full path for source files for srcline output.

--max-stack

Set the stack depth limit when parsing the callchain, anything beyond the specified depth will be ignored. This is a trade-off between information loss and faster processing especially for workloads that can have a very long callchain stack. Note that when using the --itrace option the synthesized callchain size will override this value if the synthesized callchain size is bigger.

Default: 127

--ns

Use 9 decimal places when displaying time (i.e. show the nanoseconds)

-f, --force

Don?t do ownership validation.

--time

Only analyze samples within given time window: <start>,<stop>. Times have the format seconds.nanoseconds. If start is not given (i.e. time string is ,x.y) then analysis starts at the beginning of the file. If stop time is not given (i.e. time string is x.y,) then analysis goes to end of file. Multiple ranges can be separated by spaces, which requires the argument to be quoted e.g. --time "1234.567,1234.789 1235,"

Also support time percent with multiple time ranges. Time string is 'a%/n,b%/m,...' or 'a%-b%,c%-%d,...'.

For example:

Select the second 10% time slice:

perf script --time 10%/2

Select from 0% to 10% time slice:

perf script --time 0%-10%

Select the first and second 10% time slices:

perf script --time 10%/1,10%/2

Select from 0% to 10% and 30% to 40% slices:

perf script --time 0%-10%,30%-40%

--max-blocks

Set the maximum number of program blocks to print with brstackinsn for each sample.

--reltime

Print time stamps relative to trace start.

--deltatime

Print time stamps relative to previous event.

--per-event-dump

Create per event files with a "perf.data.EVENT.dump" name instead of printing to stdout, useful, for instance, for generating flamegraphs.

--inline

If a callgraph address belongs to an inlined function, the inline stack will be printed. Each entry has function name and file/line. Enabled by default, disable with --no-inline.

--insn-trace[=<raw|disasm>]

Show instruction stream in bytes (raw) or disassembled (disasm) for intel_pt traces. The default is raw. To use xed, combine raw with --xed to show disassembly done by xed.

--xed

Run xed disassembler on output. Requires installing the xed disassembler.

-S, --symbols=symbol[,symbol...]

Only consider the listed symbols. Symbols are typically a name but they may also be hexadecimal address.

The hexadecimal address may be the start address of a symbol or

any other address to filter the trace records

For example, to select the symbol noploop or the address 0x4007a0:

perf script --symbols=noploop,0x4007a0

Support filtering trace records by symbol name, start address of

symbol, any hexadecimal address and address range.

The comparison order is:

1. symbol name comparison

2. symbol start address comparison.

3. any hexadecimal address comparison.

4. address range comparison (see --addr-range).

--addr-range

Use with -S or --symbols to list traced records within address range.

For example, to list the traced records within the address range

[0x4007a0, 0x0x4007a9]:

perf script -S 0x4007a0 --addr-range 10

--dsos=

Only consider symbols in these DSOs.

--call-trace

Show call stream for intel_pt traces. The CPUs are interleaved, but can be filtered with

-C.

--call-ret-trace

Show call and return stream for intel_pt traces.

--graph-function

For itrace only show specified functions and their callees for itrace. Multiple

functions can be separated by comma.

--switch-on EVENT_NAME

Only consider events after this event is found.

--switch-off EVENT_NAME

Stop considering events after this event is found.

--show-on-off-events

Show the --switch-on/off events too.

--stitch-lbr

Show callgraph with stitched LBRs, which may have more complete callgraph. The perf.data

file must have been obtained using perf record --call-graph lbr. Disabled by default. In

common cases with call stack overflows, it can recreate better call stacks than the

default lbr call stack output. But this approach is not foolproof. There can be cases

where it creates incorrect call stacks from incorrect matches. The known limitations

include exception handing such as setjmp/longjmp will have calls/returns not match.

--merge-callchains

Enable merging deferred user callchains if available. This is the default behavior. If you want to see separate CALLCHAIN_DEFERRED records for some reason, use --no-merge-callchains explicitly.

--guestmount=<path>

Guest OS root file system mount directory. Users mount guest OS root directories under <path> by a specific filesystem access method, typically, sshfs. For example, start 2 guest OS, one?s pid is 8888 and the other?s is 9999:

```
$ mkdir ~/guestmount
$ cd ~/guestmount
$ sshfs -o allow_other,direct_io -p 5551 localhost:/ 8888/
$ sshfs -o allow_other,direct_io -p 5552 localhost:/ 9999/
$ perf script --guestmount=~/guestmount
```

--guestkallsyms=<path>

Guest OS /proc/kallsyms file copy. perf reads it to get guest kernel symbols. Users copy it out from guest OS.

--guestmodules=<path>

Guest OS /proc/modules file copy. perf reads it to get guest kernel module information. Users copy it out from guest OS.

--guestvmlinux=<path>

Guest OS kernel vmlinux.

--guest-code

Indicate that guest code can be found in the hypervisor process, which is a common case for KVM test programs.

SEE ALSO

perf-record(1), perf-script-perl(1), perf-script-python(1), perf-intel-pt(1), perf-dlfilter(1)