Rocky Enterprise Linux 9.2 Manual Pages on command 'tmpfiles.d.5'

$ man tmpfiles.d.5

TMPFILES.D(5) tmpfiles.d TMPFILES.D(5)

NAME

tmpfiles.d - Configuration for creation, deletion and cleaning of volatile and temporary files

SYNOPSIS

/etc/tmpfiles.d/*.conf

/run/tmpfiles.d/*.conf

/usr/lib/tmpfiles.d/*.conf

~/.config/user-tmpfiles.d/*.conf

$XDG_RUNTIME_DIR/user-tmpfiles.d/*.conf

~/.local/share/user-tmpfiles.d/*.conf

...

/usr/share/user-tmpfiles.d/*.conf

#Type Path Mode User Group Age Argument

f /file/to/create mode user group - content

f+ /file/to/create-or-truncate mode user group - content

w /file/to/write-to - - - - content

w+ /file/to/append-to - - - - content

d /directory/to/create-and-cleanup mode user group cleanup-age -

D /directory/to/create-and-remove mode user group cleanup-age -

e /directory/to/cleanup mode user group cleanup-age -

v /subvolume-or-directory/to/create mode user group cleanup-age -

q /subvolume-or-directory/to/create mode user group cleanup-age -

Q /subvolume-or-directory/to/create mode user group cleanup-age -

p /fifo/to/create mode user group - -

p+ /fifo/to/[re]create mode user group - -

L /symlink/to/create - - - - symlink/target/path

L+ /symlink/to/[re]create - - - - symlink/target/path

c /dev/char-device-to-create mode user group - major:minor

c+ /dev/char-device-to-[re]create mode user group - major:minor

b /dev/block-device-to-create mode user group - major:minor

b+ /dev/block-device-to-[re]create mode user group - major:minor

C /target/to/create - - - cleanup-age /source/to/copy

x /path-or-glob/to/ignore/recursively - - - cleanup-age -

X /path-or-glob/to/ignore - - - cleanup-age -

r /empty/dir/to/remove - - - - -

R /dir/to/remove/recursively - - - - -

z /path-or-glob/to/adjust/mode mode user group - -

Z /path-or-glob/to/adjust/mode/recursively mode user group - -

t /path-or-glob/to/set/xattrs - - - - xattrs

T /path-or-glob/to/set/xattrs/recursively - - - - xattrs

h /path-or-glob/to/set/attrs - - - - file attrs

H /path-or-glob/to/set/attrs/recursively - - - - file attrs

a /path-or-glob/to/set/acls - - - - POSIX ACLs

a+ /path-or-glob/to/append/acls - - - - POSIX ACLs

A /path-or-glob/to/set/acls/recursively - - - - POSIX ACLs

A+ /path-or-glob/to/append/acls/recursively - - - - POSIX ACLs

DESCRIPTION

tmpfiles.d configuration files provide a generic mechanism to define the creation of regular files, directories, pipes, and device nodes, adjustments to their access mode, ownership, attributes, quota

assignments, and contents, and finally their time-based removal. It is mostly commonly used for volatile and temporary files and directories (such as those located under /run/, /tmp/, /var/tmp/, the API file systems such as /sys/ or /proc/, as well as some other directories below /var/).

systemd-tmpfiles uses this configuration to create volatile files and directories during boot and to do periodic cleanup afterwards. See systemd-tmpfiles(5) for the description of systemd-tmpfiles-setup.service, systemd-tmpfiles-clean.service, and associated units.

System daemons frequently require private runtime directories below /run/ to store communication sockets and similar. For these, it is better to use RuntimeDirectory= in their unit files (see systemd.exec(5) for details), if the flexibility provided by tmpfiles.d is not required. The advantages are that the configuration required by the unit is centralized in one place, and that the lifetime of the directory is tied to the lifetime of the service itself. Similarly, StateDirectory=, CacheDirectory=, LogsDirectory=, and ConfigurationDirectory= should be used to create directories under /var/lib/, /var/cache/, /var/log/, and /etc/. tmpfiles.d should be used for files whose lifetime is independent of any service or requires more complicated configuration.

CONFIGURATION DIRECTORIES AND PRECEDENCE

Each configuration file shall be named in the style of package.conf or package-part.conf. The second variant should be used when it is desirable to make it easy to override just this part of configuration.

