COMP 282 Lecture: 10/13/03
Advanced Data Structures

Announcements and Course Details

- Wiegley isn't here today
- He will be back tomorrow
- Schedule remains the same
- Southern California Linux Expo
  www.socallinuxexpo.com 11/22/03
- Please make fun of how Wiegley spells CACHING (cacheing)
- Questions?

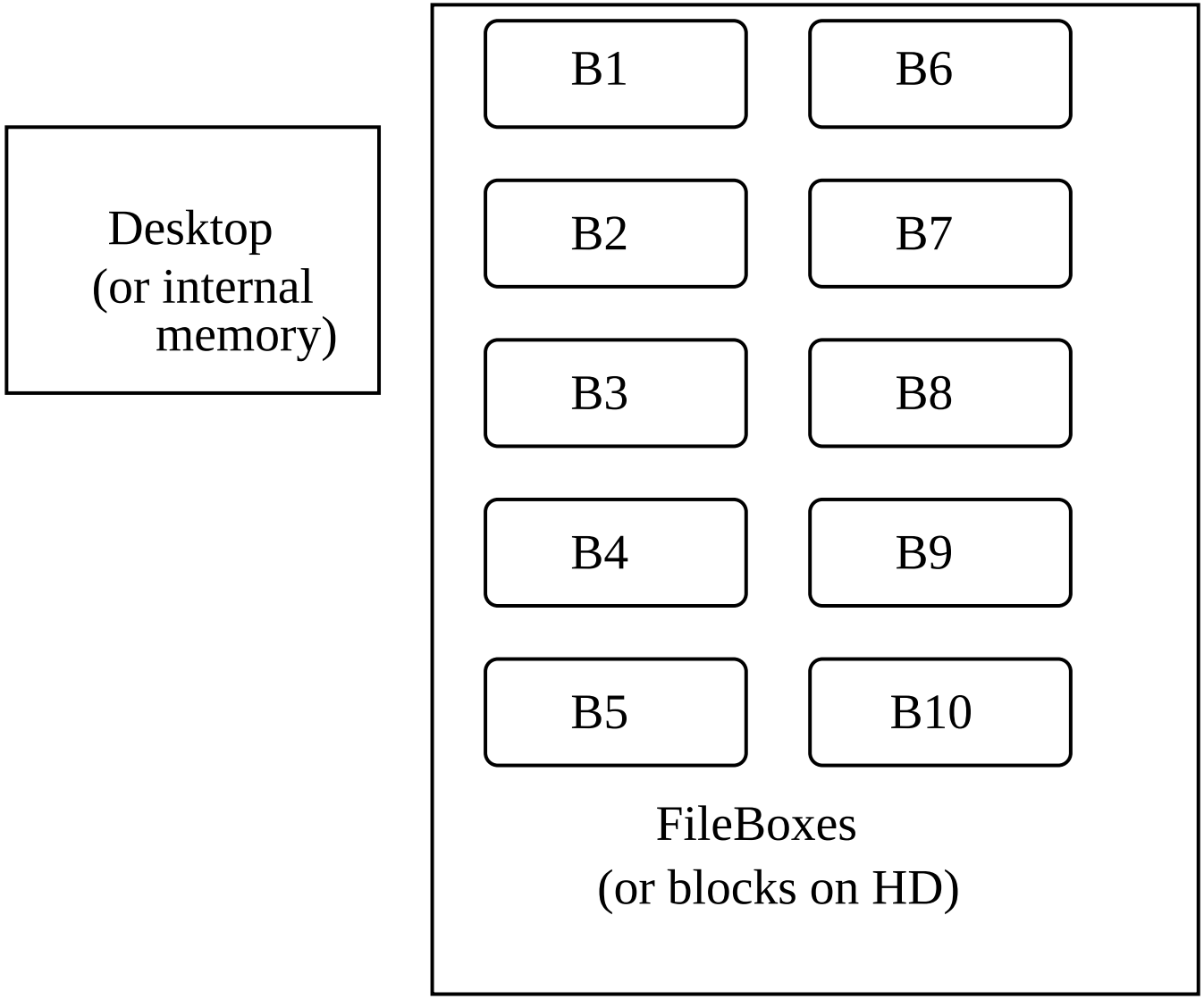
External File

- In real life data sets are ofter very large
- We often cannot store them in internal memory
- Even more often we don't want to
- Still want to be able to insert/delete/find quickly

External File - Analogy

- Each semester we teach over 100 students
- After 5 years we will have:
- Over 1000 student files
- 10 file boxes (1 for each semester)
- We could search by going through each box (really slow)
- We could alpha the students and bisection search (work and slow)
- Or we could keep an index (what box to search)

Drawing of External File



Dumb Attempt - Pretend Vector

- Figure out how many records will fit in a "block"
- Store that many items in a bunch of "consecutive" blocks
- Search: load a block, linear search the block, repeat
- Insert: (if room) load last block, put item in, write back
  (if no room) start new block
- Delete: load a block, linear search the block, write back, repeat
- Time Complexities? Count number of block loads?

