An Example Using fork and wait System Calls

Note: The following example uses both the fork and the wait system calls. As before, the child process simply prints its pid and exits with a status of 0 (zero). After the fork system call, the parent process waits for the child to complete. After the child exits, the parent process prints its own pid and the pid of the child process. This is followed by the exit status of the child (zero in this case). When you run this program, the output produced by the child will appear first and then the output produced by the parent. Thus, the outputs of the two processes won’t be interleaved.

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

int main(void) {
    pid_t child; /* Child’s pid. */
    int cstatus; /* Exit status of child. */
    pid_t c; /* Child’s pid to be returned by the wait system call. */

    if ((child = fork()) == 0) { /* Child process. */
        printf("Child: PID of Child = %ld \n", (long) getpid());
        exit(0);
    }
    else { /* Parent process. */
        if (child == (pid_t)(-1)) {
            fprintf(stderr, "Fork failed.\n");
            exit(1);
        }
        else { /* Parent will wait until the child exits. */

            c = wait(&cstatus);
            printf("Parent: PID of Child = %ld\n", (long) child);
            printf("Parent: PID of Parent = %ld\n", (long) getpid());
            printf("Parent: Child %ld exited with status = %d\n",
                   (long) c, cstatus);
        }
    }
    return 0;
} /* End of main. */
```