

**Software engineering**

**Project-work**

**NAME: Abel Amare**

**ID:1601012 SECTION:A**

## **KWORT LINUX**



## Implement System Calls

To better understand how the kernel interacts with user-level programs, I implemented a simple C program using the `execv()` system call. This function is part of the `unistd.h` library in Unix-like systems and replaces the current process image with a new one — typically another executable.

**Source File:** `exec_test.c`

### The code

```
#include <unistd.h>

#include <stdio.h>

int main() {

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

    execv(args[0], args);

    perror("exec failed");

    return 1;

}
```

### How it works:

- **Explanation:** `Execv (path, args):` This system call executes the program specified in `path` (here, `/bin/ls`) with the arguments in `args`.
- If successful, the current process is replaced and does not return to the caller.
- If it fails, `perror()` displays the error.

## Compilation & Execution

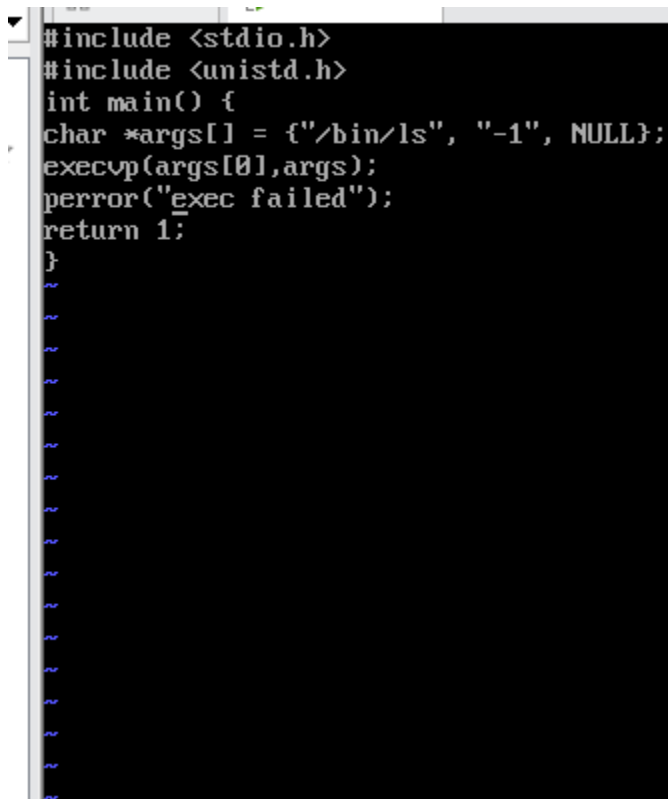
To compile: Run : vi exec\_test.c



```
root@Abel:~# vi exec_test.c
```

Next type i in order to be able to write in the compiler

now we can write the code



```
#include <stdio.h>
#include <unistd.h>
int main() {
    char *args[] = {"/bin/ls", "-1", NULL};
    execvp(args[0],args);
    perror("_exec failed");
    return 1;
}
```

Then

**To Exit and Save Changes Press: Esc**

**Type :- :wq**

**Press: Enter**

**Finally RUN:** gcc exec\_test.c -o exec\_test

Then ./test

```
root@Abel:~# gcc exec_test.c -o test
root@Abel:~# ./test
bootloaders
exec_example1.c
exec_test.c
test
root@Abel:~#
```

### What the exec() System Call Did:

Most likely, your C program (exec\_test.c) called exec() (e.g., execlp() or execvp()) to **replace the current process image** with the ls command. That means:

- The test process was started.
- It immediately used exec() to run ls, effectively replacing itself with the ls process.
- Therefore, you see the output of ls instead of anything from the original C code that might have followed the exec() call.

---

### Summary:

The exec() system call **replaces the current process** with a new one — in this case, ls. Your program successfully demonstrated this behavior by printing the contents of the directory and exiting without returning to the original C process logic.