Rocky Enterprise Linux 9.2 Manual Pages on command 'memusage.1'

$ man memusage.1

MEMUSAGE(1) Linux user manual MEMUSAGE(1)

NAME

memusage - profile memory usage of a program

SYNOPSIS

memusage [option]... program [programoption]...

DESCRIPTION

memusage is a bash script which profiles memory usage of the program, program. It preloads the libmemusage.so library into the caller's en? vironment (via the LD_PRELOAD environment variable; see ld.so(8)). The libmemusage.so library traces memory allocation by intercepting calls to malloc(3), calloc(3), free(3), and realloc(3); optionally, calls to mmap(2), mremap(2), and munmap(2) can also be intercepted.

memusage can output the collected data in textual form, or it can use memusagestat(1) (see the -p option, below) to create a PNG file con? taining graphical representation of the collected data.

Memory usage summary

The "Memory usage summary" line output by memusage contains three fields:

heap total

Sum of size arguments of all malloc(3) calls, products of arguments (nmemb*size) of all calloc(3) calls, and sum of length arguments of all mmap(2) calls. In the case of real? loc(3) and mremap(2), if the new size of an allocation is larger than the previous size, the sum of all such differ? ences (new size minus old size) is added.

heap peak

Maximum of all size arguments of malloc(3), all products of nmemb*size of calloc(3), all size arguments of realloc(3), length arguments of mmap(2), and new_size arguments of mremap(2).

stack peak

Before the first call to any monitored function, the stack pointer address (base stack pointer) is saved. After each function call, the actual stack pointer address is read and the difference from the base stack pointer computed. The maximum of these differences is then the stack peak.

Immediately following this summary line, a table shows the number calls, total memory allocated or deallocated, and number of failed calls for each intercepted function. For realloc(3) and mremap(2), the additional field "nomove" shows reallocations that changed the address of a block, and the additional "dec" field shows reallocations that de? creased the size of the block. For realloc(3), the additional field "free" shows reallocations that caused a block to be freed (i.e., the reallocated size was 0).

The "realloc/total memory" of the table output by memusage does not re? flect cases where realloc(3) is used to reallocate a block of memory to have a smaller size than previously. This can cause sum of all "total memory" cells (excluding "free") to be larger than the "free/total mem? ory" cell.

Histogram for block sizes

The "Histogram for block sizes" provides a breakdown of memory alloca?

tions into various bucket sizes.

OPTIONS

-n name, --progname=name

Name of the program file to profile.

-p file, --png=file

Generate PNG graphic and store it in file.

-d file, --data=file

Generate binary data file and store it in file.

-u, --unbuffered

Do not buffer output.

-b size, --buffer=size

Collect size entries before writing them out.

--no-timer

Disable timer-based (SIGPROF) sampling of stack pointer value.

-m, --mmap

Also trace mmap(2), mremap(2), and munmap(2).

-?, --help

Print help and exit.

--usage

Print a short usage message and exit.

-V, --version

Print version information and exit.

The following options apply only when generating graphical output:

-t, --time-based

Use time (rather than number of function calls) as the scale for

the X axis.

-T, --total

Also draw a graph of total memory use.

--title=name

Use name as the title of the graph.

-x size, --x-size=size

Make the graph size pixels wide.

-y size, --y-size=size

Make the graph size pixels high.

EXIT STATUS

Exit status is equal to the exit status of profiled program.

BUGS

To report bugs, see ?http://www.gnu.org/software/libc/bugs.html?

EXAMPLES

Below is a simple program that reallocates a block of memory in cycles that rise to a peak before then cyclically reallocating the memory in smaller blocks that return to zero. After compiling the program and running the following commands, a graph of the memory usage of the pro? gram can be found in the file memusage.png:

```
$ memusage --data=memusage.dat ./a.out
...
Memory usage summary: heap total: 45200, heap peak: 6440, stack peak: 224
        total calls  total memory  failed calls
 malloc|         1           400             0
realloc|        40         44800             0  (nomove:40, dec:19, free:0)
 calloc|         0             0             0
   free|         1           440
Histogram for block sizes:
  192-207             1   2% ================
...
 2192-2207            1   2% ================
 2240-2255            2   4% ================================
 2832-2847            2   4% ================================
 3440-3455            2   4% ================================
 4032-4047            2   4% ================================
 4640-4655            2   4% ================================
 5232-5247            2   4% ================================
 5840-5855            2   4% ================================
 6432-6447            1   2% ================
$ memusagestat memusage.dat memusage.png
```

Program source

```
#include <stdio.h>
#include <stdlib.h>
#define CYCLES 20
int
main(int argc, char *argv[])
{
    int i, j;
    size_t size;
    int *p;
    size = sizeof(*p) * 100;
    printf("malloc: %zu\n", size);
    p = malloc(size);
    for (i = 0; i < CYCLES; i++) {
        if (i < CYCLES / 2)
            j = i;
        else
            j--;
        size = sizeof(*p) * (j * 50 + 110);
        printf("realloc: %zu\n", size);
        p = realloc(p, size);
        size = sizeof(*p) * ((j + 1) * 150 + 110);
        printf("realloc: %zu\n", size);
        p = realloc(p, size);
    }
    free(p);
    exit(EXIT_SUCCESS);
}
```

SEE ALSO

memusagestat(1), mtrace(1), ld.so(8)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at

https://www.kernel.org/doc/man-pages/.