15-122: Principles of Imperative Computation, Fall 2014 Lab 11: Strings in C

Tom Cortina(`tcortina@cs`) and Rob Simmons(`rjsimmon@cs`)

Monday, November 10, 2014

For this lab, you will show your TA your answers once you complete your activities. Autolab is not used for this lab.
SETUP: Make a directory `lab11` in your private `15122` directory and copy the required lab files to your `lab11` directory:

```
cd private/15122
mkdir lab11
cp /afs/andrew.cmu.edu/usr9/tcortina/public/15122-f14/*.c lab11
```

1 Storing and using strings using C

Load the file `lab11ex1.c` into a text editor. Read through the file and write down what you think the output will be before you run the program. (The ASCII value of `'a'` is 97.) Then compile and run the program. Be sure to use all of the required flags for the C compiler. Answer the following questions on paper:

Exercise 1. When `word` is initially printed out character by character, why does only one character get printed?

Exercise 2. Change the program so `word[3] = 'd'`. Recompile and rerun. Explain the change in the output. Run valgrind on your program. How do the messages from valgrind correspond to the change we made?

Exercise 3. Change `word[3]` back. Uncomment the code that treats the four character array `word` as a 32-bit integer. Compile and run again. Based on the answer, how are bytes of an integer stored on the computer where you are running your code?

2 Arrays of strings

Load the file `lab11ex2.c` into a text editor. Read through the file and write down what you think the output will be before you run the program. Then compile and run the program. Be sure to use all of the required flags for the C compiler. Answer the following questions on paper:

Exercise 4. What do you think happens if you reset `num_states` to 7? Edit the file, compile and run it. Explain the output you see. Also use valgrind and report what issue it finds and how this relates to the error.

Exercise 5. We never free any memory in this program, yet valgrind reports no memory leaks. Why? Where are these strings stored? Where is the pointer variable `states` stored?

3 C string libraries

The header file `string.h` outlines a number of string functions that can be used (often incorrectly) in C programs. They include:

```
char *strcpy(char *dest, const char *src)
char *strncpy(char *dest, const char *src, size_t n)
size_t strlen(const char *str)
```

Read about how these functions work here:

`http://en.wikipedia.org/wiki/C_string_handling#Functions`

These functions assume that the pointers point to a NUL-terminated string (i.e. a string that ends with, `'\0'`, ASCII value 0). Load the file `lab11ex3.c` into a text editor. Read through the file and write down what you think the output will be before you run the program. Then compile and run the program. Be sure to use all of the required flags for the C compiler. Answer the following questions on paper:

Exercise 6. Did the result surprise you? Can you explain what happened? (HINT: The `buffer` array is stored on the system stack along with the return location back to `main`.)

Exercise 7. Comment out the call to `code1` and uncomment the call to `code2`. Note that `code2` uses `strncpy` instead of `strcpy`. Compile and run again. Explain the result this time.

4 Programming with C strings

Exercise 8. Write a C function that reverses a string and returns a pointer a new string with the result. The function should have the following prototype:

```
char *reverse(char *s);
```

Write a main function to test your function on a number of strings. Include only those header files that are necessary to compile your code. If you allocate memory, use `calloc` and be sure to free what you allocate.