go-criu -- Go bindings for CRIU

This repository provides Go bindings for CRIU. The code is based on the Go-based PHaul implementation from the CRIU repository. For easier inclusion into other Go projects the CRIU Go bindings have been moved to this repository.

The Go bindings provide an easy way to use the CRIU RPC calls from Go without the need to set up all the infrastructure to make the actual RPC connection to CRIU.

The following example would print the version of CRIU:

```
import (
    "log"

    "github.com/checkpoint-restore/go-criu/v5"
)

func main() {
    c := criu.MakeCriu()
    version, err := c.GetCriuVersion()
    if err != nil {
        log.Fatalln(err)
    }
    log.Println(version)
}
```

or to just check if at least a certain CRIU version is installed:

```
    c := criu.MakeCriu()
    result, err := c.IsCriuAtLeast(31100)
```

Releases

The first go-criu release was 3.11 based on CRIU 3.11. The initial plan was to follow CRIU so that go-criu would carry the same version number as CRIU.

As go-criu is imported in other projects and as Go modules are expected to follow Semantic Versioning go-criu will also follow Semantic Versioning starting with the 4.0.0 release.

The following table shows the relation between go-criu and criu versions:

Major version	Latest release	CRIU version
v5	5.2.0	3.16
v5	5.0.0	3.15
v4	4.1.0	3.14

How to contribute

While bug fixes can first be identified via an "issue", that is not required. It's ok to just open up a PR with the fix, but make sure you include the same information you would have included in an issue - like how to reproduce it.

PRs for new features should include some background on what use cases the new code is trying to address. When possible and when it makes sense, try to break-up larger PRs into smaller ones - it's easier to review smaller code changes. But only if those smaller ones make sense as stand-alone PRs.

Regardless of the type of PR, all PRs should include:

- well documented code changes
- additional testcases. Ideally, they should fail w/o your code change applied
- documentation changes

Squash your commits into logical pieces of work that might want to be reviewed separate from the rest of the PRs. Ideally, each commit should implement a single idea, and the PR branch should pass the tests at every commit. GitHub makes it easy to review the cumulative effect of many commits; so, when in doubt, use smaller commits.

PRs that fix issues should include a reference like `Closes #XXXX` in the commit message so that github will automatically close the referenced issue when the PR is merged.

Contributors must assert that they are in compliance with the Developer Certificate of Origin 1.1. This is achieved by adding a "Signed-off-by" line containing the contributor's name and e-mail to every commit message. Your signature certifies that you wrote the patch or otherwise have the right to pass it on as an open-source patch.

License and copyright

Unless mentioned otherwise in a specific file's header, all code in this project is released under the Apache 2.0 license.

The author of a change remains the copyright holder of their code (no copyright assignment). The list of authors and contributors can be retrieved from the git commit history and in some cases, the file headers.