

## **pwalk: parallel implementation of filepath.Walk**

This is a wrapper for [filepath.Walk](#) which may speed it up by calling multiple callback functions (WalkFunc) in parallel, utilizing goroutines.

By default, it utilizes  $2 * \text{runtime.NumCPU}()$  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 deprecated in favor of [pwalkdir](#), which is faster, but requires at least Go 1.16.

### **Caveats**

Please note the following limitations of this code:

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

### **Documentation**

For the official documentation, see <https://pkg.go.dev/github.com/opencontainers/selinux/pkg/pwalk?tab=doc>

### **Benchmarks**

For a WalkFunc that consists solely of the return statement, this implementation is about 10% slower than the standard library's filepath.Walk.

Otherwise (if a WalkFunc is doing something) this is usually faster, except when the WalkN(..., 1) is used.