mpatrol reference card

The mpatrol library can read certain options at run-time from an environment variable called `MPATROL_OPTIONS`. This variable must contain one or more valid option keywords from the list below and must be no longer than 1024 characters in length. If `MPATROL_OPTIONS` is unset or empty then the default settings will be used.

Library behaviour

`HELP` Displays a quick-reference option summary.

`PROGFILE=<`***string***`>` Specifies an alternative filename with which to locate the executable file containing the program's symbols.

`CHECK=<`***unsigned range***`>` Specifies a range of allocation indices at which to check the integrity of free memory and overflow buffers.

`DEFALIGN=<`***unsigned integer***`>` Specifies the default alignment for general-purpose memory allocations, which must be a power of two.

`NOPROTECT` Specifies that the mpatrol library's internal data structures should not be made read-only after every memory allocation, reallocation or deallocation.

`SAFESIGNALS` Instructs the library to save and replace certain signal handlers during the execution of library code and to restore them afterwards.

`USEMMAP` Specifies that the library should use `mmap()` instead of `sbrk()` to allocate system memory on UNIX platforms.

Logging and tracing

`LOGFILE=<`***string***`>` Specifies an alternative file in which to place all diagnostics from the mpatrol library.

`LOGALLOCS` Specifies that all memory allocations are to be logged and sent to the log file.

`LOGREALLOCS` Specifies that all memory reallocations are to be logged and sent to the log file.

`LOGFREES` Specifies that all memory deallocations are to be logged and sent to the log file.

`LOGMEMORY` Specifies that all memory operations are to be logged and sent to the log file.

`LOGALL` Equivalent to the `LOGALLOCS`, `LOGREALLOCS`, `LOGFREES` and `LOGMEMORY` options specified together.

`SHOWFREED` Specifies that a summary of all of the freed memory allocations should be displayed at the end of program execution.

`SHOWUNFREED` Specifies that a summary of all of the unfreed memory allocations should be displayed at the end of program execution.

`SHOWMAP` Specifies that a memory map of the entire heap should be displayed at the end of program execution.

`SHOWSYMBOLS` Specifies that a summary of all of the function symbols read from the program's executable file should be displayed at the end of program execution.

`SHOWALL` Equivalent to the `SHOWFREED`, `SHOWUNFREED`, `SHOWMAP` and `SHOWSYMBOLS` options specified together.

`USEDEBUG` Specifies that any debugging information in the executable file should be used to obtain additional source-level information.

General errors

`CHECKALLOCS` Checks that no attempt is made to allocate a block of memory of size zero.

`CHECKREALLOCS` Checks that no attempt is made to reallocate a `NULL` pointer or resize an existing block of memory to size zero.

`CHECKFREES` Checks that no attempt is made to deallocate a `NULL` pointer.

`CHECKALL` Equivalent to the `CHECKALLOCS`, `CHECKREALLOCS` and `CHECKFREES` options specified together.

`ALLOCBYTE=<`***unsigned integer***`>` Specifies an 8-bit byte pattern with which to prefill newly-allocated memory.

`FREEBYTE=<`***unsigned integer***`>` Specifies an 8-bit byte pattern with which to prefill newly-freed memory.

`NOFREE` Specifies that the mpatrol library should keep all reallocated and freed memory allocations.

`PRESERVE` Specifies that any reallocated or freed memory allocations should preserve their original contents.

Overwrites and underwrites

`OFLOWSIZE=<`***unsigned integer***`>` Specifies the size in bytes to use for all overflow buffers, which must be a power of two.

`OFLOWBYTE=<`***unsigned integer***`>` Specifies an 8-bit byte pattern with which to fill the overflow buffers of all memory allocations.

`OFLOWWATCH` Specifies that watch point areas should be used for overflow buffers rather than filling with the overflow byte.

`PAGEALLOC=<LOWER|UPPER>` Specifies that each individual memory allocation should occupy at least one page of virtual memory and should be placed at the lowest or highest point within these pages.

