Computer Systems

CS107

Cynthia Lee

Topics

LAST TIME:

- Pointers and arrays (including strings—arrays of char)
- Pointer arithmetic
- Strings
- Files, error() reporting

THIS TIME:

- Two tools/tricks that help us understand memory:
  - “sizeof”
  - When sizeof works on an array and when it doesn’t
- Dynamic memory: malloc and free
- Where’s my data?
  - Stack vs heap vs data segment
- Pointers to pointers

Code example: sizes.c, ptr.c

SEE SAMPLES/LECT4 DIRECTORY ON MYTH FOR CODE

Dynamic memory: `malloc` and `free`

Arrays in C (on the heap)

```
int main(int argc, char *argv[]) {
    /* one-step process for stack */
    double arr1[3];
    /* two-step process for heap */
    double *ptr;
    ptr = malloc(3 * sizeof(double));   //calloc similar but 0-fills
```

char str[PATH_MAX]

- All about **malloc**:
  - › Like "new" in C++, but more basic
    - **`void * malloc(int);`**

Returns pointer to the location it has reserved for you

Takes an integer number of bytes to allocate (you need to do the math on how much you need)

Heap memory works like a hotel registration desk

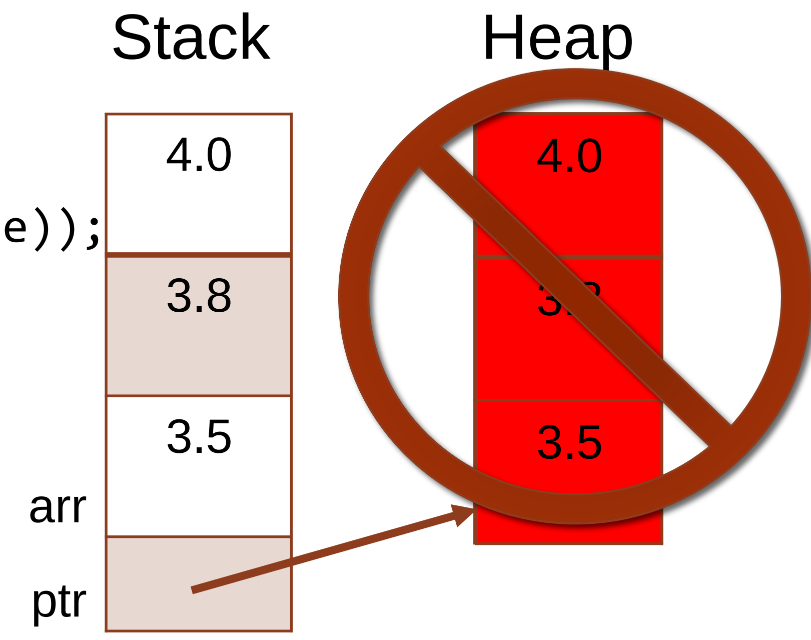
malloc's best friends: realloc and free

```
int main(int argc, char *argv[]) {
    double *ptr;
    ptr = malloc(3*sizeof(double));
    ptr[0] = 2.5;
    ptr = realloc(ptr, 5 * sizeof(double));
    free(ptr);
    ptr[0] = 7.2;
```

- All about **realloc**:
  - It gives you a larger (or smaller) space, still contiguous!
  - If the adjacent space was unused, will give you that
    - Otherwise will copy values over for you to a new, bigger space
- All about **free**:
  - Like new/delete in C++, malloc/free always needs to come in pairs!
  - Failing to free something you malloc-ed when you are done using it is a **memory leak**
- Of course, after you realloc or free memory, you never try to access that obsolete pointer again.

malloc + free example

```
int main(int argc, char *argv[]) {
    double arr[] = {3.5, 3.8, 4.0};
    double *ptr = malloc(3*sizeof(double));
    ptr[0] = 3.5;
    ptr[1] = 3.8;
    ptr[2] = 4.0;
    free(ptr);
    printf("%f\n", ptr[0]);
```



Only a horror movie character would access a hotel room that isn’t theirs (either never was, or was but checked out)

