pwalkdir: parallel implementation of filepath.WalkDir

This is a wrapper for filepath.WalkDir which may speed it up by calling multiple callback functions (WalkDirFunc) in parallel, utilizing goroutines.

By default, it utilizes 2*runtime.NumCPU() goroutines for callbacks. This can be changed by using WalkN function which has the additional parameter, specifying the number of goroutines (concurrency).

pwalk vs pwalkdir

This package is very similar to pwalk, but utilizes `filepath.WalkDir` (added to Go 1.16), which does not call stat(2) on every entry and is therefore faster (up to 3x, depending on usage scenario).

Users who are OK with requiring Go 1.16+ should switch to this implementation.

Caveats

Please note the following limitations of this code:

- Unlike filepath.WalkDir, the order of calls is non-deterministic;
- Only primitive error handling is supported:
  - fs.SkipDir is not supported;
  - no errors are ever passed to WalkDirFunc;
  - once any error is returned from any walkDirFunc instance, no more calls to WalkDirFunc are made, and the error is returned to the caller of WalkDir;
  - if more than one WalkDirFunc instance will return an error, only one of such errors will be propagated to and returned by WalkDir, others will be silently discarded.

Documentation

For the official documentation, see https://pkg.go.dev/github.com/opencontainers/selinux/pkg/pwalkdir

Benchmarks

For a WalkDirFunc that consists solely of the return statement, this implementation is about 15% slower than the standard library's filepath.WalkDir.

Otherwise (if a WalkDirFunc is actually doing something) this is usually faster, except when the WalkDirN(..., 1) is used. Run `go test -bench .` to see how different operations can benefit from it, as well as how the level of paralellism affects the speed.