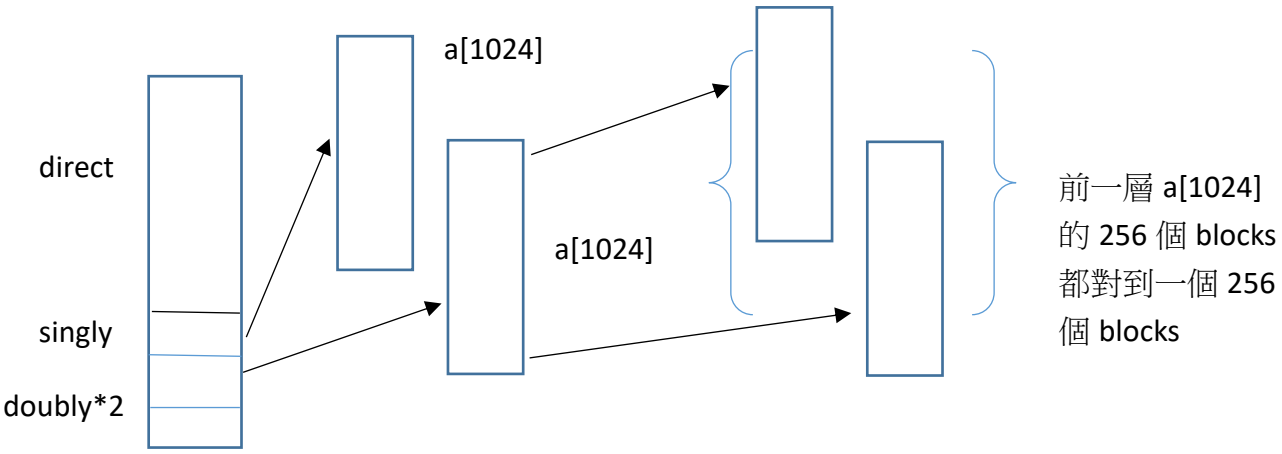
1.

關於擴大 file 的 size 到 66666 這件事，原本 xv6 是 12 個 direct block+1 個 singly direct block 所以是 12+256=268。我極為奢侈的直接用了 2 個 doubly direct block 來保證測資不會太奇怪，這樣就有至少 100000 個 blocks 了。我的 addrs 大小還是 13，10 個 direct，1 個 singly direct，2 個 doubly direct block 這樣分配。所以我的 addrs 的狀況如下圖:(8 個 uchar 換成 1 個 uint 所以 a[1024]有 256 個 blocks)



跟上面一樣是 doubly 所以結構一樣

基本上 bmap 就是觀察 singly 的做法然後把 doubly 的做法模仿出來。只要確定好接下來對應到 256 blocks 中的哪個就可以連接下去了。Itrunc 的部分不用想太多，模仿 singly 的做法然後把每層的 256 個 blocks 都 bfree 就好了，不用考慮說對應到哪個 blocks。

2.

By guideline 第 6 點，首先拿到參數，然後用 begin_op 啟用 fs，用 create 把 path 的 inode 資料建立起來，然後再把該寫的資料寫到 path 的 inode 裡面(target)，最後用 iupdate 更新資料，put 更新 reference，end_op 結束 fs 的使用，這樣就成功把 path 跟 target 連接起來了。open 的部分要檢查 ip->type 是不是 symlink 且 omode 不是 O_NOFOLLOW，都符合就要處理，這時需要不斷去用 readi 讀取 inode 的資料，再用 namei 更新 ip 的 inode，如果讀太多次，那就是有 cycle 的狀況(count>10)，就需要跳出來。直到 ip->type 不再是 symlink 就繼續下去。不存在代表 namei 讀不到東西所以 return -1。

3.

Namex 尋找 inode 要處理 symlink，如果 ip->type 是 symblink 的話，就要用 readi 把真正的 path 找出來然後繼續執行下去。當然如果有 cycle 的話(count>10)還是要跳出來。chdir 的部分跟 namex 幾乎一樣，都是 ip->type 是 symblink 的話，就要用 readi 把真正的 path 找出來然後繼續執行下去。當然如果有 cycle 的話(count>10)還是要跳出來。