Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

New York University
Tandon School of Engineering

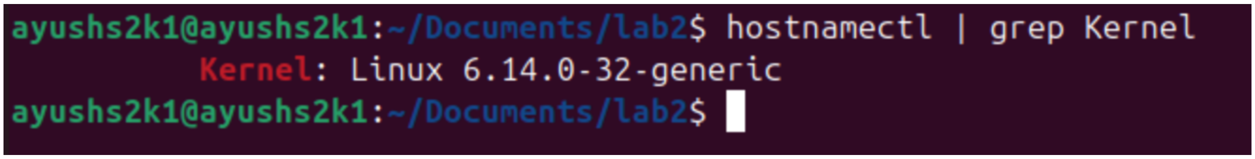
Department of Computer Science and Engineering

Introduction to Operating Systems

Fall 2025

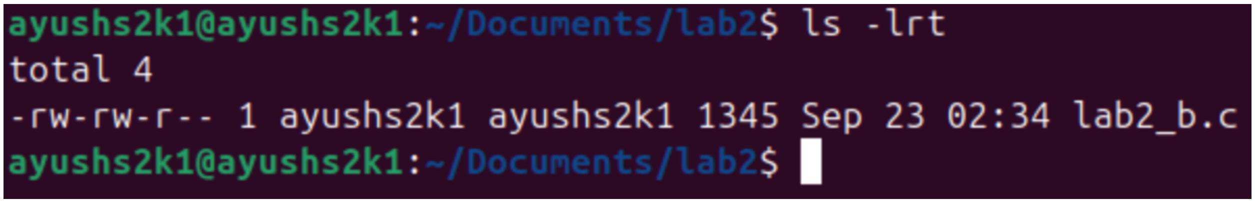
Assignment 2

Kernel Version: Linux 6.14.0-32-generic



```
ayushs2k1@ayushs2k1:~/Documents/lab2$ hostnamectl | grep Kernel
         Kernel: Linux 6.14.0-32-generic
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```

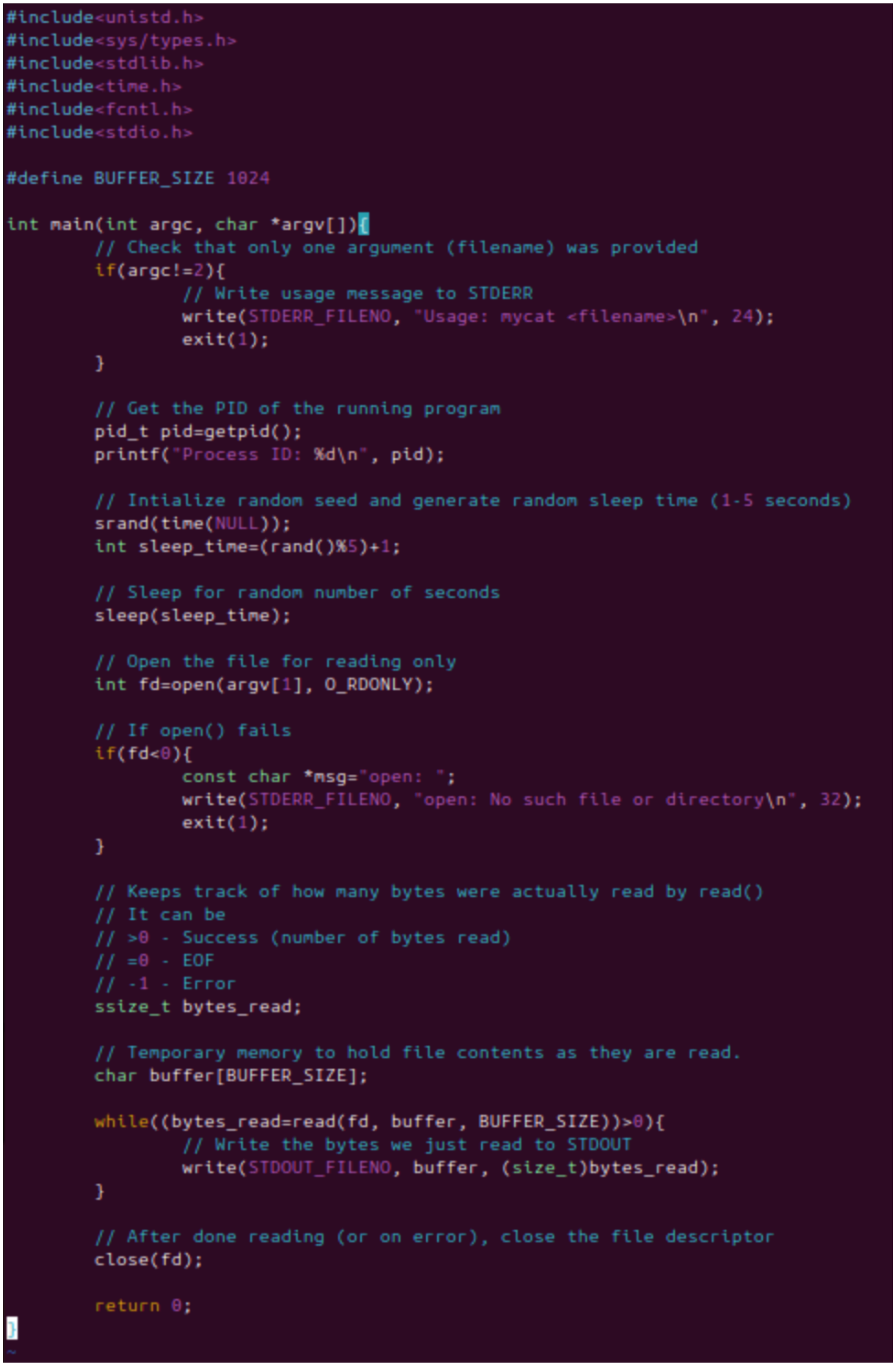
Current Directory: lab2



