Pseudo filesystem

Pseudo filesystem

Stable: 31.01.2020 - 13:23 / Revision: 31.01.2020 - 13:16

Contents

1 Introduction

When running, Linux® operating system creates and populates some filesystems which are not present on the rootfs fileystem of the Linux distribution image:

- **pseudo filesystems**: **sysfs** (/sys), **procfs** (/proc), **debugfs** (/sys/kernel/debug), **configfs** (/sys/kernel/config), **tracefs** (/sys/kernel/tracing)
- **temporary filesystems**: **tmppfs** (/dev/shm, /run, /sys/fs/cgroup, /tmp/, /var/volatile, /run/user/*<id>*), **devtmpfs** (/dev)

Pseudo filesystems contain many informations, configurations and logs about the current running kernel.
Informations from those pseudo filesystems are very helpful and any debugging session should start by browsing them.

These both filesystem groups are part of the File Hierarchy Standard (FHS) for the Linux operating system.

As they are placed in volatile memory, they are only available at run time, and they disappear at shutdown.

2 procfs (/proc) - Kernel and process information

Procfs[1] is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux® kernel configuration **CONFIG_PROC_FS**, set to yes by default.

```
Symbol: PROC_FS
Location:
  File systems --->
    Pseudo filesystems -->
      [*] /proc file system support
```

Please refer to Menuconfig or how to configure kernel article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Procfs is sometimes referred to as a process information pseudo-file system. It does not contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).

For this reason it can be seen as a control and information center for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory.

For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

Procfs is explain in details in The Linux Documentation Project[2], or Wikipedia[3].

2.1 Some procfs useful entries

Name	Description
/proc/<*PID*>/	directory containing information about the kernel process <*PID*>
/prod /cmdline	gives the boot options passed to the kernel
/proc /cpuinfo	provides information about the processor: its type, make, model, and performance
/proc /devices	lists all of the devices (divided into the "block" and "character" categories) available on the currently running system, giving also the major part of the /dev name too
/proc /device-tree	contains devices tree information for all nodes organized as defined in the device-tree source file(s).
/proc /interrupts	shows which interrupts are in use, and how many of each there have been.
/proc /iomem	gives information about memory mapping
/proc /kallsyms	contains the dynamic kernel symbol table
/proc	

Name	Description
/kmsg	holds messages output by the kernel.
/proc /meminfo	contains a summary of how the kernel is managing its memory (both physical and swap).
/proc /misc	Miscellaneous pieces of information, lists also the misc features devices
/proc /modules	one of the most important files in /proc; contains a list of the kernel modules currently loaded. It gives some indication of dependencies.

3 sysfs (/sys) - System filesystem

Sysfs[4] is a RAM-based filesystem initially based on ramfs. It provides a means to export kernel data structures, their attributes, and the linkages between them, to user space.

Sysfs is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux® kernel configuration **CONFIG_SYSFS**, set to yes by default.

```
Symbol: SYSFS
Location:
  File systems --->
    Pseudo filesystems -->
      [*] sysfs file system support
```

Please refer to Menuconfig or how to configure kernel article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Useful information also given in How to access information in sysfs article.

3.1 Top Level sysfs directory layout

Sysfs directory arrangement exposes the relationship of kernel data structures.

/sys has a sub-hierarchy file system:

sub-Directory	Description
/sys/bl	

sub-Directory	Description
ock/	Contains one symbolic link for each block device that has been discovered on the system. The symbolic links point to corresponding directories under /sys/devices.
/sys/bus/	Contains one subdirectory for each of the bus types in the kernel. Each bus's directory contains two subdirectories: ./devices/ ./drivers/
/sys/class/	Contains a single layer of further subdirectories for each of the device classes that have been registered on the system (e.g., terminals, network devices, block devices, graphics devices, sound devices, and so on). Inside each of these subdirectories are symbolic links for each of the devices in this class. These symbolic links refer to entries in the /sys/devices directory
/sys/class/net/	Each of the entries in this directory is a symbolic link representing one of the real or virtual networking devices that are visible in the network namespace of the process that is accessing the directory. Each of these symbolic links refers to entries in the /sys/devices directory
/sys/dev/	This directory contains two subdirectories block/ and char/, corresponding, respectively, to the block and character devices on the system. Inside each of these subdirectories are symbolic links with names of the form major-ID:minor-ID, where the ID values correspond to the major and minor ID of a specific device. Each symbolic link points to the sysfs directory for a device. The symbolic links inside /sys/dev thus provide an easy way to look up the sysfs interface using the device IDs returned by a call to stat tool (or similar)
/sys/devices/	Contains a filesystem representation of the kernel device tree, which is a hierarchy of device structures within the kernel
/sys/fi	

