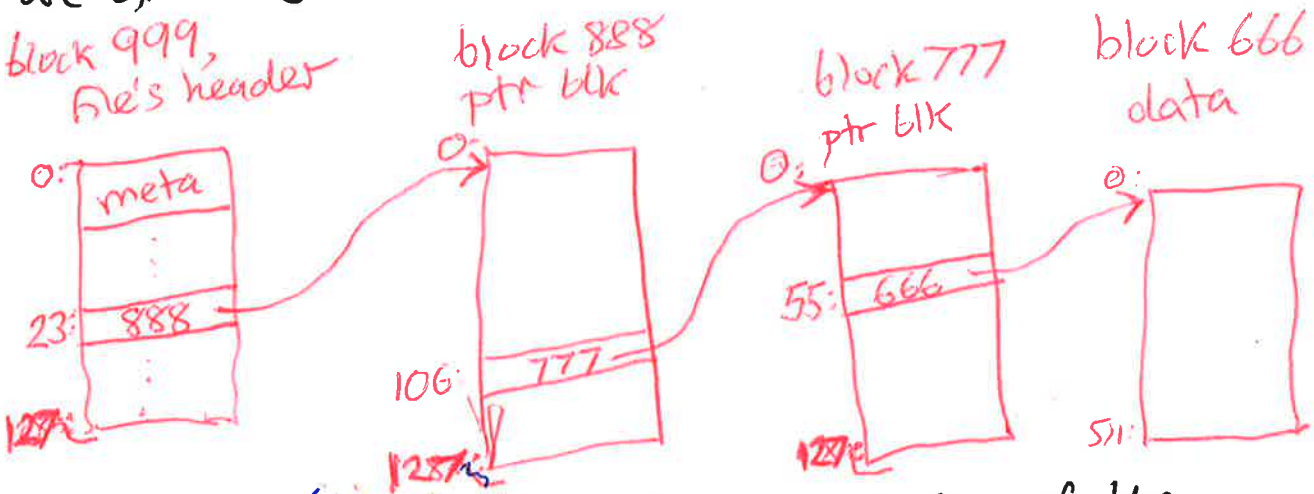
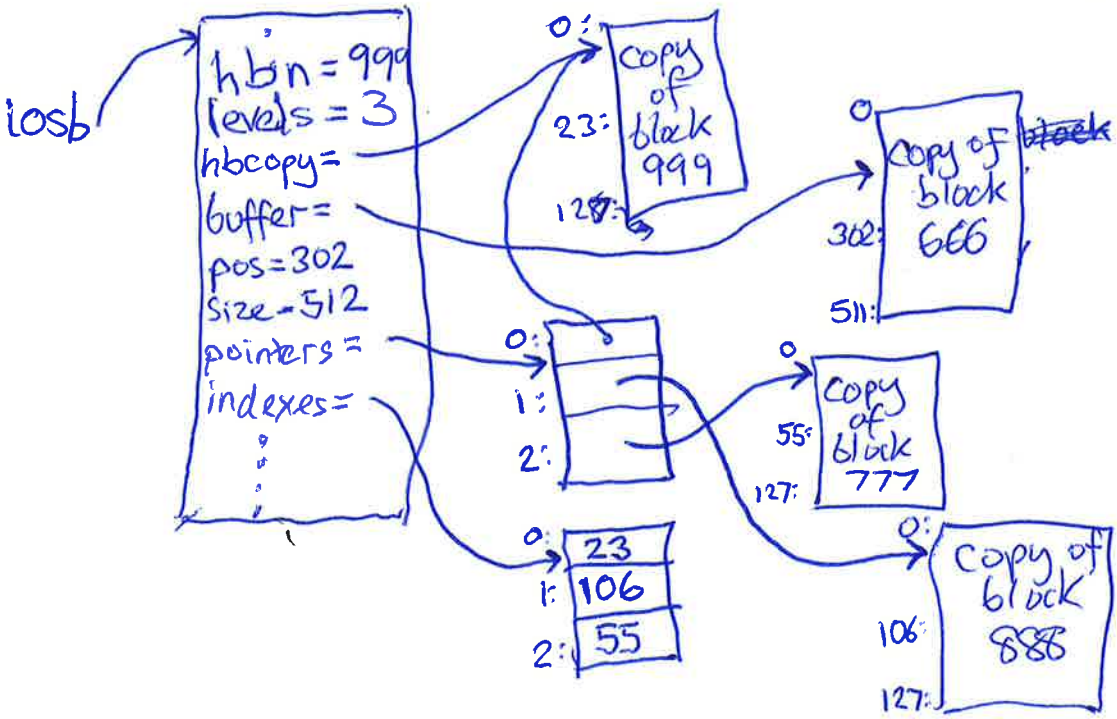
if you have a three level index file open, then on disc (red) the header block will contain block numbers for the first level of pointer blocks, which will contain blk. nums for the second lev. of ptr. blks, which will contain blk. nums for data blocks. At any one time only four things will be relevant – those on the path to the current data block. Block numbers are effectively pointers on a disc.



in memory (blue) it is best to keep a copy of the contents of all four blocks, plus the iosb. Addresses (not written) are pointers in memory.



