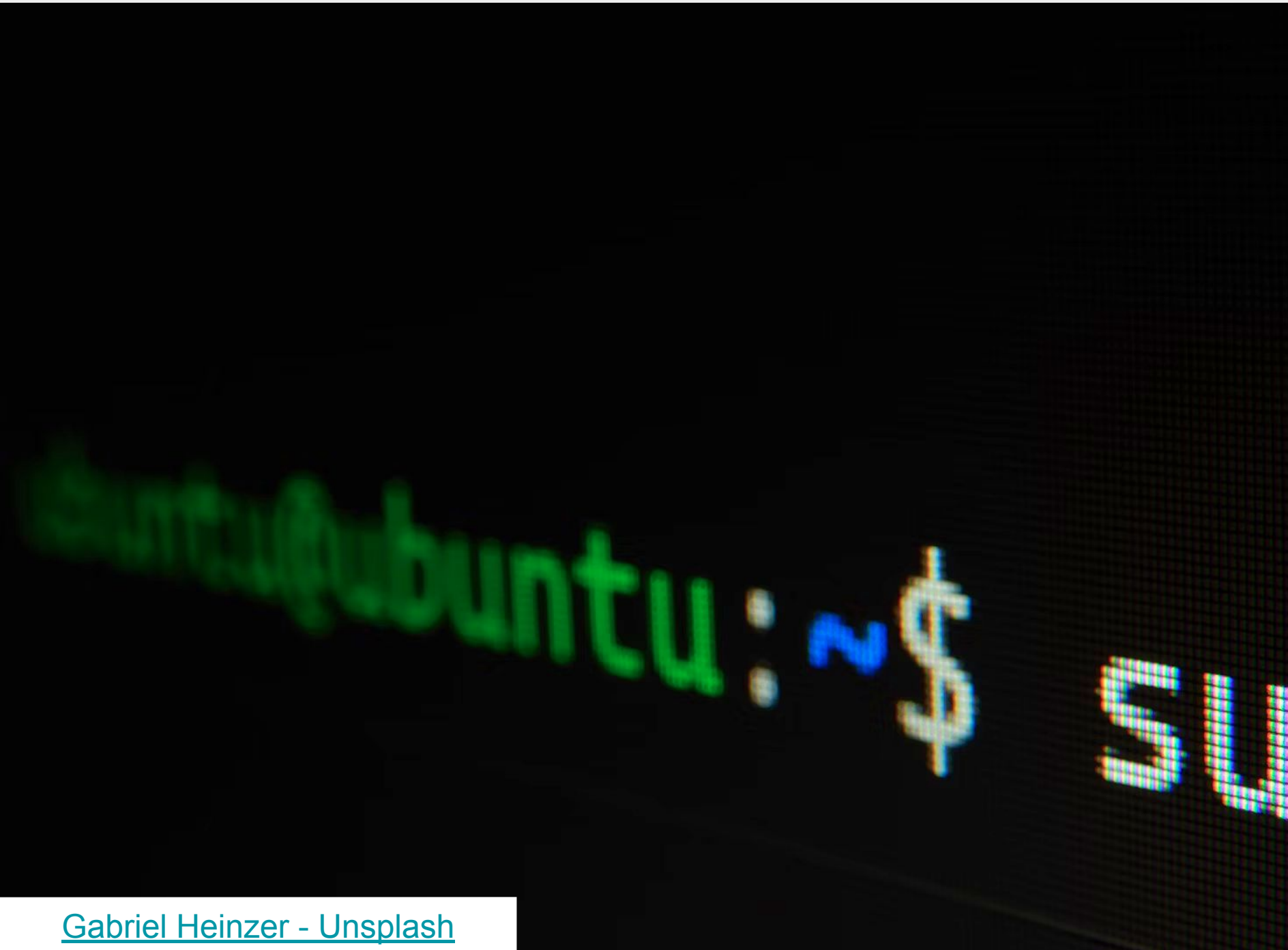
C command line args

Adapted from materials by Dr. Carrier



Gabriel Heinzer - Unsplash

The usual

So far, we have not passed in command line arguments when running our code

The usual

So far, we have not passed in command line arguments when running our code

We also have not passed any arguments to our main function:

The usual

So far, we have not passed in command line arguments when running our code

We also have not passed any arguments to our main function:

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

Command line args

We will pass command line args an array of strings

What is the actual type of a string in C?

Command line args

We will pass command line args an array of strings

What is the actual type of a string in C?

`char[]` or `char*`

Command line args

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:

Command line args

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:

1. The number of arguments (int argc)

Command line args

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:

1. The number of arguments (int argc)
2. The array of strings (`char* argv[]`)

Command line args

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:

1. The number of arguments (int argc)
2. The array of strings (`char* argv[]`)

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

Using command line arguments

First argument (index 0) is always the name of the program

Using command line arguments

First argument (index 0) is always the name of the program

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

Using command line arguments

First argument (index 0) is always the name of the program

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

`argv[0]` = the string “`./a.out`”

`argv[1]` = the string “`100`”

`argv[2]` = the string “`foo`”

Using command line arguments

First argument (index 0) is always the name of the program

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:

Using command line arguments

First argument (index 0) is always the name of the program

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]);
```

List of other conversions in stdlib.h