LAB Assignment 4

1. Implement `fork()` System Call:

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

int main() {
    pid_t child_pid;


    child_pid = fork();

    if (child_pid == 0) {

        printf("Child process: PID = %d\\n", getpid());
    } else if (child_pid > 0) {

        printf("Parent process: PID = %d, Child PID = %d\\n", getpid(), child_pid);
    } else {

        perror("Fork failed");
    }

    return 0;
}
```

2. Implement `wait()` and `exit()` System Calls:

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

```
int main() {
    pid_t child_pid;
    int status;

    child_pid = fork();

    if (child_pid == 0) {

        printf("Child process: PID = %d\\n", getpid());
        exit(42);
    } else if (child_pid > 0) {

        wait(&status);
        if (WIFEXITED(status)) {
            printf("Parent process: Child exited with status %d\\n", WEXIT
STATUS(status));
        }
    } else {

        perror("Fork failed");
    }

    return 0;
}
```

3. Implement `execv()` System Call:

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

int main() {
    char *args[] = {"/bin/ls", "-l", NULL};

    if (execv(args[0], args) == -1) {
        perror("Execv failed");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

```
}
```

4. Implement `open()`, `read()`, `write()`, and `close()` System Calls:

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

int main() {
    int fd;
    char buffer[100];

    // Open a file for writing
    fd = open("output.txt", O_WRONLY | O_CREAT, 0644);

    if (fd == -1) {
        perror("Open failed");
        exit(EXIT_FAILURE);
    }


    write(fd, "Hello, World!\\n", 14);


    close(fd);


    fd = open("output.txt", O_RDONLY);

    if (fd == -1) {
        perror("Open failed");
        exit(EXIT_FAILURE);
    }


    read(fd, buffer, sizeof(buffer));
    printf("Data from file: %s", buffer);


    close(fd);

    return 0;
}
```