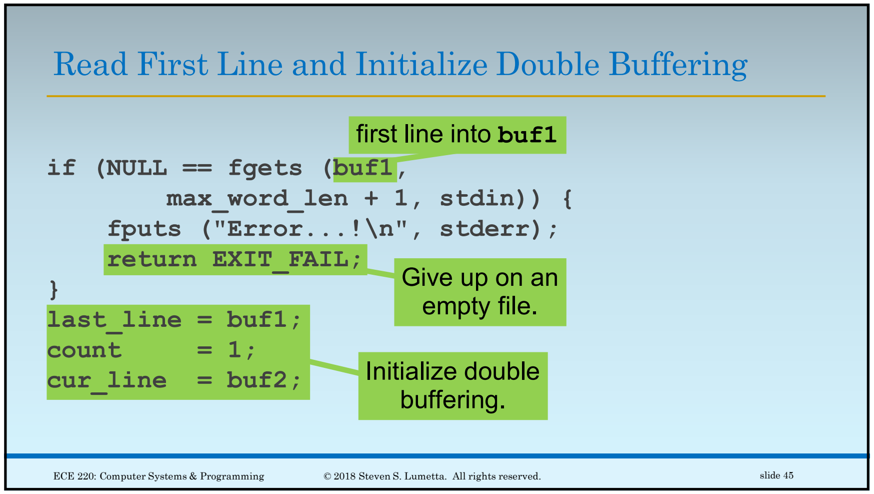


Read First Line and Initialize Double Buffering

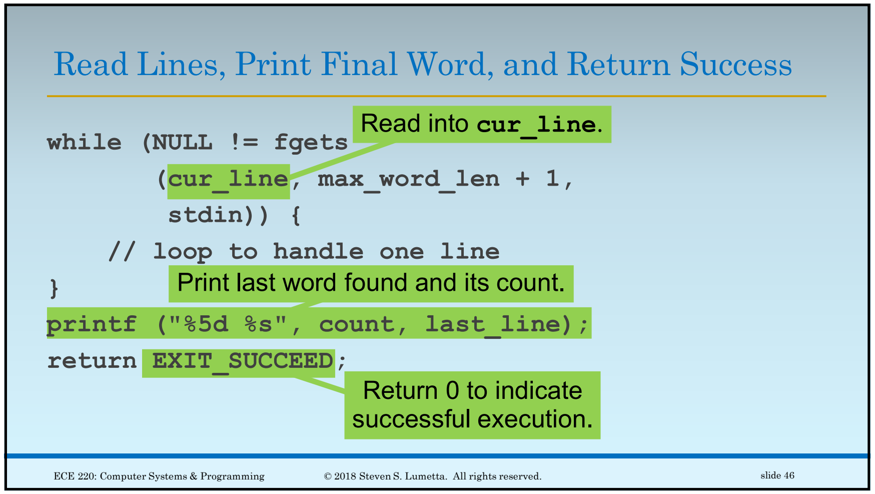
```
if (NULL == fgets (buf1,
        max_word_len + 1, stdin)) {
    fputs ("Error...!\n", stderr);
    return EXIT_FAIL;
}
last_line = buf1;
count     = 1;
cur_line  = buf2;
```

first line into **buf1**

Give up on an empty file.

Initialize double buffering.

45



Read Lines, Print Final Word, and Return Success

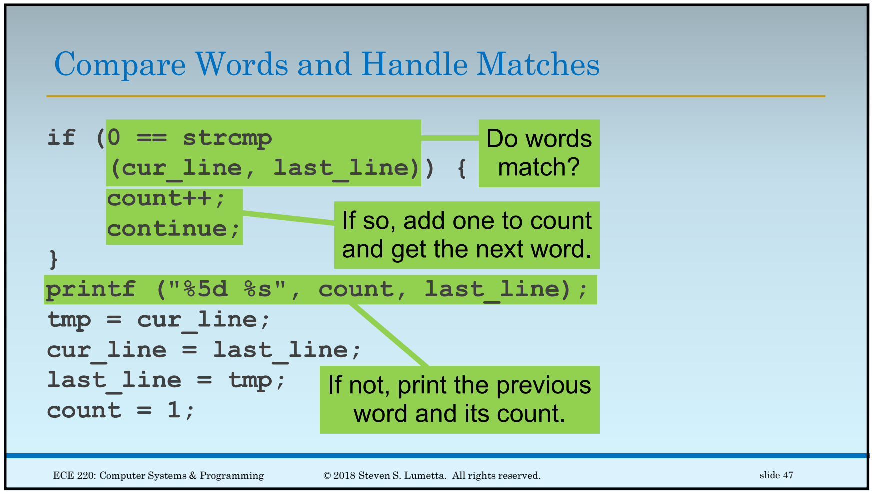
```
while (NULL != fgets
       (cur_line, max_word_len + 1,
        stdin)) {
    // loop to handle one line
}
printf ("%5d %s", count, last_line);
return EXIT_SUCCEED;
```

Read into **cur_line**.

Print last word found and its count.

Return 0 to indicate successful execution.

46



Compare Words and Handle Matches

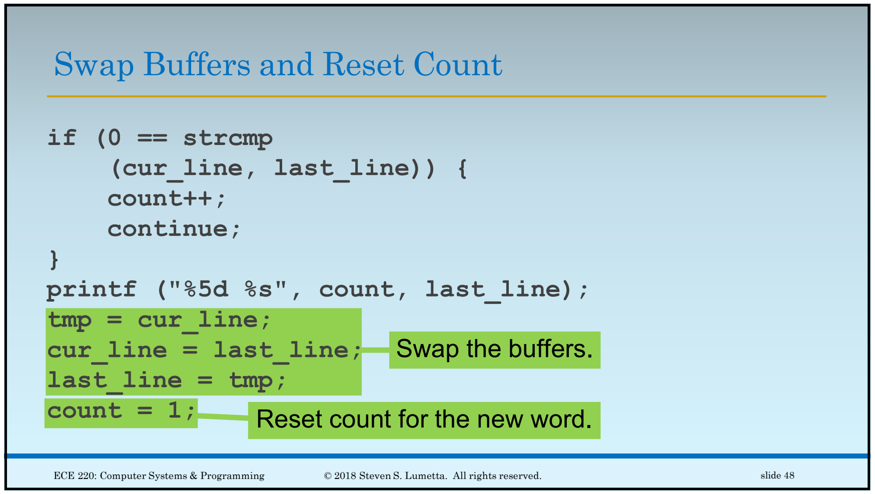
```
if (0 == strcmp
    (cur_line, last_line)) {
    count++;
    continue;
}
printf ("%5d %s", count, last_line);
tmp = cur_line;
cur_line = last_line;
last_line = tmp;
count = 1;
```

Do words match?

If so, add one to count and get the next word.

If not, print the previous word and its count.

47



Swap Buffers and Reset Count

```
if (0 == strcmp
     (cur_line, last_line)) {
    count++;
    continue;
}
printf ("%5d %s", count, last_line);
tmp = cur_line;
cur_line = last_line;
last_line = tmp;
count = 1;
```

Swap the buffers.

Reset count for the new word.

48