## NAME

brk, sbrk − change data segment size

## SYNOPSIS

**#include <unistd.h>**

**int brk(void *****addr***);**

**void *sbrk(intptr_t** *increment***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

**brk**(), **sbrk**(): _BSD_SOURCE || _SVID_SOURCE || _XOPEN_SOURCE >= 500

## DESCRIPTION

**brk**() and **sbrk**() change the location of the *program break*, which defines the end of the process's data segment (i.e., the program break is the first location after the end of the uninitialized data segment). Increasing the program break has the effect of allocating memory to the process; decreasing the break deallocates memory.

**brk**() sets the end of the data segment to the value specified by *addr*, when that value is reasonable, the system has enough memory, and the process does not exceed its maximum data size (see **setrlimit**(2)).

**sbrk**() increments the program's data space by *increment* bytes. Calling **sbrk**() with an *increment* of 0 can be used to find the current location of the program break.

## RETURN VALUE

On success, **brk**() returns zero. On error, −1 is returned, and *errno* is set to **ENOMEM**. (But see *Linux Notes* below.)

On success, **sbrk**() returns the previous program break. (If the break was increased, then this value is a pointer to the start of the newly allocated memory). On error, *(void *) −1* is returned, and *errno* is set to **ENOMEM**.

## CONFORMING TO

4.3BSD; SUSv1, marked LEGACY in SUSv2, removed in POSIX.1-2001.

## NOTES

Avoid using **brk**() and **sbrk**(): the **malloc**(3) memory allocation package is the portable and comfortable way of allocating memory.

Various systems use various types for the argument of **sbrk**(). Common are *int*, *ssize_t*, *ptrdiff_t*, *intptr_t*.

### Linux Notes

The return value described above for **brk**() is the behavior provided by the glibc wrapper function for the Linux **brk**() system call. (On most other implementations, the return value from **brk**() is the same; this return value was also specified in SUSv2.) However, the actual Linux system call returns the new program break on success. On failure, the system call returns the current break. The glibc wrapper function does some work (i.e., checks whether the new break is less than *addr*) to provide the 0 and −1 return values described above.

On Linux, **sbrk**() is implemented as a library function that uses the **brk**() system call, and does some internal bookkeeping so that it can return the old break value.

## SEE ALSO

**execve**(2), **getrlimit**(2), **end**(3), **malloc**(3)

## COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.