The cgroup freezer is useful to batch job management system which start and stop sets of tasks in order to schedule the resources of a machine according to the desires of a system administrator. This sort of program is often used on HPC clusters to schedule access to the cluster as a whole. The cgroup freezer uses cgroups to describe the set of tasks to be started/stopped by the batch job management system. It also provides a means to start and stop the tasks composing the job.

The cgroup freezer will also be useful for checkpointing running groups of tasks. The freezer allows the checkpoint code to obtain a consistent image of the tasks by attempting to force the tasks in a cgroup into a quiescent state. Once the tasks are quiescent another task can walk /proc or invoke a kernel interface to gather information about the quiesced tasks. Checkpointed tasks can be restarted later should a recoverable error occur. This also allows the checkpointed tasks to be migrated between nodes in a cluster by copying the gathered information to another node and restarting the tasks there.

Sequences of SIGSTOP and SIGCONT are not always sufficient for stopping and resuming tasks in userspace. Both of these signals are observable from within the tasks we wish to freeze. While SIGSTOP cannot be caught, blocked, or ignored it can be seen by waiting or ptracing parent tasks. SIGCONT is especially unsuitable since it can be caught by the task. Any programs designed to watch for SIGSTOP and SIGCONT could be broken by attempting to use SIGSTOP and SIGCONT to stop and resume tasks. We can demonstrate this problem using nested bash shells:

```
$ echo $$
16644
$ bash
$ echo $$
16690

From a second, unrelated bash shell:
$ kill -SIGSTOP 16690
$ kill -SIGCONT 16990

<at this point 16990 exits and causes 16644 to exit too>
```

This happens because bash can observe both signals and choose how it responds to them.

Another example of a program which catches and responds to these signals is gdb. In fact any program designed to use ptrace is likely to have a problem with this method of stopping and resuming tasks.

In contrast, the cgroup freezer uses the kernel freezer code to prevent the freeze/unfreeze cycle from becoming visible to the tasks being frozen. This allows the bash example above and gdb to run as expected.

The freezer subsystem in the container filesystem defines a file named freezer.state. Writing "FROZEN" to the state file will freeze all tasks in the cgroup. Subsequently writing "THAWED" will unfreeze the tasks in the cgroup. Reading will return the current state.

Note freezer.state doesn't exist in root cgroup, which means root cgroup
is non-freezable.

* Examples of usage :

```
# mkdir /containers
# mount -t cgroup -ofreezer freezer  /containers
# mkdir /containers/0
# echo $some_pid > /containers/0/tasks
```

to get status of the freezer subsystem :

```
# cat /containers/0/freezer.state
THAWED
```

to freeze all tasks in the container :

```
# echo FROZEN > /containers/0/freezer.state
# cat /containers/0/freezer.state
FREEZING
# cat /containers/0/freezer.state
FROZEN
```

to unfreeze all tasks in the container :

```
# echo THAWED > /containers/0/freezer.state
# cat /containers/0/freezer.state
THAWED
```

This is the basic mechanism which should do the right thing for user space task
in a simple scenario.

It's important to note that freezing can be incomplete. In that case we return
EBUSY. This means that some tasks in the cgroup are busy doing something that
prevents us from completely freezing the cgroup at this time. After EBUSY,
the cgroup will remain partially frozen -- reflected by freezer.state reporting
"FREEZING" when read. The state will remain "FREEZING" until one of these
things happens:

1) Userspace cancels the freezing operation by writing "THAWED" to
the freezer.state file
2) Userspace retries the freezing operation by writing "FROZEN" to
the freezer.state file (writing "FREEZING" is not legal
and returns EINVAL)
3) The tasks that blocked the cgroup from entering the "FROZEN"
state disappear from the cgroup's set of tasks.