MyWebUniversity
python
Red Hat Enterprise Linux Manual Pages
FPDF Library PDF generator
FPDF
Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'file-hierarchy.7' command

$ man file-hierarchy.7

FILE-HIERARCHY(7) file-hierarchy FILE-HIERARCHY(7)

NAME

file-hierarchy - File system hierarchy overview

DESCRIPTION

Operating systems using the systemd(1) system and service manager are organized based on a file system hierarchy inspired by UNIX, more specifically the hierarchy described in the File System Hierarchy[1] specification and hier(7), with various extensions, partially documented in the XDG Base Directory Specification[2] and XDG User Directories[3]. This manual page describes a more generalized, though minimal and modernized subset of these specifications that defines more strictly the suggestions and restrictions systemd makes on the file system hierarchy.

Many of the paths described here can be queried with the systemd-path(1) tool.

GENERAL STRUCTURE

/

The file system root. Usually writable, but this is not required. Possibly a temporary file system ("tmpfs"). Not shared with other hosts (unless read-only).

/boot/

The boot partition used for bringing up the system. On EFI systems, this is possibly the EFI System Partition (ESP), also see systemd-gpt-auto-generator(8). This directory is usually strictly local to the host, and should be considered read-only, except when a new kernel or boot loader is installed. This directory only exists on systems that run on physical or emulated hardware that requires boot loaders.

/efi/

If the boot partition /boot/ is maintained separately from the EFI System Partition (ESP), the latter is mounted here. Tools that need to operate on the EFI system partition should look for it at this mount point first, and fall back to /boot/ ? if the former doesn't qualify (for example if it is not a mount point or does not have the correct file system type MSDOS_SUPER_MAGIC).

/etc/

System-specific configuration. This directory may or may not be read-only. Frequently, this directory is pre-populated with vendor-supplied configuration files, but applications should not make assumptions about this directory being fully populated or populated at all, and should fall back to defaults if configuration is missing.

/home/

The location for normal user's home directories. Possibly shared with other systems, and never read-only. This directory should only be used for normal users, never for system users. This directory and possibly the directories contained within it might only become available or writable in late boot or even only after user authentication. This directory might be placed on limited-functionality network file systems, hence applications should not assume the full set of file API is available on this directory. Applications should generally not reference this directory directly, but via the per-user $HOME environment variable, or via the home directory field of the user database.

/root/

The home directory of the root user. The root user's home directory

is located outside of /home/ in order to make sure the root user
may log in even without /home/ being available and mounted.

/srv/

The place to store general server payload, managed by the administrator. No restrictions are made how this directory is organized internally. Generally writable, and possibly shared among systems. This directory might become available or writable only very late during boot.

/tmp/

The place for small temporary files. This directory is usually mounted as a "tmpfs" instance, and should hence not be used for larger files. (Use /var/tmp/ for larger files.) This directory is usually flushed at boot-up. Also, files that are not accessed within a certain time may be automatically deleted.

If applications find the environment variable $TMPDIR set, they should use the directory specified in it instead of /tmp/ (see environ(7) and IEEE Std 1003.1[4] for details).

Since /tmp/ is accessible to other users of the system, it is essential that files and subdirectories under this directory are only created with mkstemp(3), mkdtemp(3), and similar calls. For more details, see Using /tmp/ and /var/tmp/ Safely[5].

RUNTIME DATA

/run/

A "tmpfs" file system for system packages to place runtime data, socket files, and similar. This directory is flushed on boot, and generally writable for privileged programs only. Always writable.

/run/log/

Runtime system logs. System components may place private logs in this directory. Always writable, even when /var/log/ might not be accessible yet.

/run/user/

Contains per-user runtime directories, each usually individually mounted "tmpfs" instances. Always writable, flushed at each reboot and when the user logs out. User code should not reference this directory directly, but via the $XDG_RUNTIME_DIR environment variable, as documented in the XDG Base Directory Specification[2].

VENDOR-SUPPLIED OPERATING SYSTEM RESOURCES

/usr/

Vendor-supplied operating system resources. Usually read-only, but this is not required. Possibly shared between multiple hosts. This directory should not be modified by the administrator, except when installing or removing vendor-supplied packages.