Files in /etc/tmpfiles.d override files with the same name in /usr/lib/tmpfiles.d and /run/tmpfiles.d. Files in /run/tmpfiles.d override files with the same name in /usr/lib/tmpfiles.d. Packages should install their configuration files in /usr/lib/tmpfiles.d. Files in /etc/tmpfiles.d are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. All configuration files are sorted by their filename in

lexicographic order, regardless of which of the directories they reside in. If multiple files specify the same path, the entry in the file with the lexicographically earliest name will be applied (note that lines suppressed due to the "!" are filtered before application, meaning that if an early line carries the exclamation mark and is suppressed because of that, a later line matching in path will be applied). All other conflicting entries will be logged as errors. When two lines are prefix path and suffix path of each other, then the prefix line is always created first, the suffix later (and if removal applies to the line, the order is reversed: the suffix is removed first, the prefix later). Lines that take globs are applied after those accepting no globs. If multiple operations shall be applied on the same file (such as ACL, xattr, file attribute adjustments), these are always done in the same fixed order. Except for those cases, the files/directories are processed in the order they are listed.

If the administrator wants to disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in /etc/tmpfiles.d/ bearing the same filename.

CONFIGURATION FILE FORMAT

The configuration format is one line per path, containing type, path, mode, ownership, age, and argument fields. The lines are separated by newlines, the fields by whitespace:

```
#Type Path        Mode User Group Age Argument...
d     /run/user   0755 root root  10d -
L     /tmp/foobar -    -    -     -   /dev/null
```

Fields may contain C-style escapes. With the exception of the seventh field (the "argument") all fields may be enclosed in quotes. Note that any whitespace found in the line after the beginning of the argument field will be considered part of the argument field. To begin the argument field with a whitespace character, use C-style escapes (e.g. "\x20").

Type

The type consists of a single letter and optionally one or emore

modifier characters: a plus sign ("+"), exclamation mark ("!"), minus sign ("-"), equals sign ("="), tilde character ("~") and/or caret ("^").

The following line types are understood:

f, f+

f will create a file if it does not exist yet. If the argument parameter is given and the file did not exist yet, it will be written to the file. f+ will create or truncate the file. If the argument parameter is given, it will be written to the file. Does not follow symlinks.

w, w+

Write the argument parameter to a file, if the file exists. If suffixed with +, the line will be appended to the file. If your configuration writes multiple lines to the same file, use w+. Lines of this type accept shell-style globs in place of normal path names. The argument parameter will be written without a trailing newline. C-style backslash escapes are interpreted. Follows symlinks.

d

Create a directory. The mode and ownership will be adjusted if specified. Contents of this directory are subject to time based cleanup if the age argument is specified.

D

Similar to d, but in addition the contents of the directory will be removed when --remove is used.

e

Adjust the mode and ownership of existing directories and remove their contents based on age. Lines of this type accept shell-style globs in place of normal path names. Contents of the directories are subject to time based cleanup if the age argument is specified. If the age argument is "0", contents will be unconditionally deleted every time systemd-tmpfiles --clean is run.

For this entry to be useful, at least one of the mode, user, group,

or age arguments must be specified, since otherwise this entry has no effect. As an exception, an entry with no effect may be useful when combined with !, see the examples.

v

Create a subvolume if the path does not exist yet, the file system supports subvolumes (btrfs), and the system itself is installed into a subvolume (specifically: the root directory / is itself a subvolume). Otherwise, create a normal directory, in the same way as d.

A subvolume created with this line type is not assigned to any higher-level quota group. For that, use q or Q, which allow creating simple quota group hierarchies, see below.

q

Create a subvolume or directory the same as v, but assign the subvolume to the same higher-level quota groups as the parent. This ensures that higher-level limits and accounting applied to the parent subvolume also include the specified subvolume. On non-btrfs file systems, this line type is identical to d.

If the subvolume already exists, no change to the quota hierarchy is made, regardless of whether the subvolume is already attached to a quota group or not. Also see Q below. See btrfs-qgroup(8) for details about the btrfs quota group concept.

