Github repo:

https://github.com/ugol/canonical/

Exercise 1

VERBOSE=1 ./script.sh

or just

./script.sh

Init has been implemented directly as a simple C function.

Exercise 2

- Shred function implemented in go.
- Added a shredder_test.go unit test
- Added a main.go to test the function from the CLI
- Added some command line options similar to the original shredder command (to remove the file after shredding, to optionally zero the file with an additional pass to hide that's been shredded)

Notes:

This tool could be useful for:

- Secure file deletion: e.g., logs, key material, database exports.
- Compliance: To meet standards like GDPR, HIPAA, etc.
- Sanitization of temp files: Cleanup for privacy-focused tools.

Anyway, is not considered anymore an effective practice, mainly because in journaling file systems the original data may survive to a shredding (the file system may first write the new data to the journal, not to the file's original blocks)

Also, it wouldn't be effective at all for SSD (journaling or not journaling) because of wear leveling: NAND flash cells can only be written a limited number of times, so wear-leveling spreads out writes to avoid wearing out specific blocks.
The effect is that when you overwrite a file, the SSD may write the new data to a different physical block and mark the old one as invalid, but it still physically exists until garbage-collected.

The best thing to do is to use full disk encryption and not shredding tools.