/usr/bin/

Binaries and executables for user commands that shall appear in the $PATH search path. It is recommended not to place binaries in this directory that are not useful for invocation from a shell (such as daemon binaries); these should be placed in a subdirectory of /usr/lib/ instead.

/usr/include/

C and C++ API header files of system libraries.

/usr/lib/

Static, private vendor data that is compatible with all architectures (though not necessarily architecture-independent). Note that this includes internal executables or other binaries that are not regularly invoked from a shell. Such binaries may be for any architecture supported by the system. Do not place public libraries in this directory, use $libdir (see below), instead.

/usr/lib/arch-id/

Location for placing dynamic libraries into, also called $libdir. The architecture identifier to use is defined on Multiarch Architecture Specifiers (Tuples)[6] list. Legacy locations of $libdir are /usr/lib/, /usr/lib64/. This directory should not be used for package-specific data, unless this data is architecture-dependent, too. To query $libdir for the primary architecture of the system, invoke:

```
# systemd-path system-library-arch
```

/usr/share/

Resources shared between multiple packages, such as documentation, man pages, time zone information, fonts and other resources. Usually, the precise location and format of files stored below this directory is subject to specifications that ensure interoperability.

/usr/share/doc/

Documentation for the operating system or system packages.

/usr/share/factory/etc/

Repository for vendor-supplied default configuration files. This directory should be populated with pristine vendor versions of all configuration files that may be placed in /etc/. This is useful to compare the local configuration of a system with vendor defaults and to populate the local configuration with defaults.

/usr/share/factory/var/

Similar to /usr/share/factory/etc/, but for vendor versions of files in the variable, persistent data directory /var/.

PERSISTENT VARIABLE SYSTEM DATA

/var/

Persistent, variable system data. Writable during normal system operation. This directory might be pre-populated with vendor-supplied data, but applications should be able to reconstruct necessary files and directories in this subhierarchy should they be missing, as the system might start up without this directory being populated. Persistency is recommended, but optional, to support ephemeral systems. This directory might become available or writable only very late during boot. Components that are required to operate during early boot hence shall not unconditionally rely on this directory.

/var/cache/

Persistent system cache data. System components may place non-essential data in this directory. Flushing this directory should have no effect on operation of programs, except for increased runtimes necessary to rebuild these caches.

/var/lib/

Persistent system data. System components may place private data in this directory.

/var/log/

Persistent system logs. System components may place private logs in this directory, though it is recommended to do most logging via the syslog(3) and sd_journal_print(3) calls.

/var/spool/

Persistent system spool data, such as printer or mail queues.

/var/tmp/

The place for larger and persistent temporary files. In contrast to /tmp/, this directory is usually mounted from a persistent physical file system and can thus accept larger files. (Use /tmp/ for small ephemeral files.) This directory is generally not flushed at boot-up, but time-based cleanup of files that have not been accessed for a certain time is applied.

If applications find the environment variable $TMPDIR set, they should use the directory specified in it instead of /var/tmp/ (see environ(7) for details).

The same security restrictions as with /tmp/ apply: mkstemp(3), mkdtemp(3), and similar calls should be used. For further details about this directory, see Using /tmp/ and /var/tmp/ Safely[5].

VIRTUAL KERNEL AND API FILE SYSTEMS

/dev/

The root directory for device nodes. Usually, this directory is mounted as a "devtmpfs" instance, but might be of a different type in sandboxed/containerized setups. This directory is managed jointly by the kernel and systemd-udevd(8), and should not be written to by other components. A number of special purpose virtual file systems might be mounted below this directory.

/dev/shm/

Place for POSIX shared memory segments, as created via shm_open(3).

This directory is flushed on boot, and is a "tmpfs" file system.
Since all users have write access to this directory, special care should be taken to avoid name clashes and vulnerabilities. For normal users, shared memory segments in this directory are usually deleted when the user logs out. Usually, it is a better idea to use memory mapped files in /run/ (for system programs) or $XDG_RUNTIME_DIR (for user programs) instead of POSIX shared memory segments, since these directories are not world-writable and hence not vulnerable to security-sensitive name clashes.

