# CONTENTS

[CONTENTS 1](#_Toc525666685)

[1 DESCRIPTION 1](#_Toc525666686)

[2 USAGE 1](#_Toc525666687)

[2.1 Primary Modes 2](#_Toc525666688)

[2.2 Secondary Modes 2](#_Toc525666689)

[2.3 Options 3](#_Toc525666690)

[Timestamp Considerations 3](#_Toc525666691)

# DESCRIPTION

pwalk - a high-speed multi-threaded tree walker

# USAGE

Invoking pwalk with no arguments will produce a usage summary. Available options may vary between target platforms.

OSX pwalk usage;

pwalk 2.02 BETA-1 OSX

Usage: pwalk [<mode>] [<option> ...] <directory> [<directory> ...]

Where:

<mode> is at most ONE of:

-ls // create .ls outputs (like ls -l)

-xml // create .xml outputs

-fix\_times // create .fix outputs (or just enumerate with -dryrun)

-csv=<ifile> // create .csv outputs based on fields in <ifile>

<option> values are:

-cd=<relative\_path> // cd to here before evaluating directory arguments

-dop=<n> // specifies the Degree Of Parallelism (max number of workers)

-paths=<paths\_file> // specify equivalent pathname prefixes for multi-pathing

-gz // gzip output files (HANGS on OSX!!)

-dryrun // suppress making changes (with -fix\_times)

-shadow=<shadow\_d> // shadow directory; optional (with -fix\_times)

-pmode // suppress showing formatted mode bits (with -ls and -xml)

+crc // also ... calculate CRC for each file (READS ALL FILES!)

+denist // also ... read first 128 bytes of every file encountered

+tally[=<tag>] // also ... output bucketized file/space counts in .tally file

+tstat // also ... include hi-res timing statistics in some outputs

+.snapshot // also ... traverse .snapshot[s] directories (OFF by default)

<directory> ... // one or more directories to traverse

OneFS pwalk usage;

pwalk 2.02 beta OneFS

Usage: pwalk [<mode>] [<option> ...] <directory> [<directory> ...]

Where:

<mode> is at most ONE of:

-ls // create .ls outputs (like ls -l)

-xml // create .xml outputs

-audit // create .audit files based on OneFS SmartLock status

-fix\_times // create .fix outputs (or just enumerate with -dryrun)

-csv=<ifile> // create .csv outputs based on fields in <ifile>

<option> values are:

-cd=<relative\_path> // cd to here before evaluating directory arguments

-dop=<n> // specifies the Degree Of Parallelism (max number of workers)

-paths=<paths\_file> // specify equivalent pathname prefixes for multi-pathing

-gz // gzip output files (HANGS on OSX!!)

-dryrun // suppress making changes (with -fix\_times)

-shadow=<shadow\_d> // shadow directory; optional (with -fix\_times)

-pmode // suppress showing formatted mode bits (with -ls and -xml)

+crc // also ... calculate CRC for each file (READS ALL FILES!)

+denist // also ... read first 128 bytes of every file encountered

+tally[=<tag>] // also ... output bucketized file/space counts in .tally file

+rm\_acls // also ... remove non-inherited ACEs in any ACLs encountered

+tstat // also ... include hi-res timing statistics in some outputs

+.snapshot // also ... traverse .snapshot[s] directories (OFF by default)

<directory> ... // one or more directories to traverse

Linux pwalk usage;

pwalk 2.02 BETA-1 Linux

Usage: pwalk [<mode>] [<option> ...] <directory> [<directory> ...]

Where:

<mode> is at most ONE of:

-ls // create .ls outputs (like ls -l)

-xml // create .xml outputs

-fix\_times // create .fix outputs (or just enumerate with -dryrun)

-csv=<ifile> // create .csv outputs based on fields in <ifile>

<option> values are:

-cd=<relative\_path> // cd to here before evaluating directory arguments

-dop=<n> // specifies the Degree Of Parallelism (max number of workers)

-paths=<paths\_file> // specify equivalent pathname prefixes for multi-pathing

-gz // gzip output files (HANGS on OSX!!)

-dryrun // suppress making changes (with -fix\_times)

-shadow=<shadow\_d> // shadow directory; optional (with -fix\_times)

-pmode // suppress showing formatted mode bits (with -ls and -xml)

+crc // also ... calculate CRC for each file (READS ALL FILES!)