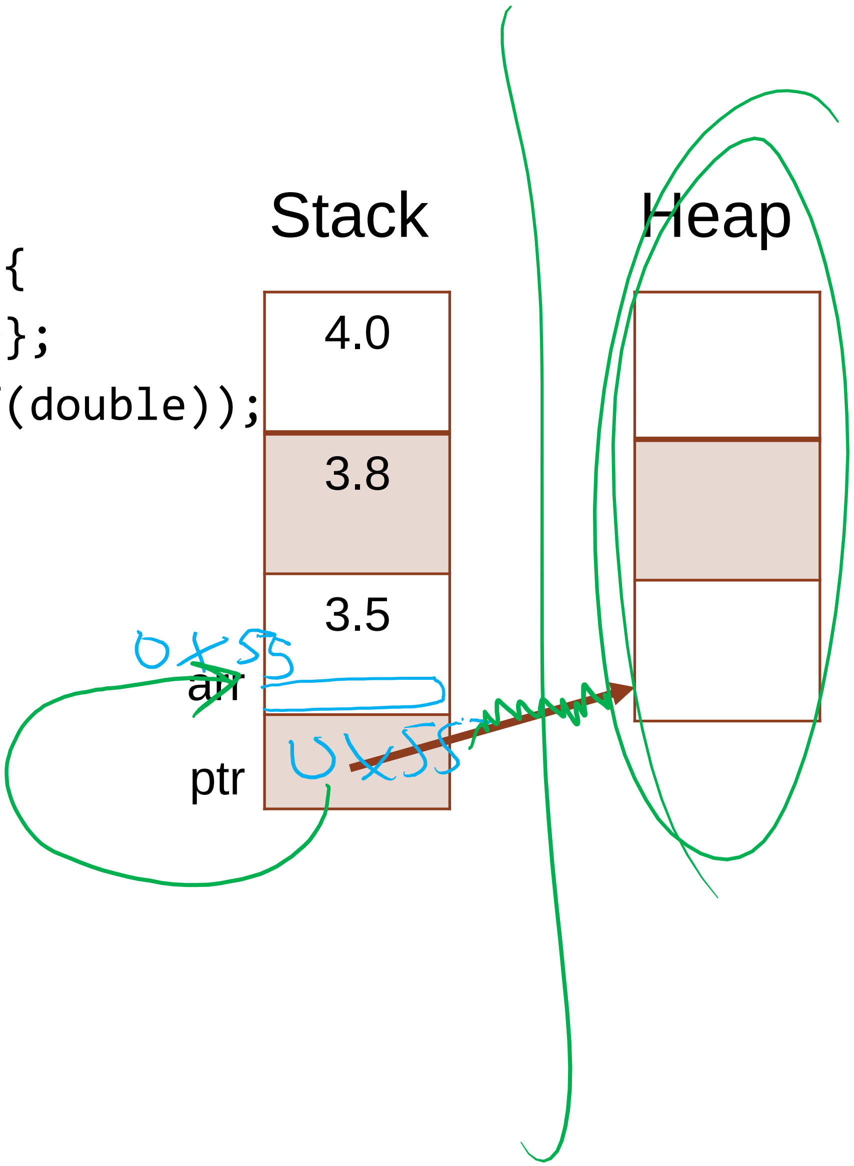
Arrays and Pointers

Stanford University

Pointers and arrays

```
int main(int argc, char *argv[]) {
    double arr[] = {3.5, 3.8, 4.0};
    double *ptr = malloc(3*sizeof(double));
    ptr = arr;   /* is this ok? */
```

