lab21.c

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

int main() {
    /*
     * Fork a new process. fork() returns:
     * - Negative value: Error in fork()
     * - Zero: Child process
     * - Positive value: Parent process (value is child's PID)
     */
    pid_t pid = fork();

    // Error handling for fork() failure
    if (pid < 0) {
        perror("fork failed");
        exit(EXIT_FAILURE);
    }

    // Child process
    if (pid == 0) {
        printf("Child process:\n");
        printf("  My PID is %d\n", getpid());
        printf("  My parent's PID is %d\n", getppid());

        // Child process exits with success status
        exit(EXIT_SUCCESS);
    }
```

Part 2:

```c
    // Parent process
    else {
        printf("Parent process:\n");
        printf("  My PID is %d\n", getpid());
        printf("  My child's PID is %d\n", pid);

        /*
         * Wait for child process to complete.
         * NULL means we don't care about the exit status.
         * This prevents zombie processes.
         */
        int status;
        pid_t child_pid = wait(&status);

        // Check if wait() failed
        if (child_pid < 0) {
            perror("wait failed");
            exit(EXIT_FAILURE);
        }

        // Check how child process terminated
        if (WIFEXITED(status)) {
            printf("Child process %d exited with status %d\n",
                   child_pid, WEXITSTATUS(status));
        } else if (WIFSIGNALED(status)) {
            printf("Child process %d killed by signal %d\n",
                   child_pid, WTERMSIG(status));
        }
    }

    return EXIT_SUCCESS;
}
```