Brandon Baker
Brendan Coffman
CIS 452-03
1/11/24

1. Description – Explains what the command does that you are currently in the manpage for. Also lists command line flags and describes what those do for the command.

   Environment – Describes what environment variables affect the execution of the given command and how.

2. write in the terminal is a command line utility that allows you to communicate with different users on the device through the terminal. You must follow up with a user to send the message to as an argument (within the same system). Optionally, you can specify the terminal to write to.

   write() is a system call that interacts with low-level code commonly used in C code. It takes 3 arguments, a file or location to write to, a pointer to the data, and the number of bytes to write.

3. SEEK_SET is the offset in bytes from the start of a file. Therefore it would be equivalent to a cursor that is set relative to the beginning of the file. On the man page for "llseek", the following statement can be found, "This new offset is a byte offset relative to the beginning of the file, the current file offset, or the end of the file, depending on whether whence is SEEK_SET, SEEK_CUR, or SEEK_END, respectively."

4. The command to long lookup files is "ls -l" to view the hidden files would be "ls -a". They can both be combined into "ls -la" to do a long lookup on all files including hidden ones.

5. The command to change directory permissions so that they are only readable, writable, and executable to only you is. The command to do this is chmod u+rwx,g-rwx,o-rwx name_of_directory.

6.
```
(gdb) b 4
Breakpoint 1 at 0x1175: file sampleProgramOne.c, line 6.
```

```
(gdb) run
Starting program: /home/bakebran/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at sampleProgramOne.c:6
6           double num = 0.0;
(gdb) n
8           printf ("Hello, world.\n");
(gdb) p num
$1 = 0
(gdb) n
Hello, world.
9           num = pow(2, 28);
(gdb) n
10          printf ("You are the %f person to write this program!\n", num);
(gdb) p num
$2 = 268435456
```

Brandon Baker
Brendan Coffman
CIS 452-03
1/11/24

7. In sample program 2, The memory leak occurs where the variable “data2” is malloced but never free’d. Because it is never free’d, this causes the memory leak spotted in valgrind. In the screenshot below, under heap summary, valgrind shows that memory was allocated on line 14 which caused the leak.

```
Please input your eos username: test1
data2 :test1:
Please input your eos username: test2
data2 :test2:
Please input your eos username: ^C==558767==
==558767== Process terminating with default action of signal 2 (SIGINT)
==558767==    at 0x498B5F2: read (read.c:26)
==558767==    by 0x4903C35: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:516)
==558767==    by 0x4904D95: _IO_default_uflow (genops.c:362)
==558767==    by 0x48DA0CF: __vfscanf_internal (vfscanf-internal.c:628)
==558767==    by 0x48D9141: __isoc99_scanf (isoc99_scanf.c:30)
==558767==    by 0x109211: main (sampleProgram2.c:11)
==558767==
==558767== HEAP SUMMARY:
==558767==     in use at exit: 2,096 bytes in 5 blocks
==558767==   total heap usage: 7 allocs, 2 frees, 2,128 bytes allocated
==558767==
==558767== 16 bytes in 1 blocks are definitely lost in loss record 3 of 5
==558767==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==558767==    by 0x109235: main (sampleProgram2.c:14)
==558767==
==558767== LEAK SUMMARY:
==558767==    definitely lost: 16 bytes in 1 blocks
==558767==    indirectly lost: 0 bytes in 0 blocks
==558767==      possibly lost: 0 bytes in 0 blocks
==558767==    still reachable: 2,080 bytes in 4 blocks
==558767==         suppressed: 0 bytes in 0 blocks
==558767== Reachable blocks (those to which a pointer was found) are not shown.
==558767== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==558767==
==558767== For lists of detected and suppressed errors, rerun with: -s
==558767== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Brandon Baker
Brendan Coffman
CIS 452-03
1/11/24

Corrected 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"))
         break;

      data2 = malloc (SIZE);
      for (i=0; i<SIZE; i++)  data2[i] = data1[i];

      free (data1);
      printf ("data2 :%s:\n", data2);
      free (data2); //data2 was not free then malloc was done a second time
   } while (1);

      return 0;
}
```

Brandon Baker
Brendan Coffman
CIS 452-03
1/11/24

Once corrected:

```
coffmanb@eos15:~/CIS/cis-452/lab1$ vim q2.c
coffmanb@eos15:~/CIS/cis-452/lab1$ gcc q2.c -o q2 -g
coffmanb@eos15:~/CIS/cis-452/lab1$ valgrind --leak-check=full ./q2
==226360== Memcheck, a memory error detector
==226360== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==226360== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==226360== Command: ./q2
==226360==
Please input your eos username: coffman
data2 :coffman:
Please input your eos username: dlkdj
data2 :dlkdj:
Please input your eos username: sldkjfdlkjf
data2 :sldkjfdlkjf:
Please input your eos username: slsss
data2 :slsss:
Please input your eos username: ^C==226360==
==226360== Process terminating with default action of signal 2 (SIGINT)
==226360==    at 0x498B5F2: read (read.c:26)
==226360==    by 0x4903C35: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:516)
==226360==    by 0x4904D95: _IO_default_uflow (genops.c:362)
==226360==    by 0x48DA0CF: __vfscanf_internal (vfscanf-internal.c:628)
==226360==    by 0x48D9141: __isoc99_scanf (isoc99_scanf.c:30)
==226360==    by 0x109211: main (q2.c:13)
==226360==
==226360== HEAP SUMMARY:
==226360==     in use at exit: 2,064 bytes in 3 blocks
==226360==   total heap usage: 11 allocs, 8 frees, 2,192 bytes allocated
==226360==
==226360== LEAK SUMMARY:
==226360==    definitely lost: 0 bytes in 0 blocks
==226360==    indirectly lost: 0 bytes in 0 blocks
==226360==      possibly lost: 0 bytes in 0 blocks
==226360==    still reachable: 2,064 bytes in 3 blocks
==226360==         suppressed: 0 bytes in 0 blocks
==226360== Reachable blocks (those to which a pointer was found) are not shown.
==226360== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==226360==
==226360== For lists of detected and suppressed errors, rerun with: -s
==226360== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

coffmanb@eos15:~/CIS/cis-452/lab1$
```

8. The formula for write system calls: Number of Calls = 2n + 1 (where n is the number of loops) The 2n is due to the write system call getting called twice through each loop. However, since it is nearly impossible to stop before the first call of write (without a breakpoint) an additional 1 is added.

9. The primary C subroutine that causes the write() system call to be invoked is the printf() method. Each time the printf() method is called, a write system call executes. So, once the loop in the source code is entered, the write system call is executed due to the printf() that occurs before taking user input and then once at the end of the main loop.