To support containers, we now allow multiple instances of devpts filesystem, such that indices of ptys allocated in one instance are independent of indices allocated in other instances of devpts.

To preserve backward compatibility, this support for multiple instances is enabled only if:

- CONFIG_DEVPTS_MULTIPLE_INSTANCES=y, and
- '-o newinstance' mount option is specified while mounting devpts

IOW, devpts now supports both single-instance and multi-instance semantics.

If CONFIG_DEVPTS_MULTIPLE_INSTANCES=n, there is no change in behavior and this referred to as the "legacy" mode. In this mode, the new mount options (-o newinstance and -o ptmxmode) will be ignored with a 'bogus option' message on console.

If CONFIG_DEVPTS_MULTIPLE_INSTANCES=y and devpts is mounted without the 'newinstance' option (as in current start-up scripts) the new mount binds to the initial kernel mount of devpts. This mode is referred to as the 'single-instance' mode and the current, single-instance semantics are preserved, i.e PTYs are common across the system.

The only difference between this single-instance mode and the legacy mode is the presence of new, '/dev/pts/ptmx' node with permissions 0000, which can safely be ignored.

If CONFIG_DEVPTS_MULTIPLE_INSTANCES=y and 'newinstance' option is specified, the mount is considered to be in the multi-instance mode and a new instance of the devpts fs is created. Any ptys created in this instance are independent of ptys in other instances of devpts. Like in the single-instance mode, the /dev/pts/ptmx node is present. To effectively use the multi-instance mode, open of /dev/ptmx must be a redirected to '/dev/pts/ptmx' using a symlink or bind-mount.

Eg: A container startup script could do the following:

```
$ chmod 0666 /dev/pts/ptmx
$ rm /dev/ptmx
$ ln -s pts/ptmx /dev/ptmx
$ ns_exec -cm /bin/bash

# We are now in new container

$ umount /dev/pts
$ mount -t devpts -o newinstance lxcpts /dev/pts
$ sshd -p 1234
```

where 'ns_exec -cm /bin/bash' calls clone() with CLONE_NEWNS flag and execs /bin/bash in the child process. A pty created by the sshd is not visible in the original mount of /dev/pts.

User-space changes

In multi-instance mode (i.e '-o newinstance' mount option is specified at least once), following user-space issues should be noted.

1. If -o newinstance mount option is never used, /dev/pts/ptmx can be ignored and no change is needed to system-startup scripts.

2. To effectively use multi-instance mode (i.e -o newinstance is specified) administrators or startup scripts should "redirect" open of /dev/ptmx to /dev/pts/ptmx using either a bind mount or symlink.

```
$ mount -t devpts -o newinstance devpts /dev/pts
```

   followed by either

```
$ rm /dev/ptmx
$ ln -s pts/ptmx /dev/ptmx
$ chmod 666 /dev/pts/ptmx
```

   or

```
$ mount -o bind /dev/pts/ptmx /dev/ptmx
```

3. The '/dev/ptmx -> pts/ptmx' symlink is the preferred method since it enables better error-reporting and treats both single-instance and multi-instance mounts similarly.

   But this method requires that system-startup scripts set the mode of /dev/pts/ptmx correctly (default mode is 0000). The scripts can set the mode by, either

   - adding ptmxmode mount option to devpts entry in /etc/fstab, or
   - using 'chmod 0666 /dev/pts/ptmx'

4. If multi-instance mode mount is needed for containers, but the system startup scripts have not yet been updated, container-startup scripts should bind mount /dev/ptmx to /dev/pts/ptmx to avoid breaking single-instance mounts.

   Or, in general, container-startup scripts should use:

```
mount -t devpts -o newinstance -o ptmxmode=0666 devpts /dev/pts
if [ ! -L /dev/ptmx ]; then
        mount -o bind /dev/pts/ptmx /dev/ptmx
fi
```

   When all devpts mounts are multi-instance, /dev/ptmx can permanently be a symlink to pts/ptmx and the bind mount can be ignored.

5. A multi-instance mount that is not accompanied by the /dev/ptmx to /dev/pts/ptmx redirection would result in an unusable/unreachable pty.

```
mount -t devpts -o newinstance lxcpts /dev/pts
```

   immediately followed by:

```
open("/dev/ptmx")
```

   would create a pty, say /dev/pts/7, in the initial kernel mount.

But /dev/pts/7 would be invisible in the new mount.

6. The permissions for /dev/pts/ptmx node should be specified when mounting /dev/pts, using the '-o ptmxmode=%o' mount option (default is 0000).

   mount -t devpts -o newinstance -o ptmxmode=0644 devpts /dev/pts

   The permissions can be later be changed as usual with 'chmod'.

   chmod 666 /dev/pts/ptmx

7. A mount of devpts without the 'newinstance' option results in binding to initial kernel mount. This behavior while preserving legacy semantics, does not provide strict isolation in a container environment. i.e by mounting devpts without the 'newinstance' option, a container could get visibility into the 'host' or root container's devpts.

   To workaround this and have strict isolation, all mounts of devpts, including the mount in the root container, should use the newinstance option.