CIS 452

Lab 1 Report

Ashley Hendrickson

Muna Gigowski

Fall 2019

Table of Contents

online help 3

Question One 3

Question Two 3

include files 3

Question Three 3

Useful UNIX Fundamentals 3

Question Four 3

Question FIve 4

Program Debugging 4

Question Six 4

Dynamic Memory Debugging 4

Question Seven 4

System profiling 5

Question Eight 5

Question Nine 6

Programming assignment 6

Source Code 6

# Online Help

## Question One

**Briefly describe the purpose of (what is found in) any two sections commonly found on a man page (e.g. the Synopsis section of the malloc() system call).**

The Synopsis: shows other related methods that are typically used together

The description: describes how to use the methods and other similar methods in the synopsis

## Question Two

**Describe the difference between the UNIX shell command time and the UNIX system call time().**

The primary difference is that the system call is not linked to a user program and executes in kernel space, without user intervention. The shell command, on the other hand, is a command that is initiated by the user (either directly or indirectly), so it is not a part of the kernel.

# Include Files

## Question Three

**What is the meaning of the stream-based SEEK\_CUR macro?**

This macro is used in relation to file positioning in a stream, which describes where in the file the stream is currently reading or writing. SEEK\_CUR is an integer constant that is often used as an argument in the fseek or fseeko functions to specify that he offset provided is relative to the current file position. The value for SEEK-CUR may be positive or negative.

# Useful UNIX Fundamentals

## Question Four

**What is the command to list the contents of a directory in long mode, including hidden files?**

ls -l -a

## Question Five

**What is the command syntax to make a directory readable/writable only by you?**

chmod u+rw -R directory/

# Program Debugging

## Question Six

**Script from our C file operations:**

gcc -g lab1.c -o lab1 // gdb lab1 // b main // run //

# Dynamic Memory Debugging

## Question Seven

**Describe precisely (nature of problem, location) the memory leak(s) in Sample Program, fix the problem(s), and submit your corrected source code.**

The first memory leak in Sample Program 2 occurs because of the line:

data2 = malloc (SIZE);

This line itself is not an issue, as it is just telling the system to allocate memory for the variable data2. The problem arises at the end of the main() function, because by the time the function exists the memory that was allocated for data2 should have been freed (aka deallocated) using the free() function, just as the program did for the data1 variable. However, since it did not also free() the memory that malloc() previously allocated for data2, a memory leak is created. Dynamically allocated memory should always be deallocated after it is used.

The second memory leak that occurs in Sample Program 2 is more subtle, but still very important. This memory leak occurs after the line:

if (!strcmp (data1, "quit"))

The issue here is that if the condition is met and the break is hit, the loop will exit before hitting the line that deallocates the memory that malloc() allocated for the data1 variable (which occurs much farther below in the loop). Even though this would only happen under one condition, it is still important that the memory gets properly freed before the loop is broken out of, otherwise a memory leak will occur.

CORRECTED SOURCE CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 16

int main()

{

char \*data1, \*data2;

int i;

do {

       data1 = malloc (SIZE);

       printf ("Please input your eos username: ");

       scanf ("%s", data1);

  if (!strcmp (data1, "quit")) {

free(data1);

      break;

   }

  data2 = malloc (SIZE);

       for (i=0; i<SIZE; i++)

          data2[i] = data1[i];

       free (data1);

       printf ("data2 :%s:\n", data2);

       free (data2);

} while (1);

return 0;

}

# System Profiling

## Question Eight

**How many times is the write() system call invoked when running Sample Program 2?**

write() gets invoked 1 time when running Sample Program 2, if we enter “quit” for the prompted username the first time, else it gets invoked 1 + the amount of times we enter a name value that is not “quit”.

## Question Nine

**Examine the source code and output to answer the question: what is the primary 'C' library subroutine that causes the write() system call to be invoked while executing Sample Program 2?**

The primary ‘C’ library subroutine that causes the write() system call to be invoked is the printf() subroutine.

# Programming Assignment

## Source Code

Here is a link to our Github source code for the Lab 1 programming assignment:

<https://github.com/hendrash/cis452/tree/master/lab/1>