Lecture 12: More `struct` and `malloc()`

CSE 29: Systems Programming and Software Tools

Olivia Weng

Announcements

- Midsummer feedback form

- Sign up for exam 2 on prairietest.com!

Review

When a variable lasts longer than a function call, we want to store it on the heap.

- `malloc()`: allocates memory on the heap

  `int *pnum = malloc(#bytes you need);`

- `free()`: frees up memory previously allocated on the heap

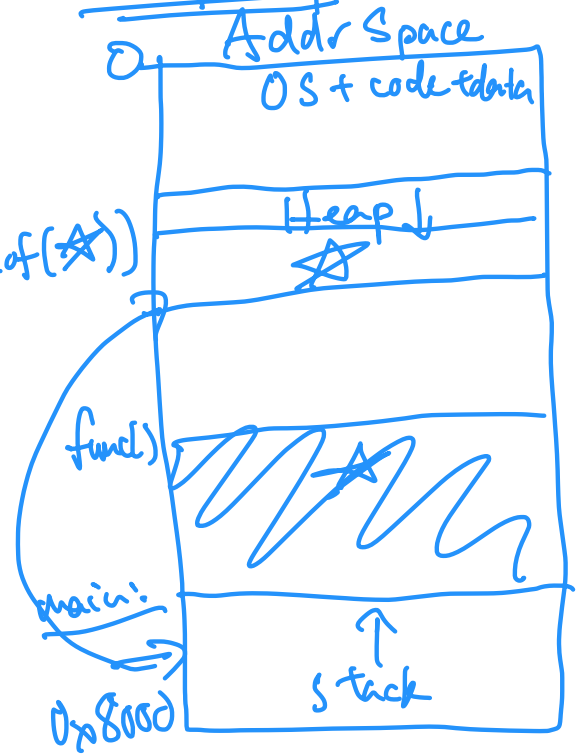
  `free(pnum);`

- `calloc()`: allocates memory on the heap AND zeroes it out

`int *arr = Calloc(10, sizeof(int));`

10 = # items

sizeof(int) = size of items in bytes



Review

- `malloc()`: allocate memory on the heap

```
uint64_t *pa = malloc(...?);
```

- `free()`: free allocated memory

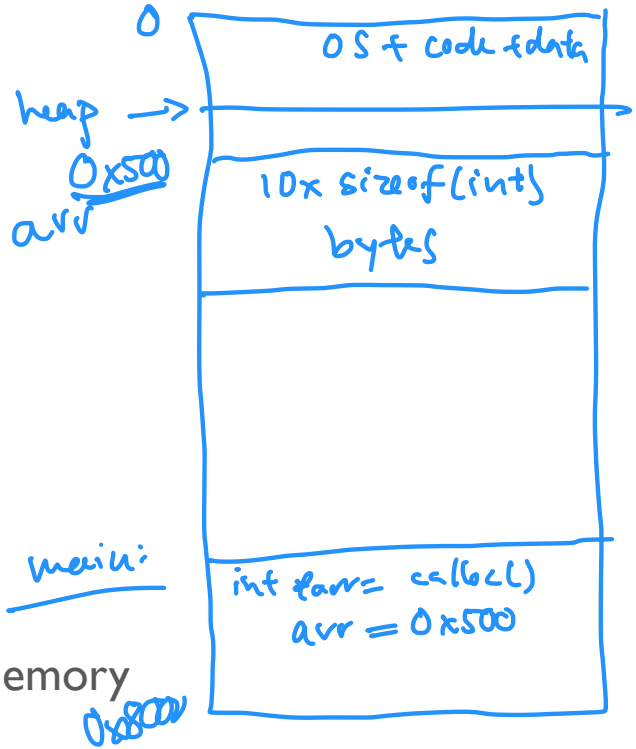
```
free(pa);
```

- `calloc()`: similar to malloc but zeros out allocated memory

```
int *pa = calloc(10, sizeof(int));
```

realloc()

main:
int *arr = calloc(10, sizeof(int));
arr[0] = 100;
arr[i] = arr[0] + arr[0];



> ./a.out 5

Review

- Fill in the …? blanks

```
int main(int argc, char *argv[]) {
    int *pnum = malloc(...?)
    *pnum = atoi(argv[1]); // assume some user integer input
    int num_squared = ...?
    printf("num_squared = %d\n", num_squared);
    printf("pnum_squared = %d\n", ...?);
    ...?
}
```

sizeof(int)

*pnum * *pnum;

*pnum * *pnum

free(pnum);

Review

- Fill in the …? blanks
  - This is a program creating a user-defined variable-length array

```
int main(int argc, char *argv[]) {

    uint32_t len = atoi(argv[1]); // assume some user integer input

    int *arr = malloc(...?);

    // filled array
    print_array(arr);  // write this function!

    square_array(arr); // write this function!

    ...?

}
```

Review

- Fill in the ...? blanks

```
typedef struct point {
    int x;
    int y;
} Point;

// write this function!
Point make_Point(int x, int y);
```

```
int main(int argc, char *argv[]) {
    uint32_t len = atoi(argv[1]);
    Point *arr = malloc(...?);
    for (int i = 0; i < len; i++) {
        arr[i] = make_Point(i, i);
    }
    ...?
}
```

len * sizeof(Point)