super_block	group_desc	block_bitmap	inode_bitmap	inode[0]	……	inode[32]
glock	glock	block	ilock	inodelock[0]		inodelock[32]

1. When I modify content of super_block or group_desc, I will lock(glock). After I have modified, I will unlock(glock).
2. When I want to request or free a block, I will lock(block). I just set corresponding bit from 0 to 1, or from 1 to 1. Then I will unlock(block).
3. When I want to request or free an ilock, I will lock(ilock). I just set corresponding bit from 0 to 1, or from 1 to 1. Then I will unlock(ilock).
4. When I modify the content of an Inode i, I will lock(inodelock[i]). After I have completed modification, I will unlock(inodelock[i]).
5. Procedure of ext2_cp():
   a) lock(glock)
   b) Judge if arguments are correct, if not, unlock(glock) return.
   c) Judge if there are free inode and block, if not, unlock(glock) return.
   d) lock(block), get a block no, set block_bitmap, unlock (block)
   e) lock(ilock), get an inode no, set inode_bitmap
   f) modify inodelock[i] and corresponding block info
   g) unlock (ilock)
   h) lock(glock), modify s_free_blocks_count, s_free_inodes_count…
   i) unlock(glock)

6. ext2_mkdir() and ext2_rm() are similar to ext2_mkidr()

7.To ensure efficient synchronization:

a) I only lock super_block and group_desc to update some integers at the end of ext2_cp(), it should be very fast.

b) I will lock corresponding inode lock when I copy data, and unlock it after I finished. For I only lock one of these locks, so it should not affect other people to use file system.

c) I only lock a block to set some bit of block_bitmap, that should be very fast. After that, I modify data on these blocks, because they have been set as ‘occupied’. So, no other thread will get them, and cannot modify them at the same time. And after saving data on these blocks, I will unlock.