CS 241
Data Organization
Lab 2: Counting Characters, Words, and Lines

Brooke Chenoweth

University of New Mexico

Fall 2014

Count lines of input

```
1  #include<stdio.h>
2  void main(void)
3  {
4     char c;
5     int numberOfLines = 0;
6
7     while((c = getchar()) != EOF)
8     {
9        if(c == '\n') numberOfLines++;
10    }
11
12    printf("%d\n", numberOfLines);
13 }
```

Lines 4 and 5:

`int c, nl = 0;` in book

Line 7:

1. Reads character from `stdin`
2. Copies character read into `c`
3. Compares `c` to `EOF`

Lines 8 and 10: Book leaves out brackets.

Count Lines of Input

```
1  #include<stdio.h>
2  void main(void)
3  {
4    int numberOfLines = 0;
5    char c = getchar();
6
7    while(c != EOF)
8    {
9      if(c == '\n') numberOfLines++;
10     c = getchar();
11   }
12   printf("%d\n", numberOfLines);
13 }
```

Why is `getchar()` called on two different lines of code?

Compile: `gcc lineCounter.c`

Run with input: `./a.out < lineCounter.c`

Output: 13

Count Characters, Lines, and Words: 1.5.4

```
#include<stdio.h>
#define IN 1
#define OUT 0
void main(void)
{ int c, nl, nw, nc, state;
  state = OUT;
  nl = nw = nc = 0;
  while((c = getchar()) != EOF) {
    ++nc;
    if (c == '\n')
      ++nl;
    if(c == ' ' || c == '\n' || c == '\t')
      state = OUT;
    else if(state == OUT) {
      state = IN;
      ++nw;
    }
  }
  printf("%d %d %d\n", nl, nw, nc);
}
```

Coding style in textbook:

- Variable names too short.
- Multiple actions in one line.
- Leaves out {} when body of block is only one statement.
- { at end of line.

Counting it all – Top level

```c
#include <stdio.h>

#define IN 1
#define OUT 0

void main(void)
{
  /* Body of function on next slides */
}
```

Counting it all – Main

```
void main(void)
{
  int charCount = 0;
  int totalCharCount = 0;
  int wordCount = 0;
  int totalWordCount = 0;
  int wordState = OUT;
  int lineCount = 1;
  char c = getchar();

  while (c != EOF)
  {
    /* BODY ON NEXT SLIDE */
  }
  printf("%d lines, %d words, %d characters\n",
         lineCount-1, totalWordCount, totalCharCount)
}
```

Counting it all – While body 1

```
if (charCount == 0)
{
  printf("%d)", lineCount);
}

if (c == '\n')
{
  printf("[%d,%d]\n", charCount, wordCount);
  charCount = 0;
  wordCount = 0;
  wordState = OUT;
  lineCount++;
}
else
```

Counting it all – While body 2

```
else /* char just read not '\n' */
{
   charCount++;
   totalCharCount++;
   printf("%c", c);
   if (c == ' ' || c == '\n' || c == '\t')
   {
      wordState = OUT;
   }
   else if (wordState == OUT)
   {
      wordState = IN;
      wordCount++;
      totalWordCount++;
   }
}
c = getchar();
```

Finding longest lines

Add some variables for char counts and line numbers.

```
int mostCharLineNum = 0;
int mostCharCount = 0;
int mostWordLineNum = 0;
int mostWordCount = 0;
```

Output results at end

```
printf("%d lines, %d words, %d characters\n",
        lineCount-1, totalWordCount, totalCharCount);
printf("Line %d has the most words (%d)\n",
        mostWordLineNum, mostWordCount);
printf("Line %d has the most characters (%d)\n",
        mostCharLineNum, mostCharCount);
```

Finding longest lines

At end of lines, check for new longest line.

```
if (c == '\n')
{
  printf("[%d,%d]\n", charCount, wordCount);

  if(charCount >= mostCharCount)
  {
    mostCharCount = charCount;
    mostCharLineNum = lineCount;
  }

  if(wordCount >= mostWordCount)
  {
    mostWordCount = wordCount;
    mostWordLineNum = lineCount;
  }

  charCount = 0;
```