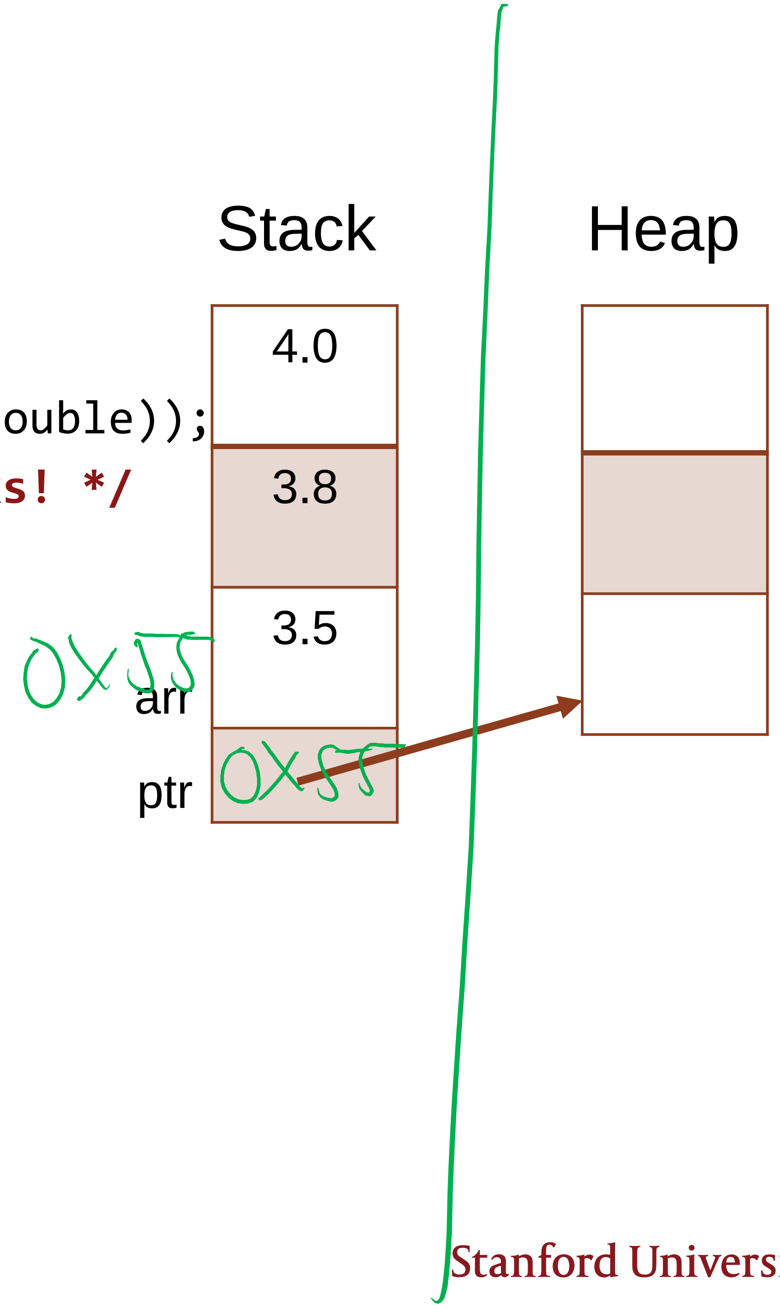
- Hmmm…what happens in this case?



Pointers and arrays

```
int main(int argc, char *argv[]) {
    double arr[] = {3.5, 3.8, 4.0};
    double *ptr = malloc(3*sizeof(double));
    ptr = arr;  /* last slide: leaks! */
    arr = ptr;  /* is this ok? */
```

- Hmmm…what happens in this case?



