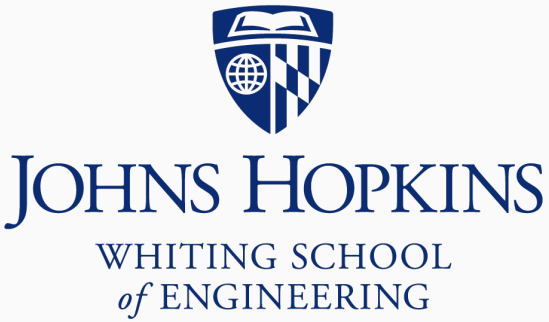
Command line arguments

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

Command line arguments

We talked about how to read input from standard in

- `scanf` & `getchar`

Another way to get input is with *command-line arguments*

Command-line arguments

You are used to running programs with command-line arguments:

- `mkdir cs220` – makes a directory called `cs220`
- `ls *.txt` – lists all the files ending in `.txt`
- `cd $HOME/programming` – changes to the `$HOME/programming` directory

C programs can take command-line arguments, but we must tell the `main` function how to receive them

Command-line arguments

```c
#include <stdio.h>

int main(int argc, char* argv[]) {
    printf("argc = %d\n", argc);
    for(int i = 0; i < argc; i++) {
        printf("argv[%d] = %s\n", i, argv[i]);
    }
    return 0;
}
```

```
$ gcc args_eg_1.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out rosebud
argc = 2
argv[0] = ./a.out
argv[1] = rosebud
```

Command-line arguments

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

- `int argc` equals the number of command-line arguments. Program name (e.g. `./a.out`) is always included first.
- `char* argv[]` is an array of strings, one per command-line argument.
    - Sometimes written as `char **argv`; means the same thing

Command-line arguments

```c
#include <stdio.h>
#include <stdlib.h> // for atof

int main(int argc, char* argv[]) {
    double total = 0.0;
    // i = 1 skips first argument, which is ./a.out
    for(int i = 1; i < argc; i++) {
        // atof converts a string to a double
        total += atof(argv[i]);
    }
    printf("Total: %.1f\n", total);
    return 0;
}
```

```
$ gcc args_eg_2.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out 1.1 2.2 3.3
Total: 6.6
```