|  |  |
| --- | --- |
| CSE422: Operating Systems Organization | Spring 2014 |
| Lab 0 | |
| Yifu Ma | Due: Jan 24th |

## Part A (20 points)

In this part, you are to answer the questions below. Feel free to code up and run these programs to test them out. You could even step through them with a debugger. However, these are more “why” questions, so these measures will help most if you are stuck or for seeking confirmation.

Try to identify the key to each problem and keep your answers concise and to the point. These questions bring up important points about pointer usage and control flow in C. Keep these in mind when working on the remainder of the lab.

1. Consider the following C program.

#include <string.h>

int main(int argc, char \*\*argv)

{

char \*temp;

strcpy(temp, argv[0]);

return 0;

}

Why is the above code incorrect (i.e. likely to crash)?

***Answer*** *no space allocated for temp*

1. Consider the following C program.

#include <string.h>

int main(int argc, char \*\*argv)

{

char \*temp;

temp = argv[0];

return 0;

}

Why does this program work when the previous question did not?

***Answer*** *this simply change the pointer to point to argv, so no memory allocation is needed*

1. Consider the following C program.

#include <string.h>

int main(int argc, char \*\*argv)

{

char temp[9];

int i = 0;

strcpy(temp, argv[1]);

return i;

}

A buffer overflow occurs when the argument is 9 characters long. Why?

***Answer*** *there is a \n at the end of string*

1. Consider the following C program.

#include <string.h>

int main(int argc, char \*\*argv)

{

char \*buffer = "Hello";

strcpy(buffer, "World");

return 0;

}

Why does this program crash?

***Answer*** *because buffer points to string literals and it is not allowed to change literals*

1. Consider the following C function.

void myfunc()

{

char b[100];

char \*buffer = &b[0];

strcpy(buffer, "World");

}

Is this correct? What data type does &b[0] refer to? What is a simpler expression for this data type? In what situation might you want to use the &b[0] notation instead?

***Answer*** *It is correct .It refers to a string. Simpler expression will be char\*. When we want to specify the length of the buffer.*

1. Consider the following C program.

#include <stdio.h>

int main(int argc, char\* argv[])

{

printf("%s %s %s\n",\*argv, (\*(argv+1)) + 2, \*(argv+2));

return 0;

}

If this code is executed with the line

program5 –n5 abc

what will be the output?

***Answer*** *ogram5 –n5*

1. Consider the following C program.

#include <stdio.h>

#include <string.h>

char \*myfunc(char \*\*argv)

{

char buffer[100];

strcpy(buffer, "hello");

return buffer;

}

int main(int argc, char \*argv[])

{

char \*s = myfunc(argv);

printf("%s\n", s);

}

What’s wrong with this?

***Answer*** *argv passed into myfunc is never used*

1. This question is not about pointers, but about control flow. Consider the following code snippet.

switch (c) {

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

printf("%c is a vowel.\n", c);

break;

case 'y':

printf("y is sometimes a vowel.\n");

default:

printf("%c is a consonant.\n", c);

}

Why are two messages printed if the variable *c* refers to the value *‘y’*? Is a message printed when *c* refers to the value *‘a’*?

***Answer*** *because there is no break statement after y case so the default case will also be executed. Yes there is, also the default case.*

Note: with these questions in mind, you should only use strncpy in your programs.

## Part B (20 points)

In this part, we will explore the accuracy of your solution. Your values need not be exact, but should be in the general ballpark. Let’s define what the ballpark is.

The commands

grep –o ‘search’ file | wc -w

and

grep –o –w ‘search’ file | wc -w

provide a good range. Your solution should probably be between these two numbers inclusively. If you want to learn about these commands, the man page provides plenty of information.

We will be searching through two books courtesy of Project Gutenberg. One is called *A Trip To Mars* by Fenton Ash. It is around 200 pages, yet can be represented as a ~450 KB text file. The second book is *Anna Karenina* by Leo Tolstoy. This book is nearly 1000 pages long, yet can be represented as a ~2 MB text file. Pretty impressive! We will have our word count program search through these **entire books**.

Run your program via the following command

lab0 –f dict1.txt TripToMars.txt

and paste its output below:

***Answer*** Mars, 47

Tom, 60

Gerald, 273

Jack, 189

meteorite, 8

Now, run the grep commands

grep –o ‘Tom’ TripToMars.txt | wc -w

and

grep –o –w ‘Tom’ TripToMars.txt | wc -w

How did the number your solution gave for *‘Tom’* compare to grep’s numbers? Where does it lie in regards to the ballpark? If it is not inside the ballpark, why do you think that is? (*Hint:* If it is not inside the ballpark, what assumptions did you make which might have contributed to that?)

***Answer*** *63 instead of 60. Numbers are lower, due to special chars not completely eliminated by strtok*

What are some possible strings that would be counted by the command without the *–w* option but not counted by the grep with the ­*–w* option? For these questions, feel free to look through the search files. You **can** think of these strings off of the top of your head, but you certainly can find the actual strings in the file.

***Answer*** *strings like* ***Jack\_ or Jack02***

Run your program with the following command

lab0 –f dict2.txt AnnaKarenina.txt

and paste its output below.

***Answer*** Oblonsky, 110

Moscow, 162

Dolly, 297

Stiva, 65

Anna, 798

Petersburg, 123

Kitty, 655

Vronsky, 837

Levin, 1561

Betsy, 108

Karenin, 49

Now, run the commands

grep –o ‘Oblonsky’ AnnaKarenina.txt | wc -w

and

grep –o –w ‘Oblonsky’ AnnaKarenina.txt | wc -w

How did the number your solution gave for *‘Oblonsky’* compare to grep’s numbers? Where does it lie in regards to the ballpark? If it is not inside the ballpark, why do you think that is?

***Answer*** *127 compared to my 110.*

What are some possible strings that would be counted by the command without the *–w* option but not counted by the grep with the ­*–w* option? For these questions, feel free to look through the search files. You **can** think of these strings off of the top of your head, but you certainly can find the actual strings in the file.

***Answer***

Lastly, run the commands

grep –o ‘Levin’ AnnaKarenina.txt | wc -w

and

grep –o –w ‘Levin’ AnnaKarenina.txt | wc -w

How did the number your solution gave for *‘Levin’* compare to grep’s numbers? Where does it lie in regards to the ballpark? If it is not inside the ballpark, why do you think that is?

***Answer*** *0 compared to 1627. Might be s problem with parsing dict file.*

Now that you have used it quite a bit, explain in your own words what the grep command with the *–w* option accomplishes.

***Answer*** *it only searches for whole words that are not substrings of any other string*

In your own words, explain what the grep command without the *–w* option accomplishes.

***Answer*** *it searches for any matching string regardless of being a substring or not*