Strings in C

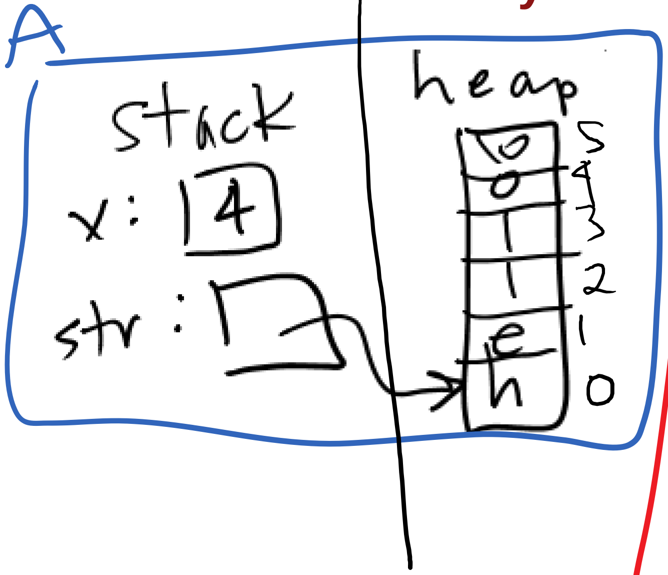
RECAP OF MEMORY DIAGRAMS OF THEIR POSSIBLE LOCATIONS IN MEMORY

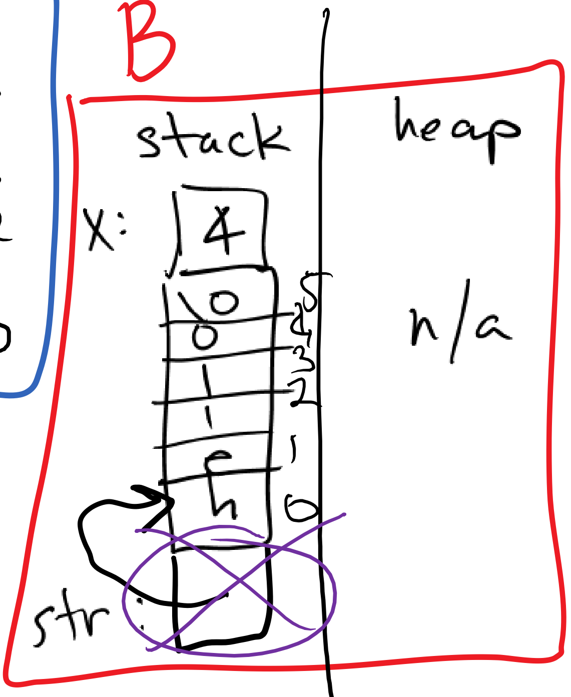
Stanford University

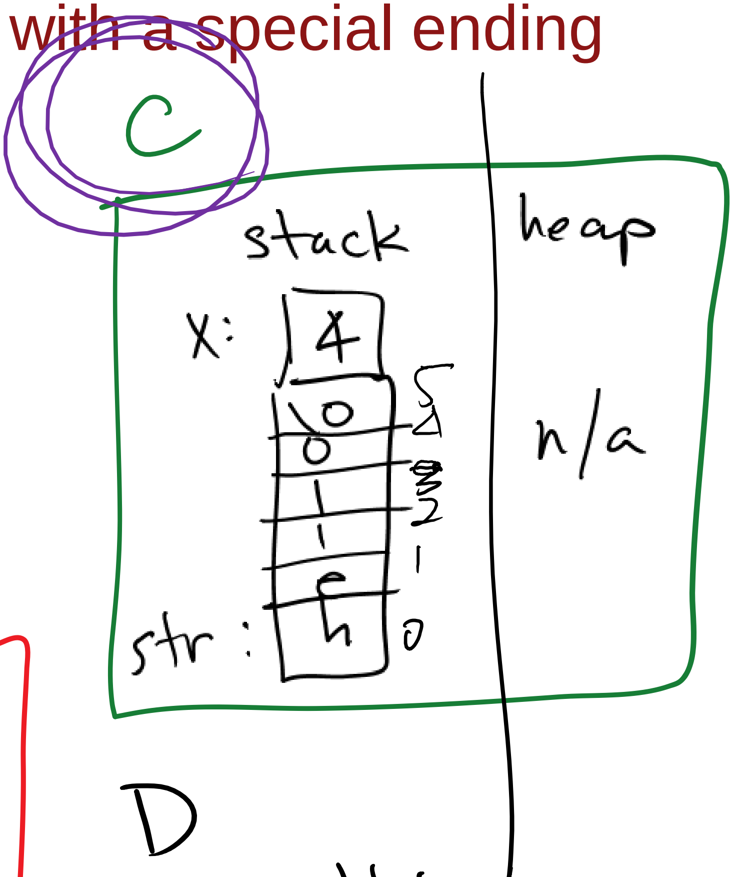
Strings in C: just an array of chars, but with a special ending sentinel value

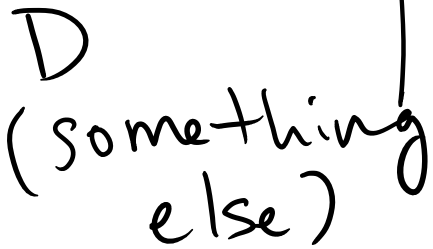
```
int main(int argc, char *argv[]) {
    int x = 4;
    char str[] = "hello";
```

