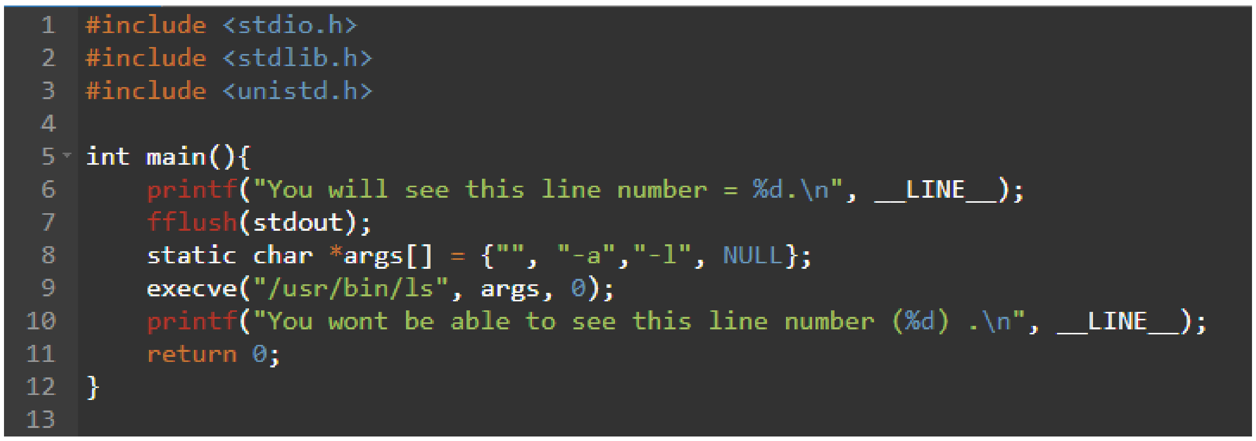
No E-Mail submissions will be accepted.
Submission formats and file naming:
File name : firstName_lastName_lab_6
File format: pdf or MS Word format
e.g. Donald_Trump_lab_6.pdf

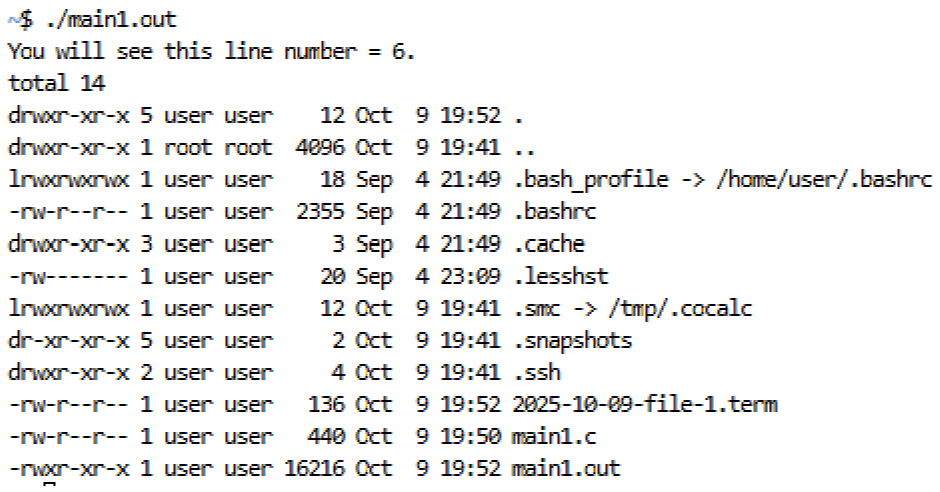
1) Consider the following c code, first compile your code to obtain the binary file main1.out and then run it.

main1.c



```c
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <unistd.h>
 4
 5  int main(){
 6      printf("You will see this line number = %d.\n", __LINE__);
 7      fflush(stdout);
 8      static char *args[] = {"", "-a","-l", NULL};
 9      execve("/usr/bin/ls", args, 0);
10      printf("You wont be able to see this line number (%d) .\n", __LINE__);
11      return 0;
12  }
13
```

- Attach a screenshot of your output.

