

Start as Before, with Command-Line Arguments

```
Let's take a look at the code. limit on word size
static const
    int32_t max_word_len = 500;
int main (int argc, char* argv[])
{
    char buf1[max_word_len + 1];
    char buf2[max_word_len + 1];
    char* last_line; command-line
    char* cur_line; arguments
```

ECE 220: Computer Systems & Programming

© 2018 Steven S. Lumetta. All rights reserved.

slide 41

41

Declare Buffers and Pointers to Buffers

```
static const include space for NUL
    int32_t max_word_len = 500;
int main (int argc, char* argv[])
{
    char buf1[max_word_len + 1];
    char buf2[max_word_len + 1];
    char* last_line; pointers for
    char* cur_line; double-buffering
```

ECE 220: Computer Systems & Programming

© 2018 Steven S. Lumetta. All rights reserved.

slide 42

42

Need Variables for Swapping and Counting

```
char* tmp; Swapping requires
int32_t count; a temporary.
if (1 != argc) { count of current word
    fprintf (stderr,
        "syntax: %s\n",
        argv[0]);
    return EXIT_BAD_ARGS;
}
```

ECE 220: Computer Systems & Programming

© 2018 Steven S. Lumetta. All rights reserved.

slide 43

43

Start by Checking Command Syntax

```
char* tmp; must be called
int32_t count; with one argument
if (1 != argc) { error messages
    fprintf (stderr,
        "syntax: %s\n",
        argv[0]); sent to stderr
    return EXIT_BAD_ARGS;
} Lumetta's favorite return value for
                                bad arguments (enumerated, not 0)
```

ECE 220: Computer Systems & Programming

© 2018 Steven S. Lumetta. All rights reserved.

slide 44

44