- **What does memory look like?**







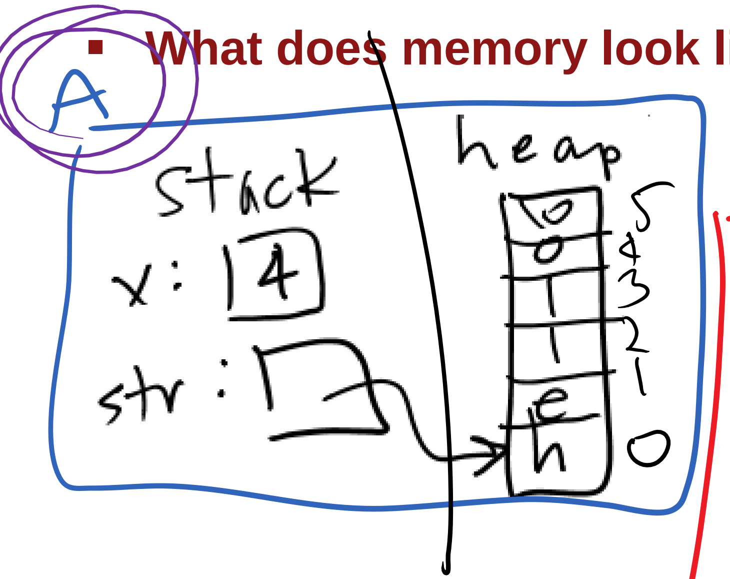


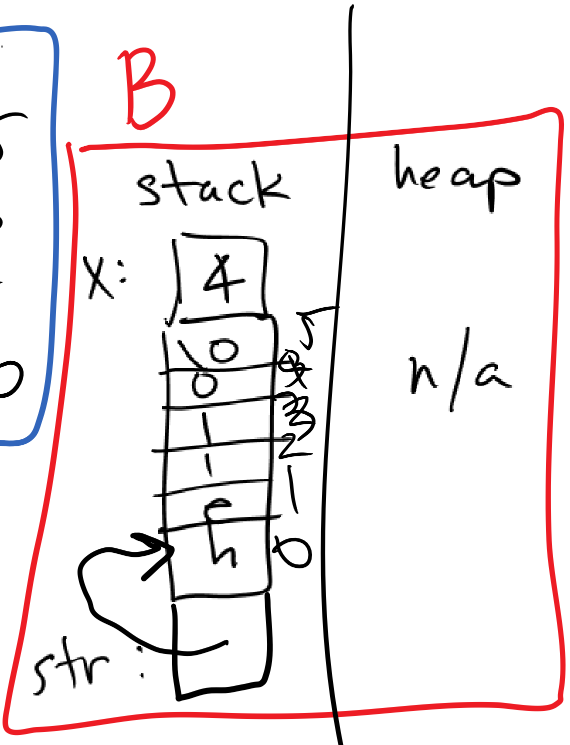
Strings and strdup: the gory details

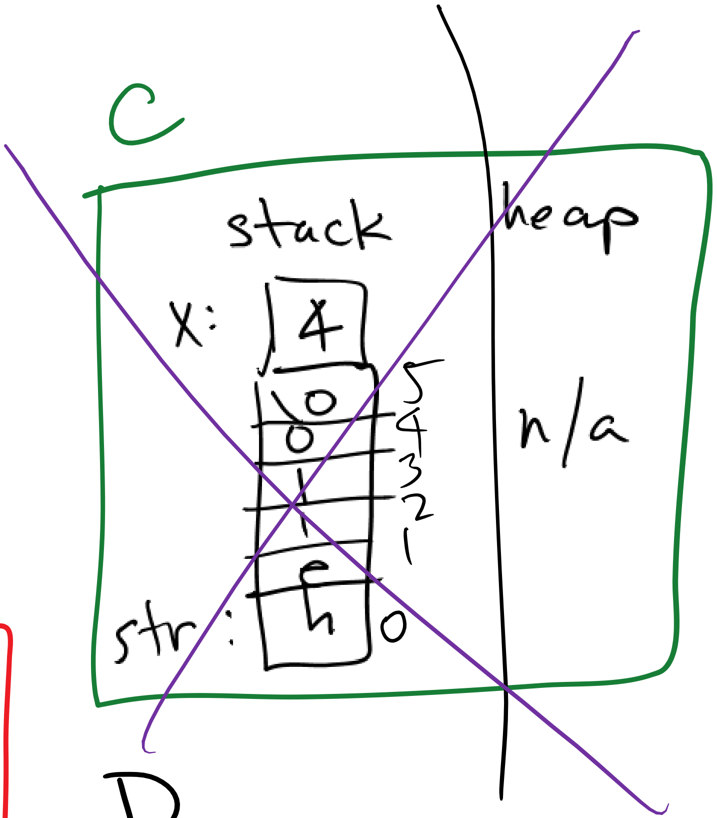
```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = strdup("hello");
```

malloc inside

- **What does memory look like?**







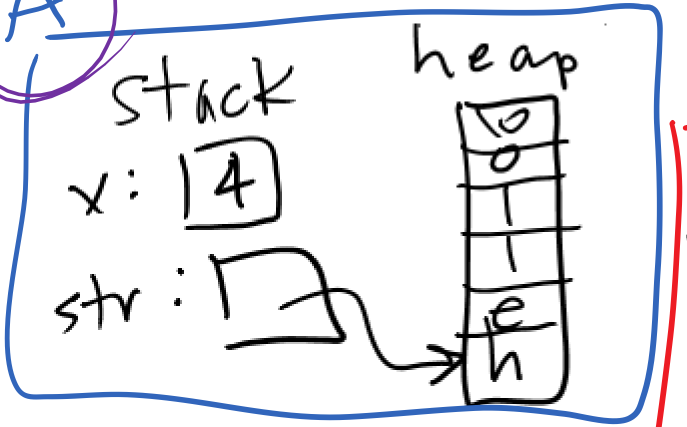
D
(something else)

