To write C programs using the following system calls of Unix operating systems

Fork, Exec, Getpid, Exit, Wait, Close, Stat, Opendir, Readdir

**Overview of System Calls**

1. **fork()**
   * Creates a new process by duplicating the calling process.
   * Returns 0 to the child process and the child's PID to the parent.
2. **exec()**
   * Replaces the current process image with a new process image.
   * Variants: execl(), execv(), execvp(), etc.
3. **getpid()**
   * Returns the process ID (PID) of the calling process.
4. **exit()**
   * Terminates the calling process.
5. **wait()**
   * Makes the parent process wait for the child process to terminate.
6. **close()**
   * Closes a file descriptor.
7. **stat()**
   * Retrieves information about a file (e.g., size, permissions, etc.).
8. **opendir()**
   * Opens a directory stream.
9. **readdir()**
   * Reads entries in a directory stream.

Code :

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <dirent.h>

int main() {

    pid\_t pid;

    struct stat fileStat;

    DIR \*dir;

    struct dirent \*entry;

    printf("Parent PID: %d\n", getpid());

    pid = fork();

    if (pid < 0) {

        perror("fork failed");

        exit(EXIT\_FAILURE);

    } else if (pid == 0) {

        printf("Child PID: %d\n", getpid());

        execl("/bin/ls", "ls", "-l", NULL);

        perror("execl failed");

        exit(EXIT\_FAILURE);

    } else {

        printf("Waiting for child process to finish...\n");

        wait(NULL);

        if (stat("example.txt", &fileStat) == 0) {

            printf("File: example.txt\n");

            printf("Size: %ld bytes\n", fileStat.st\_size);

            printf("Permissions: %o\n", fileStat.st\_mode & 0777);

        } else {

            perror("stat failed");

        }

        dir = opendir(".");

        if (dir) {

            printf("Directory contents:\n");

            while ((entry = readdir(dir)) != NULL) {

                printf("  %s\n", entry->d\_name);

            }

            closedir(dir);

        } else {

            perror("opendir failed");

        }

        printf("Parent process exiting.\n");

    }

    return 0;

}

Output:

