NAME

eatmydata − transparently disable fsync() and other data-to-disk synchronization calls

SYNOPSIS

eatmydata [--] *command* [*command arguments ...*]

DESCRIPTION

eatmydata runs a command in the environment where data-to-disk synchronization calls (like fsync(), fdatasync(), sync(), msync() and open() O_SYNC / O_DSYNC flags) have no effect. LD_PRELOAD library **libeatmydata** overrides respective C library calls with custom functions that don't trigger synchronization but return success nevertheless.

You may use **eatmydata** in two ways. In normal mode, just execute **eatmydata** directly and pass a command−to−be−run and its arguments via command line. In order to use symlink mode, create a symlink to */usr/bin/eatmydata* with the filename (a.k.a basename) of another program in the PATH and execute **eatmydata** via that symlink. Then **eatmydata** will find that program in the PATH and run it in the libeatmydata environment repassing all command line options.

OPTIONS

Please note that **eatmydata** does not process any command line options in symlink mode. All command line options will be repassed to the underlying executable as-is.

command
: The command to execute. It may be either a full path or the name of the command in PATH. In case command cannot be found in PATH, **eatmydata** will fail.

command arguments
: Arbitrary number of arguments to pass to the command being executed.

--
: Optional command separator for compatibility with similar utilities. Ignored at the moment.

EXAMPLES

Given PATH is /usr/bin and both /usr/bin/aptitude and /usr/bin/eatmydata are installed, the following:

```
$ ln −s /usr/bin/eatmydata ./aptitude
$ ./aptitude moo
```

is equivalent to:

```
$ eatmydata −− aptitude moo
```

Therefore, you may use symlink mode to automatically run specific programs in the libeatmydata environment whenever you run them from PATH. For example, given standard PATH settings, just do:

```
# ln −s /usr/bin/eatmydata /usr/local/bin/aptitude
```

and enjoy sync−free aptitude system-wide.

CAVEAT

When using **eatmydata** with **setarch** (including alias such as **linux32**), or anyway with chroots with a different architectures than the host's, make sure to install the matching architecture of **libeatmydata1** both in the **setarch** environment and host's.

Trying to load libeatmydata manually (without using the wrapper script) and using it through a chroot, especially if the eatmydata version differ between outside and inside, is probably going to fail do the different position of the library on the file system.
The safest way to manually load libeatmydata is by setting the following two environment variables (shell syntax):

```
LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+"$LD_LIBRARY_PATH:"}/usr/lib/libeatmydata
LD_PRELOAD=${LD_PRELOAD:+"$LD_PRELOAD "}libeatmydata.so
```

These two variables accounts the case of a Debian Jessie host with a Debian Wheezy chroot, where the position of the library changed.