The two little vectors are

pointers: where to find the in-memory copy of any block that contains the useful block numbers

indexes: the positions in those blocks that are currently relevant to us.

There are of course many variations on this basic method.

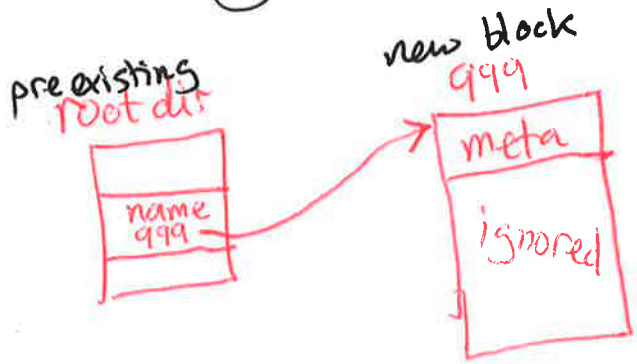
The indexes 23, 106, 55, 302 mean that we are currently at byte 199,913,262 of the file -

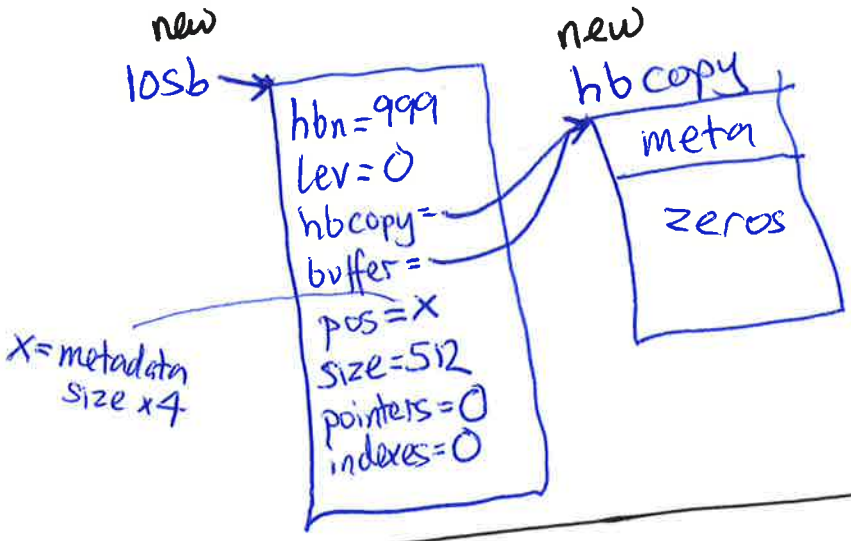
*** 23 × 128 × 128 × 512 + 106 × 128 × 512 + 55 × 512 + 302

The 23 should be 13, I forgot to subtract the size of the metadata.

