Initial address space:
    * pages for code and globals
    * a few (empty) heap pages
    * a page for the initial stack frame

The stack is extended down automatically

The heap is extended only when the process asks for more pages
    * standard allocators (malloc/new) do this for you

How are freelists on the heap organized?

How are objects allocated?

How are objects deallocated?

What happens if the free list is empty?

Most OSes provide other mechanisms to manipulate virtual address spaces

* mmap(filename, addr, pages)

* mprotect(addr, pages, protection)

You can use this to find dynamic storage bugs!

Most common bugs:
- * write off end of object
- * use-after-free
- * double-free

Less common bugs:
- * random pointer dereference
- * write off beginning of object

How can you use mmap/mprotect to detect all common bugs and many less common ones?

* malloc

* free

* what happens for each of these?

    * write-off-end

    * use-after-free

    * double-free

What are the performance implications of this scheme?