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.

`ALLOWOFLOW` Specifies that a warning rather than an error should be produced if any memory operation function overflows the boundaries of a memory allocation, and that the operation should still be performed.

Using with a debugger

`ALLOCSTOP=`<*unsigned integer*> Specifies an allocation index at which to stop the program when it is being allocated.

`REALLOCSTOP=`<*unsigned integer*> Specifies an allocation index at which to stop the program when a memory allocation is being reallocated.

`FREESTOP=`<*unsigned integer*> Specifies an allocation index at which to stop the program when it is being freed.

Testing

`LIMIT=`<*unsigned integer*> Specifies the limit in bytes at which all memory allocations should fail if the total allocated memory should increase beyond this.

`FAILFREQ=`<*unsigned integer*> Specifies the frequency at which all memory allocations will randomly fail.

`FAILSEED=`<*unsigned integer*> Specifies the random number seed which will be used when determining which memory allocations will randomly fail.

`UNFREEDABORT=`<*unsigned integer*> Specifies the minimum number of unfreed allocations at which to abort the program just before program termination.

Profiling

`PROF` Specifies that all memory allocations are to be profiled and sent to the profiling output file.

`PROFFILE=`<*string*> Specifies an alternative file in which to place all memory allocation profiling information from the mpatrol library.

`AUTOSAVE=`<*unsigned integer*> Specifies the frequency at which to periodically write the profiling data to the profiling output file.

`SMALLBOUND=`<*unsigned integer*> Specifies the limit in bytes up to which memory allocations should be classified as small allocations for profiling purposes.

`MEDIUMBOUND=`<*unsigned integer*> Specifies the limit in bytes up to which memory allocations should be classified as medium allocations for profiling purposes.

`LARGEBOUND=`<*unsigned integer*> Specifies the limit in bytes up to which memory allocations should be classified as large allocations for profiling purposes.

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	See `NOPROTECT`.
–n	See `NOFREE`.
–O *<unsigned integer>*	See `OFLOWSIZE`.
–o *<unsigned integer>*	See `OFLOWBYTE`.
–P *<string>*	See `PROFFILE`.
–p	See `PROF`.
–Q *<unsigned integer>*	See `AUTOSAVE`.
–R *<unsigned integer>*	See `REALLOCSTOP`.
–S	See `SHOWMAP` and `SHOWSYMBOLS`.
–s	See `SHOWFREED` and `SHOWUNFREED`.
–U *<unsigned integer>*	See `UNFREEDABORT`.
–V	Displays the version number of the **mpatrol** command.
–v	See `PRESERVE`.
–w	See `OFLOWWATCH`.
–X	See `PAGEALLOC=UPPER`.
–x	See `PAGEALLOC=LOWER`.
–Z *<unsigned integer>*	See `FAILSEED`.
–z *<unsigned integer>*	See `FAILFREQ`.

mprof command options

–a Specifies that different call sites from within the same function are to be differentiated and that the names of all functions should be displayed with their call site offset in bytes.

–c Specifies that certain tables should be sorted by the number of allocations or deallocations rather than the total number of bytes allocated or deallocated.

–n *<depth>* Specifies the maximum stack depth to use when calculating if one call site has the same call stack as another call site. This also specifies the maximum number of functions to display in a call stack.

–V Displays the version number of the **mprof** command.

mleak command options

–V Displays the version number of the **mleak** command.