~~writing a byte:~~

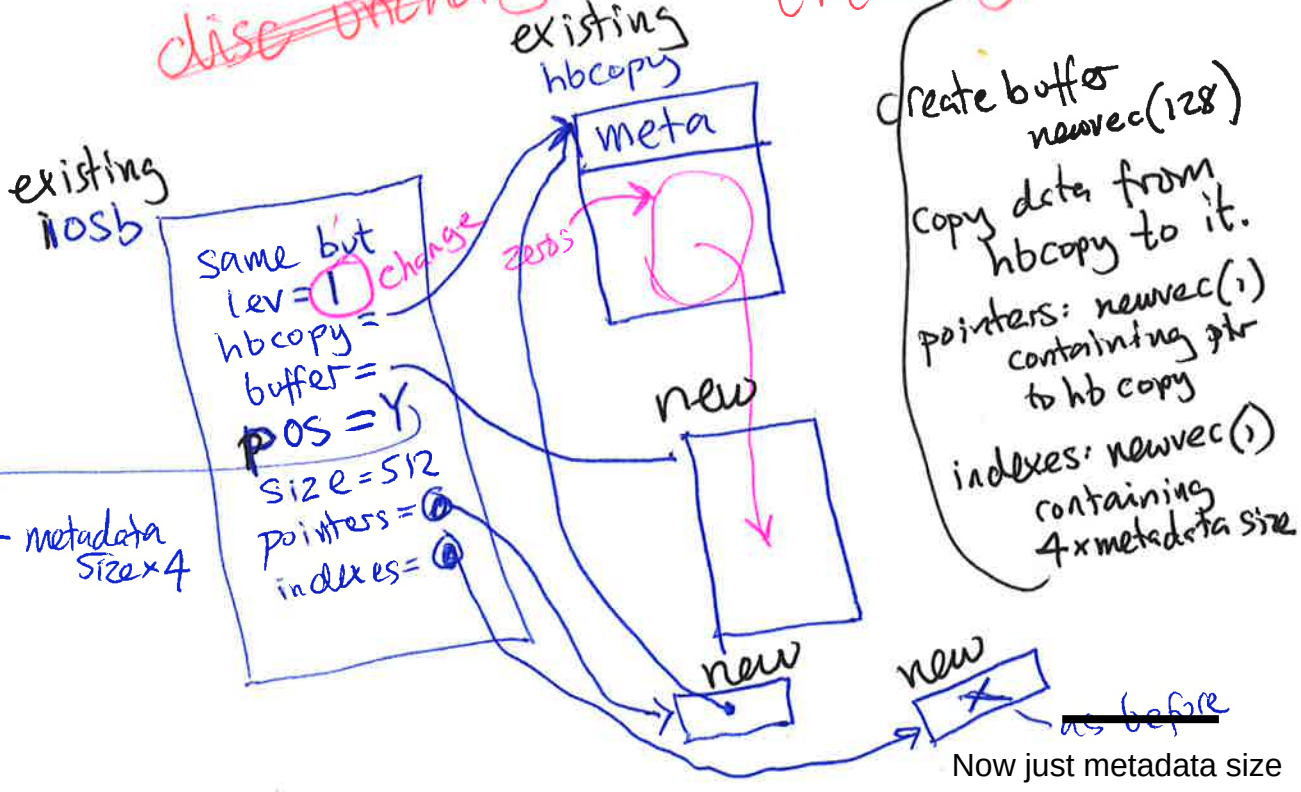
immediately on creating = opening for writing





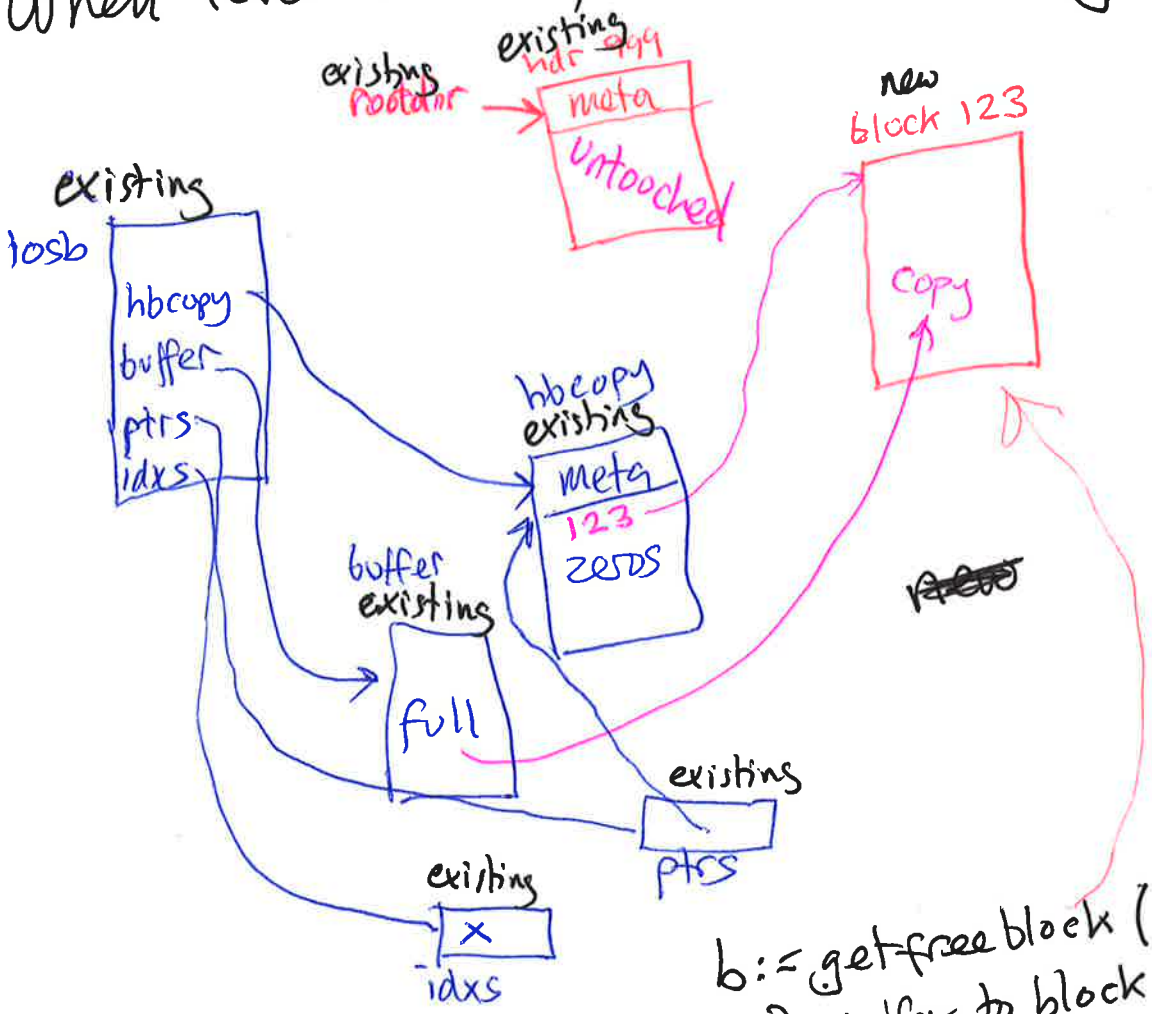
When Level 0 and pos = size, grow to level 1

