Lecture 14: ArrayList

CSE 29: Systems Programming and Software Tools

Olivia Weng

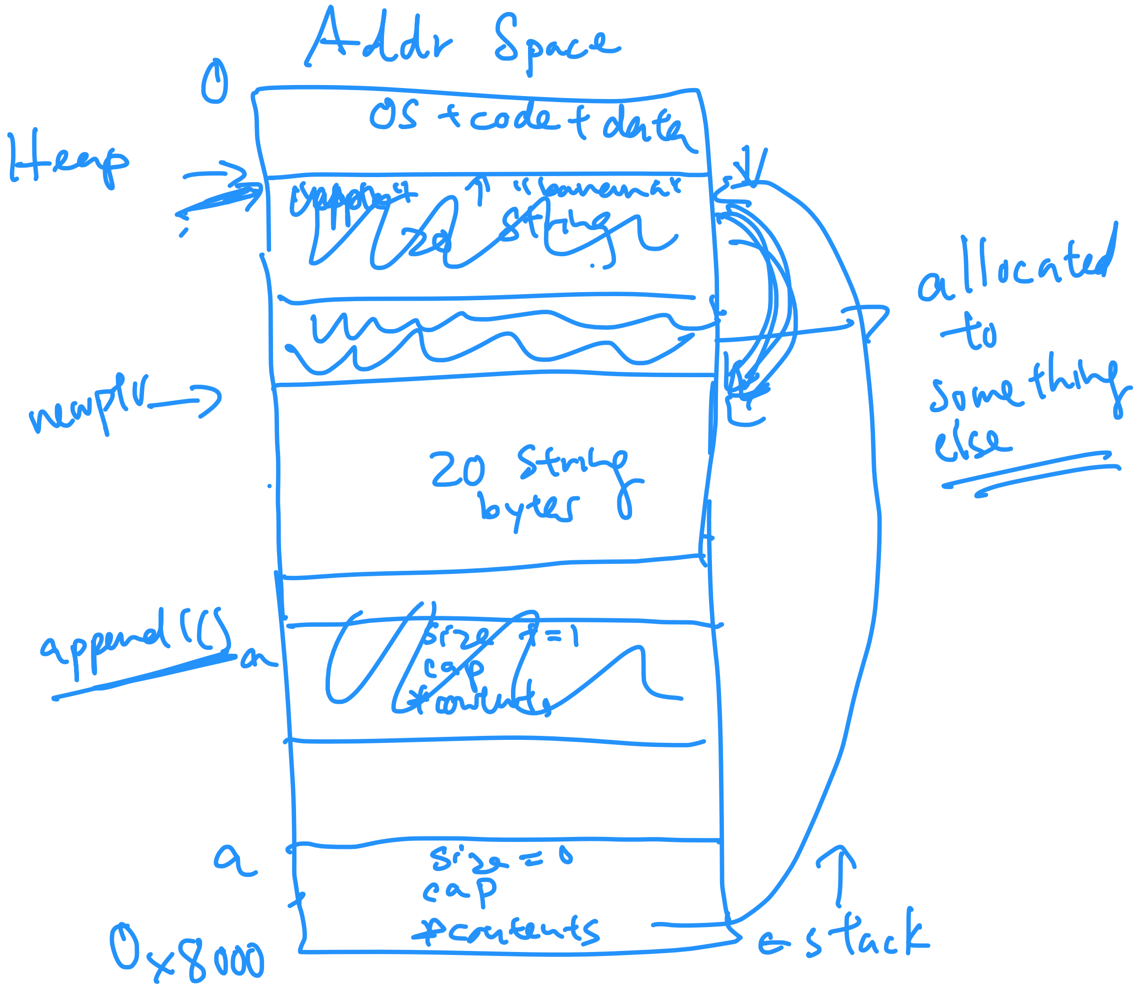
How could we implement an ArrayList in C?

How could we implement an ArrayList class in C?

```
struct list {

    uint32_t size;

    uint32_t capacity;

    String *contents; // the actual list of strings

};
```

Demo

- new_List()
- append1()
- expandCapacity()
- destroy_List()



Pointer arithmetic

pa+1 == &a[1]

*(pa+1) == a[1]

- We can do addition on pointers to get a new address!

0x100 + 1 = 0x101 ✗

0x100 + 4 = 0x104 ✓

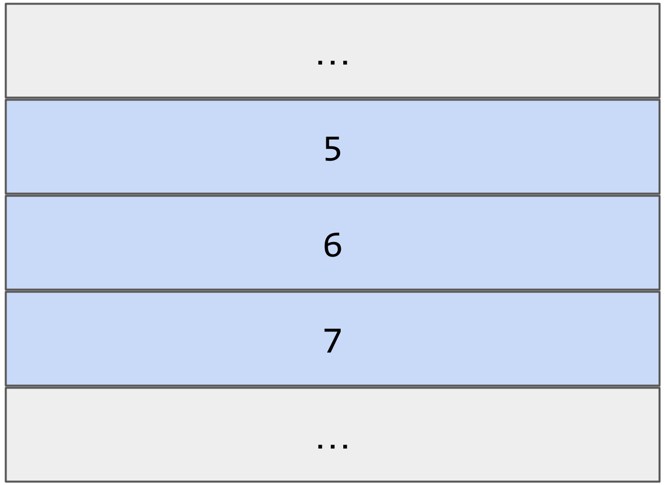
```
int a[3] = {5, 6, 7};

int *pa = a;
```

pa + (1 * sizeof(int))

under the hood

C does this for you

```
pa = 0x100

pa + 1 = 0x104

pa + 2 = 0x108

0x10C
```



Rule: `pa + n =` pa + n * sizeof(int)

