WIKIPEDIA

sbrk

brk and **sbrk** are basic memory management system calls used in Unix and Unix-like operating systems to control the amount of memory allocated to the data segment of the process.[1] These functions are typically called from a higher-level memory management library function such as `malloc`. In the original Unix system, `brk` and `sbrk` were the only ways in which applications could acquire additional data space; later versions allowed this to also be done using the `mmap` call.[2][3]

Contents

Description

The brk and sbrk calls dynamically change the amount of space allocated for the data segment of the calling process. The change is made by resetting the program break of the process, which determines the maximum space that can be allocated. The program break is the address of the first location beyond the current end of the data region. The amount of available space increases as the break value increases. The available space is initialized to a value of zero, unless the break is lowered and then increased, as it may reuse the same pages in some unspecified way. The break value can be automatically rounded up to a size appropriate for the memory management architecture.[4]

sbrk and brk were considered legacy even by 1997 standards (Single UNIX Specification v2 or POSIX.1-1998).[5] They were removed in POSIX.1-2001.[6]

Function signatures and behavior

```
#include <unistd.h>

int brk(void* end_data_segment);
void *sbrk(intptr_t increment);
```

sbrk is used to adjust the program break value by adding a possibly negative size, while brk is used to set the break value to the value of a pointer. Set *increment* parameter to zero to fetch the current value of the program break.

Upon successful completion, the brk subroutine returns a value of 0, and the sbrk subroutine returns the prior value of the program break (if the available space is increased then this prior value also points to the start of the new area). If either subroutine is unsuccessful, a value of −1 is returned and the errno global

variable is set to indicate the error.[5]

Not every Unix-like system entertains the concept of having the user control the data segment. The Mac OS X implementation of `sbrk` is an emulation and has a maximum allocation of 4 megabytes. On first call an area of exactly this large is allocated to hold the simulated segment. When this limit is reached, −1 is returned and the `errno` is set to ENOMEM. `brk` always errors.[7]

Error codes

The error ENOMEM is set and the allocated space remains unchanged if one or more of the following are true:

- The requested change allocates more space than is allowed by a system-imposed maximum.
- The requested change sets the break value to a value greater than or equal to the start address of any attached shared memory segment.

See also

- Exec (computing)
- Memory address#Address space in application programming

References

1. "Process Memory Concepts" (https://www.gnu.org/software/libc/manual/html_node/Memory-Concepts.html). Free Software Foundation. Retrieved 9 October 2015.
2. "A new malloc(3) for OpenBSD" (http://www.openbsd.org/papers/eurobsdcon2009/otto-malloc.pdf) (PDF). Retrieved 13 June 2018.
3. "POSIX Memory Management" (http://blog.pr4tt.com/2016/02/01/posix-memory-management/). blog.pr4tt.com.
4. X/Open CAE Specification, System Interfaces and Headers (http://pubs.opengroup.org/onlinepubs/9695969499/toc.pdf) (PDF) (2 ed.). X/Open Company Ltd., U.K. September 1994. p. 64. Retrieved 9 October 2015.
5. "brk, sbrk - change space allocation (LEGACY)" (https://pubs.opengroup.org/onlinepubs/7908799/xsh/brk.html). The Single UNIX ® Specification, Version 2. 1997. Retrieved 30 November 2019.
6. brk(2) (https://man7.org/linux/man-pages/man2/brk.2.html) – Linux Programmer's Manual – System Calls
7. "BRK.c" (https://opensource.apple.com/source/Libc/Libc-1272.250.1/emulated/brk.c.auto.html).