Файловая система tmpfs

Автор:
Christoph Rohland <cr@sap.com>, 1.12.01

Обновление:
Hugh Dickins, 4 June 2007

Обновление:
KOSAKI Motohiro, 16 Mar 2010

Файловая система tmpfs служит для хранения файлов в виртуальной памяти.

Содержимое tmpfs является временным в том смысле, что на «жесткие» диски диски компьютера файлы этой системы не записываются. При размонтировании экземпляра tmpfs все хранящиеся в этой файловой системе данные будут потеряны.

Файловая система tmpfs размещается во внутреннем кэше ядра и меняет свой размер (расширяется или сужается) в соответствии с обънмом хранящейся информации. Ненужные в настоящее время страницы могут «сбрасываться» в область подкачки (swap). Максимальный размер файловой системы можно исзменять в процессе работы с помощью команды **mount -o remount …**

В отличие от файловой системы ramfs (прообраз tmpfs) поддерживается возможность свопинга и контроля размера. Другим похожим объектом являются виртуальные диски в ОЗУ
(RAM-диск, /dev/ram*), которые имитируют «жесткий» диск фиксированного размера, размещаемый в физической памяти компьютера с организованной на нем обычной файловой системой. Для виртуальных дисков свопинг и изменение размера не поддерживаются.

Поскольку tmpfs может размещаться только в страницах кэша и области подкачки, все страницы tmpfs, находящиеся в памяти, будут выглядеть, как кэшированные. Они не будут представляться чем-либо типа разделяемых страниц. Более того, вы можете проверить реальное размещение данных tmpfs в ОЗУ и области подкачки с помощью команд df и du.

Файловая система tmpfs служит для решения ряда задач:

1) В ядре используется несколько внутренних точек монтирования, невидимых для пользователя и служащих для организации разделяемых анонимных отображений и разделяемой памяти SYSV.

   На эти монтирования опция ядра CONFIG_TMPFS не указывает влияния. Если эта опция не была установлена при сборке ядра, видимая пользователю часть tmpfs не создается, но внутренние механизмы продолжают использоваться.

2) glibc, начиная с версии 2.2, предполагает, что файловая система tmpfs примонтирована, как **/dev/shm** для разделяемой памяти POSIX (**shm_open**, **shm_unlink**). Для реализации этого в файле **/etc/fstab** указывается строка:

   ```
   tmpfs  /dev/shm      tmpfs  defaults      0 0
   ```

   Не забудьте создать на диске каталог, куда вы планируете монтировать tmpfs.

   Эта точка монтирования не требуется для разделяемой памяти SYSV, поскольку для этого служит внутреннее монтирование в ядре (в ядре версии 2.3 требовалось монтировать предшественника tmpfs - shm fs - для использования разделяемой памяти SYSV).

3) Некоторые считают весьма удобным монтировать файловую систему tmpfs как **/tmp** и/или **/var/tmp** при наличии большого раздела подкачки. В большинстве дистрибутивов mkinitrd позволяет работать с tmpfs, примонтированной, как **/tmp**.

4) Можно придумать еще множество вариантов применения tmpfs.

Файловая система tmpfs поддерживает три опции монтирования для управления размером:

size задает предельный размер экземпляра tmpfs в байтах (по умолчанию составляет половину размера имеющейся в системе физической памяти - ОЗУ). Создание tmpfs слишком большого размера может приводить к «зависанию» системы, поскольку обработчик OOM не сможет освободить эту память.

nr_blocks задает предельный размер числом блоков PAGE_CACHE_SIZE.

nr_inodes задает максимальное число узлов inode для данного экземпляра (по умолчанию совпадает меньшим из значений половины от числа страниц ОЗУ или на машинах с highmem половины от числа страниц ОЗУ lowmem).

При задании этих параметров можно использовать суффиксы **k** (kilo), **m** (mega) и **g** (giga). Для параметра **size** можно также использовать суффикс **%**, который показывает задание размера tmpfs в процентах от физического размера ОЗУ. По умолчанию, если ни один из параметров **size**, **nr_blocks** не задан, будет использоваться значение **size=50%**.
Если **nr_blocks=0** (или **size=0**), число блоков для данного экземпляра не ограничивается; если **nr_inodes=0**, не ограничивается число узлов inode. В общем случае монтирование с такими опциями нежелательно, поскольку при этом каждый пользователь с правом записи сможет занять всю оперативную память системы, однако такие значения позволяют повысить уровень масштабируемости tmpfs в многопроцессорных машинах, интенсивно использующих файловую систему tmpfs.

Для tmpfs поддерживается опция монтирования **mpol**, позволяющая использовать политику выделения памяти NUMA для всех файлов данного экземпляра (при включенной опции CONFIG_NUMA). Значение данной опции также можно менять в процессе работы с помощью команды **mount -o remount …**

mpol=default использовать политику выделения для процессов (см. set_mempolicy(2));

mpol=prefer:Node предпочтительно выделять память из данного узла (Node);

mpol=bind:NodeList выделять память только из узлов списка NodeList;

mpol=interleave предпочтительно выделять память из каждого узла поочередно;

mpol=interleave:NodeList память выделяется из каждого узла NodeList поочередно;

mpol=local предпочтительно выделять память из локального узла.

NodeList предсатвляет собой список разделенных запятыми десятичных значений или диапазонов (два десятичных числа, начиная с меньшего, через дефис). Например, **mpol=bind:0-3,5,7,9-15**.

Заданная при монтировании политика выделения памяти сохраняется для использования при создании файлов. Когда та или иная задача создает файл в tmpfs, политика выделения памяти, заданная при монтировании, применяется с соответствующим списком NodeList (если он задан) и с учетом ограничений набора процессоров (см. Documentation/cgroups/cpusets.txt в дистрибутиве ядра) и флагов, перечисленных ниже. Если в результате список NodeLists становится пустым, используется принятая по умолчанию политика выделения памяти (mpol=default).

Политика распределения памяти NUMA поддерживает ряд дополнительных флагов, используемых вместе с заданным при монтировании режимом. Эти флаги могут быть заданы при монтировании tmpfs путем указания после режима но перед списком NodeList. Полный список флагов можно найти в документе Documentation/vm/numa_memory_policy.txt, где описано также воздействие этих флагов на политику выделения памяти.

=static эквивалентно MPOL_F_STATIC_NODES

=relative эквивалентно MPOL_F_RELATIVE_NODES

Например, **mpol=bind=static:NodeList** эквивалентно политике выделения **MPOL_BIND | MPOL_F_STATIC_NODES**.

Отметим, что попытка монтирование tmpfs с опцией mpol будет приводить к отказу, если используемое ядро не поддерживает NUMA, а также в тех случаях, когда в списке присутствует неактивный узел. Если в вашей системе используется tmpfs, но некоторые варианты ядра могут не поддерживать NUMA (например, ядро, используемое при восстановлении системы) или отдельные узлы могут лказываться неактивными, рекомендуется исключить опцию mpol из параметров автоматического монтирования. Вы можете добавить нужную опцию позднее, когда tmpfs уже примонтирована (как MountPoint) с помощью команды **mount -o remount,mpol=Policy:NodeList MountPoint**.

Для задания изначального корневого каталога можно использовать приведенные ниже опции монтирования:

mode права доступа в восьмеричном формате;

uid идентификатор пользователя;

gid идентификатор группы.

Эти опции не меняются при перемонтировании файловой системы, но исх можно поменять с помощью команд **chmod**, **chown** и **chgrp** на смонтированной файловой системе.

Так, команда **mount -t tmpfs -o size=10G,nr_inodes=10k,mode=700 tmpfs /mytmpfs** будет создавать экземпляр tmpfs с точкой монтирования **/mytmpfs**, для которого будет выделено 10 Гбайт RAM/SWAP с 10240 узлами inode, доступный только пользователю root.

Перевод на русский язык

Николай Малых

nmalykh@protocols.ru