Strings and malloc: the gory details

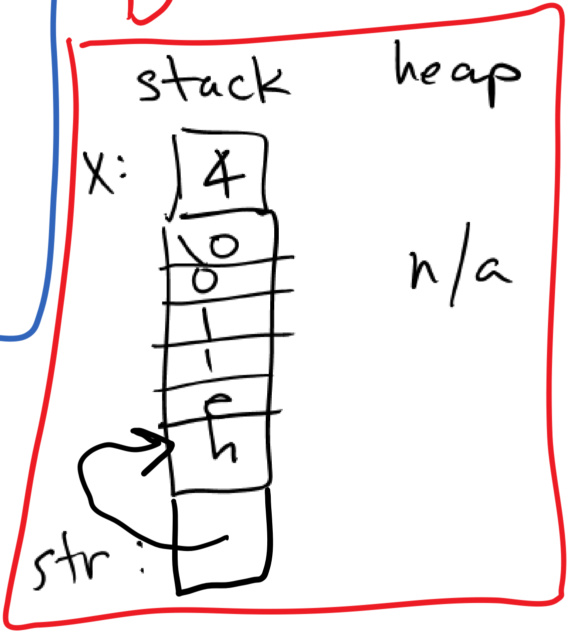
```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = malloc(6); //why not 5?
    strcpy(str, "hello");
```

- **What does memory look like?**

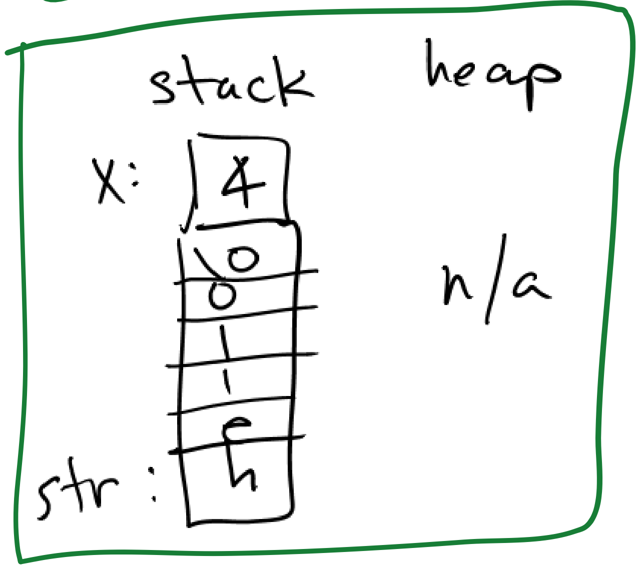
A



B



C

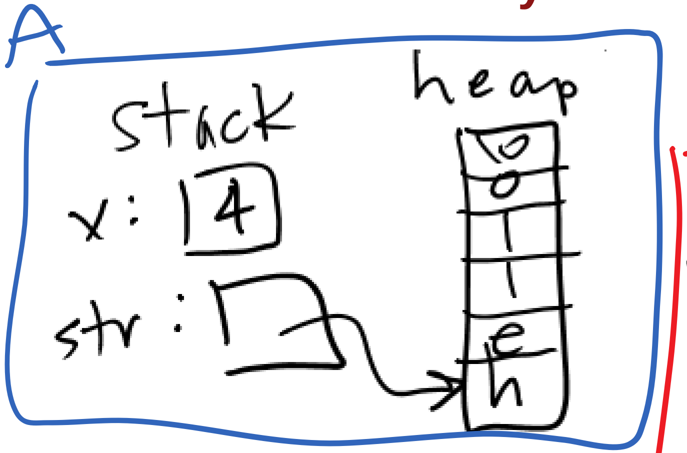


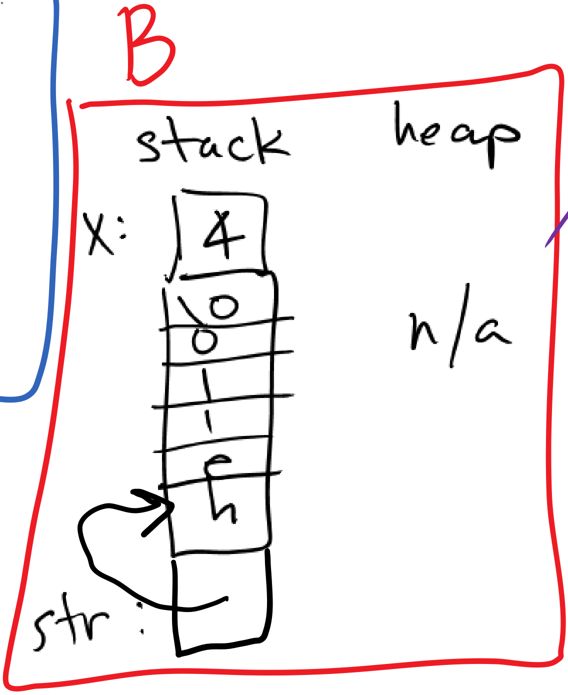
D
(something else)

