Core dump

A **core dump** is a file containing a process's address space (memory) when the process terminates unexpectedly. Core dumps may be produced on-demand (such as by a **debugger**), or automatically upon termination. Core dumps are triggered by the kernel in response to program crashes, and may be passed to a helper program (such as **systemd-coredump (https://www.freedesktop.org/software/systemd/man/systemd-coredump.html)**) for further processing. A core dump is not typically used by an average user, but may be passed on to developers upon request where it can be invaluable as a post-mortem snapshot of the program's state at the time of the crash, especially if the fault is hard to reliably reproduce.

Contents

Disabling automatic core dumps

Users may wish to disable automatic core dumps for a number of reasons:

- Performance: generating core dumps for memory-heavy processes can waste system resources and delay the cleanup of memory.
- Disk space: core dumps of memory-heavy processes may consume disk space equal to, if not greater, than the process's memory footprint if not compressed.
- Security: core dumps, although typically readable only by root, may contain sensitive data (such as passwords or cryptographic keys), which are written to disk following a crash.

Using sysctl

sysctl can be used to set the `kernel.core_pattern` to nothing to disable core dump handling. Create this file

```
/etc/sysctl.d/50-coredump.conf
```

```
kernel.core_pattern=/dev/null
```

To apply the setting immediately, use `sysctl`:

```
# sysctl -p /etc/sysctl.d/50-coredump.conf
```

Using systemd

systemd's default behavior is defined in `/usr/lib/sysctl.d/50-coredump.conf`, which sets `kernel.core_pattern` to call `systemd-coredump`. It generates core dumps for all processes in `/var/lib/systemd/coredump`. `systemd-coredump` behavior can be overridden by creating a configuration snippet in the `/etc/systemd/coredump.conf.d/` directory with the following content**[1] (https://www.freedesktop.org/software/systemd/man/coredump.conf.html#Description)[2] (https://bbs.archlinux.org/viewtopic.php?id=214207)**:

```
/etc/systemd/coredump.conf.d/custom.conf
```

```
[Coredump]
Storage=none
```

Note: Do not forget to include the `[Coredump]` section name, otherwise this option will be ignored:
`systemd-coredump[1728]: [/etc/systemd/coredump.conf.d/custom.conf:1] Assignment outside of section. Ignoring.`

Then **reload** systemd's configuration.

This method alone is usually sufficient to disable userspace core dumps, so long as no other programs enable automatic core dumps on the system, but the coredump is still generated in memory and systemd-coredump run.

Using PAM limits

The maximum core dump size for users logged in via **PAM** is enforced by **limits.conf**. Setting it to zero disables core dumps entirely. **[3] (https://www.cyberciti.biz/faq/linux-disable-core-dumps/)**

```
/etc/security/limits.conf
```

```
* hard core 0
```

Using ulimit

Command-line shells such as *bash* or *zsh* provide a builtin *ulimit* command which can be used to report or set resource limits of the shell and the processes started by the shell. See **bash(1) § SHELL BUILTIN COMMANDS (https://man.archlinux.org/man/bash.1#SHELL_BUILTIN_COMMANDS)** or **zshbuiltins(1) (https://man.archlinux.org/man/zshbuiltins.1)** for details.

To disable core dumps in the current shell:

```
$ ulimit -c 0
```

Making a core dump

To generate a core dump of an arbitrary process, first **install** the **gdb (https://archlinux.org/packages/?name=gdb)** package. Then find the PID of the running process, for example with *pgrep*:

```
$ pgrep -f firefox
```

```
2071 firefox
```

Attach to the process:

```
$ gdb -p 2071
```

Then at the `(gdb)` prompt:

```
(gdb) generate-core-file
Saved corefile core.2071
(gdb) quit
```

Now you have a coredump file called `core.2071`.

Where do they go?

The `kernel.core_pattern` **sysctl** decides where automatic core dumps go. By default, core dumps are sent to *systemd-coredump* which can be configured in `/etc/systemd/coredump.conf`. By default, all core dumps are stored in `/var/lib/systemd/coredump` (due to `Storage=external`) and they are compressed with `zstd` (due to `Compress=yes`). Additionally, various size limits for the storage can be configured.

Note: The default value for `kernel.core_pattern` is set in `/usr/lib/sysctl.d/50-coredump.conf`. This file may be masked or overridden to use a different setting following normal **sysctl.d(5) (https://man.archlinux.org/man/sysctl.d.5)** rules.

To retrieve a core dump from the journal, see **coredumpctl(1) (https://man.archlinux.org/man/coredumpctl.1)**.

Examining a core dump

Use *coredumpctl* to find the corresponding dump:

```
# coredumpctl list
```

You need to uniquely identify the relevant dump. This is possible by specifying a `PID`, name of the executable, path to the executable or a **journalctl** predicate (see **coredumpctl(1) (https://man.archlinux.org/man/coredumpctl.1)** and **journalctl(1) (https://man.archlinux.org/man/journalctl.1)** for details). To see details of the core dumps:

```
# coredumpctl info match
```

Pay attention to "Signal" row, that helps to identify crash cause. For deeper analysis you can examine the backtrace using *gdb*:

```
# coredumpctl gdb match
```

When *gdb* is started, use the `bt` command to print the backtrace:

```
(gdb) bt
```

See **Debugging/Getting traces** if debugging symbols are requested, but not found.

Cleanup of core dump files

The core dump files stored in `/var/lib/systemd/coredump/` will be automatically cleaned by `systemd-tmpfiles --clean`, which is triggered daily with `systemd-tmpfiles-clean.timer`. Core dumps are configured to persist for at least 3 days, see `systemd-tmpfiles --cat-config`.

See also

- **american fuzzy lop (https://lcamtuf.coredump.cx/afl/)** - A tool for automated tests of the kernel and programs
- **Filesystem fuzzing (https://lwn.net/Articles/637151/)** - LWN article about testing filesystems for bugs