Faculty of Engineering and Applied Science

SOFE 3200U Systems Programming

Lab Report 1

Name: Devante Wilson

Student ID: 100554361

Date: September 20, 2016

Lab Section CRN: Section 3, 44210

Questions

ARGS.c

1. Explain what argc and *argv[] represent.

   argc denotes the amount of command line arguments passed into the main() method.

   ***argv[] passes the literal command line arguments and stores them in an array.**

2. For the code snippet below, explain the overview of what the loop does and what variables i and j are keeping track of.

```
for (i=0; i<argc; i++) {
        j=0;
        while (argv[i][j] != '\0') {
                printf("%c",argv[i][j]);
                j++;
        }
        printf("\n");
}
```

   The loop iterates through and prints out the command line arguments passed in at run time.

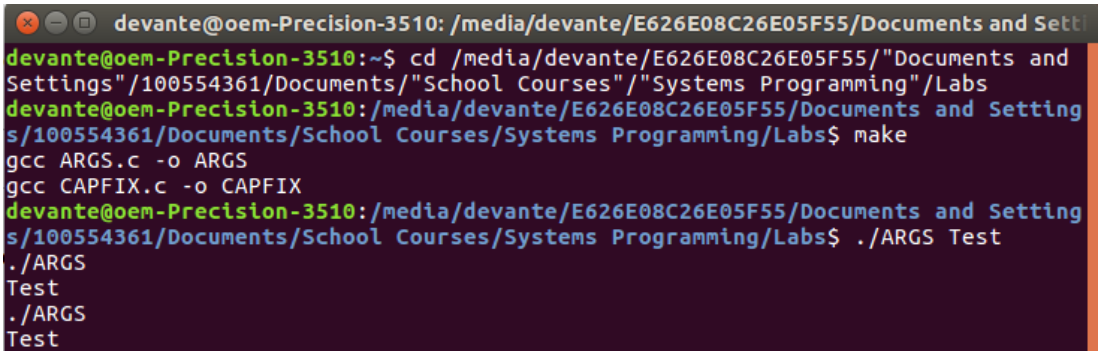
   The variable "i" stores the amount of iterations up to the amount of command line arguments (argc).

   Similarly, "j" keeps track of each literal command line argument character.

3. For the code snippet below, explain the overview of what the loop does and how it is different from the previous loop.

```
for (i=0; i<argc; i++)
        printf("%s\n",argv[i]);
```

   This loop prints the entire literal command line arguments as a character sequence (i.e., string) and not individual characters. This is evident through the % format specifier (%s).



CAPFIX.c

1. Explain what each variable listed in the table below represents.

Variable	Description
*fpt	Pointer to a file object in memory
first[20][30]	Character sequence array to store first names in file
last[20][30]	Character sequence array to store last names in file
i	Keep track of the command line argument amount (up to argc)
j	Keep track of each literal command line argument character
total	Keep track of total number of complete names in the text file (ex. bennett Mike counts as 1 complete name)

2. For the code snippet below, explain the overview of what it does.

```
total=0;
while (1) {
        if (fscanf(fpt,"%s %s",last[total],first[total]) != 2)
                break;
        total++;
}
```

This reads from the text file data and feeds it into each array using a delimiter (“%s %s”) to split up the tokens.

3. For the code snippet below, explain what it does in detail.

```
if (first[i][0] >= 'a' && first[i][0] <= 'z')
        first[i][0]=first[i][0]-'a'+'A';
```

Checks for lower case letters and converts to uppercase (for first names). Each character in the name is iterated through and uses ASCII to convert.

4. For the code snippet below, explain what it does in detail.

```
for (j=1; j<strlen(first[i]); j++)
        if (first[i][j] >= 'A' && first[i][j] <= 'Z')
                first[i][j]=first[i][j]-'A'+'a';
```

The names are iterated through and every letter except the first letter are converted to lower case. (ex. BENNETT → Bennett)