NAME

libhugetlbfs − preload library to back text, data, malloc() or shared memory with hugepages

SYNOPSIS

export [environment options]
[LD_PRELOAD=libhugetlbfs.so] target_application

DESCRIPTION

libhugetlbfs is a library that can back application text, data, malloc() and shared memory with hugepages. This is of benefit to applications that use large amounts of address space and suffer a performance hit due to TLB misses. Wall-clock time or oprofile can be used to determine if there is a performance benefit from using **libhugetlbfs** or not. In all cases but shared memory, a hugetlbfs mount must exist and a hugepage pool defined for hugepages to be used.

Some limited functionality is available for unmodified dynamically linked applications. By preloading the library, the library can back malloc() and shared memory, and text and data segments can be partially backed if they are large enough.

For the effective backing of text and data with huge pages, the application must be linked to the library and the ELF segments correctly aligned using the ld helpers. Once linked, malloc or shared memory can still be backed but no pre-loading is required. See /usr/share/doc/libhugetlbfs/HOWTO and ld.hugetlbfs(1) for detailed instructions on relinking applications.

For applications that are hugepage-aware and linked to the library **get_huge_pages()** can be used for the direct allocation of hugepage-backed regions.

Unless otherwise specified, **libhugetlbfs** will use the default hugepage size to back memory regions. The default size is the value of Hugepagesize displayed in /proc/meminfo. The size can be specified in bytes or in kilobytes, megabytes, or gigabytes by appending K, M, or G respectively. It is an error to specify a invalid, unsupported, or otherwise unconfigured huge page size. Kernel 2.6.27 or later is required to specify any pagesize other than the default.

See /usr/share/docs/libhugetlbfs/HOWTO for detailed instructions on how the library should be used, particularly when relinking the application. This manual page provides a brief synopsis of the environment variables as a quick reference.

The following variables affect what memory regions are backed by hugepages. In all cases, the environment being unset implies the feature should remain disabled.

HUGETLB_DEFAULT_PAGE_SIZE=<pagesize>

This sets the default hugepage size to be used by libhugetlbfs. If not set, libhugetlbfs will use the kernel's default hugepage size.

HUGETLB_MORECORE=[yes|<pagesize>]

This enables the hugepage malloc() feature, instructing libhugetlbfs to override glibc's normal morecore() function with a hugepage version and use it for malloc(). All application malloc() memory should come from hugepage memory until it runs out, it will then fallback to base pages. Note that applications that use custom allocators may not be able to back their heaps using hugepages and this environment variable. It may be necessary to modify the custom allocator to use **get_huge_pages()**.

HUGETLB_SHM=yes

When this environment variable is set, the SHM_HUGETLB flag is added to the shmget() call and the size parameter is aligned to back the shared memory segment with hugepages. In the event

hugepages cannot be used, base pages will be used instead and a warning will be printed to explain the failure. The pagesize cannot be specified with this parameter. To change the kernels default hugepage size, use the pagesize= kernel boot parameter (2.6.26 or later required).

HUGETLB_ELFMAP=[no|[R[<=pagesize>]:[W[<=pagesize>]]]

If the application has been relinked (see the HOWTO for instructions), this environment variable determines whether read-only, read-write, both or no segments are backed by hugepages and what pagesize should be used. If the recommended relinking method has been used, then **hugeedit** can be used to automatically back the text or data by default.

HUGETLB_FORCE_ELFMAP=yes

Force the use of hugepages for text and data segments even if the application has not been relinked to align the ELF segments on a hugepage boundary. Partial segment remapping is not guaranteed to work and the segments must be large enough to contain at least one hugepage for the remapping to occur.

The following options affect how libhugetlbfs behaves.

HUGETLB_RESTRICT_EXE=e1:e2:...:eN

By default, libhugetlbfs will act on any program that it is loaded with, either via LD_PRELOAD or by explicitly linking with -lhugetlbfs.