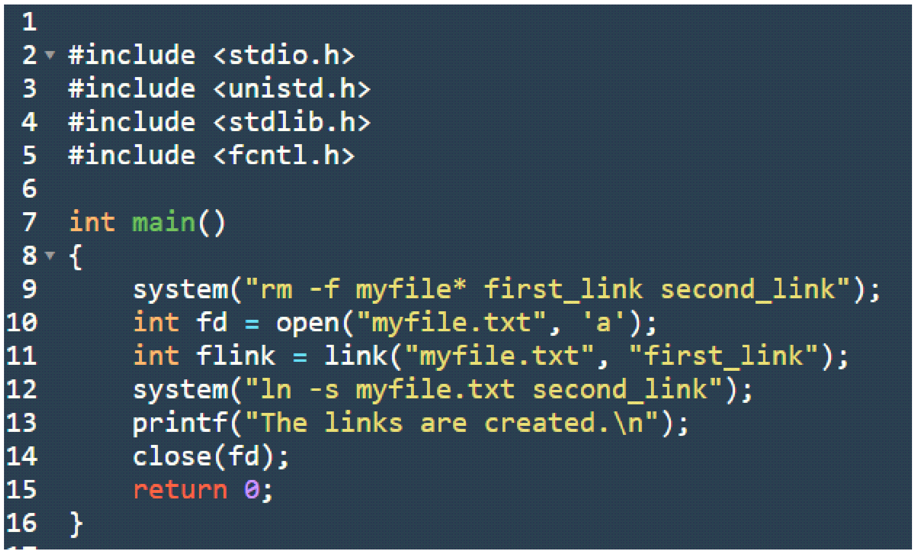


```
~$ ./main1.out
You will see this line number = 6.
total 14
drwxr-xr-x 5 user user    12 Oct  9 19:52 .
drwxr-xr-x 1 root root  4096 Oct  9 19:41 ..
lrwxrwxrwx 1 user user    18 Sep  4 21:49 .bash_profile -> /home/user/.bashrc
-rw-r--r-- 1 user user  2355 Sep  4 21:49 .bashrc
drwxr-xr-x 3 user user     3 Sep  4 21:49 .cache
-rw------- 1 user user    20 Sep  4 23:09 .lesshst
lrwxrwxrwx 1 user user    12 Oct  9 19:41 .smc -> /tmp/.cocalc
dr-xr-xr-x 5 user user     2 Oct  9 19:41 .snapshots
drwxr-xr-x 2 user user     4 Oct  9 19:41 .ssh
-rw-r--r-- 1 user user   136 Oct  9 19:52 2025-10-09-file-1.term
-rw-r--r-- 1 user user   440 Oct  9 19:50 main1.c
-rwxr-xr-x 1 user user 16216 Oct  9 19:52 main1.out
```

- Is line 10 in the output? Why?
  No
  execve is a system call that creates a new process. If execve succeeds it enters the new process and the print statement does not print because it no longer exists.

2) Consider the following c code, first compile your code to obtain the binary file main2.out and then run it.

main2.c

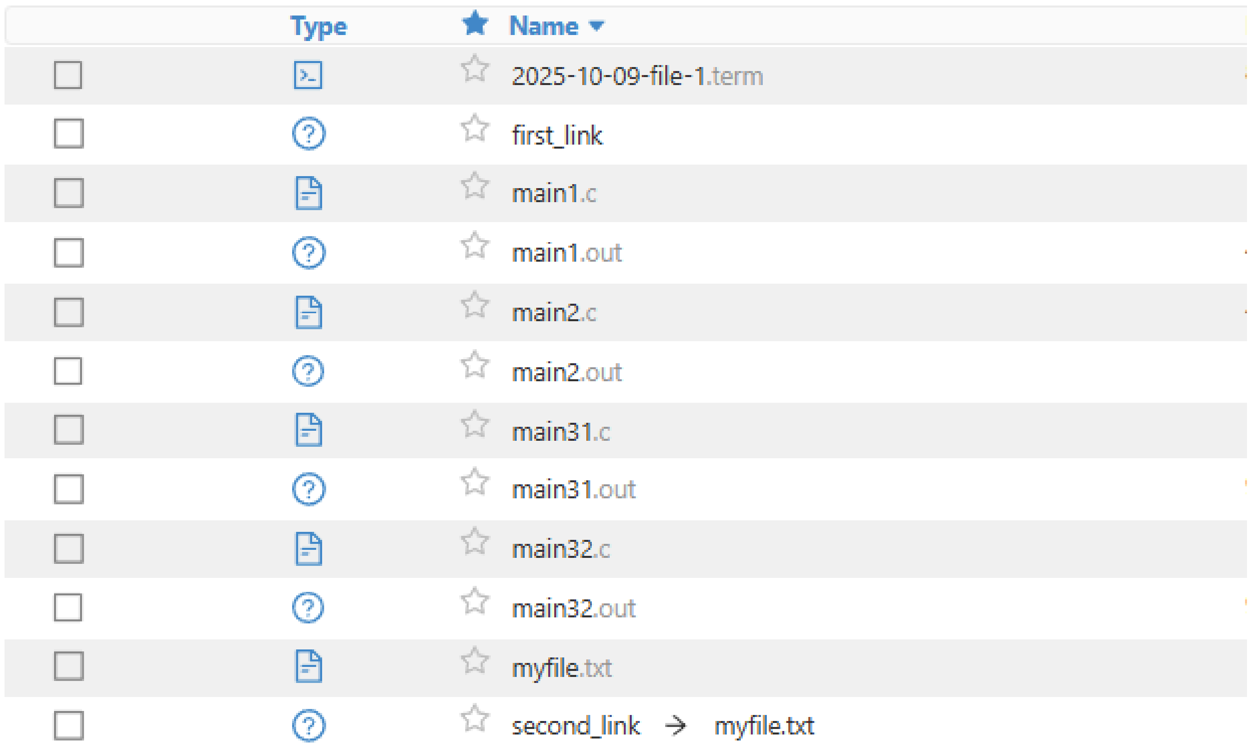


```

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int main()
{
    system("rm -f myfile* first_link second_link");
    int fd = open("myfile.txt", 'a');
    int flink = link("myfile.txt", "first_link");
    system("ln -s myfile.txt second_link");
    printf("The links are created.\n");
    close(fd);
    return 0;
}
```

- Attach a screenshot of your output.



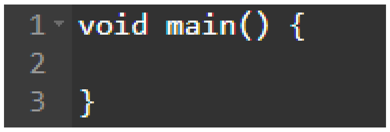
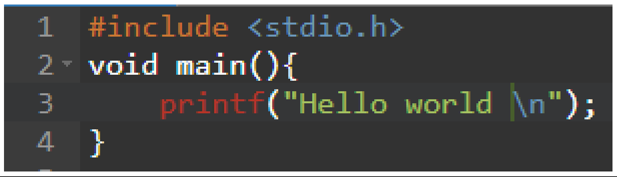
```
Type    Name
        2025-10-09-file-1.term
        first_link
        main1.c
        main1.out
        main2.c
        main2.out
        main31.c
        main31.out
        main32.c
        main32.out
        myfile.txt
        second_link → myfile.txt
```

- Complete the following table and discuss your finding.

	myfile.txt	first_link	second_link
Number of links	2	2	1
type of link	hard	hard	soft

(hard/soft)			

3) Examine the provided C code snippets. Initially, compile the code to generate the binary files named main31.out and main32.out. Execute the subsequent commands to generate a list of system calls utilized in each scenario.