~~disc unchanged~~ disc unchanged

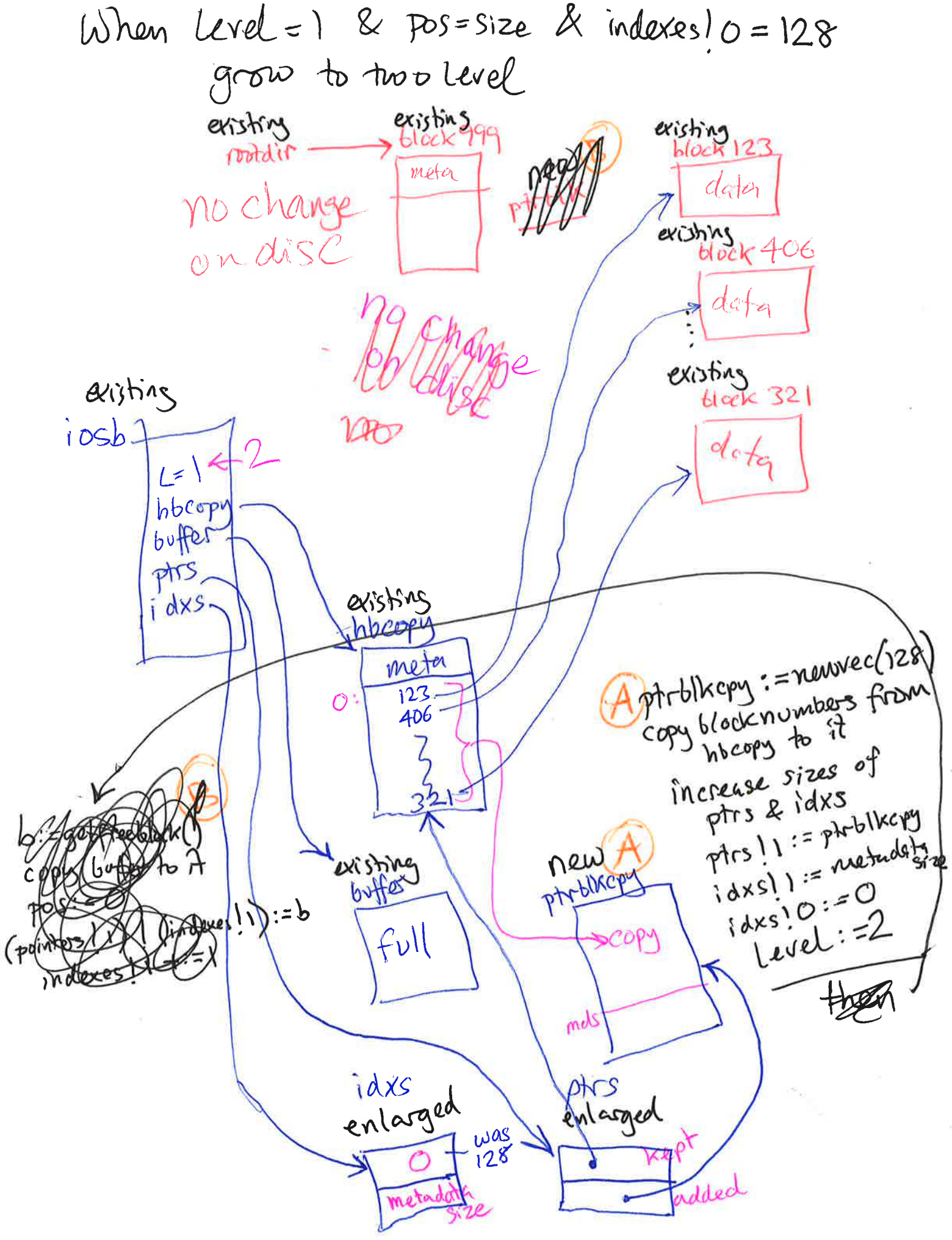


Now just metadata size

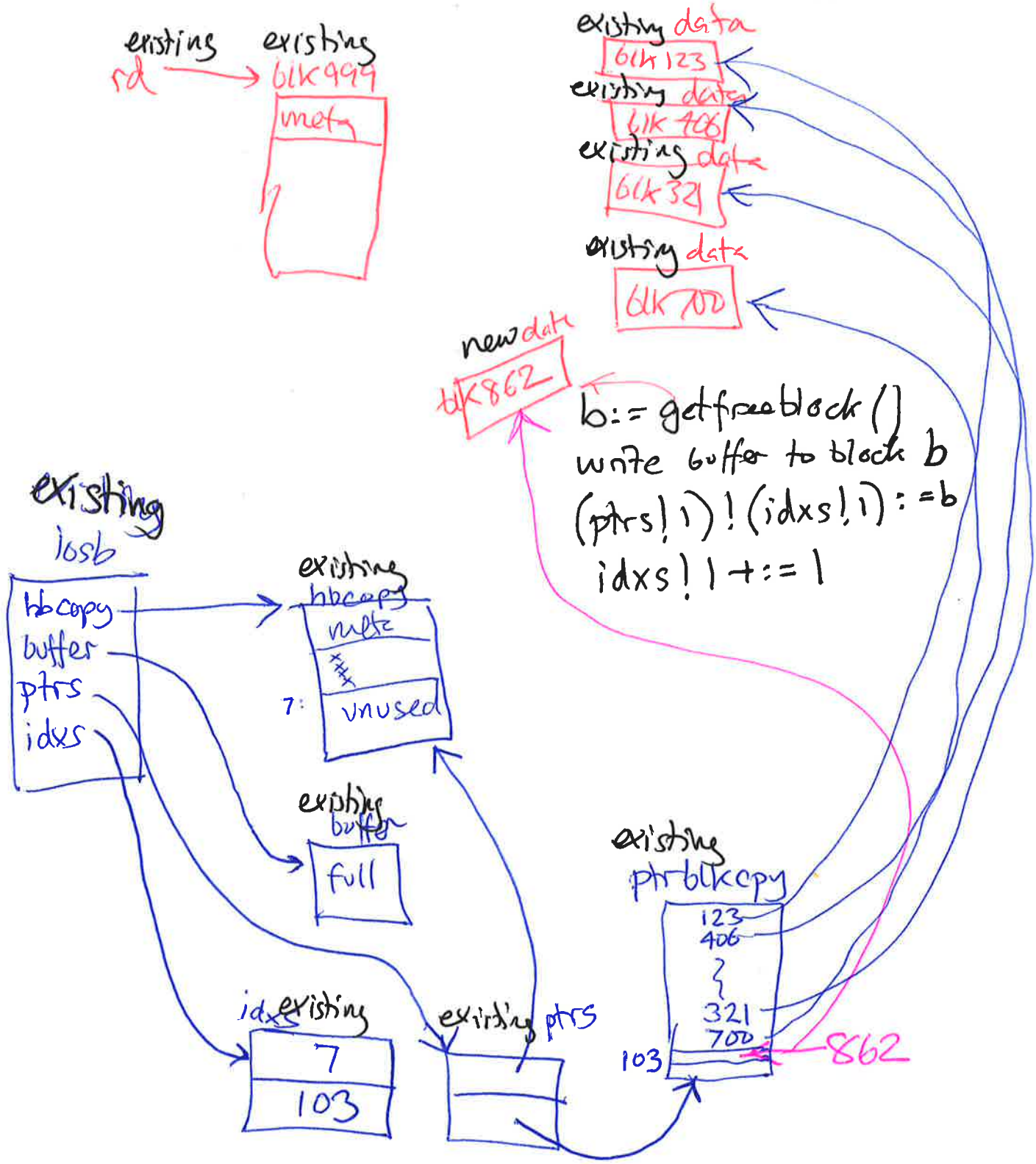
When level 1 and pos = size normally



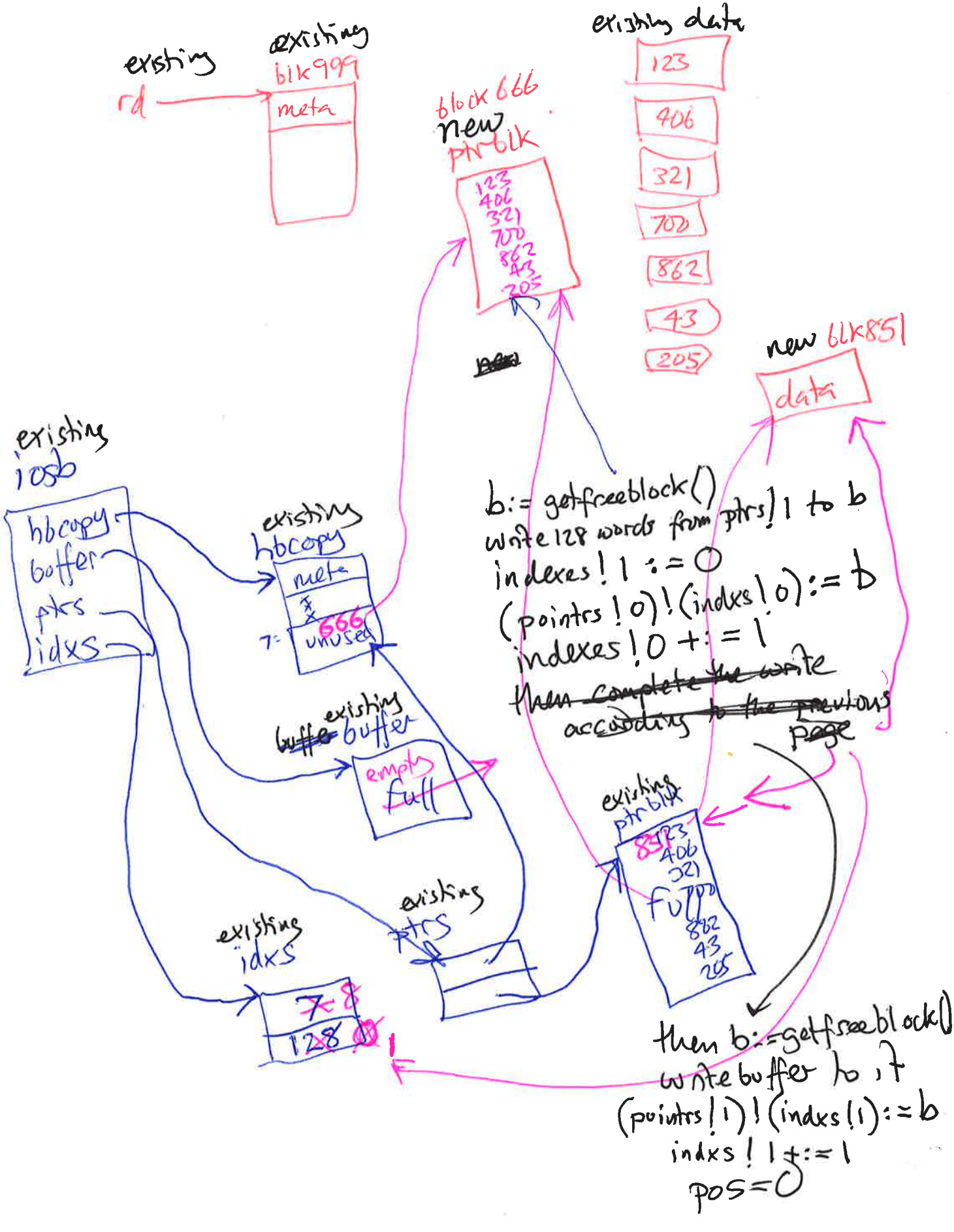
b := getfreeblock()
write buffer to block b
(pointers!0)!(indexes!0) := b
indexes!0 +:= 1
pos := 0


When Level = 1 & pos = size & indexes!0 = 128
grow to two level
existing rootdir
existing block 999
meta
no change on disc
new ptrblk
existing block 123
data
existing block 406
data
existing block 321
data
no change disc
existing iosb
L=1 ← 2
hbcopy
buffer
ptrs
idxs
existing hbcopy
meta
0:
123
406
321
A ptrblkcopy := newvec(128)
copy blocknumbers from hbcopy to it
increase sizes of ptrs & idxs
ptrs!1 := ptrblkcopy
idxs!1 := metadata size
idxs!0 := 0
Level := 2
then
B
b := getfreeblock()
copy buffer to b
pos := 0
(pointers!1 := b; indexes!1 := 1)
existing buffer
full
new A
ptrblkcopy
copy
mds
idxs enlarged
0
metadata size
was 128
ptrs enlarged
kept
added

