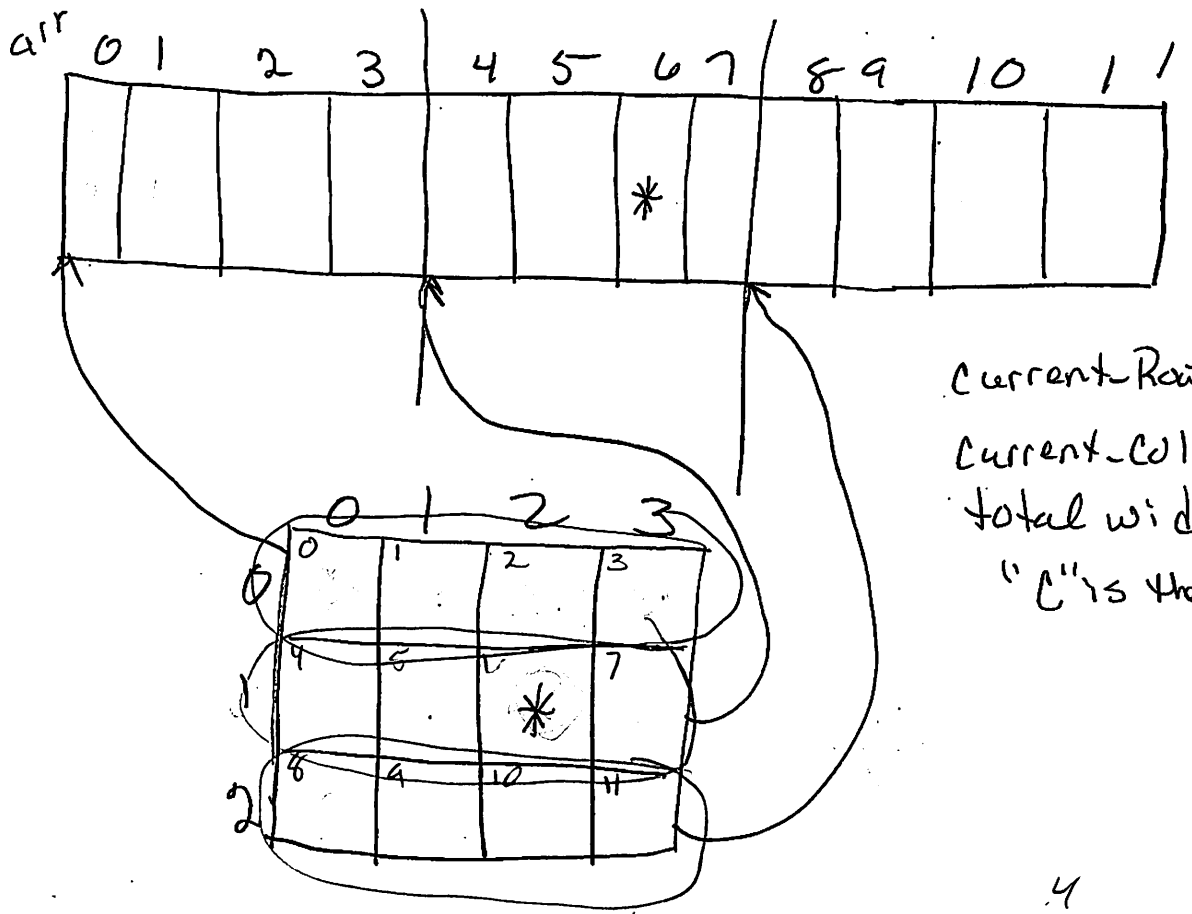
malloc1.c

Accessing 1D array as if it were a 2D.

