

Sebastian Torrejon

Fabian Kirberg

Lab1

CIS 452

Fall 2023

1. 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)

Synopsis: Shows how to call the related commands and what arguments can be passed in.

Description: Gives detailed descriptions of what the commands listed in the synopsis do.

2. Describe the difference between the UNIX shell command write and the UNIX system call write()

The shell command write is used to communicate with other users by copying lines from one terminal to another. The system call write() is used to manipulate data. The shell command is a high-level interaction and can be used by any user, and the system call is a low-level operation used by programs.

- They both write data but at a different level of the UNIX environment.
- The shell command is a higher-level interaction for sending messages between users, while the system call is a lower-level operation for writing data to file descriptors, which can encompass a wide range of data manipulation scenarios.

3. What is the meaning of the stream-based SEEK_SET macro?

Seek from the beginning of the file

- `>man -k seek` will show fseek function which relates to a stream.
- `>man -3 fseek` will show the include file `<stdio.h>` that requires this function to work
- `>cd /usr/include/` navigate to directory where stdio.h is
- `>cat stdio.h` will print its contents and we can find the meaning of seek_set macro

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

`ls -al`

- `-l` is an option that specifies long mode
- `-a` is an option that includes hidden files

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

`chmod 700 <directory_name>`

- The owner (you) is granted read (r), write (w), and execute (x) permissions using 7. Group and others have no permissions, represented by 0.

6. Perform the following operations:

- Create the above sample program (via copy/paste)
- Compile the program (remember to include debugging information and link all necessary libraries)
- Run the sample program
- Start the debugger on your program (gdb sampleProgramOne)
- Set a breakpoint at the function main
- Take a screenshot (screenshotOne)

```
kirbergf@DESKTOP-2B8TP1M:~/cis452$ ls
a.out sampleProgramOne.c
kirbergf@DESKTOP-2B8TP1M:~/cis452$ gdb a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(No debugging symbols found in a.out)
(gdb)
(gdb)
(gdb)
(gdb) b main
Breakpoint 1 at 0x1169
(gdb) _
```

- Run your program within the debugger (run) and step through it
- Use the debugger to print the value of num before and after it changes
- Take a screenshot (screenshotTwo)

```
Reading symbols from a.out...
(gdb) b main
Breakpoint 1 at 0x1169: file sampleProgramOne.c, line 5.
(gdb) b 9
Breakpoint 2 at 0x1197: file sampleProgramOne.c, line 9.
(gdb) run
Starting program: /home/kirbergf/cis452/a.out


Breakpoint 1, main () at sampleProgramOne.c:5
5      {
(gdb) print num
$1 = 0
(gdb) continue
Continuing.
Hello, world.

Breakpoint 2, main () at sampleProgramOne.c:9
9      printf ("You are the %f person to write this program!\n", num);
(gdb) print num
$2 = 268435456
(gdb) _
```

7. Describe precisely (nature of problem, location) the memory leak(s) in

sample program2. Fix the problem(s) and submit your corrected source code. Note: it would be a good idea to use valgrind to verify that you have fixed the problem(s)!

The memory leak issues occurred when the loop executed multiple times because the memory for data2 was not being freed before being reallocated, we solved this by adding a free(data2) statement at the end of the do-while loop. Also the minor issue of unrecovered memory was that data1 was not being freed after entering "quit", we solved this by adding a free(data1) statement after the do-while loop.



```
kirbergf@DESKTOP-2B8TP1M: ~/cis452
#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);
    } while (1);
    free (data1);
    return 0;
}
```

8. How many times is the write() system call invoked when running sample program 2? Experiment with executing the loop a different number of times. Then express your answer as a formula.

$2x - 1$ where x is the number of times the do-while loop is run. Or x is the number of inputs we give the program.

9. 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 printf() subroutines, such as printf("Please input your eos username: ") and printf("data2 :%s:\n", data2).

If we delete a printf from the code, the amount of write() calls will decrease.