```
ayushs2k1@ayushs2k1:~/Documents/lab2$ ls -lrt
total 4
-rw-rw-r-- 1 ayushs2k1 ayushs2k1 1345 Sep 23 02:34 lab2_b.c
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

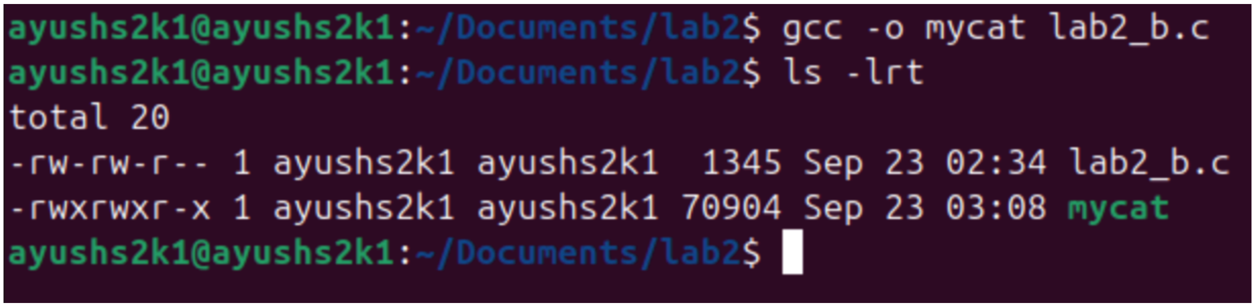
C program whose main routine accepts an input text file from the user, prints the PID of the running program, sleeps for a random number of seconds (1-5 seconds), and prints the contents of the input file.



```
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<time.h>
#include<fcntl.h>
#include<stdio.h>

#define BUFFER_SIZE 1024

int main(int argc, char *argv[]){
        // Check that only one argument (filename) was provided
        if(argc!=2){
                // Write usage message to STDERR
                write(STDERR_FILENO, "Usage: mycat <filename>\n", 24);
                exit(1);
        }

        // Get the PID of the running program
        pid_t pid=getpid();
        printf("Process ID: %d\n", pid);

        // Intialize random seed and generate random sleep time (1-5 seconds)
        srand(time(NULL));
        int sleep_time=(rand()%5)+1;

        // Sleep for random number of seconds
        sleep(sleep_time);

        // Open the file for reading only
        int fd=open(argv[1], O_RDONLY);

        // If open() fails
        if(fd<0){
                const char *msg="open: ";
                write(STDERR_FILENO, "open: No such file or directory\n", 32);
                exit(1);
        }

        // Keeps track of how many bytes were actually read by read()
        // It can be
        // >0 - Success (number of bytes read)
        // =0 - EOF
        // -1 - Error
        ssize_t bytes_read;

        // Temporary memory to hold file contents as they are read.
        char buffer[BUFFER_SIZE];

        while((bytes_read=read(fd, buffer, BUFFER_SIZE))>0){
                // Write the bytes we just read to STDOUT
                write(STDOUT_FILENO, buffer, (size_t)bytes_read);
        }

        // After done reading (or on error), close the file descriptor
        close(fd);

        return 0;
}
```

