NAME

poolset − persistent memory pool configuration file format

SYNOPSIS

```
mypool.set
```

DESCRIPTION

Depending on the configuration of the system, the available non−volatile memory space may be divided into multiple memory devices. In such case, the maximum size of the transactional object store could be limited by the capacity of a single memory device. Therefore,**libpmemobj**(7), **libpmemblk**(7) and **libpmemlog**(7) allow building object stores spanning multiple memory devices by creation of persistent memory pools consisting of multiple files, where each part of such a *pool set* may be stored on a different pmem−aware filesystem.

To improve reliability and eliminate single point of failure, **libpmemobj**(7) also allows all the data written to a persistent memory pool to be copied to local or remote pool *replicas*, thereby providing backup for the persistent memory pool by producing a *mirrored pool set*. In practice, the pool replicas may be considered as binary copies of the "master" pool set. Data replication is not supported in **libpmemblk**(7) and **libpmemlog**(7).

The *set* file for each type of pool is a plain text file. Lines in the file are formatted as follows:

- The first line of the file must be the literal string "PMEMPOOLSET"
- The pool parts are specified, one per line, in the format:

  size pathname
- *Replica* sections, if any, start with the literal string "REPLICA". See **REPLICAS**, below, for further details.
- Pool set options, if any, start with literal string *OPTION*. See**POOL SET OPTIONS** belo w for details.
- Lines starting with "#" are considered comments and are ignored.

The *size* must be compliant with the format specified in IEC 80000−13, IEEE 1541 or the Metric Interchange Format. These standards accept SI units with obligatory B − kB, MB, GB, ... (multiplier by 1000) suffixes, and IEC units with optional "iB" − KiB, MiB, GiB, ..., K, M, G, ... − (multiplier by 1024) suffixes.

pathname must be an absolute pathname.

The *pathname* of a part can point to a Device DAX. Device DAX is the device−centric analogue of Filesystem DAX. It allows memory ranges to be allocated and mapped without need of an intervening file system.

Pools created on Device DAX have additional options and restrictions:

- The *size* may be set to "AUTO", in which case the size of the device will be automatically resolved at pool creation time.
- To concatenate more than one Device DAX device into a single pool set, the configured internal alignment of the devices must be 4KiB, unless the *SINGLEHDR* or *NOHDRS* option is used in the pool set file. See**POOL SET OPTIONS** belo w for details.

Please see **ndctl−create−namespace**(1) for more information on Device DAX, including how to configure desired alignment.

The minimum file size of each part of the pool set is defined as follows:

- For block pools, as **PMEMBLK_MIN_PART** in **<libpmemblk.h>**
- For object pools, as **PMEMOBJ_MIN_PART** in **<libpmemobj.h>**
- For log pools, as **PMEMLOG_MIN_PART** in **<libpmemlog.h>**

The net pool size of the pool set is equal to:

```
net_pool_size = sum_over_all_parts(page_aligned_part_size − 4KiB) + 4KiB
```

where

```
page_aligned_part_size = part_size & ~(page_size − 1)
```

Note that page size is OS specific. For more information please see **sysconf**(3).

The minimum net pool size of a pool set is defined as follows:

- For block pools, as **PMEMBLK_MIN_POOL** in <**libpmemblk.h**>
- For object pools, as **PMEMOBJ_MIN_POOL** in <**libpmemobj.h**>
- For log pools, as **PMEMLOG_MIN_POOL** in <**libpmemlog.h**>

Here is an example "mypool.set" file:

```
PMEMPOOLSET
OPTION NOHDRS
100G /mountpoint0/myfile.part0
200G /mountpoint1/myfile.part1
400G /mountpoint2/myfile.part2
```

The files in the set may be created by running one of the following commands. To create a block pool:

```
$ pmempool create blk <bsize> mypool.set
```

To create a log pool:

```
$ pmempool create log mypool.set
```

REPLICAS

Sections defining replica sets are optional. There may be multiple replica sections.

Local replica sections begin with a line containing only the literal string "REPLICA", followed by one or more pool part lines as described above.

Remote replica sections consist of the *REPLICA* keyword, followed on the same line by the address of a remote host and a relative path to a remote pool set file:

```
REPLICA [<user>@]<hostname> [<relative-path>/]<remote-pool-set-file>
```

- *hostname* must be in the format recognized by the **ssh**(1) remote login client
- *pathname* is relative to the root config directory on the target node − see **librpmem**(7)

There are no other lines in the remote replica section − the REPLICA line defines a remote replica entirely.