`ALLOW`

Usin

`ALLOC`

`REALL`

`FREES`

Test

`LIMIT`

`FAILF`

`FAILS`

`UNFRE`

Prof

`PROF`

`PROFF`

`AUTOS`

`SMALL`

`MEDIU`

`LARGE`

All of the function definitions in `mpatrol.h` can be disabled by defining the `NDEBUG` preprocessor macro, which is the same macro used to control the behaviour of the `assert()` function. If `NDEBUG` is defined then no macro redefinition of functions will take place and all special mpatrol library functions will evaluate to empty statements. It is intended that the `NDEBUG` preprocessor macro be defined in release builds.

C dynamic memory allocation functions

`malloc()`	Allocates memory.
`calloc()`	Allocates zero-filled memory.
`memalign()`	Allocates memory with a specified alignment.
`valloc()`	Allocates page-aligned memory.
`pvalloc()`	Allocates a number of pages.
`strdup()`	Duplicates a string.
`strndup()`	Duplicates a string with a maximum length.
`strsave()`	Duplicates a string.
`strnsave()`	Duplicates a string with a maximum length.
`realloc()`	Resizes memory.
`recalloc()`	Resizes memory allocated by `calloc()`.
`expand()`	Resizes memory but does not relocate it.
`free()`	Frees memory.
`cfree()`	Frees memory allocated by `calloc()`.

C++ dynamic memory allocation functions

`operator new`	Allocates memory.
`operator new[]`	Allocates memory for an array.
`operator delete`	Frees memory.
`operator delete[]`	Frees memory allocated by `new[]`.
`set_new_handler()`	Sets up an allocation failure handler.

C memory operation functions

`memset()`	Fills memory with a specific byte.
`bzero()`	Fills memory with the zero byte.
`memccpy()`	Copies memory up to a specific byte.
`memcpy()`	Copies non-overlapping memory.
`memmove()`	Copies possibly-overlapping memory.
`bcopy()`	Copies possibly-overlapping memory.
`memcmp()`	Compares two blocks of memory.
`bcmp()`	Compares two blocks of memory.
`memchr()`	Searches memory for a specific byte.
`memmem()`	Searches memory for specific bytes.

mpatrol library functions

`__mp_info()`	Returns information for an allocation.
`__mp_printinfo()`	Displays information for an allocation.
`__mp_memorymap()`	Displays a map of memory in the heap.
`__mp_summary()`	Displays a summary of library statistics.
`__mp_check()`	Validates memory in the heap.
`__mp_prologue()`	Sets up an allocation prologue handler.
`__mp_epilogue()`	Sets up an allocation epilogue handler.
`__mp_nomemory()`	Sets up an allocation failure handler.

The commands that are distributed with the mpatrol library all parse their command line options in a similar way to the UNIX `getopt()` function. Options that accept numeric arguments can have their value specified in binary, octal, decimal or hexadecimal notation.

mpatrol command options

−1 <*unsigned integer*> See `SMALLBOUND`.

−2 <*unsigned integer*> See `MEDIUMBOUND`.

−3 <*unsigned integer*> See `LARGEBOUND`.

−A <*unsigned integer*> See `ALLOCSTOP`.

−a <*unsigned integer*> See `ALLOCBYTE`.

−C <*unsigned range*> See `CHECK`.

−c See `CHECKALL`.

−D <*unsigned integer*> See `DEFALIGN`.

−d Specifies that programs which were not linked with the mpatrol library should also be traced, but only if they were dynamically linked.

−e <*string*> See `PROGFILE`.

−F <*unsigned integer*> See `FREESTOP`.

−f <*unsigned integer*> See `FREEBYTE`.

−G See `SAFESIGNALS`.

−g See `USEDEBUG`.

−L <*unsigned integer*> See `LIMIT`.

−l <*string*> See `LOGFILE`.

−M See `ALLOWOFLOW`.

−m See `USEMMAP`.

−N

−n

−O <*u*

−o <*u*

−P <*s*

−p

−Q <*u*

−R <*u*

−S

−s

−U <*u*

−V D

−v

−w

−X

−x

−Z <*u*

−z <*u*

mpr

−a Sp

−c Sp

−n <*d*

−V D

mlea

−V D

Copyr

This r
of the