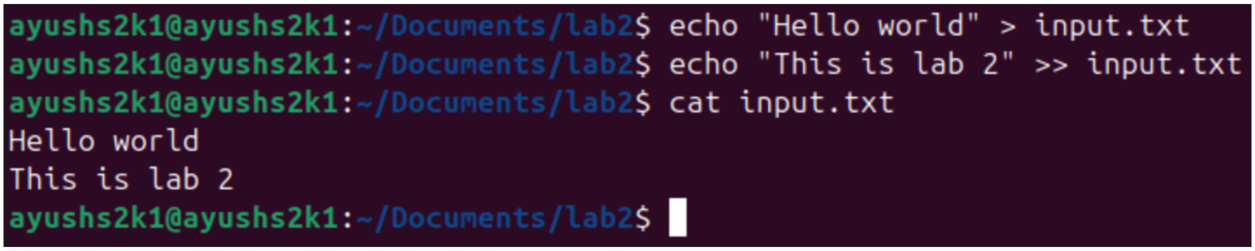
Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Command used to compile the program: gcc -o mycat lab2_b.c
After compiling, it creates an executable file, mycat, as shown below.



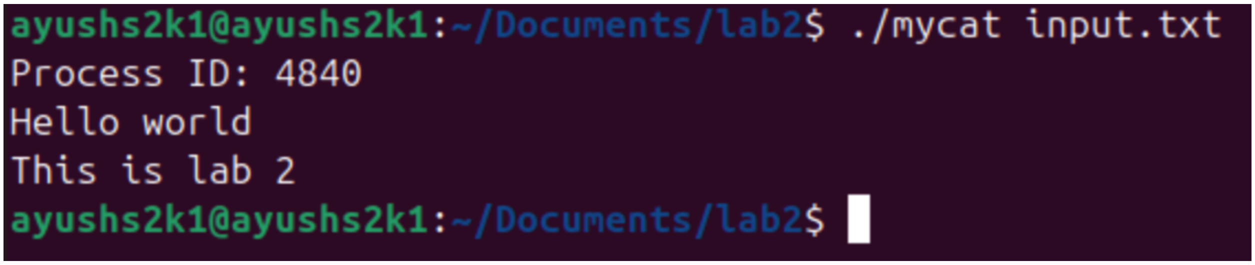
```
ayushs2k1@ayushs2k1:~/Documents/lab2$ gcc -o mycat lab2_b.c
ayushs2k1@ayushs2k1:~/Documents/lab2$ ls -lrt
total 20
-rw-rw-r-- 1 ayushs2k1 ayushs2k1  1345 Sep 23 02:34 lab2_b.c
-rwxrwxr-x 1 ayushs2k1 ayushs2k1 70904 Sep 23 03:08 mycat
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```

