Examples of Arguments Passed to `main`

What does `main` receive if you type …

`./word_split` ?

`argc` is `1`, `argv` is `{"./word_split", NULL}`

`./word_split word_split.c` ?

`argc` is `2`, `argv` is `{"./word_split", "word_split.c", NULL}`

`./word_split a b c d` ?

`argc` is `5`, `argv` is `{"./word_split", "a", "b", "c", "d", NULL}`

 © 2018 Steven S. Lumetta. All rights reserved.

Graphical Interfaces Produce Arguments to `main`

What about graphical interfaces?

Double-clicking on an **application produces an `argc` of 1**.

Double-clicking on an **associated file produces an `argc` of 2** (the program and the file).

Dropping N files into an **application** icon **produces an `argc` of N + 1**.

 © 2018 Steven S. Lumetta. All rights reserved.

System Calls Provide a Reason for Failure in `errno`

When system calls (or functions that call system calls, such as **`fopen`**) **fail**,
- **they set** a (thread-specific) variable
- called **`errno`**
- **to an error number** indicating the reason.

Code can check error numbers:
- for example, ENOENT means
- that the file does not exist.

 © 2018 Steven S. Lumetta. All rights reserved.

Use `perror` to Print a Human-Readable Error Message

Sometimes it's better to tell the human.

Humans don't know error numbers.

To **print**
- **a human-readable error message**
- to **`stderr`**, use **`perror`**:

```
void perror (const char* prefix);
```

The prefix, a colon, and a space are printed before the error message, and a newline is printed after the error message.

 © 2018 Steven S. Lumetta. All rights reserved.