Set_G_Lab6

Question 1 (1 point) ✓ *Saved*

https://cocalc.com/

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

main1.out

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(){
    printf("You will see this line number = %d.\n", __LINE__);
    fflush(stdout);
    static char *args[] = {"", "-a","-l", NULL};
    execve("/usr/bin/ls", args, 0);
    printf("You wont be able to see this line number (%d) .\n", __LINE__);
    return 0;
}

```

Insert a screenshot of your output.

Paragraph

```
~$ ./main1.out
You will ssee this line number = 6.
total 514
drwxr-xr-x  6 user user    26 Feb 18 18:48 .
drwxr-xr-x  1 root root  4096 Feb 18 18:43 ..
lrwxrwxrwx  1 user user    18 Jan 14 17:25 .bash_profile -> /home/user/.bashrc
-rw-r--r--  1 user user  2355 Jan 14 17:25 .bashrc
drwxr-xr-x  3 user user     3 Jan 14 17:25 .cache
-rw-------  1 user user    20 Jan 14 18:42 .lesshst
lrwxrwxrwx  1 user user    12 Feb 18 18:43 .smc -> /tmp/.cocalc
dr-xr-xr-x 12 user user     2 Feb 11 19:16 .snapshots
drwxr-xr-x  2 user user     4 Feb 18 18:43 .ssh
-rw-------  1 user user  1742 Feb 18 18:48 2026-01-14-terminal-2.term
d---------  2 user user     2 Jan 28 19:00 TEST
```

Add a File | Record Audio | Record Video

Set_G_Lab6

Question 2 (1 point) ✓ *Saved*

Referring to your output from **Question 1**, does line 10 appear in the output (Y/N)? N

Question 3 (1 point) ✓ *Saved*

What is the primary purpose of the `execve()` system call?

- ○ To terminate the running process
- ○ To put the process to sleep
- ○ To create a new child process
- ● To replace the current process image with a new program

Question 4 (1 point) ✓ *Saved*

What happens to the current process after a successful call to `execve()`?

- ○ It forks a new process and returns to the caller
- ● It is completely replaced by the new program
- ○ It enters a waiting state
- ○ It continues executing the next instruction

Set_G_Lab6

Question 5 (1 point) ✓ *Saved*

If `execve()` fails, what value does it return?

- ○ 0
- ● -1
- ○ 1
- ○ It never returns a value

Question 6 (1 point) ✓ *Saved*

Which statement correctly describes the return behavior of `execve()` on success?

- ○ It returns a file descriptor for the new executable
- ● It never returns because the current process image is replaced
- ○ It returns 0 and continues executing the original program
- ○ It returns 1 to indicate a new program is loaded

Set_G_Lab6

Question 7 (1 point) ✓ *Saved*

https://cocalc.com/

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

main2.out

```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;
}
```

Insert a screenshot of your output.

Paragraph

```
~$ gcc main2.c -o main2.out
~$ ./main2.out
The links are created.
~$
```

Set_G_Lab6

Question 8 (2.5 points) ✓ *Saved*

Referring to your output from Question 7, complete the following table.

	myfile.txt	first_link	second_link
Number of links	N1	N2	N3
type of link (hard/soft)	N/A	T1	T2

N1 = ? e.g. 4 [2]

N2 = ? e.g. 4 [2]

N3 = ? e.g. 4 [1]

T1 e.g. hard or soft [hard]

T2 e.g. hard or soft [soft]

Set_G_Lab6

Question 9 (1 point) ✓ *Saved*

https://cocalc.com/

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.

case 1: strace -c ./main31.out	case 2: strace -c ./main32.out
main31.c	main32.c

```c
void main() {

}
```

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

Insert a screenshot of your output for each case.

Paragraph

