CIS241

System-Level Programming and Utilities

C Command Line Arguments

Erik Fredericks, frederer@gvsu.edu

Fall 2025

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**

Let's get argumentative

We've not done anything with comand line args yet, we should fix that

Go from:

```
int  main()
{
      return  0;
}
```

to

```
int  main(int argc, char* argv[])
{
      return  0;
}
```

The `main` function prototype:

We will pass command line args an array of strings

What is the actual type of a string in C?

- `char[]` or `char*`

Thus, we pass two arguments to main:

- The number of arguments: `int argc`
- The array of strings: `char* argv[]`

Using command line arguments

First argument (index `0`) is always the name of the program (just like in bash)

For example: `./a.out 100 foo`

```
argv[0] = the string "./a.out"
argv[1] = the string "100"
argv[2] = the string "foo"
```

Sometimes we need to convert to numeric types:

- `int x = atoi(argv[1]); // need stdlib.h for this!`

List of other conversions in stdlib.h:

- https://cplusplus.com/reference/cstdlib/

A sample - get and print command line args

```c
#include <stdio.h>
#include <stdlib.h>

int  main(int argc, char* argv[])  {
     if  (argc  >  1)  {
          for  (int  i  =  0;  i  <  argc;  i++)  {
               printf("Argument %d: %s\n",  i,  argv[i]);
          }
     }  else  {
          printf("Arguments required\n");
     }
     return  0;
}
```

Curly braces on same line to get them all on one slide ಠ_ಠ