Q

Create the subvolume or directory the same as v, but assign the new subvolume to a new leaf quota group. Instead of copying the higher-level quota group assignments from the parent as is done with q, the lowest quota group of the parent subvolume is determined that is not the leaf quota group. Then, an "intermediary" quota group is inserted that is one level below this level, and shares the same ID part as the specified subvolume. If no higher-level quota group exists for the parent subvolume, a new quota group at level 255 sharing the same ID as the specified subvolume is inserted instead. This new intermediary quota group is

then assigned to the parent subvolume's higher-level quota groups, and the specified subvolume's leaf quota group is assigned to it. Effectively, this has a similar effect as q, however introduces a new higher-level quota group for the specified subvolume that may be used to enforce limits and accounting to the specified subvolume and children subvolume created within it. Thus, by creating subvolumes only via q and Q, a concept of "subtree quotas" is implemented. Each subvolume for which Q is set will get a "subtree" quota group created, and all child subvolumes created within it will be assigned to it. Each subvolume for which q is set will not get such a "subtree" quota group, but it is ensured that they are added to the same "subtree" quota group as their immediate parents. It is recommended to use Q for subvolumes that typically contain further subvolumes, and where it is desirable to have accounting and quota limits on all child subvolumes together. Examples for Q are typically /home/ or /var/lib/machines/. In contrast, q should be used for subvolumes that either usually do not include further subvolumes or where no accounting and quota limits are needed that apply to all child subvolumes together. Examples for q are typically /var/ or /var/tmp/.

As with q, Q has no effect on the quota group hierarchy if the subvolume already exists, regardless of whether the subvolume already belong to a quota group or not.

p, p+

Create a named pipe (FIFO) if it does not exist yet. If suffixed with + and a file already exists where the pipe is to be created, it will be removed and be replaced by the pipe.

L, L+

Create a symlink if it does not exist yet. If suffixed with + and a file or directory already exists where the symlink is to be created, it will be removed and be replaced by the symlink. If the argument is omitted, symlinks to files with the same name residing in the directory /usr/share/factory/ are created. Note that

permissions and ownership on symlinks are ignored.

c, c+

Create a character device node if it does not exist yet. If suffixed with + and a file already exists where the device node is to be created, it will be removed and be replaced by the device node. It is recommended to suffix this entry with an exclamation mark to only create static device nodes at boot, as udev will not manage static device nodes that are created at runtime.

b, b+

Create a block device node if it does not exist yet. If suffixed with + and a file already exists where the device node is to be created, it will be removed and be replaced by the device node. It is recommended to suffix this entry with an exclamation mark to only create static device nodes at boot, as udev will not manage static device nodes that are created at runtime.

C

Recursively copy a file or directory, if the destination files or directories do not exist yet or the destination directory is empty. Note that this command will not descend into subdirectories if the destination directory already exists and is not empty. Instead, the entire copy operation is skipped. If the argument is omitted, files from the source directory /usr/share/factory/ with the same name are copied. Does not follow symlinks. Contents of the directories are subject to time based cleanup if the age argument is specified.

x

Ignore a path during cleaning. Use this type to exclude paths from clean-up as controlled with the Age parameter. Note that lines of this type do not influence the effect of r or R lines. Lines of this type accept shell-style globs in place of normal path names.

X

Ignore a path during cleaning. Use this type to exclude paths from clean-up as controlled with the Age parameter. Unlike x, this parameter will not exclude the content if path is a directory, but

only directory itself. Note that lines of this type do not influence the effect of r or R lines. Lines of this type accept shell-style globs in place of normal path names.

r

Remove a file or directory if it exists. This may not be used to remove non-empty directories, use R for that. Lines of this type accept shell-style globs in place of normal path names. Does not follow symlinks.

R

Recursively remove a path and all its subdirectories (if it is a directory). Lines of this type accept shell-style globs in place of normal path names. Does not follow symlinks.

z

Adjust the access mode, user and group ownership, and restore the SELinux security context of a file or directory, if it exists. Lines of this type accept shell-style globs in place of normal path names. Does not follow symlinks.

Z

Recursively set the access mode, user and group ownership, and restore the SELinux security context of a file or directory if it exists, as well as of its subdirectories and the files contained therein (if applicable). Lines of this type accept shell-style globs in place of normal path names. Does not follow symlinks.

t

Set extended attributes, see attr(5) for details. The argument field should take one or more assignment expressions in the form namespace.attribute=value, for examples see below. Lines of this type accept shell-style globs in place of normal path names. This can be useful for setting SMACK labels. Does not follow symlinks. Please note that extended attributes settable with this line type are a different concept from the Linux file attributes settable with h/H, see below.

