systemd.mstack

- Description
- Examples
- See Also

systemd

systemd.mstack

7

systemd.mstack

Mount stacks in self descriptive directories

Description

Directories with the `.mstack/` suffix may encode 'mount stacks' for assembling OS mount hierarchies based on bind and overlay mounts, for use in `systemd-nspawn(1)`'s `--mstack=` switch or the service manager's `RootMStack=` setting for services. `.mstack/` directories may contain various files and subdirectories, where each will effect one layer of an `overlayfs` mount, or a bind mount. The name of the file or subdirectory indicates how it shall be used in the mount hierarchy. Specifically, the following names are defined:

1. A `layer@id/` directory will be turned into a layer of an overlayfs mount. The `id` identifier is used to define the order of the layers: a version sort is executed, with the first entry being the bottom layer in the `overlayfs` stack, and the last entry becoming the highest layer (precisely: highest "lowerdir") in the `overlayfs` stack.
2. Similar, a `layer@id.raw` regular file will be mounted as a DDI, and the resulting mount will be turned into an overlayfs layer, following the same sorting rules.
3. An `rw` directory will be turned into a writable layer at the very top of the `overlayfs` stack. A subdirectory `data` of it will become the "upperdir", a subdirectory `work` will become the "workdir". Note that these two subdirectories do not need to be created explicitly, they are created automatically on first use should they be missing.

4. A `bind@location/` directory will be bind mounted to the mount point indicated by the `location` identifier, in read-write fashion. The location is encoded via the same escaping logic used for naming `.mount` units, i.e. slashes become dashes.
5. Similar, a `bind@location.raw` file will be mounted as a DDI, and the resulting mount bind mounted to the specified location.
6. A `robind@location/` is treated very similar to `bind@location/`, but the resulting bind mount is read-only.
7. Similar, `robind@location.raw` creates a read-only bind mount from a DDI.
8. If a `root/` subdirectory exists, it is used as root of the resulting mount hierarchy, and only the `usr/` subtree of the overlayfs mount will be bound to `usr/` in the hierarchy.

Note that each of the entry types above may be a symbolic link pointing to a directory or image file, instead a directory or image file itself.

On each listed file or subdirectory type the `systemd.v(7)` functionality may be used, for automatic selection of versioned resources.

Use the `systemd-mstack(1)` tool to process or mount `.mstack/` directories from the command line.

Examples

The following `.mstack/` consists of two read-only overlayfs layers as DDI, plus one writable directory on top. The read-only layers are symlinked:

1. `foobar.mstack/layer@0.raw` → `../base.raw`
2. `foobar.mstack/layer@1.raw` → `../app.raw`
3. `foobar.mstack/rw/`

The following `.mstack/` consists of a read-only DDI mounted to `/usr/` and writable root:

1. `waldo.mstack/layer@0.raw` → `../vendor.raw`
2. `waldo.mstack/root/`

The following `.mstack/` consists of a read-only DDI mounted as root, but a writable `/var/` mounted on top:

1. `quux.mstack/layer@0.raw` → `../myapp1.raw`
2. `quux.mstack/bind:var` → `../myapp1-var/`

See Also

`systemd(1)`, `systemd-mstack(1)`, `systemd-nspawn(1)`, `systemd.exec(5)`, `systemd.v(7)`, `systemd-vpick(1)`

□□□ #2
□ □□□ □□□ 2026-10-31 15:06:28 UTC
□ □□□ □□□ 2026-10-31 15:06:49 UTC