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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'coredumpctl.1' command

$ man coredumpctl.1

COREDUMPCTL(1) coredumpctl COREDUMPCTL(1)

NAME

coredumpctl - Retrieve and process saved core dumps and metadata

SYNOPSIS

coredumpctl [OPTIONS...] {COMMAND} [PID|COMM|EXE|MATCH...]

DESCRIPTION

coredumpctl is a tool that can be used to retrieve and process core dumps and metadata which were saved by systemd-coredump(8).

COMMANDS

The following commands are understood:

list

List core dumps captured in the journal matching specified characteristics. If no command is specified, this is the implied default.

The output is designed to be human readable and contains a table with the following columns:

TIME

The timestamp of the crash, as reported by the kernel.

PID

The identifier of the process that crashed.

UID, GID

The user and group identifiers of the process that crashed.

SIGNAL

The signal that caused the process to crash, when applicable.

COREFILE

Information whether the coredump was stored, and whether it is still accessible: "none" means the core was not stored, "-" means that it was not available (for example because the process was not terminated by a signal), "present" means that the core file is accessible by the current user, "journal" means that the core was stored in the "journal", "truncated" is the same as one of the previous two, but the core was too large and was not stored in its entirety, "error" means that the core file cannot be accessed, most likely because of insufficient permissions, and "missing" means that the core was stored in a file, but this file has since been removed.

EXE

The full path to the executable. For backtraces of scripts this is the name of the interpreter.

It's worth noting that different restrictions apply to data saved in the journal and core dump files saved in /var/lib/systemd/coredump, see overview in systemd-coredump(8). Thus it may very well happen that a particular core dump is still listed in the journal while its corresponding core dump file has already been removed.

info

Show detailed information about the last core dump or core dumps matching specified characteristics captured in the journal.

dump

Extract the last core dump matching specified characteristics. The core dump will be written on standard output, unless an output file is specified with --output=.

debug

Invoke a debugger on the last core dump matching specified characteristics. By default, gdb(1) will be used. This may be changed using the --debugger= option or the $SYSTEMD_DEBUGGER

environment variable. Use the --debugger-arguments= option to pass extra command line arguments to the debugger.

OPTIONS

The following options are understood:

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--no-pager

Do not pipe output into a pager.

--no-legend

Do not print the legend, i.e. column headers and the footer with hints.

--json=MODE

Shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks), "pretty" (for a pretty version of the same, with indentation and line breaks) or "off" (to turn off JSON output, the default).

-1

Show information of the most recent core dump only, instead of listing all known core dumps. Equivalent to --reverse -n 1.

-n INT

Show at most the specified number of entries. The specified parameter must be an integer greater or equal to 1.

-S, --since

Only print entries which are since the specified date.

-U, --until

Only print entries which are until the specified date.

-r, --reverse

Reverse output so that the newest entries are displayed first.

-F FIELD, --field=FIELD

Print all possible data values the specified field takes in

matching core dump entries of the journal.

-o FILE, --output=FILE

Write the core to FILE.

--debugger=DEBUGGER

Use the given debugger for the debug command. If not given and $SYSTEMD_DEBUGGER is unset, then gdb(1) will be used.

-A ARGS, --debugger-arguments=ARGS

Pass the given ARGS as extra command line arguments to the debugger. Quote as appropriate when ARGS contain whitespace. (See Examples.)

--file=GLOB

Takes a file glob as an argument. If specified, coredumpctl will operate on the specified journal files matching GLOB instead of the default runtime and system journal paths. May be specified multiple times, in which case files will be suitably interleaved.

-D DIR, --directory=DIR

Use the journal files in the specified DIR.

--root=ROOT

Use root directory ROOT when searching for coredumps.

--image=image

Takes a path to a disk image file or block device node. If specified, all operations are applied to file system in the indicated disk image. This option is similar to --root=, but operates on file systems stored in disk images or block devices. The disk image should either contain just a file system or a set of file systems within a GPT partition table, following the Discoverable Partitions Specification[1]. For further information on supported disk images, see systemd-nspawn(1)'s switch of the same name.