Creating a text file and using it to test the program.



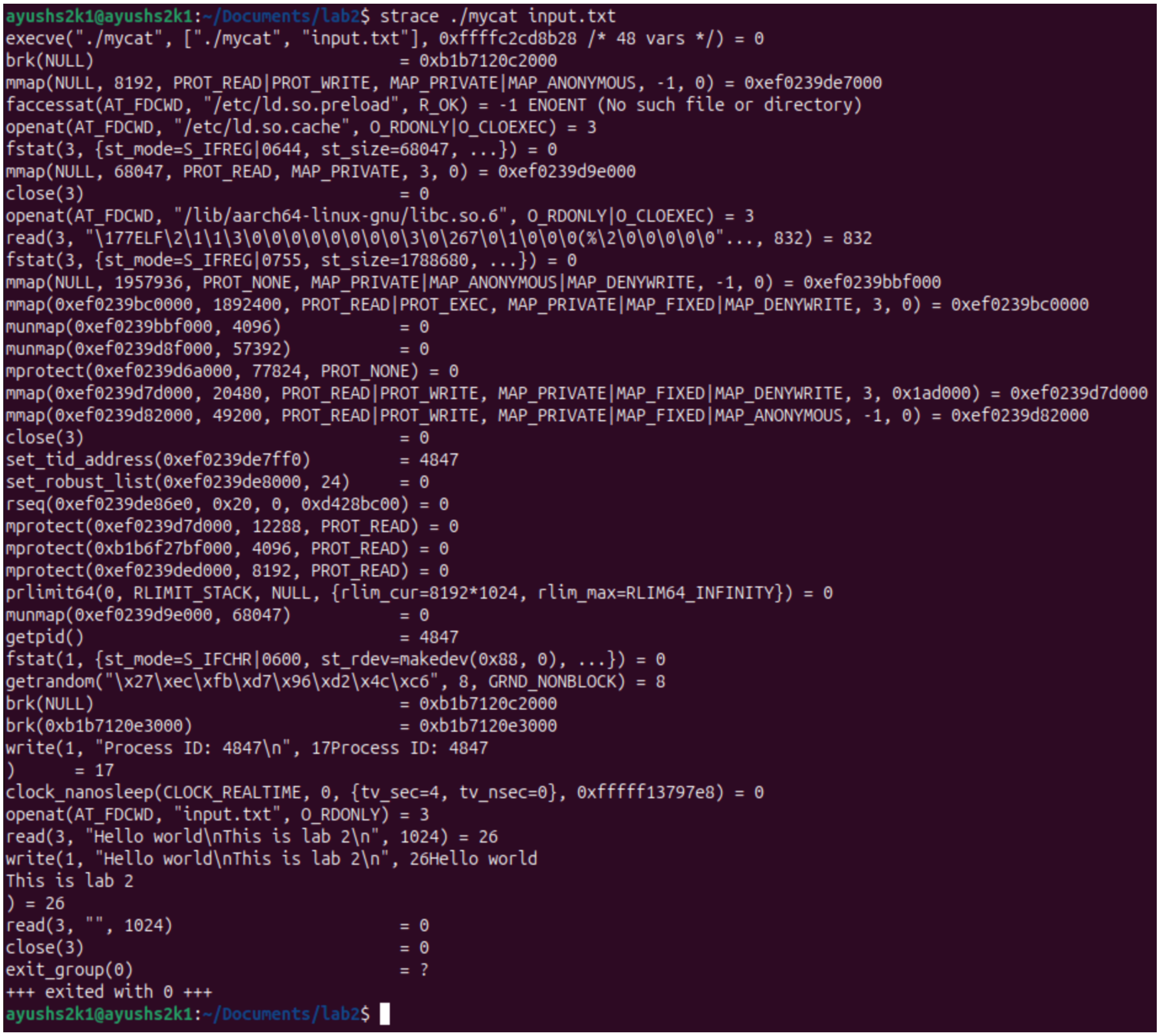
```
ayushs2k1@ayushs2k1:~/Documents/lab2$ echo "Hello world" > input.txt
ayushs2k1@ayushs2k1:~/Documents/lab2$ echo "This is lab 2" >> input.txt
ayushs2k1@ayushs2k1:~/Documents/lab2$ cat input.txt
Hello world
This is lab 2
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```

Running the executable to print the PID of the running program, sleeps for a random number of seconds (1-5 seconds), and prints the contents of the input file

