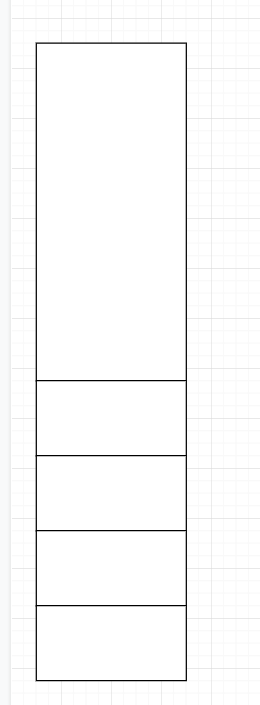
file management是OS的一部分，用来组织抽象化FILE

Allocation Strategies for File Blocks

1.continuous strategy



所有的block一个接一个

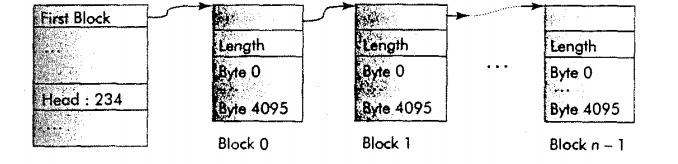
优势：容易与file保持联系keep contract:你只需要这一串block开头，以及block size，然后你就可以进入对应block

2.快：O(1) to go to block f:go to block 1 and calculate block size to go to block f

劣势:即使有enough space，也不见得是连续的space，那么你就无法放下这一串block

|  |  |  |  |
| --- | --- | --- | --- |
|  | 11 |  | 1 |
|  | 1 |  | 1 |
|  | 1 |  | 1 |
|  | 1 |  | 11 |
|  | 1 |  | 1 |

linked list strategy

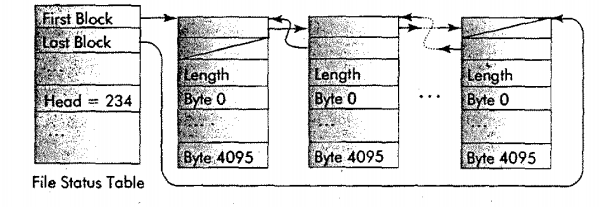


每个block都有对下一个block的pointer

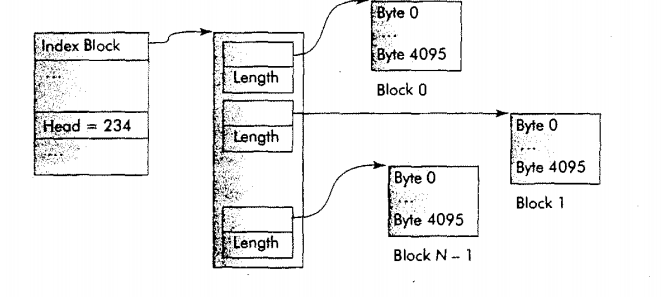
但我们需要第一个来行来存储来存储对第一个block的pointer

64位Lost 8byte ,

缺点O（n） to get block n，如果你lost了一个Pointer,就lost 了剩下的所有file  
解决方案，double linked list

如果lost，可以回去，重新回来，但是多存一个回来的linked list，更慢

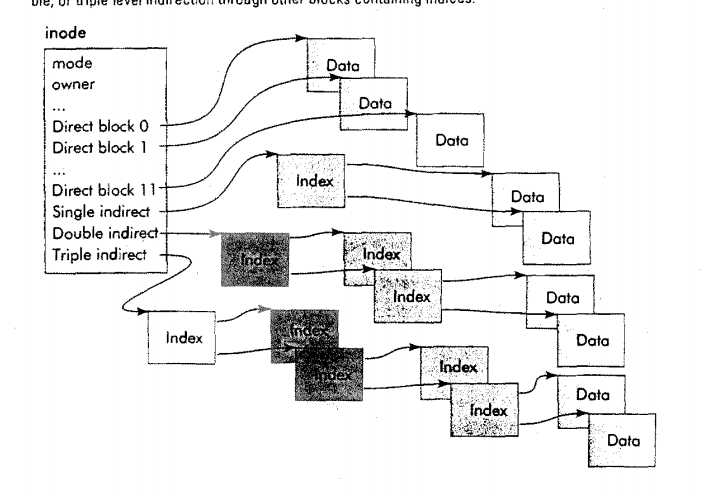
indexed strategy



每一个file都有一个index table，每个pointer指向一个block，speedO(1),如果一个table不够用，最后一格存下一个table的pointer

缺陷，会造成很多浪费如果你有很多小file，因为index table的size是固定的

最好的是Unix file



0到11各存一个block

12开始存一个pointer block，假设一个pointer长度为4 byte//32 bit，而一个block为1k默认，那么12这里能存储1024/4=256个BLOCK

13就是256\*256个block

14就是256\*256\*256个block

complexity O(1)即使用第四阶，你也只需要用四次linked list

所以总共能存储的容量：0到11等于block size

12，block/pointer\*block size

13.block/pointer\*block/pointer\*block

能支持最大4000G file，即使超过4000G，加一个entry又能够用了

注意计算大小只用data block没必要管index block。

unallocated file system:

还没分配位置的file system（暂时为空的file system）

1.linked list

把所有的空block unallocated block用linked list连接起来。

lost counter没关系，找下一个empty block，又重新连上了

问题，他从一开始就会占据过多hard disk,记录他的index table也很大，占据过多hard disk

2.bit map

他直接记录下一个block是不是free. 10 1 01 01 10 1这种，即使文件很大，这个也很小