-q, --quiet

Suppresses informational messages about lack of access to journal files and possible in-flight coredumps.

--all

Look at all available journal files in /var/log/journal/ (excluding journal namespaces) instead of only local ones.

MATCHING

A match can be:

PID

Process ID of the process that dumped core. An integer.

COMM

Name of the executable (matches COREDUMP_COMM=). Must not contain slashes.

EXE

Path to the executable (matches COREDUMP_EXE=). Must contain at least one slash.

MATCH

General journalctl match filter, must contain an equals sign ("=").

See journalctl(1).

EXIT STATUS

On success, 0 is returned; otherwise, a non-zero failure code is returned. Not finding any matching core dumps is treated as failure.

ENVIRONMENT

$SYSTEMD_DEBUGGER

Use the given debugger for the debug command. See the --debugger= option.

EXAMPLES

Example 1. List all the core dumps of a program

```
$ coredumpctl list /usr/lib64/firefox/firefox
TIME       PID  UID  GID SIG     COREFILE EXE                      SIZE
Tue ...   8018 1000 1000 SIGSEGV missing  /usr/lib64/firefox/firefox     -
Wed ... 251609 1000 1000 SIGTRAP missing  /usr/lib64/firefox/firefox     -
Fri ... 552351 1000 1000 SIGSEGV present  /usr/lib64/firefox/firefox 28.7M
```

The journal has three entries pertaining to /usr/lib64/firefox/firefox, and only the last entry still has an available core file (in external storage on disk).

Note that coredumpctl needs access to the journal files to retrieve the

relevant entries from the journal. Thus, an unprivileged user will normally only see information about crashing programs of this user.

Example 2. Invoke gdb on the last core dump

```
$ coredumpctl debug
```

Example 3. Use gdb to display full register info from the last core dump

```
$ coredumpctl debug --debugger-arguments="-batch -ex 'info all-registers'"
```

Example 4. Show information about a core dump matched by PID

```
$ coredumpctl info 6654
           PID: 6654 (bash)
           UID: 1000 (user)
           GID: 1000 (user)
        Signal: 11 (SEGV)
     Timestamp: Mon 2021-01-01 00:00:01 CET (20s ago)
  Command Line: bash -c $'kill -SEGV $$'
    Executable: /usr/bin/bash
 Control Group: /user.slice/user-1000.slice/...
          Unit: user@1000.service
     User Unit: vte-spawn-....scope
         Slice: user-1000.slice
     Owner UID: 1000 (user)
       Boot ID: ...
    Machine ID: ...
      Hostname: ...
       Storage: /var/lib/systemd/coredump/core.bash.1000.....zst (present)
  Size on Disk: 51.7K
       Message: Process 130414 (bash) of user 1000 dumped core.
                Stack trace of thread 130414:
                #0  0x00007f398142358b kill (libc.so.6 + 0x3d58b)
                #1  0x0000558c2c7fda09 kill_builtin (bash + 0xb1a09)
                #2  0x0000558c2c79dc59 execute_builtin.lto_priv.0 (bash + 0x51c59)
                #3  0x0000558c2c79709c execute_simple_command (bash + 0x4b09c)
                #4  0x0000558c2c798408 execute_command_internal (bash + 0x4c408)
```

#5 0x0000558c2c7f6bdc parse_and_execute (bash + 0xaabdc)

#6 0x0000558c2c85415c run_one_command.isra.0 (bash + 0x10815c)

#7 0x0000558c2c77d040 main (bash + 0x31040)

#8 0x00007f398140db75 __libc_start_main (libc.so.6 + 0x27b75)

#9 0x0000558c2c77dd1e _start (bash + 0x31d1e)

Example 5. Extract the last core dump of /usr/bin/bar to a file named bar.coredump

```
$ coredumpctl -o bar.coredump dump /usr/bin/bar
```

SEE ALSO

systemd-coredump(8), coredump.conf(5), systemd-journald.service(8), gdb(1)

NOTES

1. Discoverable Partitions Specification

   https://systemd.io/DISCOVERABLE_PARTITIONS