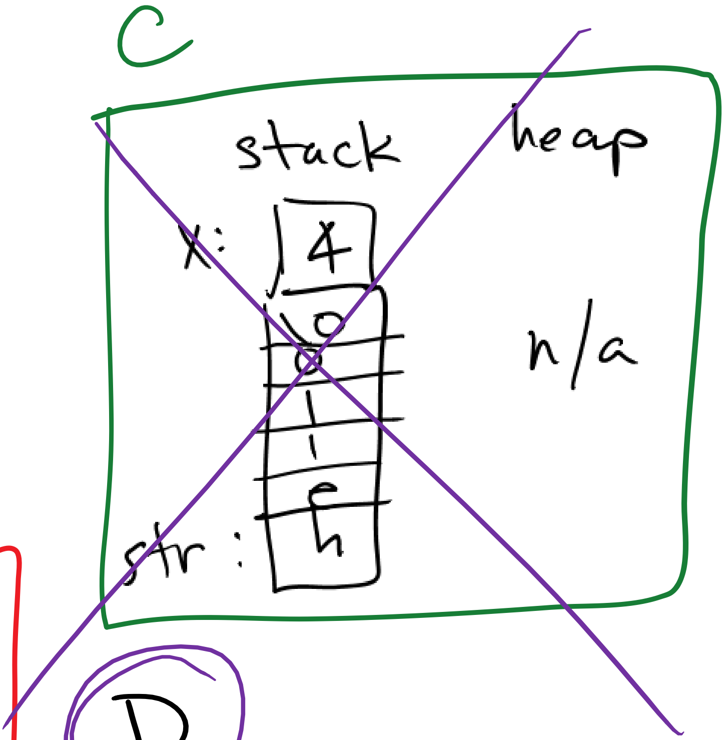
Strings in C: **even gorier** details

```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = "hello";
```

- **What does memory look like?**



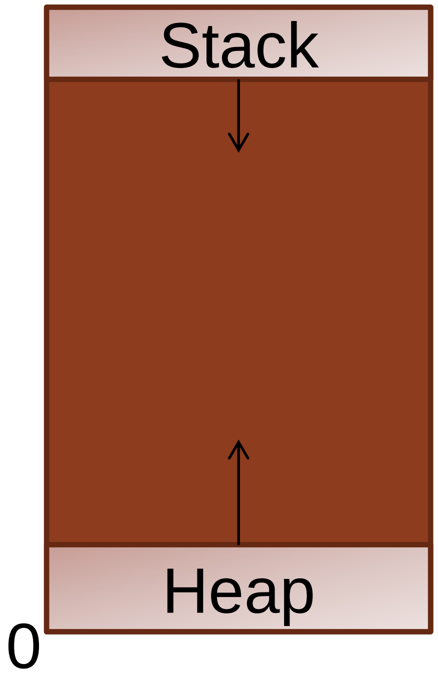


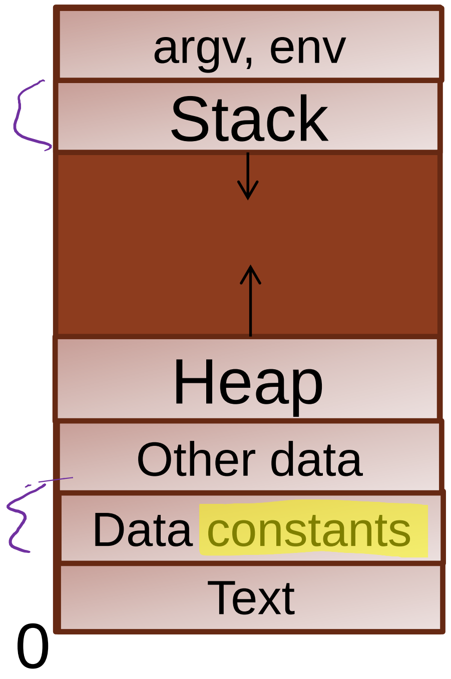


Strings in C: more gory details

```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = "hello";
```