sub-Directory	Description
rmware/	Contains interfaces for viewing and manipulating firmware-specific objects and attributes
/sys/fs/	Contains subdirectories for some filesystems. A filesystem will have a subdirectory here only if it chose to explicitly create the subdirectory
/sys/kernel/	Contains various files and subdirectories that provide information about the running kernel
/sys/module/	Contains one subdirectory for each module that is loaded into the kernel. The name of each directory is the name of the module

3.2 Zoom to debugfs (/sys/kernel/debug)

Please refer to debugfs article.

3.3 Zoom to configfs (/sys/kernel/config)

Please refer to configfs article.

3.4 Zoom to tracefs (/sys/kernel/tracing)

Tracefs is used with the Linux kernel tracing framework.

Example of usage is given in Ftrace article.

- Command to mount tracefs:

```
Board $> mount -t tracefs nodev /sys/kernel/tracing
```

To find out which tracers are available, simply read available_tracers file in the tracing directory:

```
Board $> cat /sys/kernel/tracing/available_tracers
 function_graph function nop
```

More tracers can be added by kernel build configurations. Please refer to Ftrace#More_tracers paragraph.

4 Information about temporary filesystem

4.1 tmpfs

Tmpfs[5] is a file system which keeps all files in virtual memory.

It is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux® kernel configuration CONFIG_TMPFS, set to yes by default.

```
Symbol: TMPFS
Location:
  File systems --->
    Pseudo filesystems -->
      [*] Tmpfs virtual memory file system support (former shm fs)
```

Please refer to Menuconfig or how to configure kernel article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Everything in tmpfs is temporary in the sense that no files will be created on your hard drive. If you unmount a tmpfs instance, everything stored therein is lost.

On the board target, you can check for the directory path mount with the tmpfs:

```
Board $> mount | grep tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
tmpfs on /var/volatile type tmpfs (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=43812k,mode=700)
```

For all details you can refer to the Linux documentation about tmpfs[5].

4.2 devtmpfs

Devtmpfs is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux® kernel configuration **CONFIG_DEVTMPFS** and **CONFIG_DEVTMPFS_MOUNT**, set to yes by default.

```
Symbol: DEVTMPFS
Location:
  Device Drivers --->
    Generic Driver Options -->
      [*] Maintain a devtmpfs filesystem to mount at /dev

Symbol: DEVTMPFS_MOUNT
Location:
  Device Drivers --->
    Generic Driver Options -->
      [*] Maintain a devtmpfs filesystem to mount at /dev
        [*] Automount devtmpfs at /dev, after the kernel mounted the rootfs
```

- **/dev** - special or device files

Devtmpfs is mounted on /dev which is the location of special or device files. Many of these are generated at boot time or even on the fly.

It is a very interesting directory that highlights one important aspect of the Linux filesystem: everything is a file or a directory.

Look through this directory and you can see device file system entries which represent the various partitions on the first master drive of the system:
For example:

- mmcblk0p<*id*> (microSD Card),
- mmcblk1p<*id*> (eMMC),
- sda<*id*>,
- sdb<*id*> (NAND or USB Key),
- ttySTM<*id*> (tty Serial link),
- etc...

These entries can be both read from and written to.
Take /dev/ttyUSB0, for instance. This file represents the USB Serial port. Sending data to and reading from /dev/ttyUSB0 will allow you to communicate with host PC through the minicom application (or equivalent).

/dev is very helpful, more info could be found in the Linux Documentation Project[6].

5 References

- Documentation/filesystems/proc.txt
- http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html
- https://en.wikipedia.org/wiki/Procfs
- Documentation/filesystems/sysfs.txt
- 5.05.1 Documentation/filesystems/tmpfs.txt
- http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html

System File System (See https://en.wikipedia.org/wiki/Sysfs for more details)

Process File System (See https://en.wikipedia.org/wiki/Procfs for more details)

Debug File System (See https://en.wikipedia.org/wiki/Debugfs for more details)

Configuration File System (See https://en.wikipedia.org/wiki/Configfs for more details)

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today forvolatile random-accesssemiconductor memories.)

Read Only

former spelling for e•MMC ('e' in italic)