Here is an example "myobjpool.set" file with replicas:

```
PMEMPOOLSET
100G /mountpoint0/myfile.part0
200G /mountpoint1/myfile.part1
400G /mountpoint2/myfile.part2

# local replica
REPLICA
500G /mountpoint3/mymirror.part0
200G /mountpoint4/mymirror.part1

# remote replica
REPLICA user@example.com remote-objpool.set
```

The files in the object pool set may be created by running the following command:

```
$ pmempool create --layout="mylayout" obj myobjpool.set
```

Remote replica cannot have replicas, i.e. a remote pool set file cannot define any replicas.

POOL SET OPTIONS

Pool set options can appear anywhere after the line with *PMEMPOOLSET* string. Pool set file can contain several pool set options. The following options are supported:

- *SINGLEHDR*
- *NOHDRS*

If the *SINGLEHDR* option is used, only the first part in each replica contains the pool part internal metadata. In that case the effective size of a replica is the sum of sizes of all its part files decreased once by 4096 bytes.

The *NOHDRS* option can appear only in the remote pool set file, when **librpmem** does not serve as a means of replication for **libpmemobj** pool. In that case none of the pool parts contains internal metadata. The effective size of such a replica is the sum of sizes of all its part files.

Options *SINGLEHDR* and *NOHDRS* are mutually exclusive. If both are specified in a pool set file, creating or opening the pool will fail with an error.

When using the *SINGLEHDR* or *NOHDRS* option, one can concatenate more than one Device DAX devices with any internal alignments in one replica.

The *SINGLEHDR* option concerns only replicas that are local to the pool set file. That is if one wants to create a pool set with the *SINGLEHDR* option and with remote replicas, one has to add this option to the local pool set file as well as to every single remote pool set file.

Using the *SINGLEHDR* and *NOHDRS* options has important implications for data integrity checking and recoverability in case of a pool set damage. See **pmempool_sync**() API for more information about pool set recovery.

DIRECTORIES

Providing a directory as a part's *pathname* allows the pool to dynamically create files and consequently removes the user−imposed limit on the size of the pool.

The *size* argument of a part in a directory poolset becomes the size of the address space reservation required for the pool. In other words, the size argument is the maximum theoretical size of the mapping. This value can be freely increased between instances of the application, but decreasing it below the real required space will result in an error when attempting to open the pool.

The directory must NOT contain user created files with extension *.pmem*, otherwise the behavior is undefined. If a file created by the library within the directory is in any way altered (resized, renamed) the behavior is undefined.

A directory poolset must exclusively use directories to specify paths − combining files and directories will result in an error. A single replica can consist of one or more directories. If there are multiple directories, the address space reservation is equal to the sum of the sizes.

The order in which the files are created is unspecified, but the library will try to maintain equal usage of the directories.

By default pools grow in 128 megabyte increments.

Only poolsets with the *SINGLEHDR* option can safely use directories.

NOTES

Creation of all the parts of the pool set and the associated replica sets can be done with the **pmemobj_create**(3), **pmemblk_create**(3) or **pmemlog_create**(3) function, or by using the **pmempool**(1) utility.

Restoring data from a local or remote replica can be done by using the **pmempool−sync**(1) command or the **pmempool_sync**() API from the **libpmempool**(7) library.

Modifications of a pool set file configuration can be done by using the **pmempool−transform**(1) command or the **pmempool_transform**() API from the **libpmempool**(7) library.

When creating a pool set consisting of multiple files, or when creating a replicated pool set, the *path* argument passed to **pmemobj_create**(3), **pmemblk_create**(3) or **pmemlog_create**(3) must point to the special

set file that defines the pool layout and the location of all the parts of the pool set.

When opening a pool set consisting of multiple files, or when opening a replicated pool set, the *path* argument passed to **pmemobj_open**(3), **pmemblk_open**(3) or **pmemlog_open**(3) must point to the same *set* file that was used for pool set creation.

SEE ALSO

ndctl−create−namespace(1), **pmemblk_create**(3), **pmemlog_create**(3), **pmemobj_create**(3), **sysconf**(3), **libpmemblk**(7), **libpmemlog**(7), **libpmemobj**(7) and **<https://pmem.io>**