r = 3 c = 4



current_Row = 1
current_col = 2
total width = 4
"C" is the width

formula: (current_row * total width) + current_col

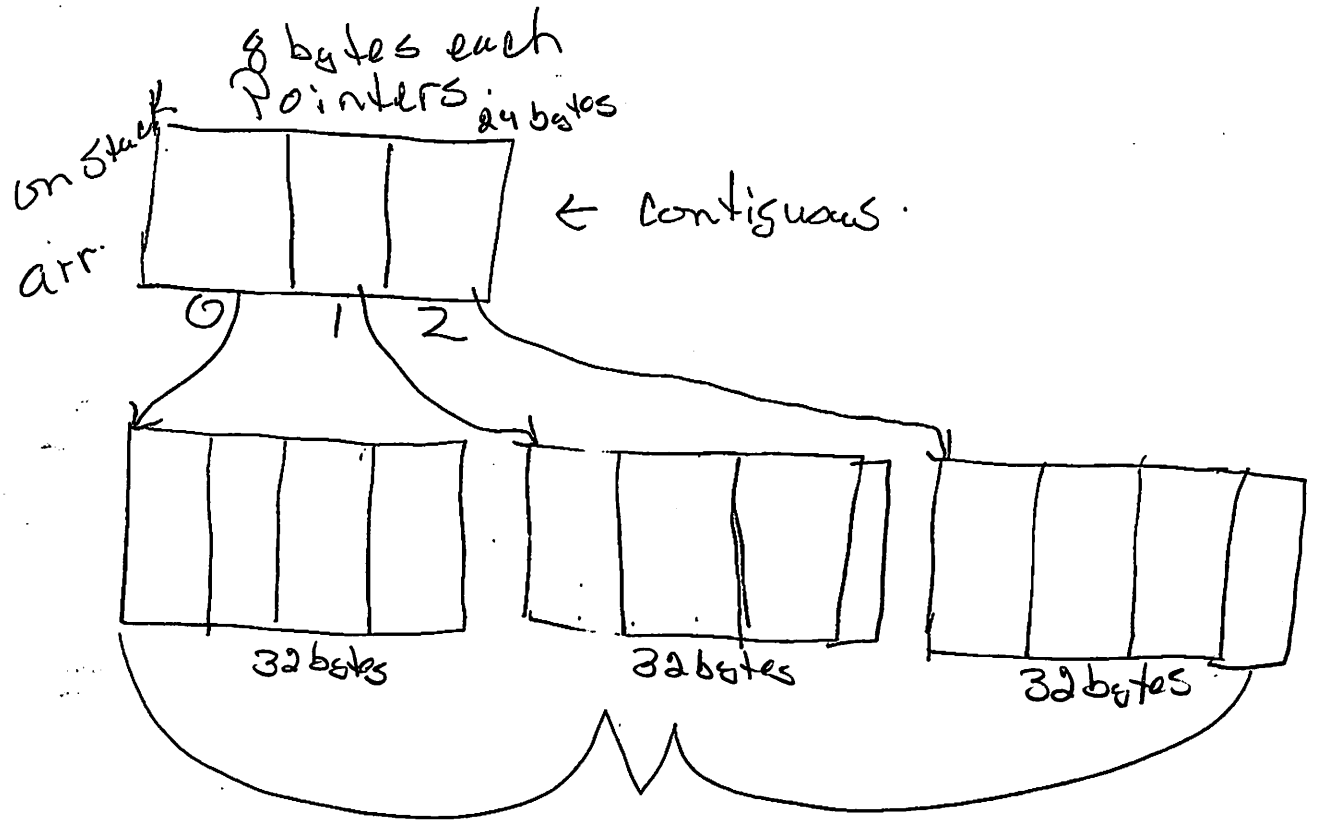
(1 * 4) + 2 = 6

arr[(curr_row * totalWidth) + curr_col] is the same as arr[(1*4)+2] or arr[6]

malloc 2 R=3 C=4

int *arr[R]

