An Example Using `execvp` System Call

**Note:** The following example uses three system calls, namely, `fork`, `wait` and `execvp`. The child process sets up an argument array (for the "ls" command with the "-l" option) and calls `execvp`. After the `fork` system call, the parent process waits for the child to complete. When you run this program, the output produced by the "ls -l" command from the child will appear first and then the output produced by the parent.

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

int main(void) {
    pid_t child;
    int cstatus; /* Exit status of child. */
    pid_t c; /* Pid of child to be returned by wait. */
    char *args[3]; /* List of arguments for the child process. */

    /* Set up arguments to run an exec in the child process. */
    /* (This example runs the "ls" program with "-l" option.) */
    args[0] = "ls"; args[1] = "-l";
    args[2] = NULL; /* Indicates the end of arguments. */

    if ((child = fork()) == 0) { /* Child process. */
        printf("Child: PID of Child = %ld\n", (long) getpid());
        execvp(args[0], args); /* arg[0] has the command name. */

        /* If the child process reaches this point, then */
        /* execvp must have failed. */

        fprintf(stderr, "Child process could not do execvp.\n");
        exit(1);
    }
    else { /* Parent process. */
        if (child == (pid_t)(-1)) {
            fprintf(stderr, "Fork failed.\n"); exit(1);
        }
        else {
            c = wait(&cstatus); /* Wait for child to complete. */
            printf("Parent: Child %ld exited with status = %d\n", (long) c, cstatus);
        }
    }
    return 0;
} /* End of main. */
```