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

int main() {
    int fd[2] = {-1,-1};
    int rfd[2] = {-1,-1};
    int ret = pipe(fd);
    if (-1 == ret) {
        perror("error");
        return 1;
    }
    ret = pipe(rfd);
    if (-1 == ret) {
        perror("error pipe2");
        return 2;
    }
    printf("fd: %d %d\n", fd[0], fd[1]);
    printf("rfd: %d %d\n", rfd[0], rfd[1]);

    char buffer_in[20];
    ssize_t nread2 = read(0, buffer_in, 20);
    buffer_in[nread2] = '\0';

    // protocol
    // parent sends message to child first
    // child responds with AdCK
    // parent recognises ACK

    int pid = fork();
    if (0 == pid) {

        // child
        close(fd[1]);
        close(rfd[0]);

        char buffer[20];
        strncpy(buffer, "This is a long sentence, much
         longer than 20 characters", 20);

        printf("Child is waiting for input..\n");
        ssize_t nread = read(fd[0], buffer, 19);
        if (-1 == nread) {
            perror("child read error");
        } else {
```

```
            buffer[nread] = '\0';
            printf("nread: %zd\n", nread);
            printf("%s\n", buffer);

            char response[25];
            snprintf(response, 25, "ACK:%s", buffer);
            size_t resp_len = strlen(response) + 1;
            ssize_t nwritten = write(rfd[1], response,
             resp_len );
            if (-1 == nwritten) {
                perror("child write error");
            } else {
                printf("child wrote %zd bytes\n", nwritten);
            }

        }

        close(fd[0]);
        close(rfd[1]);

    } else {

        sleep(1);

        close(fd[0]);
        close(rfd[1]);

        // parent
        printf("child pid is %d\n", pid);

        ssize_t nwritten = write(fd[1], buffer_in,
         strlen(buffer_in) );
        if (-1 == nwritten) {
            perror("parent write error");
        } else {
            printf("%zd bytes written\n", nwritten);

            char child_response[25];
            ssize_t nread = read(rfd[0], child_response,
             25);
            if (-1 == nread) {
                perror("parent read error");
            } else {
                child_response[nread] = '\0';
```

```
                char expected[25];
                snprintf(expected, 25, "ACK:%s", buffer_in);

                printf("parent received: %s [ %d ]\n",
                 child_response,
                    strncmp(expected, child_response,
                     strlen(expected)) );
            }

        }
        printf("parent has completed\n");

        close(fd[1]);
        close(rfd[0]);

    }

    return 0;
}
```