14 Memory Allocators

14.1 Allocation Problem
14.2 Allocator Interface
14.3 The Purely Sbrk Solution
14.4 The Heap Solution
14.5 Organizing A Region
14.6 Gluing Regions Together
14.7 Invariants (Rules)
14.8 Allocate First-Order Pseudo-Code
14.9 Free First-Order Pseudo-Code
14.10 Best Fit or First Fit

14.1 Allocation Problem

Problem: To allocate blocks of memory, then free some of them, and reuse the freed memory.

Restriction: Once allocated a block cannot be moved.

14.2 Allocator Interface

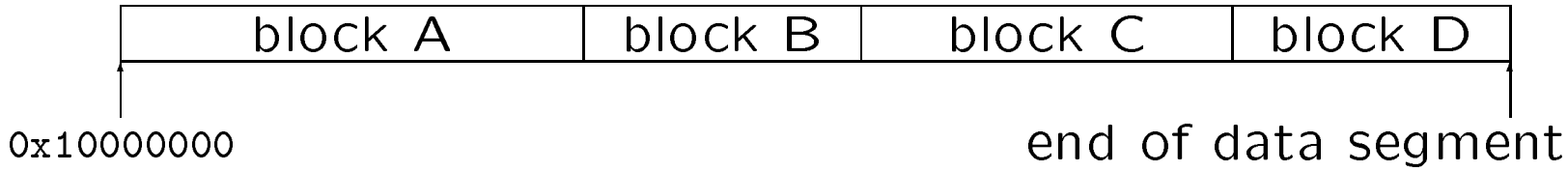
Definition: A block is a contiguous set of zero or more bytes. Blocks are denoted by the address of their first byte. No two blocks have the same address. A zero length block has an a unique address but no bytes.

Allocate Operation: Given a block size in bytes, allocate a block of equal or larger size and return its address.

Free Operation: Given the address of a block, make the memory occupied by the block available to future allocate operations.

14.3 The Purely Sbrk Solution

The Data Segment:

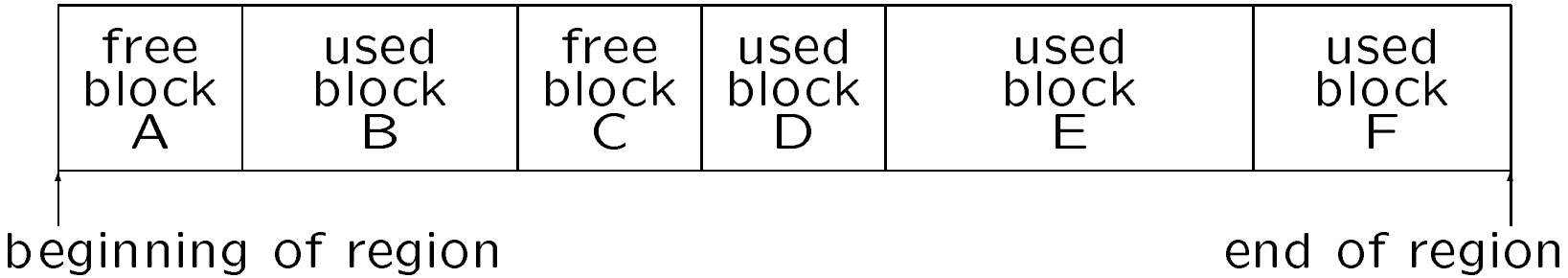


Allocate blocks to end of data segment.

Never free blocks.

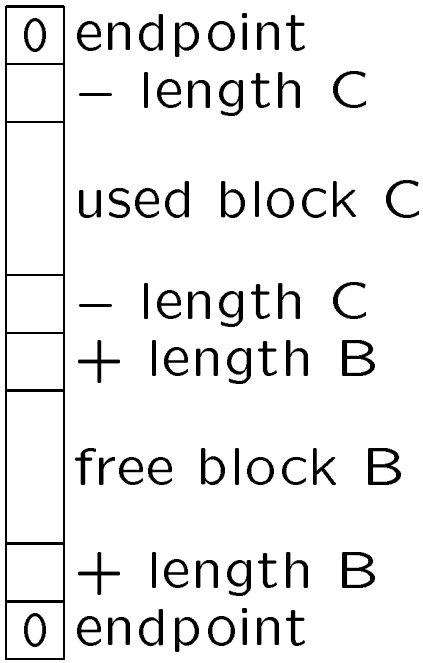
14.4 The Heap Solution

Allocate biggish regions of memory using the sbrk method, and then suballocate each of these regions:



If a block to be allocated fits inside a free block, divide the free block into a new used block and a free reminder block with whatever is left over.