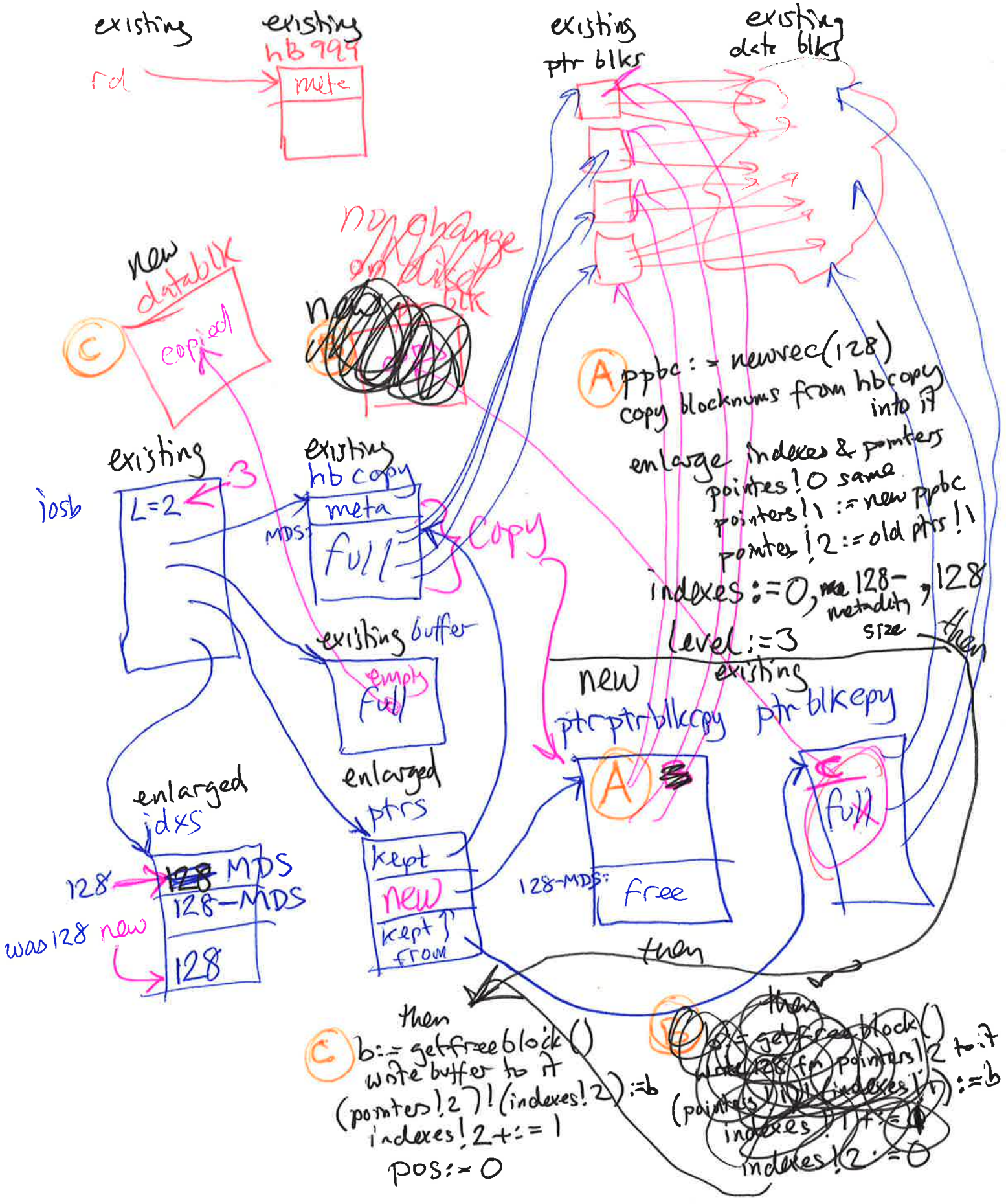
When level = 2 & pos = size normally

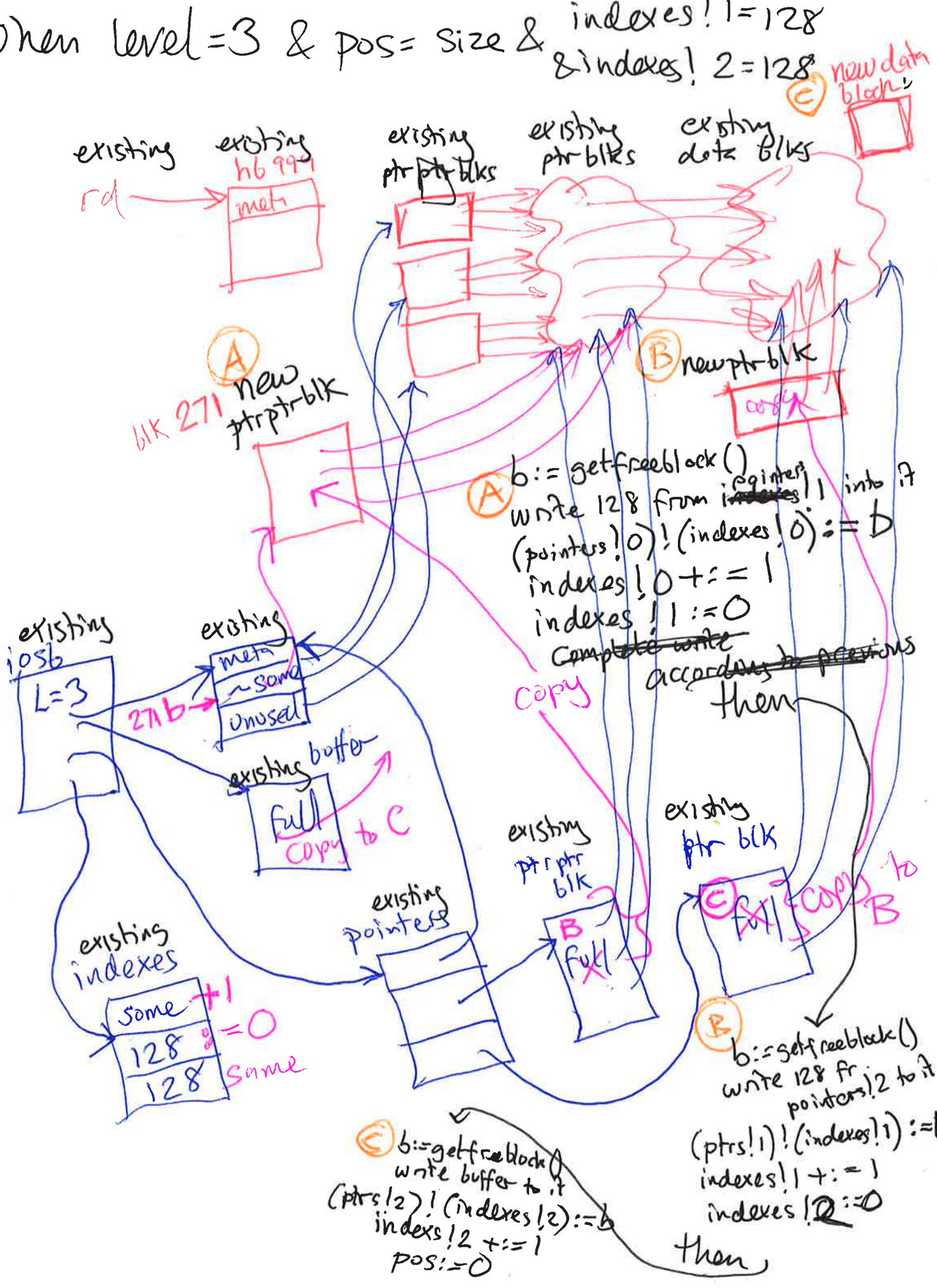


When level=2 & pos=size & indexes!1=128



When level=2 & pos=size & indexes!1 = 128 & indexes!0 = 128 grow to level 3




When level=3 & pos= size & indexes!1=128 & indexes!2=128
existing
rd
existing
hb 999
meta
existing
ptr ptr blks
existing
ptr blks
existing
data blks
C
new data block
A
new ptr ptr blk
blk 271
B
new ptr blk
copy
A
b:=getfreeblock()
write 128 from pointers!1 into it
(pointers!0)!(indexes!0):=b
indexes!0+:=1
indexes!1:=0
Complete write
according to previous
then
copy
existing
iosb
L=3
271 b
existing
meta
some
unused
existing buffer
full
copy to C
existing
indexes
some +1
128 =0
128 same
existing
pointers
existing
ptr ptr
blk
B
full
existing
ptr blk
C
full
copy to B
B
b:=getfreeblock()
write 128 fr pointers!2 to it
(ptrs!1)!(indexes!1):=b
indexes!1+:=1
indexes!2:=0
then
C
b:=getfreeblock()
write buffer to it
(ptrs!2)!(indexes!2):=b
indexes!2+:=1
pos:=0

on close (level 3)

Update header block copy info

write hb copy to block iosb!hbn

for i = 0 to level-2 (i.e. #)

write all 128 words from where pointers!(i+1) points

to block ~~(pointers!(i-1))~~

(pointers!i)!(indexes!i)

Free everything

Reading a byte - (level 3)

Normally: c := byte pos of buffer, pos +:= 1, result is c

but if pos = size:

if we're on the last block signal EOF.

otherwise first do this—

Normally ~~bzzz pois~~
indexes!2 +:= 1
b := (pointers!2)!(indexes!2)
read block b into the buffer
pos := 0
size := 512 unless very close to end of file

But if indexes!2 = 127 do this first:

Normally indexes!1 +:= 1
b := (pointers!1)!(indexes!1)
read blk b to 128 words from pointers!2
indexes!1 +:= 1
indexes!2 := 0

but if indexes!1 = 127 do this first

Normally indexes!0 +:= 1
b := (pointers!0)!(indexes!0)
read blk b to pointers!1
indexes!0 +:= 1
indexes!1 := 0

but if indexes!0 = 127:
Can't happen.

Closing a file that is open for writing:

Apart from the usual stuff:
~~write buffer to block (pointers!2)!~~
do all of the above, but always take the "but if" option regardless of the conditions.