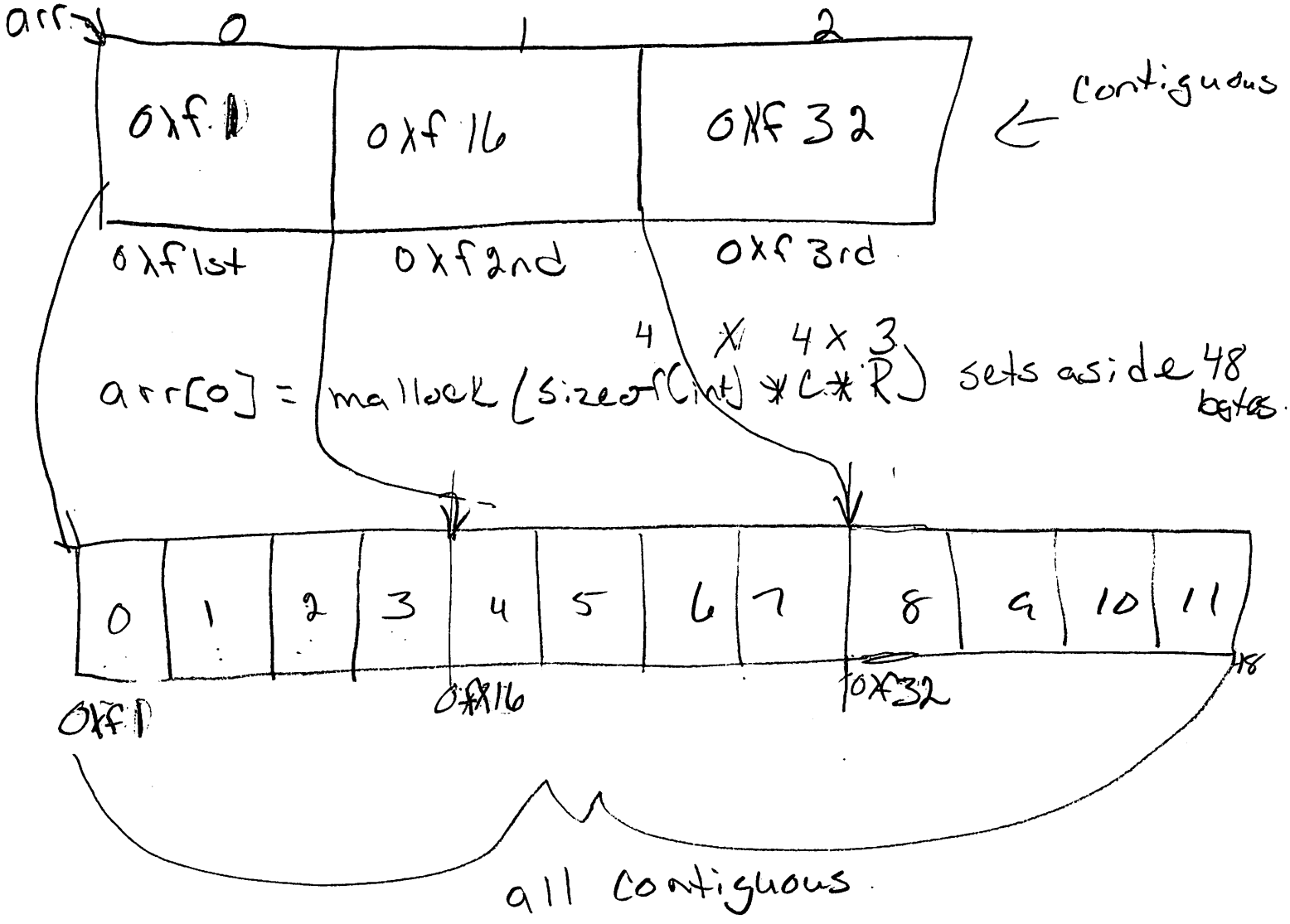
Part is stored on stack & part dynamically allocated.



malloc 4.c

2D Array using only 2 calls to malloc and 2 calls to free.