T

Same as t, but operates recursively.

h

Set Linux file/directory attributes. Lines of this type accept shell-style globs in place of normal path names.

The format of the argument field is [+-=][aAcCdDeijPsStTu]. The prefix + (the default one) causes the attributes to be added; - causes the attributes to be removed; = causes the attributes to be set exactly as the following letters. The letters "aAcCdDeijPsStTu" select the new attributes for the files, see chattr(1) for further information.

Passing only = as argument resets all the file attributes listed above. It has to be pointed out that the = prefix limits itself to the attributes corresponding to the letters listed here. All other attributes will be left untouched. Does not follow symlinks.

Please note that the Linux file attributes settable with this line type are a different concept from the extended attributes settable with t/T, see above.

H

Sames as h, but operates recursively.

a, a+

Set POSIX ACLs (access control lists), see acl(5). If suffixed with +, the specified entries will be added to the existing set. systemd-tmpfiles will automatically add the required base entries for user and group based on the access mode of the file, unless base entries already exist or are explicitly specified. The mask will be added if not specified explicitly or already present. Lines of this type accept shell-style globs in place of normal path names. This can be useful for allowing additional access to certain files. Does not follow symlinks.

A, A+

Same as a and a+, but recursive. Does not follow symlinks.

Type Modifiers

If the exclamation mark ("!") is used, this line is only safe to

execute during boot, and can break a running system. Lines without the exclamation mark are presumed to be safe to execute at any time, e.g. on package upgrades. systemd-tmpfiles will take lines with an exclamation mark only into consideration, if the --boot option is given.

For example:

```
# Make sure these are created by default so that nobody else can
d /tmp/.X11-unix 1777 root root 10d
# Unlink the X11 lock files
r! /tmp/.X[0-9]*-lock
```

The second line in contrast to the first one would break a running system, and will only be executed with --boot.

If the minus sign ("-") is used, this line failing to run successfully during create (and only create) will not cause the execution of systemd-tmpfiles to return an error.

For example:

```
# Modify sysfs but don't fail if we are in a container with a read-only /proc
w- /proc/sys/vm/swappiness - - - - 10
```

If the equals sign ("=") is used, the file types of existing objects in the specified path are checked, and removed if they do not match. This includes any implicitly created parent directories (which can be either directories or directory symlinks). For example, if there is a FIFO in place of one of the parent path components it will be replaced with a directory.

If the tilde character ("~") is used, the argument (i.e. 6th) column is Base64 decoded[1] before use. This modifier is only supported on line types that can write file contents, i.e. f, f+, w, +. This is useful for writing arbitrary binary data (including newlines and NUL bytes) to files. Note that if this switch is used, the argument is not subject to specifier expansion, neither before nor after Base64 decoding.

If the caret character ("^") is used, the argument (i.e. 6th) column takes a service credential name to read the argument data from. See System and Service Credentials[2] for details about the credentials

concept. This modifier is only supported on line types that can write file contents, i.e. f, f+, w, w+. This is useful for writing arbitrary files with contents sourced from elsewhere, including from VM or container managers further up. If the specified credential is not set for the systemd-tmpfiles service, the line is silently skipped. If "^" and "~" are combined Base64 decoding is applied to the credential contents.

Note that for all line types that result in creation of any kind of file node (i.e. f/F, d/D/v/q/Q, p, L, c/b and C) leading directories are implicitly created if needed, owned by root with an access mode of 0755. In order to create them with different modes or ownership make sure to add appropriate d lines.

Path

The file system path specification supports simple specifier expansion, see below. The path (after expansion) must be absolute.

Mode

The file access mode to use when creating this file or directory. If omitted or when set to "-", the default is used: 0755 for directories, 0644 for all other file objects. For z, Z lines, if omitted or when set to "-", the file access mode will not be modified. This parameter is ignored for x, r, R, L, t, and a lines.

Optionally, if prefixed with "~", the access mode is masked based on the already set access bits for existing file or directories: if the existing file has all executable bits unset, all executable bits are removed from the new access mode, too. Similarly, if all read bits are removed from the old access mode, they will be removed from the new access mode too, and if all write bits are removed, they will be removed from the new access mode too. In addition, the sticky/SUID/SGID bit is removed unless applied to a directory. This functionality is particularly useful in conjunction with Z.

