Program memory: arrays & pointers

COSC 208, Introduction to Computer Systems, 2021-02-22

Announcements

- Project 1 Part 2 (and revisions to Part 1) due Thursday at 11pm

Outline

- Warm-up
- Arrays & pointers

Warm-up

Q1: *What is the output of this program?*

```
void increment1(int a) {
    a = a + 1;
}

void increment2(int *b) {
    *b = *b + 1;
}

int main() {
    int x = 1;
    int *y = &x;
    increment1(x);
    printf("%d %d\n", x, *y);
    increment2(y);
    printf("%d %d\n", x, *y);
}
```

```
1 1
2 2
```

Stack memory layout

Q2: *What is the output of this program?*

```
int main() {
    int a = 1; // Assume at 0x4
    int *x = &a; // Assume at 0x8
    int **y = &x; // Assume at 0xC
    printf("%p %p %p\n", a, x ,y);
    printf("%p %p\n", *x , *y);
}
```

Output

```
0x1 0x4 0x8
0x1 0x4
```

Arrays & pointers

- An array variable is a pointer to a region of memory where the items in the array are stored
    - Example

    ```
    int main() {
        char word[] = "fat";
        printf("word = %s\n", word);
        char *ptr = word;
        printf("ptr = %s\n", ptr);
        if (ptr == word) {
            printf("ptr == word\n");
        }
        else {
            printf("ptr != word\n");
        }
        word[1] = 'i';
        printf("word = %s\n", word);
        *ptr = 's';
        printf("word = %s\n", word);
        ptr[1] = 'a';
        printf("word = %s\n", word);
    }
    ```

- This explains why there is no out-of-bounds checks for arrays: the memory addresses in pointers are never checked to see if they are valid
- This also explains why you can change an array within a function and have those changes reflected outside of the function

```
int update(char str[]) {
    str[0] = 'p';
}
int main() {
    char word[] = "hat";
    update(word);
    printf("%s\n", word);
}
```

- Q3: *What is the output of this program?*

```
int main() {
    int nums[4] = {1,2,3,4};
    printf("%d %d\n", *nums, nums[1]);
    int *ptr = &nums[1];
    nums[1] += 4;
    printf("%d %d\n", *ptr, nums[0]);
    ptr = (nums + 2);
    printf("%d\n", *ptr);
    ptr++; // num++ is illegal
    printf("%d\n", *ptr);
}
```

```
1 2
6 1
3
4
```

- Q4: *What is the output of this program?*

```
int main() {
    char *first = "Colgate";
    char second[10] = "Univ";
    char *f = &first[3];
    printf("%d\n", strlen(f));
    char *s = second;
    *s = 'K';
    s++;
    *(s+2) = 't';
    printf("%s %s\n", second, s);
}
```

```
4
Knit nit
```

Extra practice

- Q5: *What do the following two functions do? How are they different?*

```
void swap1(int *m, int *n) {
    int tmp = *n;
    *n = *m;
    *m = tmp;
}
void swap2(int **x, int **y) {
    int *tmp = *y;
    *y = *x;
    *x = tmp;
}
```

- Q6: *What is the output of this program?*

```
int main() {
    int a = 1;
    int b = 2;
    int *ptrA = &a;
    int *ptrB = &b;
    swap1(ptrA, ptrB);
    printf("%d %d\n", a, b);
    swap2(&ptrA, &ptrB);
    printf("%d %d %d %d\n", a, b, *ptrA, *ptrB);
}
```

```
2 1
2 1 1 2
```

Q7: *Draw a memory diagram that displays the program's variables and their values when the program reaches the comment* `STOP HERE`.

```c
char *split(char *str, char delim) {
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] == delim) {
            str[i] = '\0';
            return &str[i+1];
        }
    }
    return NULL;
}

void parse(char *url) {
    char separator = '/';
    char *path = split(url, separator);
    int domainlen = strlen(url);
    int pathlen = strlen(path);
    // STOP HERE
    printf("Domain (%d chars): %s\n", domainlen, url);
    printf("Path (%d chars): %s\n", pathlen, path);
}

int main() {
    char input[] = "colgate.edu/lgbtq"
    parse(input);
}
```

Worksheet created by Professor Aaron Gember-Jacobson