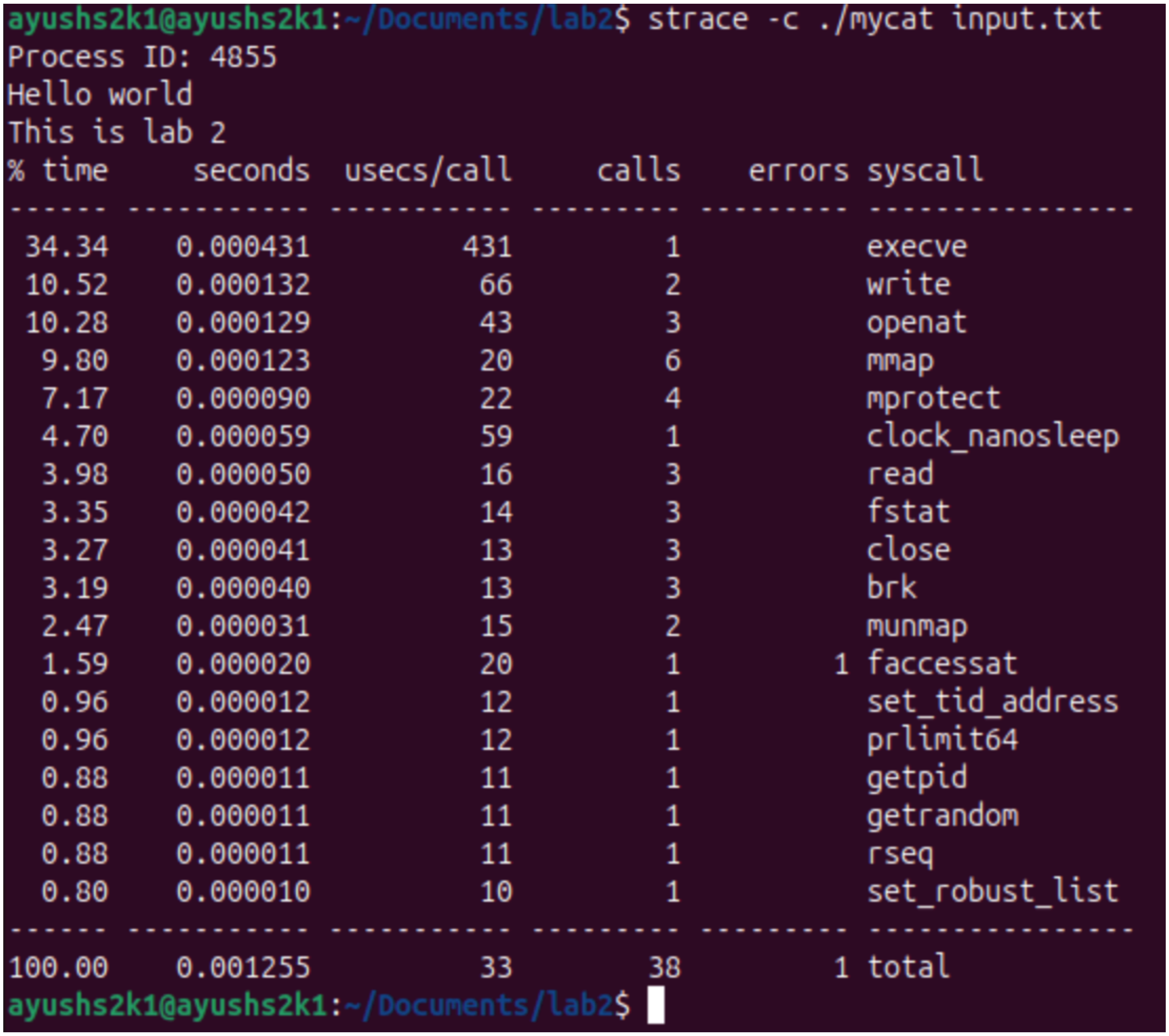


```
ayushs2k1@ayushs2k1:~/Documents/lab2$ ./mycat input.txt
Process ID: 4840
Hello world
This is lab 2
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```

Running the same program with strace analysis:



```
ayushs2k1@ayushs2k1:~/Documents/lab2$ strace ./mycat input.txt
execve("./mycat", ["./mycat", "input.txt"], 0xffffc2cd8b28 /* 48 vars */) = 0
brk(NULL)                               = 0xb1b7120c2000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xef0239de7000
faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=68047, ...}) = 0
mmap(NULL, 68047, PROT_READ, MAP_PRIVATE, 3, 0) = 0xef0239d9e000
close(3)                                = 0
openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0(%\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1788680, ...}) = 0
mmap(NULL, 1957936, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xef0239bbf000
mmap(0xef0239bc0000, 1892400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xef0239bc0000
munmap(0xef0239bbf000, 4096)            = 0
munmap(0xef0239d8f000, 57392)           = 0
mprotect(0xef0239d6a000, 77824, PROT_NONE) = 0
mmap(0xef0239d7d000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ad000) = 0xef0239d7d000
mmap(0xef0239d82000, 49200, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xef0239d82000
close(3)                                = 0
set_tid_address(0xef0239de7ff0)         = 4847
set_robust_list(0xef0239de8000, 24)     = 0
rseq(0xef0239de86e0, 0x20, 0, 0xd428bc00) = 0
mprotect(0xef0239d7d000, 12288, PROT_READ) = 0
mprotect(0xb1b6f27bf000, 4096, PROT_READ) = 0
mprotect(0xef0239ded000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0xef0239d9e000, 68047)           = 0
getpid()                                = 4847
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...}) = 0
getrandom("\x27\xec\xfb\xd7\x96\xd2\x4c\xc6", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0xb1b7120c2000
brk(0xb1b7120e3000)                     = 0xb1b7120e3000
write(1, "Process ID: 4847\n", 17Process ID: 4847
)       = 17
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=4, tv_nsec=0}, 0xfffff13797e8) = 0
openat(AT_FDCWD, "input.txt", O_RDONLY) = 3
read(3, "Hello world\nThis is lab 2\n", 1024) = 26
write(1, "Hello world\nThis is lab 2\n", 26Hello world
This is lab 2
) = 26
read(3, "", 1024)                       = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
ayushs2k1@ayushs2k1:~/Documents/lab2$ 
```



```
ayushs2k1@ayushs2k1:~/Documents/lab2$ strace -c ./mycat input.txt
Process ID: 4855
Hello world
This is lab 2
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 34.34    0.000431         431         1           execve
 10.52    0.000132          66         2           write
 10.28    0.000129          43         3           openat
  9.80    0.000123          20         6           mmap
  7.17    0.000090          22         4           mprotect
  4.70    0.000059          59         1           clock_nanosleep
  3.98    0.000050          16         3           read
  3.35    0.000042          14         3           fstat
  3.27    0.000041          13         3           close
  3.19    0.000040          13         3           brk
  2.47    0.000031          15         2           munmap
  1.59    0.000020          20         1         1 faccessat
  0.96    0.000012          12         1           set_tid_address
  0.96    0.000012          12         1           prlimit64
  0.88    0.000011          11         1           getpid
  0.88    0.000011          11         1           getrandom
  0.88    0.000011          11         1           rseq
  0.80    0.000010          10         1           set_robust_list
------ ----------- ----------- --------- --------- ----------------
100.00    0.001255          33        38         1 total
ayushs2k1@ayushs2k1:~/Documents/lab2$
```

Question: What are the system call names for getting the process ID, opening a file, closing a file, reading a file, printing to the console, and sleeping?

Answer: The system call names are as follows:
Getting the process ID - getpid
Opening a file - openat
Closing a file - close
Reading a file - read
Printing to the console - write
Sleeping - clock_nanosleep

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Question: What are the number of system calls for opening, closing, and reading the file(s) (i.e., how many times each was called)?

Answer: The number of system calls made for only the input.txt file:
Opening the file(s) (openat): As seen from the strace command, there was one call made to open the input.txt file, i.e.,

- openat(AT_FDCWD, "input.txt", O_RDONLY) = 3

Closing the file(s) (close): As seen from the strace command, there was one call made to close the input.txt file, i.e.,

- close(3) = 0

Reading the file(s) (read): As seen from the strace command, there were two calls made to read the input.txt file, i.e.,

- read(3, "Hello world\nThis is lab 2\n", 1024) = 26
- read(3, "", 1024) = 0

The second read returns 0, indicating the end of file (EOF).

Note: There were additional openat, close, and read system calls made, but those were for program initialization and not for the input.txt file.

Question: What are the number of system calls for printing to the screen? (Count each individually. You may either use strace options to aid you in doing so, or you may use grep).

Answer: There are two calls to write for printing to the screen (file descriptor 1 = stdout)
write(1, "Process ID: 4847\n", 17Process ID: 4847)= 17,
write(1, "Hello world\nThis is lab 2\n", 26Hello world This is lab 2) = 26

Question: What was the value of the file descriptor of your read file?

Answer: The value of the file descriptor of the input file is 3. This can be seen by:
openat(AT_FDCWD, "input.txt", O_RDONLY) = 3

The return value 3 indicates that file descriptor 3 was assigned to the input file. This is because:
File descriptor 0 - stdin
File descriptor 1 - stdout

File descriptor 2 - stderr
File descriptor 3 - first available descriptor for user-opened files.

-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-**END**-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x