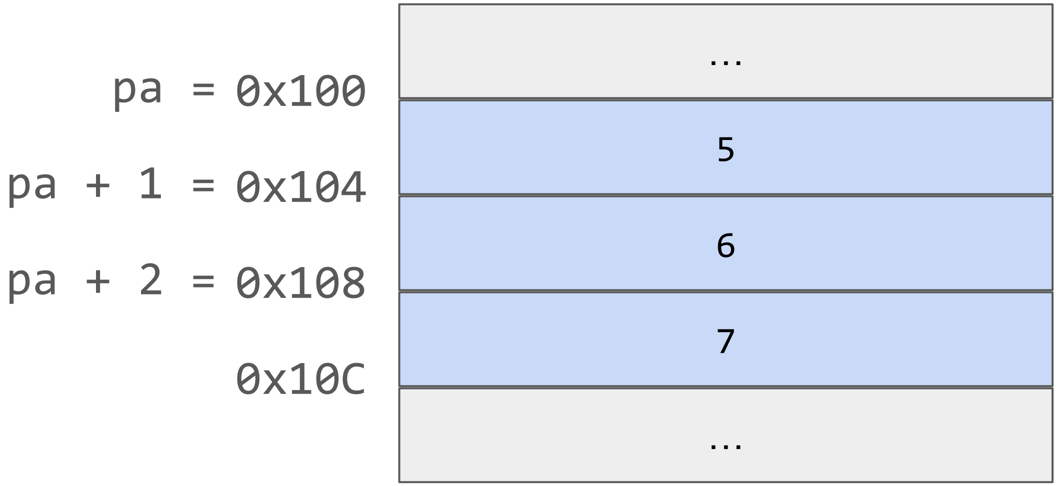
↳ whatever type was declared

Pointer arithmetic

- We can do addition on pointers to get a new address!

```
int a[3] = {5, 6, 7};

int *pa = a;


int b = *(pa + 1);

printf("%d\n", b);
```

↳ 6



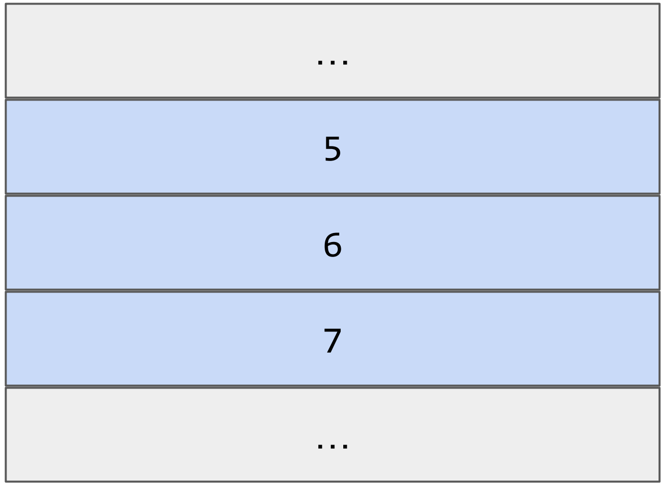
```
Rule: pa + n = pa + n * (sizeof(int))
```

↳ for int * only!

Pointer arithmetic

- We can do addition on pointers to get a new address!

```
int a[3] = {5, 6, 7};

int *pa = a;                          pa = 0x100

                                  pa + 1 = 0x104

int b = *(pa + 1);                pa + 2 = 0x108

printf("%d\n", b); -> 6                    0x10C
                → 13        6              7
printf("%d\n", *(pa + 1) + *(pa + 2));

printf("%d\n", pa[1] + pa[2]);
          13        6       7
```

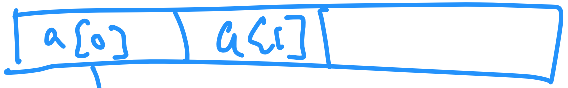


Pointer arithmetic

- General rule: $\overbrace{\texttt{ptr + n}}^{C} = \underbrace{\texttt{ptr + n * sizeof(type)}}_{\text{how C interprets "under the hood"}}$

struct
char
int
$\downarrow$

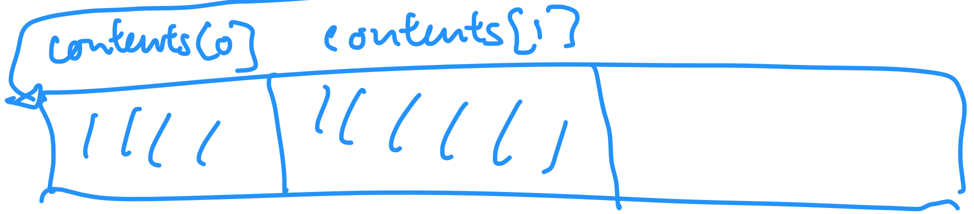
List *a;



(*(a+1).contents) + 1

struct List:

String * contents;



contents + 1 $\Rightarrow$ contents + 1 * sizeof(String)

Pointer arithmetic

- General rule: `ptr + n = ptr + n * sizeof(type)`

```
char str[] = "Hi CSE29!";
             ↑1 23
                ↑


printf("%c\n", *(str + 1));      → i

printf("%c\n", str[1]));         → i

printf("%s\n", str + 3);
```

CSE29!

Pointer arithmetic

- General rule: `ptr + n = ptr + n * sizeof(type)`

30

```
char str[] = "Hi CSE29!";
```

0123456

```
char str1[] = "30";
```

6 2

```
strncpy(str + 6, str1, strlen(str1));


printf("%s\n", str);
```

Hi CSE30!