Optionally, if prefixed with ":", the configured access mode is only used when creating new inodes. If the inode the line refers to already exists, its access mode is left in place unmodified.

User, Group

The user and group to use for this file or directory. This may either be a numeric ID or a user/group name. If omitted or when set to "-", the user and group of the user who invokes systemd-tmpfiles is used. For z and Z lines, when omitted or when set to "-", the file ownership will not be modified. These parameters are ignored for x, r, R, L, t, and a lines.

This field should generally only reference system users/groups, i.e. users/groups that are guaranteed to be resolvable during early boot. If this field references users/groups that only become resolveable during later boot (i.e. after NIS, LDAP or a similar networked directory service become available), execution of the operations declared by the line will likely fail. Also see Notes on Resolvability of User and Group Names[3] for more information on requirements on system user/group definitions.

Optionally, if prefixed with ":", the configured user/group information is only used when creating new inodes. If the inode the line refers to already exists, its user/group is left in place unmodified.

Age

The date field, when set, is used to decide what files to delete when cleaning. If a file or directory is older than the current time minus the age field, it is deleted. The field format is a series of integers each followed by one of the following suffixes for the respective time units: s, m or min, h, d, w, ms, and us, meaning seconds, minutes, hours, days, weeks, milliseconds, and microseconds, respectively. Full names of the time units can be used too.

If multiple integers and units are specified, the time values are summed. If an integer is given without a unit, s is assumed.

When the age is set to zero, the files are cleaned unconditionally.

The age field only applies to lines starting with d, D, e, v, q, Q, C, x and X. If omitted or set to "-", no automatic clean-up is done.

If the age field starts with a tilde character "~", clean-up is only applied to files and directories one level inside the directory

specified, but not the files and directories immediately inside it.

The age of a file system entry is determined from its last modification timestamp (mtime), its last access timestamp (atime), and (except for directories) its last status change timestamp (ctime). By default, any of these three (or two) values will prevent cleanup if it is more recent than the current time minus the age field. To restrict the deletion based on particular type of file timestamps, the age-by argument can be used.

The age-by argument overrides the timestamp types to be used for the age check. It can be specified by prefixing the age argument with a sequence of characters to specify the timestamp types and a colon (":"): "age-by...:cleanup-age". The argument can consist of a (A for directories), b (B for directories), c (C for directories), or m (M for directories). Those respectively indicate access, creation, last status change, and last modification time of a file system entry. The lower-case letter signifies that the given timestamp type should be considered for files, while the upper-case letter signifies that the given timestamp type should be considered for directories. See statx(2) file timestamp fields for more details about timestamp types.

If not specified, the age-by field defaults to abcmABM, i.e. by default all file timestamps are taken into consideration, with the exception of the last status change timestamp (ctime) for directories. This is because the aging logic itself will alter the ctime whenever it deletes a file inside it. To ensure that running the aging logic does not feed back into the next iteration of itself, ctime for directories is ignored by default.

For example:

```
# Files created and modified, and directories accessed more than
# an hour ago in "/tmp/foo/bar", are subject to time-based cleanup.
d /tmp/foo/bar - - - - bmA:1h -
```

Note that while the aging algorithm is run a 'shared' BSD file lock (see flock(2)) is taken on each directory the algorithm descends into (and each directory below that, and so on). If the aging algorithm

finds a lock is already taken on some directory, it (and everything below it) is skipped. Applications may use this to temporarily exclude certain directory subtrees from the aging algorithm: the applications can take a BSD file lock themselves, and as long as they keep it aging of the directory and everything below it is disabled.

Argument

For L lines determines the destination path of the symlink. For c and b, determines the major/minor of the device node, with major and minor formatted as integers, separated by ":", e.g. "1:3". For f, F, and w, the argument may be used to specify a short string that is written to the file, suffixed by a newline. For C, specifies the source file or directory. For t and T, determines extended attributes to be set. For a and A, determines ACL attributes to be set. For h and H, determines the file attributes to set. Ignored for all other lines.

This field can contain specifiers, see below.

SPECIFIERS

Specifiers can be used in the "path" and "argument" fields. An unknown or unresolvable specifier is treated as invalid configuration. The following expansions are understood:

Table 1. Specifiers available

Specifier	Meaning	Details
"%a"	Architecture	A short string identifying the architecture of the local system. A string such as x86, x86-64 or arm64. See the architectures defined for ConditionArchitecture=

? ? ? in systemd.unit(5) ?