/proc/

A virtual kernel file system exposing the process list and other functionality. This file system is mostly an API to interface with the kernel and not a place where normal files may be stored. For details, see proc(5). A number of special purpose virtual file systems might be mounted below this directory.

/proc/sys/

A hierarchy below /proc/ that exposes a number of kernel tunables. The primary way to configure the settings in this API file tree is via sysctl.d(5) files. In sandboxed/containerized setups, this directory is generally mounted read-only.

/sys/

A virtual kernel file system exposing discovered devices and other functionality. This file system is mostly an API to interface with the kernel and not a place where normal files may be stored. In sandboxed/containerized setups, this directory is generally mounted read-only. A number of special purpose virtual file systems might be mounted below this directory.

/sys/fs/cgroup/

A virtual kernel file system exposing process control groups (cgroups). This file system is an API to interface with the kernel and not a place where normal files may be stored. On current systems running in the default "unified" mode, this directory serves as the mount point for the "cgroup2" filesystem, which provides a unified cgroup hierarchy for all resource controllers. On systems with non-default configurations, this directory may instead be a tmpfs filesystem containing mount points for various "cgroup" (v1) resource controllers; in such configurations, if "cgroup2" is mounted it will be mounted on /sys/fs/cgroup/unified/, but cgroup2 will not have resource controllers attached. In sandboxed/containerized setups, this directory may either not exist or may include a subset of functionality.

COMPATIBILITY SYMLINKS

/bin/, /sbin/, /usr/sbin/

These compatibility symlinks point to /usr/bin/, ensuring that scripts and binaries referencing these legacy paths correctly find their binaries.

/lib/

This compatibility symlink points to /usr/lib/, ensuring that programs referencing this legacy path correctly find their resources.

/lib64/

On some architecture ABIs, this compatibility symlink points to $libdir, ensuring that binaries referencing this legacy path correctly find their dynamic loader. This symlink only exists on architectures whose ABI places the dynamic loader in this path.

/var/run/

This compatibility symlink points to /run/, ensuring that programs referencing this legacy path correctly find their runtime data.

HOME DIRECTORY

User applications may want to place files and directories in the user's home directory. They should follow the following basic structure. Note that some of these directories are also standardized (though more weakly) by the XDG Base Directory Specification[2]. Additional locations for high-level user resources are defined by xdg-user-dirs[3].

~/.cache/

Persistent user cache data. User programs may place non-essential data in this directory. Flushing this directory should have no effect on operation of programs, except for increased runtimes necessary to rebuild these caches. If an application finds $XDG_CACHE_HOME set, it should use the directory specified in it instead of this directory.

~/.config/

Application configuration and state. When a new user is created, this directory will be empty or not exist at all. Applications should fall back to defaults should their configuration or state in this directory be missing. If an application finds $XDG_CONFIG_HOME set, it should use the directory specified in it instead of this directory.

~/.local/bin/

Executables that shall appear in the user's $PATH search path. It is recommended not to place executables in this directory that are not useful for invocation from a shell; these should be placed in a subdirectory of ~/.local/lib/ instead. Care should be taken when placing architecture-dependent binaries in this place, which might be problematic if the home directory is shared between multiple hosts with different architectures.

~/.local/lib/

Static, private vendor data that is compatible with all architectures.

~/.local/lib/arch-id/

Location for placing public dynamic libraries. The architecture identifier to use is defined on Multiarch Architecture Specifiers (Tuples)[6] list.

~/.local/share/

Resources shared between multiple packages, such as fonts or artwork. Usually, the precise location and format of files stored below this directory is subject to specifications that ensure interoperability. If an application finds $XDG_DATA_HOME set, it should use the directory specified in it instead of this directory.

WRITE ACCESS

Unprivileged Write Access

Unprivileged processes generally lack write access to most of the hierarchy.

