生成新的节点

- x=NewNode(key, height)
- 这里需要分配内存

内存分配模块的实现

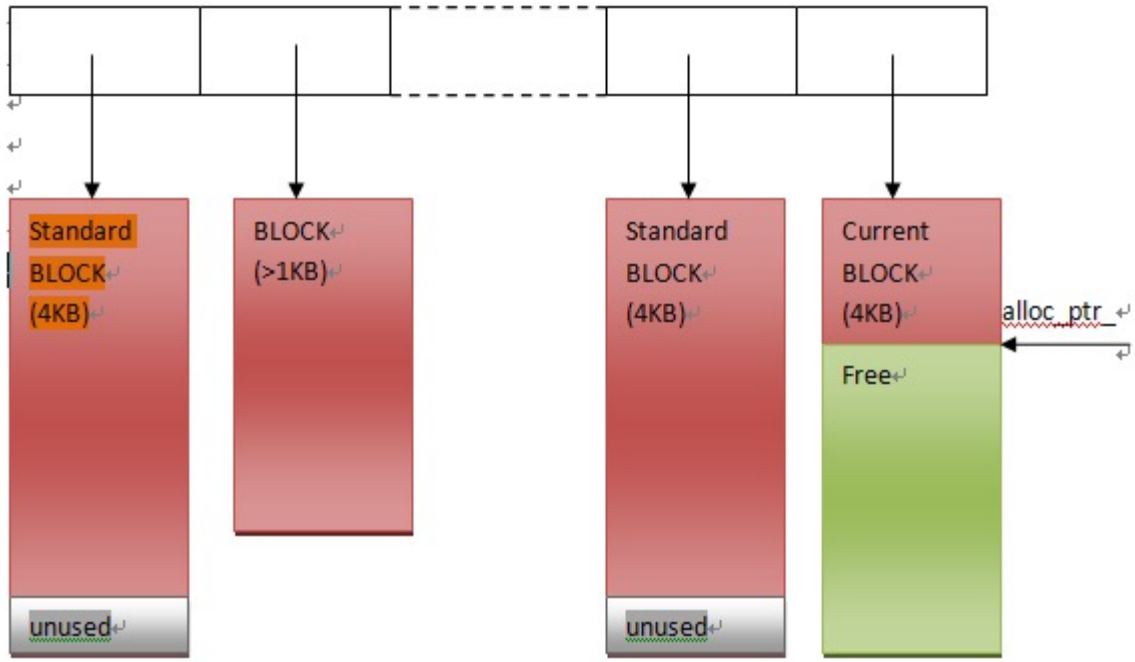
- 为何使用统一内存管理
  - 避免new/delete多次的申请和释放引起的内存碎片。
- 一次申请大块的内存，多次分给客户

内存管理模块(Arena.h)

- std::vector<char*> blocks_;
  - 来保存所有的内存分配记录
- char* Arena::Allocate(size_t bytes)
  - 普通分配
- char* Arena::AllocateAligned(size_t bytes)
  - 对齐分配

Allocate函数

- 默认每次申请4k的内存，记录下剩余指针和剩余内存字节数，每当有新的申请，如果当前剩余的字节能满足需要，则直接返回给用户，如果不能，对于超过1k的请求，直接new返回，小于1K的请求，则申请一个新的4k块，从中分配一部分给用户。



AllocateAligned函数

- 什么是内存对齐
  - 地址对齐
    - 首地址是4或者8的倍数
  - 大小对齐
- AllocateAligned函数做到地址对齐

AllocateAligned函数

```
char* Arena::AllocateAligned(size_t bytes) {
  const int align = (sizeof(void*) > 8) ? sizeof(void*) : 8;
  assert((align & (align-1)) == 0);   // Pointer size should be a power of 2
  size_t current_mod = reinterpret_cast<uintptr_t>(alloc_ptr_) & (align-1);    //计算8的余数，alloc_ptr_表示当前地址
  size_t slop = (current_mod == 0 ? 0 : align - current_mod);
  size_t needed = bytes + slop;
  char* result;
  if (needed <= alloc_bytes_remaining_) {
    result = alloc_ptr_ + slop;
    alloc_ptr_ += needed;
    alloc_bytes_remaining_ -= needed;
  } else {
    // AllocateFallback always returned aligned memory
    result = AllocateFallback(bytes);
  }
  assert((reinterpret_cast<uintptr_t>(result) & (align-1)) == 0);
  return result;
}
```