? ? ? for a full list. ?

??

?"%A" ? Operating system ? The operating system ?

? ? image version ? image version ?

? ? ? identifier of the ?

? ? ? running system, as ?

? ? ? read from the ?

? ? ? IMAGE_VERSION= field ?

? ? ? of /etc/os-release. If ?

? ? ? not set, resolves to ?

? ? ? an empty string. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%b" ? Boot ID ? The boot ID of the ?

? ? ? running system, ?

? ? ? formatted as string. ?

? ? ? See random(4) for more ?

? ? ? information. ?

??

?"%B" ? Operating system ? The operating system ?

? ? build ID ? build identifier of ?

? ? ? the running system, as ?

? ? ? read from the ?

? ? ? BUILD_ID= field of ?

? ? ? /etc/os-release. If ?

? ? ? not set, resolves to ?

? ? ? an empty string. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%C" ? System or user ? In --user mode, this ?

? ? cache directory ? is the same as ?

? ? ? $XDG_CACHE_HOME, and ?

? ? ? /var/cache otherwise. ?

??

?"%g" ? User group ? This is the name of ?

? ? ? the group running the ?

? ? ? command. In case of ?

? ? ? the system instance ?

? ? ? this resolves to ?

? ? ? "root". ?

??

?"%G" ? User GID ? This is the numeric ?

? ? ? GID of the group ?

? ? ? running the command. ?

? ? ? In case of the system ?

? ? ? instance this resolves ?

? ? ? to 0. ?

??

?"%h" ? User home directory ? This is the home ?

? ? ? directory of the user ?

? ? ? running the command. ?

? ? ? In case of the system ?

? ? ? instance this resolves ?

? ? ? to "/root". ?

??

?"%H" ? Host name ? The hostname of the ?

? ? ? running system. ?

??

?"%l" ? Short host name ? The hostname of the ?

? ? ? running system, ?

? ? ? truncated at the first ?

? ? ? dot to remove any ?

? ? ? domain component. ?

??

?"%L" ? System or user log ? In --user mode, this ?

? ? directory ? is the same as ?

? ? ? $XDG_CONFIG_HOME with ?

? ? ? /log appended, and ?

? ? ? /var/log otherwise. ?

??

?"%m" ? Machine ID ? The machine ID of the ?

? ? ? running system, ?

? ? ? formatted as string. ?

? ? ? See machine-id(5) for ?

? ? ? more information. ?

??

?"%M" ? Operating system ? The operating system ?

? ? image identifier ? image identifier of ?

? ? ? the running system, as ?

? ? ? read from the ?

? ? ? IMAGE_ID= field of ?

? ? ? /etc/os-release. If ?

? ? ? not set, resolves to ?

? ? ? an empty string. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%o" ? Operating system ID ? The operating system ?

? ? ? identifier of the ?

? ? ? running system, as ?

? ? ? read from the ID= ?

? ? ? field of ?

? ? ? /etc/os-release. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%S" ? System or user ? In --user mode, this ?

? ? state directory ? is the same as ?

? ? ? $XDG_CONFIG_HOME, and ?

? ? ? /var/lib otherwise. ?

??

?"%t" ? System or user ? In --user mode, this ?

? ? runtime directory ? is the same ?

? ? ? $XDG_RUNTIME_DIR, and ?

? ? ? /run/ otherwise. ?

??

?"%T" ? Directory for ? This is either /tmp or ?

? ? temporary files ? the path "$TMPDIR", ?

? ? ? "$TEMP" or "$TMP" are ?

? ? ? set to. (Note that the ?

? ? ? directory may be ?

? ? ? specified without a ?

? ? ? trailing slash.) ?

??

?"%u" ? User name ? This is the name of ?

? ? ? the user running the ?

? ? ? command. In case of ?

? ? ? the system instance ?

? ? ? this resolves to ?

? ? ? "root". ?

??

?"%U" ? User UID ? This is the numeric ?

? ? ? UID of the user ?

? ? ? running the command. ?

? ? ? In case of the system ?

? ? ? instance this resolves ?

? ? ? to 0. ?

??

?"%v" ? Kernel release ? Identical to uname -r ?

? ? ? output. ?

??

?"%V" ? Directory for ? This is either ?

? ? larger and ? /var/tmp or the path ?