- **What memory looks like, updated version with more detail:**





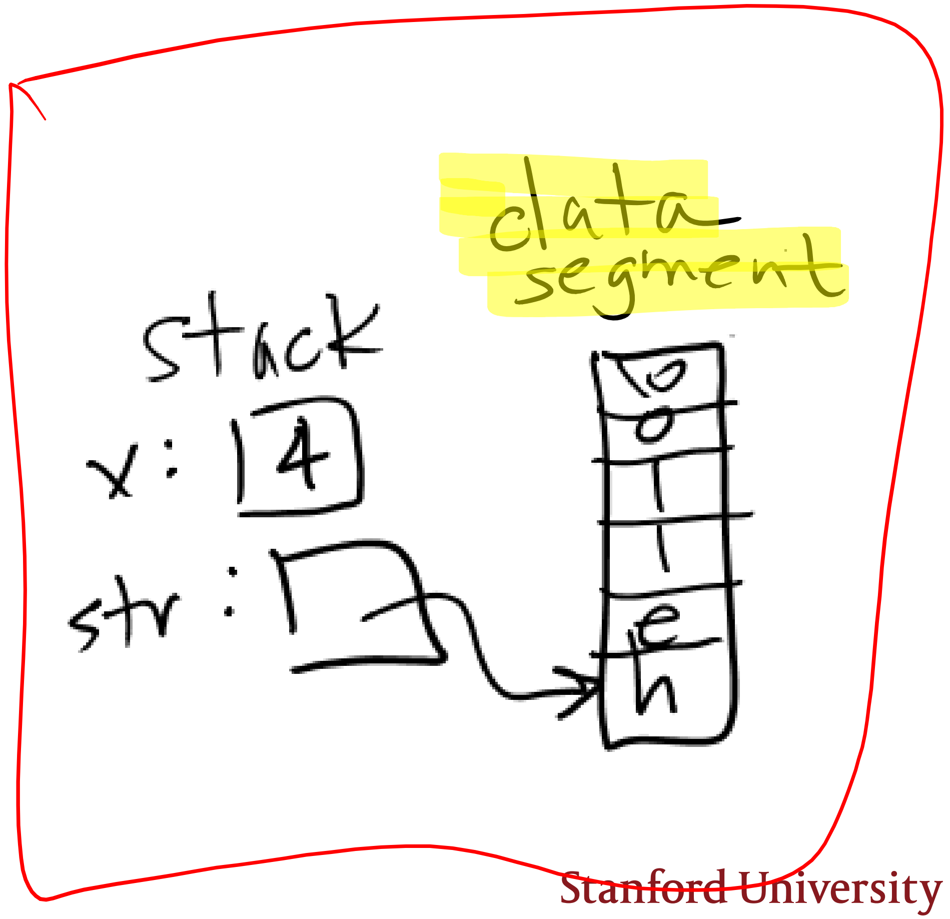
Stanford University

Strings in C: **even gorier** details [CORRECT ANSWER]

```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = "hello";
```

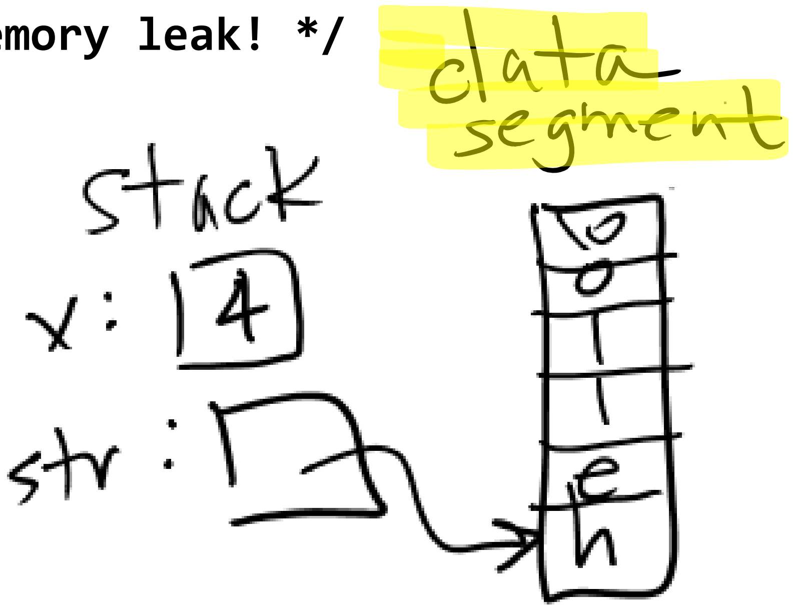
- **What does memory look like?**

D
(something
else)



Strings in C: *Leonardo DiCaprio cauterizing his own wound in the Revenant level of gory* details*

```
int main(int argc, char *argv[]) {
    int x = 4;
    char *str = "hello";
    str[4] = 'a'; /* not allowed - read only */
    str = NULL;   /* ok! not a memory leak! */
```



* confession: I haven't seen it, only heard about it