PRACTICAL : 13

AIM : Write a program which demonstrate the use of fork, join, and exec and wait system calls.

Source code :

Fork:

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

int main() {
   printf("Before fork, PID: %d\n", getpid());

   pid_t pid = fork();

   if (pid < 0) {
      perror("Fork failed");
      return 1;
   } else if (pid == 0) {
      printf("Child process: PID = %d, Parent PID = %d\n", getpid(), getppid());
   } else {
      printf("Parent process: PID = %d, Child PID = %d\n", getpid(), pid);
   }
   printf("This message is from PID = %d\n", getpid());
   return 0;
}
```

Output:

```
Before fork, PID: 14489
Parent process: PID = 14489, Child PID = 14490
This message is from PID = 14489
Child process: PID = 14490, Parent PID = 14489
This message is from PID = 14490


=== Code Execution Successful ===
```

join :

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

void *thread_function(void *arg) {
    printf("Thread %ld started.\n", pthread_self());
    sleep(2);
    printf("Thread %ld finished.\n", pthread_self());
    return NULL;
}

int main() {
    pthread_t thread;
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Thread creation failed");
        return 1;
```

```
    }
    printf("Main thread waiting for child thread...\n");
    pthread_join(thread, NULL);
    printf("Main thread exiting.\n");

    return 0;
}
```

Output:

```
Main thread waiting for child thread...
Thread 140564004746944 started.
Thread 140564004746944 finished.
Main thread exiting.


=== Code Execution Successful ===
```

exec :

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

int main() {
    printf("Executing `ls -l` using execvp()\n");

    char *args[] = {"/bin/ls", "-l", NULL};
    execvp(args[0], args);
```

```
    perror("Exec failed");
    return 1;
}
```

Output:

```
Executing `ls -l` using execvp()
total 0


=== Code Execution Successful ===
```

wait :

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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        printf("Child process: PID = %d\n", getpid());
        sleep(2);
        printf("Child process exiting...\n");
        exit(42);
```

```
    } else {
        int status;
        printf("Parent waiting for child to complete...\n");
        wait(&status);

        if (WIFEXITED(status)) {
            printf("Child exited with status %d\n", WEXITSTATUS(status));
        }

        printf("Parent process finished.\n");
    }

    return 0;
}
```

Output:

```
Parent waiting for child to complete...
Child process: PID = 19433
Child process exiting...
Child exited with status 42
Parent process finished.


=== Code Execution Successful ===
```