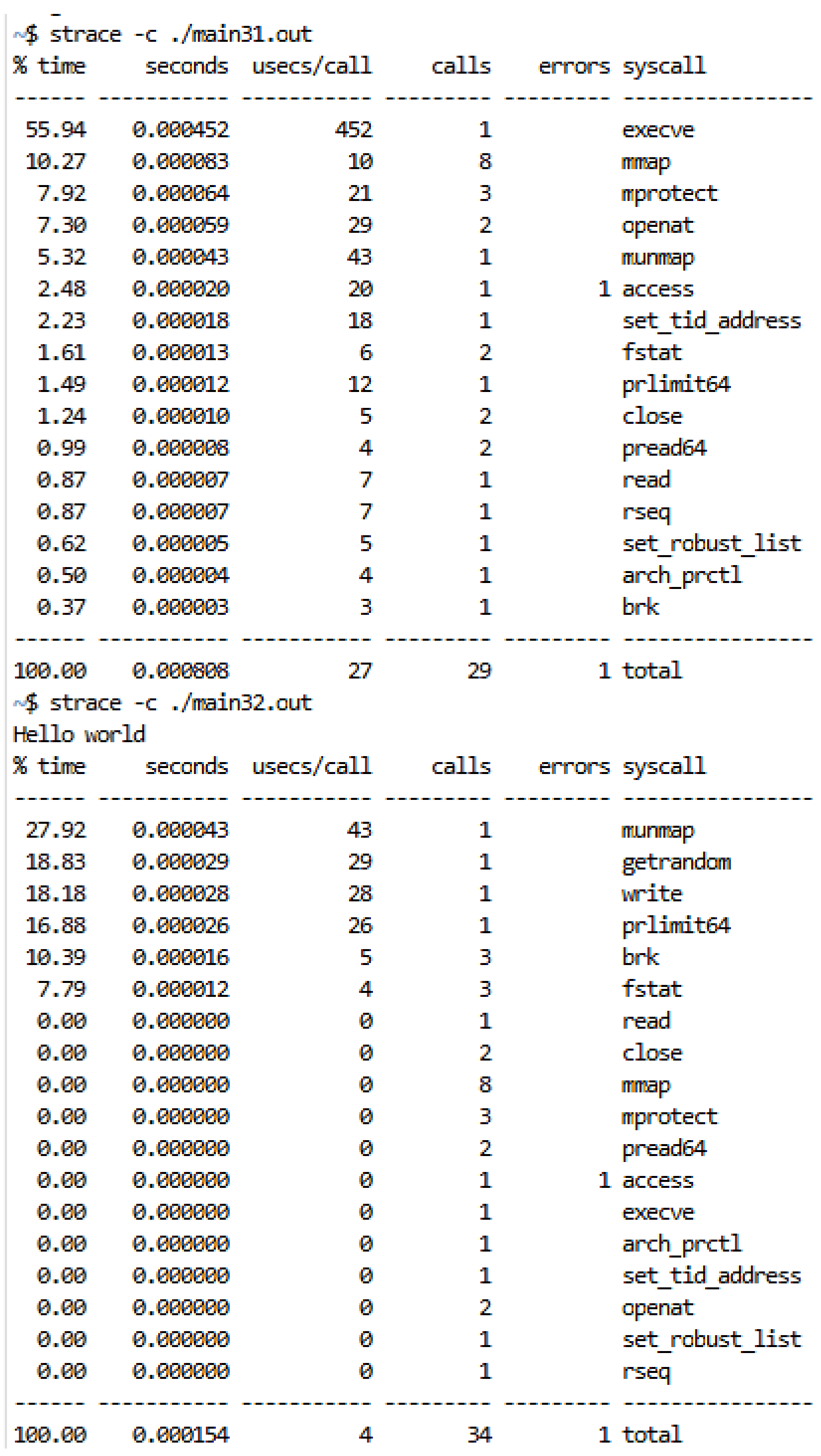
strace -c ./main31.out	strace -c ./main32.out
main31.c 	main32.c 

```c
void main() {

}
```

```c
#include <stdio.h>
void main(){
    printf("Hello world \n");
}
```

- Attach a screenshot of your output for each case.



```
~$ strace -c ./main31.out
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 55.94    0.000452         452         1           execve
 10.27    0.000083          10         8           mmap
  7.92    0.000064          21         3           mprotect
  7.30    0.000059          29         2           openat
  5.32    0.000043          43         1           munmap
  2.48    0.000020          20         1         1 access
  2.23    0.000018          18         1           set_tid_address
  1.61    0.000013           6         2           fstat
  1.49    0.000012          12         1           prlimit64
  1.24    0.000010           5         2           close
  0.99    0.000008           4         2           pread64
  0.87    0.000007           7         1           read
  0.87    0.000007           7         1           rseq
  0.62    0.000005           5         1           set_robust_list
  0.50    0.000004           4         1           arch_prctl
  0.37    0.000003           3         1           brk
------ ----------- ----------- --------- --------- ----------------
100.00    0.000808          27        29         1 total
~$ strace -c ./main32.out
Hello world
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 27.92    0.000043          43         1           munmap
 18.83    0.000029          29         1           getrandom
 18.18    0.000028          28         1           write
 16.88    0.000026          26         1           prlimit64
 10.39    0.000016           5         3           brk
  7.79    0.000012           4         3           fstat
  0.00    0.000000           0         1           read
  0.00    0.000000           0         2           close
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         2           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           arch_prctl
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         2           openat
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           rseq
------ ----------- ----------- --------- --------- ----------------
100.00    0.000154           4        34         1 total
```

- Did you obtain identical outcomes in both scenarios? Explain your answer.
  No
  main32 includes a write system call since it calls printf

- Based on the output you provided, which system call was responsible for displaying **“Hello, world!”** and which file descriptor was utilized?

  Write System call

  file descriptor 1