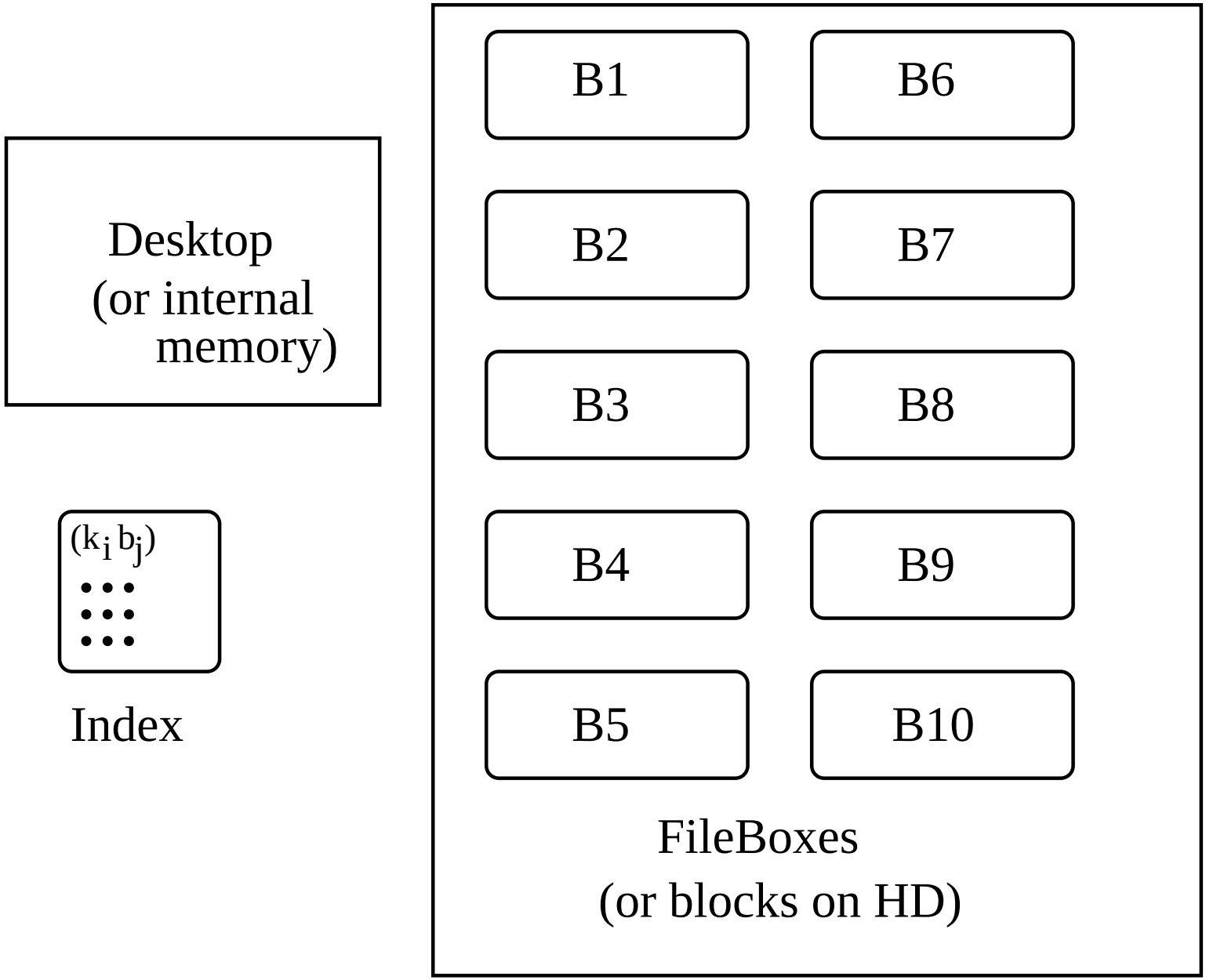
Dumb Attempt 2 - Pretend Ordered Vector

- Figure out how many records will fit in a "block"
- Totally sort the items
- Store items sorted in a bunch of "consecutive" blocks
- Search: load middle block, decide before/in/after (bisection)
- Insert: search to find where, shift
- Delete: search to find, shift back?
- Time complexities? Count number of block loads?

Indexing an External File (not dumb, but not end-all/be-all)

- Figure out how many record will fit in a "block"
- Put that many items in a bunch of "consecutive" blocks
- Keep an index of where each item is (hopefully small)
- Search: check index for block, search block
- Insert: put in last block, modify index
- Delete: check index for block, search block, modify index
- Index could be vect, ord vect, hash table
- Index order of magnitude smaller
- Time Complexities? Count number of block loads?

Drawing of Indexing Scheme



What if the index is too large for internal memory

External Hashing (pretty smart, but complicated to do)

- Create array that fits in internal memory
  this will be a hash table (with chaining of blocks!?)
- Break index into blocks
  within block must have same hash value
- Search: hash key, check index block by block, load block given by index
- Insert: hash key, go to end of ll, add to index, and block of file
- Delete: search, modify block of index, modify block of file
- Time Complexities? Count number of block loads?

Drawing of External Hashing