? ? persistent ? "$TMPDIR", "$TEMP" or ?

? ? temporary files ? "$TMP" are set to. ?

? ? ? (Note that the ?

? ? ? directory may be ?

? ? ? specified without a ?

? ? ? trailing slash.) ?

??

?"%w" ? Operating system ? The operating system ?

? ? version ID ? version identifier of ?

? ? ? the running system, as ?

? ? ? read from the ?

? ? ? VERSION_ID= field of ?

? ? ? /etc/os-release. If ?

? ? ? not set, resolves to ?

? ? ? an empty string. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%W" ? Operating system ? The operating system ?

? ? variant ID ? variant identifier of ?

? ? ? the running system, as ?

? ? ? read from the ?

? ? ? VARIANT_ID= field of ?

? ? ? /etc/os-release. If ?

? ? ? not set, resolves to ?

? ? ? an empty string. See ?

? ? ? os-release(5) for more ?

? ? ? information. ?

??

?"%%" ? Single percent sign ? Use "%%" in place of ?

? ? ? "%" to specify a ?

? ? ? single percent sign. ?

??

EXAMPLES

Example 1. Create directories with specific mode and ownership

screen(1), needs two directories created at boot with specific modes

and ownership:

```
# /usr/lib/tmpfiles.d/screen.conf
d /run/screens  1777 root screen 10d
d /run/uscreens 0755 root screen 10d12h
```

Contents of /run/screens and /run/uscreens will be cleaned up after 10

and 10? days, respectively.

Example 2. Create a directory with a SMACK attribute

```
D /run/cups - - - -
t /run/cups - - - - security.SMACK64=printing user.attr-with-spaces="foo bar"
```

The directory will be owned by root and have default mode. Its contents

are not subject to time based cleanup, but will be obliterated when

systemd-tmpfiles --remove runs.

Example 3. Create a directory and prevent its contents from cleanup

abrt(1), needs a directory created at boot with specific mode and

ownership and its content should be preserved from the automatic

cleanup applied to the contents of /var/tmp:

```
# /usr/lib/tmpfiles.d/tmp.conf
d /var/tmp 1777 root root 30d
# /usr/lib/tmpfiles.d/abrt.conf
d /var/tmp/abrt 0755 abrt abrt -
```

Example 4. Apply clean up during boot and based on time

```
# /usr/lib/tmpfiles.d/dnf.conf
r! /var/cache/dnf/*/*/download_lock.pid
r! /var/cache/dnf/*/*/metadata_lock.pid
r! /var/lib/dnf/rpmdb_lock.pid
e  /var/cache/dnf/ - - - 30d
```

The lock files will be removed during boot. Any files and directories in /var/cache/dnf/ will be removed after they have not been accessed in 30 days.

Example 5. Empty the contents of a cache directory on boot

```
# /usr/lib/tmpfiles.d/krb5rcache.conf
e! /var/cache/krb5rcache - - - 0
```

Any files and subdirectories in /var/cache/krb5rcache/ will be removed on boot. The directory will not be created.

Example 6. Provision SSH public key access for root user via Credentials in QEMU

```
-smbios type=11,value=io.systemd.credential.binary:tmpfiles.extra=$(echo "f~ /root/.ssh/authorized_keys 700 root root - $(ssh-add -L | base64 -w 0)" | base64 -w 0)
```

By passing this line to QEMU, the public key of the current user will be encoded in base64, added to a tmpfiles.d line that tells systemd-tmpfiles to decode it into /root/.ssh/authorized_keys, encode that line itself in base64 and pass it as a Credential that will be picked up by systemd from SMBIOS on boot.

/RUN/ AND /VAR/RUN/

/var/run/ is a deprecated symlink to /run/, and applications should use the latter. systemd-tmpfiles will warn if /var/run/ is used.

SEE ALSO

systemd(1), systemd-tmpfiles(8), systemd-delta(1), systemd.exec(5), attr(5), getfattr(1), setfattr(1), setfacl(1), getfacl(1), chattr(1), btrfs-subvolume(8), btrfs-qgroup(8)

NOTES

1. Base64 decoded
   https://www.rfc-editor.org/rfc/rfc4648.html
2. System and Service Credentials
   https://systemd.io/CREDENTIALS
3. Notes on Resolvability of User and Group Names
   https://systemd.io/UIDS-GIDS/#notes-on-resolvability-of-user-and-group-names