+denist // also ... read first 128 bytes of every file encountered

+tally[=<tag>] // also ... output bucketized file/space counts in .tally file

+acls // also ... include ACL info in some outputs, eg: '+'

+wacls=<command> // also ... write derived binary NFS4 ACLs to <command>

+xacls=[bin|nfs|chex] // also ... create .acl4bin, .acl4nfs, .acl4chex outputs

+tstat // also ... include hi-res timing statistics in some outputs

+.snapshot // also ... traverse .snapshot[s] directories (OFF by default)

<directory> ... // one or more directories to traverse

## Primary Modes

**-ls** - For each directory, ls displays the names of files contained within that directory, as well as any requested, associated information. Per-directory subtotals are calculated and included the output.

## Secondary Modes

**+rm\_acls**

The now-poorly-named +rm\_acls option enables the new ACL-clearing functionality;

It is assumed this will be run by root.

All non-inherited ACEs of all ACLs encountered will be removed

BEWARE: pwalk +rm\_acls will also remove the heritable ACEs from the parent directory if the parent directory is included in the pwalk scan. To work around this and avoid issues with passing long argument lists to pwalk, it’s probably best to just let that happen, and then re-instate the heritable ACEs on the parent directory as a followup step.

For all actions performed by pwalk +rm\_acls, a line will be written the workerNNN.err file

No output is ever generated for files or directories that do not have actual OneFS ACLs. For each file or directory with an ACL, each output line will be in the format of ‘@ <action> <pathname>’, where <action> is;

* NOP - no action taken
* MOD - ACL modified
* REM - ACL removed  (converted to mode bits)
* FIX - ACL ‘fixed’ (converted to mode bits) - If a OneFS ACL was previously removed ('chmod -N’) or emptied (‘chmod -D’) it will be ‘FIXed’ by replacing it with whatever were the apparent POSIX mode bits as far as stat(2) was concerned

NOTES:

* Whenever an ACL is replaced by POSIX mode bits, its setuid/setgid/sticky bits will be preserved.
* Any file or directory that is reported as REM or FIX will subsequently show as having a SYNTHETIC ACL when a native OneFS ‘ls -le’ command is used.
* The -dryrun option can be used to prevent any net changes arising from +rm\_acls operation, but the output will appear as though all actions succeeded
* The +debug option can be used to spew tons of debug output to stderr and the pwalk log file
* The potential performance impact of running pwalk +rm\_acls natively on OneFS should be properly considered

## Options

-dop=<N>

All pwalk worker threads run on a single node; it’s multi-threaded, but not ‘clusterized’

pwalk concurrency as high as -dop=16 should be OK on any modern Isilon node

Beware using any concurrency level that drives CPU usage beyond 80%

At high concurrency, pwalk could have a negative impact on competing workloads

# Timestamp Considerations

OneFS atime setting ...

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Operation** | **OneFS Native** | **OSX NFSv3** | **OSX NFSv4** | **OSX SMB** | **Linux NFSv3** | **Linux NFSv4** | **Linux SMB** |
| **Read atime, mtime, ctime** | ✓ | ✓ | ✓ | ✓[1] | ✓ | ✓ | ✓[1] |
| **Read birthtime** | ✓ | ✗[2] | ? | ✓[1] | ✗[2] | ? | ✓[1] |
| **Set atime, mtime** | ✓[5] | ✓[3][4] | ✓[3][4] | ✓[1][3][4] | ✓[3][4] | ✓[3][4] | ✓[1][3][4] |
| **Set birthtime** | ✓[5] | ✗ | ✗ | ✓[1][2] | ✗ | ✗ | ✓[1][2] |
| **utimes(2) reverts birthtime when mtime predates birthtime** | ✓[7] | ✗[8] | ✗[8] | ✗[8] | ✗[8] | ✗[8] | ✗[8] |

[1] values subject to mangling to and from Microsoft FILETIME format (BUG: ====)

[2] btime presented as a blind copy of ctime; not the actual btime

[3] values limited to microsecond precision by utimes(2)

[4] atime set by OneFS rather than by value passed to utimes(2) (BUG: ====)

[5] second utimes() call not required to set atime and mtime (BUG: ====)