FILE LAYOUTS

The layout of a file controls how its contents are mapped to Ceph RADOS objects. You can read and write a file's layout using *virtual extended attributes* or xattrs.

The name of the layout xattrs depends on whether a file is a regular file or a directory. Regular files' layout xattrs are called `ceph.file.layout`, whereas directories' layout xattrs are called `ceph.dir.layout`. Where subsequent examples refer to `ceph.file.layout`, substitute `dir` as appropriate when dealing with directories.

Tip: Your linux distribution may not ship with commands for manipulating xattrs by default, the required package is usually called `attr`.

LAYOUT FIELDS

pool
: String, giving ID or name. Which RADOS pool a file's data objects will be stored in.

pool_namespace
: String. Within the data pool, which RADOS namespace the objects will be written to. Empty by default (i.e. default namespace).

stripe_unit
: Integer in bytes. The size (in bytes) of a block of data used in the RAID 0 distribution of a file. All stripe units for a file have equal size. The last stripe unit is typically incomplete–i.e. it represents the data at the end of the file as well as unused "space" beyond it up to the end of the fixed stripe unit size.

stripe_count
: Integer. The number of consecutive stripe units that constitute a RAID 0 "stripe" of file data.

object_size
: Integer in bytes. File data is chunked into RADOS objects of this size.

Tip: RADOS enforces a configurable limit on object sizes: if you increase CephFS object sizes beyond that limit then writes may not succeed. The OSD setting is `osd_max_object_size`, which is 128MB by default. Very large RADOS objects may prevent smooth operation of the cluster, so increasing the object size limit past the default is not recommended.

READING LAYOUTS WITH GETFATTR

Read the layout information as a single string:

```
$ touch file
$ getfattr -n ceph.file.layout file
# file: file
ceph.file.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304 pool=cephfs_data"
```

Read individual layout fields:

```
$ getfattr -n ceph.file.layout.pool file
# file: file
ceph.file.layout.pool="cephfs_data"
$ getfattr -n ceph.file.layout.stripe_unit file
# file: file
ceph.file.layout.stripe_unit="4194304"
$ getfattr -n ceph.file.layout.stripe_count file
# file: file
ceph.file.layout.stripe_count="1"
$ getfattr -n ceph.file.layout.object_size file
# file: file
ceph.file.layout.object_size="4194304"
```

Note: When reading layouts, the pool will usually be indicated by name. However, in rare cases when pools have only just

been created, the ID may be output instead.

Directories do not have an explicit layout until it is customized. Attempts to read the layout will fail if it has never been modified: this indicates that layout of the next ancestor directory with an explicit layout will be used.

```
$ mkdir dir
$ getfattr -n ceph.dir.layout dir
dir: ceph.dir.layout: No such attribute
$ setfattr -n ceph.dir.layout.stripe_count -v 2 dir
$ getfattr -n ceph.dir.layout dir
# file: dir
ceph.dir.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304 pool=cephfs_data"
```

WRITING LAYOUTS WITH SETFATTR

Layout fields are modified using `setfattr`:

```
$ ceph osd lspools
0 rbd,1 cephfs_data,2 cephfs_metadata,

$ setfattr -n ceph.file.layout.stripe_unit -v 1048576 file2
$ setfattr -n ceph.file.layout.stripe_count -v 8 file2
$ setfattr -n ceph.file.layout.object_size -v 10485760 file2
$ setfattr -n ceph.file.layout.pool -v 1 file2  # Setting pool by ID
$ setfattr -n ceph.file.layout.pool -v cephfs_data file2  # Setting pool by name
```

Note: When the layout fields of a file are modified using `setfattr`, this file must be empty, otherwise an error will occur.

```
# touch an empty file
$ touch file1
# modify layout field successfully
$ setfattr -n ceph.file.layout.stripe_count -v 3 file1

# write something to file1
$ echo "hello world" > file1
$ setfattr -n ceph.file.layout.stripe_count -v 4 file1
setfattr: file1: Directory not empty
```

CLEARING LAYOUTS

If you wish to remove an explicit layout from a directory, to revert to inherting the layout of its ancestor, you can do so:

```
setfattr -x ceph.dir.layout mydir
```

Similarly, if you have set the `pool_namespace` attribute and wish to modify the layout to use the default namespace instead:

```
# Create a dir and set a namespace on it
mkdir mydir
setfattr -n ceph.dir.layout.pool_namespace -v foons mydir
getfattr -n ceph.dir.layout mydir
ceph.dir.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304 pool=cephfs_data_a po

# Clear the namespace from the directory's layout
setfattr -x ceph.dir.layout.pool_namespace mydir
getfattr -n ceph.dir.layout mydir
ceph.dir.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304 pool=cephfs_data_a"
```

INHERITANCE OF LAYOUTS

Files inherit the layout of their parent directory at creation time. However, subsequent changes to the parent directory's layout do not affect children.

```
$ getfattr -n ceph.dir.layout dir
# file: dir
ceph.dir.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304 pool=cephfs_data"

# Demonstrate file1 inheriting its parent's layout
$ touch dir/file1
$ getfattr -n ceph.file.layout dir/file1
# file: dir/file1
ceph.file.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304 pool=cephfs_data"

# Now update the layout of the directory before creating a second file
$ setfattr -n ceph.dir.layout.stripe_count -v 4 dir
$ touch dir/file2

# Demonstrate that file1's layout is unchanged
$ getfattr -n ceph.file.layout dir/file1
# file: dir/file1
ceph.file.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304 pool=cephfs_data"

# ...while file2 has the parent directory's new layout
$ getfattr -n ceph.file.layout dir/file2
# file: dir/file2
ceph.file.layout="stripe_unit=4194304 stripe_count=4 object_size=4194304 pool=cephfs_data"
```

Files created as descendents of the directory also inherit the layout, if the intermediate directories do not have layouts set:

```
$ getfattr -n ceph.dir.layout dir
# file: dir
ceph.dir.layout="stripe_unit=4194304 stripe_count=4 object_size=4194304 pool=cephfs_data"
$ mkdir dir/childdir
$ getfattr -n ceph.dir.layout dir/childdir
dir/childdir: ceph.dir.layout: No such attribute
$ touch dir/childdir/grandchild
$ getfattr -n ceph.file.layout dir/childdir/grandchild
# file: dir/childdir/grandchild
ceph.file.layout="stripe_unit=4194304 stripe_count=4 object_size=4194304 pool=cephfs_data"
```

ADDING A DATA POOL TO THE MDS

Before you can use a pool with CephFS you have to add it to the Metadata Servers.

```
$ ceph fs add_data_pool cephfs cephfs_data_ssd
$ ceph fs ls  # Pool should now show up
.... data pools: [cephfs_data cephfs_data_ssd ]
```

Make sure that your cephx keys allows the client to access this new pool.

You can then update the layout on a directory in CephFS to use the pool you added:

```
$ mkdir /mnt/cephfs/myssddir
$ setfattr -n ceph.dir.layout.pool -v cephfs_data_ssd /mnt/cephfs/myssddir
```

All new files created within that directory will now inherit its layout and place their data in your newly added pool.

You may notice that object counts in your primary data pool (the one passed to `fs new`) continue to increase, even if files are being created in the pool you added. This is normal: the file data is stored in the pool specified by the layout, but a small amount of metadata is kept in the primary data pool for all files.