1st call to malloc allocates the memory for the pointers, 8 bytes each.



If the 1st for loop it starts at arr[1] b/c arr[0] is pointing to the allocated memory.

(*arr + (C*i)) calculates where in memory arr[i] will point.

malloc4b.c - another way of allocating memory for a 2D array. Uses only 1 malloc.

rows = 3 cols = 4

Calculate the memory needed for the pointers.

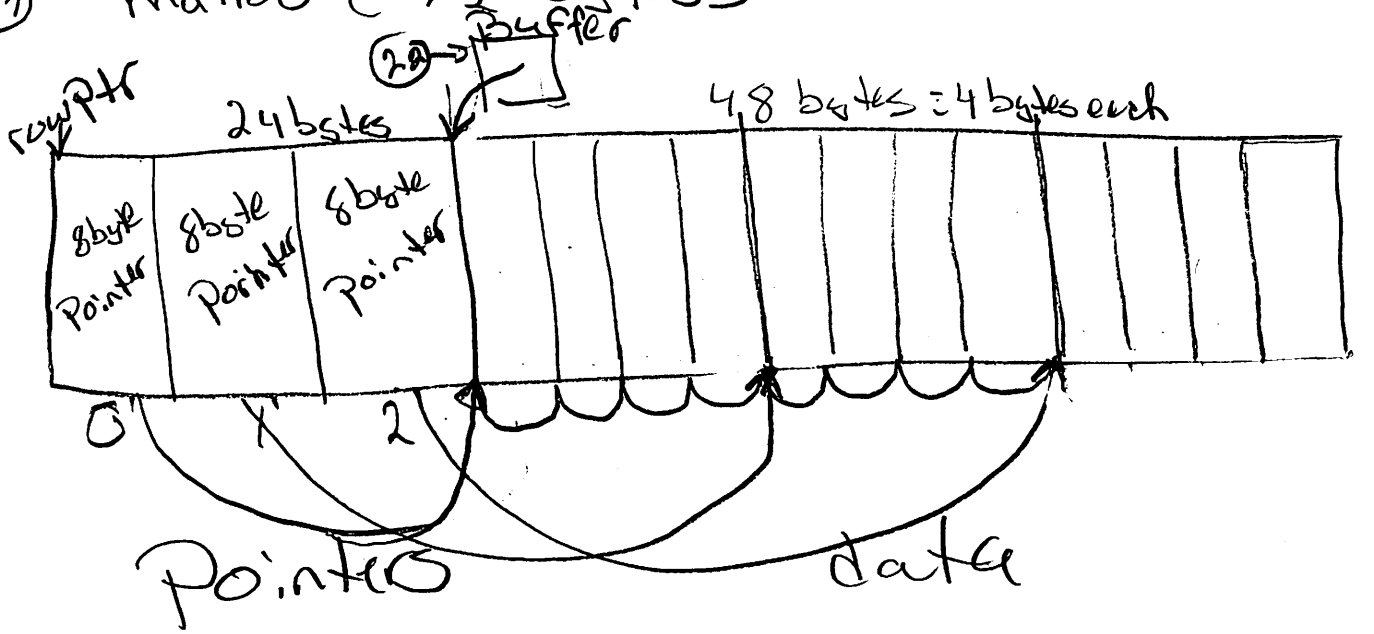
(12) header = 3 * sizeof pointer = 3*8 = 24 bytes

Calculate the amount of memory needed for the data.

(15) data = 3*4*sizeof int = 3*4*4 = 48 bytes.

malloc the memory for the pointers & the data.

(17) malloc (72 bytes) 48+24



buffer + (0 * 4 cols) = i = 0 (0*4) = 0 = buffer + 0

buffer + (1 * 4 cols) = i = 1 => (1*4) = 4 = buffer + 4

buffer + (2 * 4 cols) = i = 2 => (2*4) = 8 = buffer + 8