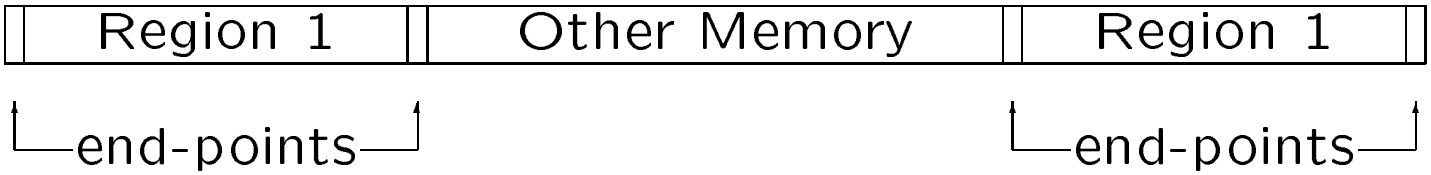
14.5 Organizing A Region



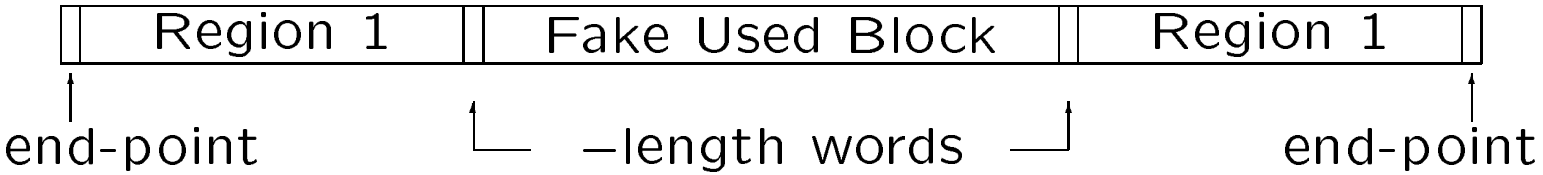

```
region ::= end-point block* end-point
end-point ::= 0 word
block     ::= free-block | used-block

free-block ::= length space length
used-block ::= -length data -length
length ::= word
space  ::= word*       data ::= word*
```

A `length` is the length of a block in bytes including the length words at the end of a block (so each length ≥ 8).

14.6 Gluing Regions Together



is changed to:



14.7 Invariants (Rules)

Blocks are 2 or more words long, including the length words at each end.

A length word contains the length of a block in bytes, including the length words, negated if the block is used.

Length words occur in matched pairs.

Free blocks are never adjacent to each other. They are merged when freed.

A zero word is placed at each end of a region to mark the ends of the region.

There is only one region. Two regions are glued together with a fake used block including the intervening space.

14.8 Allocate First-Order Pseudo-Code

(1) Round size up to multiple of 4 bytes and add 8.
(2) Search for free block of $\geq$ size.
(2a) If no free block found, create a new region large enough to hold the new block.
(2b) If a free block found less than 8 bytes larger than requested size, return free block.
(2c) If a free block found 8 bytes or more bytes larger than requested size, divide the free block into a block of the requested size to be returned, and a smaller free block.

14.9 Free First-Order Pseudo-Code

(1) Given the address of the block to free, find its length in the previous word.
(2) If block being freed is adjacent to an already free block, glue the blocks together.

General Rule: Whenever you read a length word of a block, read the matching length word at the other end of the block and check for equality. If <u>not</u> equal, memory has been damaged!

14.10 Best Fit or First Fit

If there are several free blocks large enough, which one should allocate use?

Best Fit: Use the smallest one that will work.

First Fit: Use the first one in order of increasing address.

The first fit algorithm often works slightly better because it tends to group free blocks at the end of a region, where they tend to merge.