CS481 HW1

Ryan Scherbarth & Danny Metcalfe

September 2024

1. Create and compile C program

(a)

Without creating file *myTstFile* (no such a file in your working dir), run "strace ./P01", read your output, pick the first five of most frequently invoked system calls, and explain which among the five system calls caused the program to fail.

- The most frequent system call is to **mmap()** which can be used to map files or devices into a process's memory. I would not say that mmap caused any issues in the program and there are no error returns from any mmap here.
- The second most frequent system call is to **mprotect()** which is used to change the memory access protections of memory regions and there is no error return in any of them in this program. So there is no failure here.
- The third most frequent call is to **pread()** read data from a file descriptor at a certain offset and this does not seem to cause any errors.
- The fourth most frequent is **brk()** which I believe is used for memory heap allocation and this does not seem to be causing any errors.
- The fifth most frequent call was to **openat()** which is to open a file and here it shows a return of -1 for file not found. I believe this is the problem causing the program to fail and enter the if statement since the system call does not find the file being asked for. It returns -1 for an error or file not found error.

Overall I would say the main system call causing problems is going to be **openat()** since it could not find any file in the last call to openat and returned an error value of -1.

(b)

With file $myTstFile$ (a dummy file) being created, run "strace ./PA01" again, read your output to pick the most frequently invoked system call.

After creating the file it is clear that the most frequent system call is **write()** and some of these write calls will end up printing to the terminal and the others seem to be used to write to the file "myTstFile".

(c) is "fopen" a system call? if not, which system call it mainly correlates with?

"fopen" is not a system call, it is a c library function call. A similar system function call would be "openat" and it looks like the system call is used to open a file given some descriptors, flags, and a file name.

(d) is "printf" a system call? if not, which system call it mainly correlates with?

"printf" is not a system call, as with fopen it is a c library function. A similar system call would be "write". The write function can write to files and an output like the terminal.

2. Run "strace -c cal", capture the output, and then pick the top three system calls which consumed the system time and briefly descirbe their functionality

   1:

   write

   The write system call is what we use to write data to a file descriptor, typically outputting text to the terminal or another file.

   2:

   newfstatat

   Used to retrieve information about a file (metadata like size, permissions etc.). Usually used to check the status of files opened or accessed by the program.

   3:

   mmap

   system call maps files or devices into memory, allowing for efficient access to file contents or shared memory regions. We probably use it in cal to load the other libraries we need into memory for quicker access.

3. Strace / Ltrace linux utility commands "ls"

(a) Open the current directory

The primary calls are the openat command;

openat(AT_FDCWD, ".", O_RDONLY—O_NONBLOCK—O_CLOEXEC—O_DIRECTORY)

This command opens the directory, "." in this case, with the permissions as read-only.

and the second;

newfstatat(3, "", st_mode=S_IFDIR—0775, st_size=4096, ..., AT_EMPTY_PATH)

takes place after opening the directory. fstab is used to retrieve the metadata about the directory file. For example, you can do "ls -lah" and view some of this metadata such as the file size, modification date, etc.

(b) Get the list of directory entries

1:

getdents64(3, 0x5cef50df29f0 /* 11 entries */, 32768)

2:

getdents64(3, 0x5cef50df29f0 /* 0 entries */, 32768)

3:

readdir(0x5e87a2daf9c0)

4:

opendir(".")

1 is the system call we use to retrieve all of the directories, in this case there are 11. Following, 2 is what we use to determine that the end of the directory has been reached, as it has found 0.
3 is a library function, which is what we use to iterate through each of the directories. We then use 4, another library function, to open each of the directories.

(c) Print the output to your screen

1:

write(1, "481 cs251 cs341 cs351 cs357 "..., 66)

2:

fwrite_unlocked("...", 1, length, stdout)

3:

fflush(stdout)

1 is the system call we use to write the directory list directly to standard output. This is used together with the ioctl function to determine screen size. 2 and 3 are library functions, 2 helping write formatteed output to the terminal, and 3 writes and clears the buffer.

4. Team work distribution

Danny Compiled the C program and answered the follow up questions for question 1. Ryan ran the commands and evaluated the output for questions 2 and 3. We then both read over each others' work and suggested changes before the final submission.