```
~$ strace -c ./main31.out
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0         1           read
  0.00    0.000000           0         2           close
  0.00    0.000000           0         2           fstat
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         1           brk
  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
```

Add a File | Record Audio | Record Video

Set_G_Lab6

Question 10 (1.5 points) ✓ *Saved*

Referring to your output from **Question 9**,

Did you obtain identical outcomes in both scenarios (Y/N)?

N

Which system call was responsible for displaying "Hello, world!" ?

e.g. read write

Which file descriptor was utilized for displaying "Hello world!"?

e.g. 5 1

Set_G_Lab6

Question 11 (1 point) ✓ *Saved*

https://www.onlinegdb.com/online_c_compiler

Mini Linux Terminal

Consider the following C program. Execute the commands below sequentially, one at a time. After running them, capture and insert a screenshot showing the output. Ensure that the output corresponding to each command is clearly visible in your screenshot.

1) date
2) whoami
3) ls -al
4) uname -a
5) exit

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

#define MAX_INPUT 1024
#define MAX_ARGS 64

int main() {
    char input[MAX_INPUT];
    char *args[MAX_ARGS];

    while (1) {
        printf("COMP_4736> ");
        fflush(stdout);
        if (fgets(input, MAX_INPUT, stdin) == NULL) {
            break;
        }
        input[strcspn(input, "\n")] = '\0';
        if (strcmp(input, "exit") == 0) {
```

```c
            break;
        }
        int i = 0;
        args[i] = strtok(input, " ");
        while (args[i] != NULL && i < MAX_ARGS - 1) {
            i++;
            args[i] = strtok(NULL, " ");
        }
        pid_t pid = fork();

        if (pid == 0) {
            execvp(args[0], args);
            perror("execvp failed");
            exit(EXIT_FAILURE);
        }
        else if (pid > 0) {
            wait(NULL);
        }
        else {
            perror("fork failed");
        }
    }
    return 0;
}
```

Paragraph

```
COMP_4736> date
Wed Feb 18 07:37:12 PM UTC 2026
COMP_4736> whoami
runner87
COMP_4736> ls -al
total 28
drwxrwxrwt 2 runner87 runner87    80 Feb 18 19:37 .
drwxr-xr-x 1 root     root      4096 Feb 18 06:54 ..
-rwxr-xr-x 1 runner87 runner87 16528 Feb 18 19:37 a.out
-rw-r--r-- 1 runner87 runner87  1011 Feb 18 19:37 main.c
COMP_4736> uname -a
Linux Check 6.8.0-1047-gcp #50~22.04.2-Ubuntu SMP Wed Jan 28 01:43:28 UTC 2026 x86_64 x86_
64 x86_64 GNU/Linux
COMP_4736> exit
```

Add a File | Record Audio | Record Video

Set_G_Lab6

Question 12 (1 point) ✓ *Saved*

Referring to your output from **Question 11**, what is the main advantage of execvp() over execve()?

- ○ It creates a new process
- ○ It runs faster
- ○ It allocates memory
- ● It searches the PATH variable

Question 13 (1 point) ✓ *Saved*

Referring to your output from **Question 11**, **if** execvp() executes successfully, it:

- ○ Returns PID
- ● Does not return
- ○ Returns 1
- ○ Returns 0

Question 14 (1 point) ✓ *Saved*

Referring to your output from **Question 11**, why does the parent call wait()?

- ○ To restart the shell
- ○ To stop itself
- ● To wait for child process termination
- ○ To free memory

Set_G_Lab6

Question 15 (1 point) ✓ *Saved*

Referring to your output from **Question 11**, what happens if fork() returns 0?

- ○ Parent process
- ○ System crash
- ○ Error occurred
- ◉ Child process

Question 16 (1 point) ✓ *Saved*

File descriptor numbers are allocated by open() starting from:

- ○ 2
- ○ 0
- ○ 1
- ◉ The lowest unused descriptor

Question 17 (1 point) ✓ *Saved*

Under normal conditions, calling open() will not return 0, 1, or 2 because:

- ◉ They are already in use
- ○ They are illegal values
- ○ The kernel blocks them

Set_G_Lab6

Question 18 (1 point) ✓ *Saved*

What does the directory /proc/<pid>/fd represent?

- ○ File system mount points
- ○ Running services
- ● Open file descriptors of a process
- ○ Kernel modules

Question 19 (1 point) ✓ *Saved*

Which statement is CORRECT regarding file descriptor allocation?

- ● open() returns the lowest unused descriptor
- ○ open() always returns descriptor 3 or higher
- ○ Descriptor numbers are random
- ○ open() returns the highest available descriptor

Question 20 (1 point) ✓ *Saved*

In Linux, file descriptors 0, 1, and 2 are:

- ○ Assigned only when a program opens files manually
- ○ Reserved numbers that cannot appear in /proc/<pid>/fd
- ● Automatically given to each process by the OS and can be seen in /proc/<pid>/fd
- ○ Only used by the kernel internally and invisible to users