The exceptions for normal users are /tmp/, /var/tmp/, /dev/shm/, as well as the home directory $HOME (usually found below /home/) and the runtime directory $XDG_RUNTIME_DIR (found below /run/user/) of the user, which are all writable.

For unprivileged system processes, only /tmp/, /var/tmp/ and /dev/shm/ are writable. If an unprivileged system process needs a private writable directory in /var/ or /run/, it is recommended to either create it before dropping privileges in the daemon code, to create it via tmpfiles.d(5) fragments during boot, or via the StateDirectory= and RuntimeDirectory= directives of service units (see systemd.unit(5) for details).

/tmp/, /var/tmp/ and /dev/shm/ should be mounted nosuid and nodev, which means that set-user-id mode and character or block special devices are not interpreted on those file systems. In general it is not possible to mount them noexec, because various programs use those directories for dynamically generated or optimized code, and with that flag those use cases would break. Using this flag is OK on special-purpose installations or systems where all software that may be installed is known and doesn't require such functionality. See the discussion of nosuid/nodev/noexec in mount(8) and PROT_EXEC in mmap(2).

Lack of Write Access on Read-Only Systems and during System Recovery

As noted above, some systems operate with the /usr and /etc hierarchies mounted read-only, possibly only allowing write access during package upgrades. Other part of the hierarchy are generally mounted read-write (in particular /var and /var/tmp), but may be read-only when the kernel remounts the file system read-only in response to errors, or when the system is booted read-only for recovery purposes. To the extent reasonable, applications should be prepared to execute without write access, so that for example, failure to save non-essential data to /var/cache/ or failure to create a custom log file under /var/log does not prevent the application from running.

The /run/ directory is available since the earliest boot and is always writable. It should be used for any runtime data and sockets, so that write access to e.g. /etc or /var is not needed.

NODE TYPES

Unix file systems support different types of file nodes, including regular files, directories, symlinks, character and block device nodes, sockets and FIFOs.

It is strongly recommended that /dev/ is the only location below which device nodes shall be placed. Similarly, /run/ shall be the only location to place sockets and FIFOs. Regular files, directories and symlinks may be used in all directories.

SYSTEM PACKAGES

Developers of system packages should follow strict rules when placing their files in the file system. The following table lists recommended locations for specific types of files supplied by the vendor.

Table 1. System package vendor files locations

Directory	Purpose
/usr/bin/	Package executables that shall appear in the $PATH executable search path, compiled for any of the supported architectures compatible with the operating system. It is not recommended to place internal binaries or binaries that are not commonly invoked from the

```
?                   ? shell in this directory,   ?
?                   ? such as daemon binaries.   ?
?                   ? As this directory is       ?
?                   ? shared with most other     ?
?                   ? packages of the system,    ?
?                   ? special care should be     ?
?                   ? taken to pick unique names ?
?                   ? for files placed here,     ?
?                   ? that are unlikely to clash ?
?                   ? with other package's       ?
?                   ? files.                     ?
??????????????????????????????????????????????????
?/usr/lib/arch-id/      ? Public shared libraries of ?
?                   ? the package. As above, be  ?
?                   ? careful with using too     ?
?                   ? generic names, and pick    ?
?                   ? unique names for your      ?
?                   ? libraries to place here to ?
?                   ? avoid name clashes.        ?
??????????????????????????????????????????????????
?/usr/lib/package/      ? Private static vendor      ?
?                   ? resources of the package,  ?
?                   ? including private binaries ?
?                   ? and libraries, or any      ?
?                   ? other kind of read-only    ?
?                   ? vendor data.               ?
??????????????????????????????????????????????????
?/usr/lib/arch-id/package/ ? Private other vendor       ?
?                   ? resources of the package   ?
?                   ? that are                   ?
?                   ? architecture-specific and  ?
?                   ? cannot be shared between   ?
?                   ? architectures. Note that   ?
?                    ? this generally does not   ?
?                    ? include private           ?
?                    ? executables since binaries ?
?                    ? of a specific architecture ?
?                    ? may be freely invoked from ?
?                    ? any other supported system ?
?                    ? architecture.             ?
??????????????????????????????????????????????????
?/usr/include/package/   ? Public C/C++ APIs of    ?
?                    ? public shared libraries of ?
?                    ? the package.              ?
??????????????????????????????????????????????????
```

Additional static vendor files may be installed in the /usr/share/

hierarchy to the locations defined by the various relevant

specifications.