There are situations in which it is desirable to restrict libhugetlbfs' actions to specific programs. For example, some ISV applications are wrapped in a series of scripts that invoke bash, python, and/or perl. It is more convenient to set the environment variables related to libhugetlbfs before invoking the wrapper scripts, yet this has the unintended and undesirable consequence of causing the script interpreters to use and consume hugepages. There is no obvious benefit to causing the script interpreters to use hugepages, and there is a clear disadvantage: fewer hugepages are available to the actual application.

To address this scenario, set HUGETLB_RESTRICT_EXE to a colon-separated list of programs to which the other libhugetlbfs environment variables should apply. (If not set, libhugetlbfs will attempt to apply the requested actions to all programs.) For example,

HUGETLB_RESTRICT_EXE=hpcc:long_hpcc

will restrict libhugetlbfs' actions to programs named /home/fred/hpcc and /bench/long_hpcc but not /bin/hpcc_no.

HUGETLB_MORECORE_SHRINK=yes

By default, the hugepage heap does not shrink. Shrinking is enabled by setting this environment variable. It is disabled by default as glibc occasionally exhibits strange behaviour if it mistakes the heap returned by **libhugetlbfs** as a foreign brk().

HUGETLB_NO_PREFAULT

By default **libhugetlbfs** will prefault regions it creates to ensure they can be referenced without receiving a SIGKILL. On kernels older than 2.6.27, this was necessary as the system did not guarantee that future faults would succeed on regions mapped MAP_PRIVATE. Prefaulting impacts the performance of malloc() and can result in poor placement on NUMA systems. If it is known the hugepage pool is large enough to run the application or the kernel is 2.6.27 or later, this environment variable should be set.

HUGETLB_NO_RESERVE=yes

By default, the kernel will reserve huge pages at mmap() time to ensure that future faults will succeed. This avoids unexpected application failure at fault time but some applications depend on memory overcommit to create large sparse mappings. For this type of application, setting this environment variable will create huge page backed mappings without a reservation. Use this option with extreme care as in the event huge pages are not available when the mapping is used, the application will be killed. On older kernels, the use of this feature can trigger the OOM killer. Hence, even with this variable set, reservations may still be used for safety.

HUGETLB_MORECORE_HEAPBASE=address

libhugetlbfs normally picks an address to use as the base of the heap for malloc() automatically. This environment variable fixes which address is used.

HUGETLB_PATH=<path>

The path to the hugetlbfs mount is automatically determined at run-time. In the event there are multiple mounts and the wrong one is being selected, use this option to select the correct one. This may be the case if an application-specific mount with a fixed quota has been created for example.

HUGETLB_SHARE=1

By default, **libhugetlbfs** uses unlinked hugetlbfs files to store remapped program segment data. If the same program is started multiple times using hugepage segments, multiple hugepages will be used to store the same program data. The reduce this wastage, setting this environment variable will share read-only segments between multiple invocations of a program at the cost of the memory being used whether the applications are running or not. It is also possible that a malicious application inferfere with other applications executable code. See the HOWTO for more detailed information on this topic.

The following options control the verbosity of **libhugetlbfs**.

HUGETLB_VERBOSE=<level>

The default value for this is 1 and the range of the value is from 0 to 99. The higher the value, the more verbose the output is. 0 is quiet and 3 will output much debugging information.

HUGETLB_DEBUG

Once set, this will give very detailed output on what is happening in the library and run extra diagnostics.

FILES

[DESTDIR|/usr/share]/doc/libhugetlbfs/HOWTO

SEE ALSO

oprofile(1), ld.hugetlbfs(1), hugectl(8), hugeedit(8), gethugepagesize(3), gethugepagesizes(3), getpagesizes(3), hugetlbfs_test_path(3), hugetlbfs_find_path(3), hugetlbfs_find_path_for_size(3), hugetlbfs_test_path(3), hugetlbfs_test_path_for_size(3), hugetlbfs_unlinked_fd(3), hugetlbfs_unlinked_fd_for_size(3), get_huge_pages(3), free_huge_pages(3)

AUTHORS

libhugetlbfs was written by various people on the libhugetlbfs-devel mailing list.