



Linux Ubuntu 26.04 Manual Pages on command 'mallinfo2.3'

\$ man mallinfo2.3

mallinfo(3) Library Functions Manual mallinfo(3)

NAME

mallinfo, mallinfo2 - obtain memory allocation information

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <malloc.h>

struct mallinfo mallinfo(void);

struct mallinfo2 mallinfo2(void);
```

DESCRIPTION

These functions return a copy of a structure containing information about memory allocations performed by malloc(3) and related functions. The structure returned by each function contains the same fields. However, the older function, mallinfo(), is deprecated since the type used for the fields is too small (see BUGS).

Note that not all allocations are visible to these functions; see BUGS and consider using malloc_info(3) instead.

The mallinfo2 structure returned by mallinfo2() is defined as follows:

```
struct mallinfo2 {
    size_t arena; /* Non-mmapped space allocated (bytes) */
    size_t ordblks; /* Number of free chunks */
    size_t smlbks; /* Number of free fastbin blocks */
    size_t hblks; /* Number of mmapped regions */
    size_t hblkhd; /* Space allocated in mmapped regions
```


and then every second allocated block is freed:

```
$ ./a.out 1000 100 2;
```

```
===== Before allocating blocks =====
```

```
Total non-mmapped bytes (arena):    0
```

```
# of free chunks (ordblks):         1
```

```
# of free fastbin blocks (smbblks):  0
```

```
# of mapped regions (hblks):        0
```

```
Bytes in mapped regions (hblkhd):    0
```

```
Max. total allocated space (usmbblks): 0
```

```
Free bytes held in fastbins (fsmbblks): 0
```

```
Total allocated space (uordblks):    0
```

```
Total free space (fordblks):         0
```

```
Topmost releasable block (keepcost): 0
```

```
===== After allocating blocks =====
```

```
Total non-mmapped bytes (arena):    135168
```

```
# of free chunks (ordblks):         1
```

```
# of free fastbin blocks (smbblks):  0
```

```
# of mapped regions (hblks):        0
```

```
Bytes in mapped regions (hblkhd):    0
```

```
Max. total allocated space (usmbblks): 0
```

```
Free bytes held in fastbins (fsmbblks): 0
```

```
Total allocated space (uordblks):    104000
```

```
Total free space (fordblks):         31168
```

```
Topmost releasable block (keepcost): 31168
```

```
===== After freeing blocks =====
```

```
Total non-mmapped bytes (arena):    135168
```

```
# of free chunks (ordblks):         501
```

```
# of free fastbin blocks (smbblks):  0
```

```
# of mapped regions (hblks):        0
```

```
Bytes in mapped regions (hblkhd):    0
```

```
Max. total allocated space (usmbblks): 0
```

```
Free bytes held in fastbins (fsmbblks): 0
```

```
Total allocated space (uordblks):    52000
```

Total free space (fordblks): 83168

Topmost releasable block (keepcost): 31168

Program source

```
#include <malloc.h>

#include <stdlib.h>

#include <string.h>

static void

display_mallinfo2(void)

{

    struct mallinfo2 mi;

    mi = mallinfo2();

    printf("Total non-mmapped bytes (arena):   %zu\n", mi.arena);

    printf("# of free chunks (ordblks):       %zu\n", mi.ordblks);

    printf("# of free fastbin blocks (smblocks): %zu\n", mi.smblocks);

    printf("# of mapped regions (hblks):        %zu\n", mi.hblks);

    printf("Bytes in mapped regions (hblkhd):   %zu\n", mi.hblkhd);

    printf("Max. total allocated space (usmblocks): %zu\n", mi.usmblocks);

    printf("Free bytes held in fastbins (fsmblocks): %zu\n", mi.fsmblocks);

    printf("Total allocated space (uordblks):   %zu\n", mi.uordblks);

    printf("Total free space (fordblks):        %zu\n", mi.fordblks);

    printf("Topmost releasable block (keepcost): %zu\n", mi.keepcost);

}

int

main(int argc, char *argv[])

{

#define MAX_ALLOCS 500000

    char *alloc[MAX_ALLOCS];

    size_t blockSize, numBlocks, freeBegin, freeEnd, freeStep;

    if (argc < 3 || strcmp(argv[1], "--help") == 0) {

        fprintf(stderr, "%s num-blocks block-size [free-step "

            "[start-free [end-free]]\n", argv[0]);

        exit(EXIT_FAILURE);

    }
```

```

numBlocks = atoi(argv[1]);
blockSize = atoi(argv[2]);
freeStep = (argc > 3) ? atoi(argv[3]) : 1;
freeBegin = (argc > 4) ? atoi(argv[4]) : 0;
freeEnd = (argc > 5) ? atoi(argv[5]) : numBlocks;
printf("===== Before allocating blocks =====\n");
display_mallinfo2();
for (size_t j = 0; j < numBlocks; j++) {
    if (numBlocks >= MAX_ALLOCS) {
        fprintf(stderr, "Too many allocations\n");
        exit(EXIT_FAILURE);
    }
    alloc[j] = malloc(blockSize);
    if (alloc[j] == NULL) {
        perror("malloc");
        exit(EXIT_FAILURE);
    }
}
printf("\n===== After allocating blocks =====\n");
display_mallinfo2();
for (size_t j = freeBegin; j < freeEnd; j += freeStep)
    free(alloc[j]);
printf("\n===== After freeing blocks =====\n");
display_mallinfo2();
exit(EXIT_SUCCESS);
}

```

SEE ALSO

mmap(2), malloc(3), malloc_info(3), malloc_stats(3), malloc_trim(3), mallopt(3)