The following directories shall be used by the package for local

configuration and files created during runtime:

Table 2. System package variable files locations

```
???????????????????????????????????????????????
?Directory        ? Purpose                   ?
???????????????????????????????????????????????
?/etc/package/     ? System-specific           ?
?                ? configuration for the     ?
?                ? package. It is recommended ?
?                ? to default to safe        ?
?                ? fallbacks if this         ?
?                ? configuration is missing,  ?
?                ? if this is possible.      ?
?                ? Alternatively, a          ?
?                ? tmpfiles.d(5) fragment may ?
?                ? be used to copy or symlink ?
?                ? the necessary files and   ?
?                ? directories from          ?
?                    ? /usr/share/factory/ during ?
?                    ? boot, via the "L" or "C"   ?
?                    ? directives.                ?
????????????????????????????????????????????????????
?/run/package/       ? Runtime data for the       ?
?                    ? package. Packages must be  ?
?                    ? able to create the         ?
?                    ? necessary subdirectories   ?
?                    ? in this tree on their own, ?
?                    ? since the directory is     ?
?                    ? flushed automatically on   ?
?                    ? boot. Alternatively, a     ?
?                    ? tmpfiles.d(5) fragment may ?
?                    ? be used to create the      ?
?                    ? necessary directories      ?
?                    ? during boot, or the        ?
?                    ? RuntimeDirectory=          ?
?                    ? directive of service units ?
?                    ? may be used to create them ?
?                    ? at service startup (see    ?
?                    ? systemd.unit(5) for        ?
?                    ? details).                  ?
????????????????????????????????????????????????????
?/run/log/package/   ? Runtime log data for the   ?
?                    ? package. As above, the     ?
?                    ? package needs to make sure ?
?                    ? to create this directory   ?
?                    ? if necessary, as it will   ?
?                    ? be flushed on every boot.  ?
????????????????????????????????????????????????????
?/var/cache/package/ ? Persistent cache data of   ?
?                    ? the package. If this       ?
?                    ? directory is flushed, the  ?
?                  ? application should work    ?
?                  ? correctly on next          ?
?                  ? invocation, though         ?
?                  ? possibly slowed down due   ?
?                  ? to the need to rebuild any ?
?                  ? local cache files. The     ?
?                  ? application must be        ?
?                  ? capable of recreating this ?
?                  ? directory should it be     ?
?                  ? missing and necessary. To  ?
?                  ? create an empty directory, ?
?                  ? a tmpfiles.d(5) fragment   ?
?                  ? or the CacheDirectory=     ?
?                  ? directive of service units ?
?                  ? (see systemd.unit(5)) may  ?
?                  ? be used.                   ?
???????????????????????????????????????????????????
?/var/lib/package/ ? Persistent private data of ?
?                  ? the package. This is the   ?
?                  ? primary place to put       ?
?                  ? persistent data that does  ?
?                  ? not fall into the other    ?
?                  ? categories listed.         ?
?                  ? Packages should be able to ?
?                  ? create the necessary       ?
?                  ? subdirectories in this     ?
?                  ? tree on their own, since   ?
?                  ? the directory might be     ?
?                  ? missing on boot. To create ?
?                  ? an empty directory, a      ?
?                  ? tmpfiles.d(5) fragment or  ?
?                  ? the StateDirectory=        ?
?                  ? directive of service units ?
?                   ? (see systemd.unit(5)) may  ?
?                   ? be used.                   ?
????????????????????????????????????????????????
?/var/log/package/   ? Persistent log data of the ?
?                   ? package. As above, the     ?
?                   ? package should make sure   ?
?                   ? to create this directory   ?
?                   ? if necessary, possibly     ?
?                   ? using tmpfiles.d(5) or     ?
?                   ? LogsDirectory= (see        ?
?                   ? systemd.unit(5)), as it    ?
?                   ? might be missing.          ?
????????????????????????????????????????????????
?/var/spool/package/ ? Persistent spool/queue     ?
?                   ? data of the package. As    ?
?                   ? above, the package should  ?
?                   ? make sure to create this   ?
?                   ? directory if necessary, as ?
?                   ? it might be missing.       ?
????????????????????????????????????????????????
```

