```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

// child is doing something, occasionally checking the pipe
void child(int readfd) {

    fd_set allfds;
    char cbuffer[100];
    int nfds = readfd+1;

    // how often to check?
    long delay_sec = 5;
    struct timeval tv;

    while (1)
    {
        // look at file for changes for reading
        FD_ZERO(&allfds);
        FD_SET(readfd, &allfds); // stdin

        // set delay
        tv.tv_sec = delay_sec;
        tv.tv_usec = 0;

        // select
        int ret = select(nfds, &allfds, NULL, NULL, &tv);
        if (-1 == ret) {
            perror("select() failed");
        } else if (ret) {
            if (FD_ISSET(readfd, &allfds)) {
                ssize_t nread;
                printf("message from the mothership: ");
                nread = read(readfd, cbuffer, 10);
                cbuffer[nread]= '\0';
                printf("%s\n", cbuffer);
                if (strncmp(cbuffer, "kamikaze", 8) == 0)
                    break;
```

```
            }
        }

        // child does something
        sleep(2);
        printf("CHILD: all is well\n");
    }
    printf("CHILD is no more!\n");
}

void parent(int writefd) {

    // receive and relay commands from stdin
    char buffer[100];

    while (1)
    {
        printf("> ");
        fflush(stdout);

        int nread = read(0, buffer, 100);
        if (-1 == nread)
            perror("could not read stdin");
        else if ( nread == 0 ) {
            perror("end of file for stdin?");
        } else {
            buffer[nread] = '\0';
            if (strncmp(buffer, "child", 5) == 0) {
                if (strncmp(buffer, "child die", 9) == 0)
                    // specific instruction
                    write(writefd, "kamikaze", 9);
                else {
                    // general message for child
                    size_t blen = strlen(buffer) + 1;
                    size_t max_len =
                            blen > 95 ? 95 : blen;

                    write(writefd, buffer+5, max_len);
                }
                printf("Child command issued\n");

            } else if (strncmp(buffer, "quit", 4) == 0) {

                printf("terminating child\n");
                // specific instruction
```

```
                write(writefd, "kamikaze", 8);

                printf("terminating self\n");
                break;
            } else {
                printf("%s ignored\n", buffer);
            }
        }
    }
    printf("PARENT is no more!\n");
}

int main(void)
{
    int notify[2];

    int ret = pipe(notify);
    printf("pipe created: %d %d\n", notify[0], notify[1]);

    int pid = fork();
    if (0 == pid) {
        close(notify[1]);
        child(notify[0]);
        close(notify[0]);
    } else {
        close(notify[0]);
        parent(notify[1]);
        close(notify[1]);
    }

    return 0;
}
```