USER PACKAGES

Programs running in user context should follow strict rules when placing their own files in the user's home directory. The following table lists recommended locations in the home directory for specific types of files supplied by the vendor if the application is installed in the home directory. (User applications installed system-wide are covered by the rules outlined above for vendor files.)

Table 3. Vendor package file locations under the home directory of the user

```
?????????????????????????????????????????????????????????
?Directory                  ? Purpose                   ?
?????????????????????????????????????????????????????????
?~/.local/bin/              ? Package executables that   ?
```

```
?                        ? shall appear in the $PATH  ?
?                        ? executable search path. It ?
?                        ? is not recommended to      ?
?                        ? place internal executables ?
?                        ? or executables that are    ?
?                        ? not commonly invoked from  ?
?                        ? the shell in this          ?
?                        ? directory, such as daemon  ?
?                        ? executables. As this       ?
?                        ? directory is shared with   ?
?                        ? most other packages of the ?
?                        ? user, special care should  ?
?                        ? be taken to pick unique    ?
?                        ? names for files placed     ?
?                        ? here, that are unlikely to ?
?                        ? clash with other package's ?
?                        ? files.                     ?
???????????????????????????????????????????????????????
?~/.local/lib/arch-id/   ? Public shared libraries of ?
?                        ? the package. As above, be  ?
?                        ? careful with using overly  ?
?                        ? generic names, and pick    ?
?                        ? unique names for your      ?
?                        ? libraries to place here to ?
?                        ? avoid name clashes.        ?
???????????????????????????????????????????????????????
?~/.local/lib/package/   ? Private, static vendor     ?
?                        ? resources of the package,  ?
?                        ? compatible with any        ?
?                        ? architecture, or any other ?
?                        ? kind of read-only vendor   ?
?                        ? data.                      ?
???????????????????????????????????????????????????????
```

~/.local/lib/arch-id/package/	Private other vendor resources of the package that are architecture-specific and cannot be shared between architectures.

Additional static vendor files may be installed in the ~/.local/share/ hierarchy, mirroring the subdirectories specified in the section "Vendor-supplied operating system resources" above.

The following directories shall be used by the package for per-user local configuration and files created during runtime:

Table 4. User package variable file locations

Directory	Purpose
~/.config/package/	User-specific configuration and state for the package. It is required to default to safe fallbacks if this configuration is missing.
$XDG_RUNTIME_DIR/package/	User runtime data for the package.
~/.cache/package/	Persistent cache data of the package. If this directory is flushed, the application should work correctly on next invocation, though possibly slowed down due

```
?                          ? to the need to rebuild any ?
?                          ? local cache files. The     ?
?                          ? application must be        ?
?                          ? capable of recreating this ?
?                          ? directory should it be     ?
?                          ? missing and necessary.     ?
????????????????????????????????????????????????????????????
```

SEE ALSO

systemd(1), hier(7), systemd-path(1), systemd-gpt-auto-generator(8),

sysctl.d(5), tmpfiles.d(5), pkg-config(1), systemd.unit(5)

NOTES

1. File System Hierarchy

   http://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html

2. XDG Base Directory Specification

   https://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html

3. XDG User Directories

   https://www.freedesktop.org/wiki/Software/xdg-user-dirs

4. IEEE Std 1003.1

   http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html#tag_08_03

5. Using /tmp/ and /var/tmp/ Safely

   https://systemd.io/TEMPORARY_DIRECTORIES

6. Multiarch Architecture Specifiers (Tuples)

   https://wiki.debian.org/Multiarch/Tuples

systemd 252   FILE-HIERARCHY(7)