

Re: Serial Programming trouble

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.misc/2006-09/msg00793.html>

- *From:* floyd@xxxxxxxxxx (Floyd L. Davidson)
 - *Date:* Thu, 14 Sep 2006 09:20:19 -0800
-

dieter.verslype@xxxxxxxxxx wrote:

I got it to work, i needed to wait for some small interval between sending characters.

Here's my code:

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

/*
 * Opens the serial port
 */
void open_port(int* fd) {
    *fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY);
```

You want to also flag that with O_NONBLOCK.

```
if (*fd == -1) {
    perror("open_port: Unable to open /dev/ttyS0 - ");
} else
    fcntl(*fd, F_SETFL, FNDELAY);
```

You want to **clear** FNDELAY once the port is open, not set it.

Note that O_NONBLOCK is defined by POSIX, and is therefore the same on all platforms. O_NDELAY and FNDELAY on linux are **exactly** the same thing as O_NONBLOCK, but on other systems can be defined slightly different. For that reason O_NONBLOCK should be used, because it is portable and the others are not.

Re: Serial Programming trouble

```
}

/*
 * Closes the serial port
 */
void close_port(int* fd) {
    close(*fd);
}
```

I don't see much point in that particular abstraction.

```
/*
 * Prepares the serial port for communication
 */
```

This function has several problems that will bite you sooner or later. What I'm going to suggest is not strictly kosher with POSIX, but it is safe, and much easier than trying to be POSIX correct, which leads to what you have below.

```
void configure_communication(int fd) {
    struct termios options;

    // get current options
    tcgetattr(fd, &options);
```

The easy way to pre-configure the termios struct, is to just simply **zero** it out. POSIX says otherwise, but then you **must** be sure to set each and every parameter correctly, which is **not** what you have done below (though it is an interesting effort to do so).

```
memset(&options, 0, sizeof options);
```

```
// set i/o baud rate
cfsetispeed(&options, B9600);
cfsetospeed(&options, B9600);
```

The speed values should be set just before calling `tcsetattr()`, because on some systems the values may be part of one of the flags, and setting other values might (when not done right)

Re: Serial Programming trouble

change the speed value. If done correctly, it is not a problem, but trouble shooting difficulty is reduced with a bit of defensive programming...

```
// cflag options
options.c_cflag = ~CSIZE;
```

This is a gross error. You have indeed cleared CSIZE, but in the process you have *set* every other bit in that flag! (On some systems that would have overwritten the input speed value just set above.)

```
options.c_cflag &= ~CSIZE;
```

is the *only* way to use that construct.

However, if you use the memset() call above, everything has already been cleared, and there is no need to do anything except *set* bits. For example:

```
options.c_cflag = CS8
options.c_cflag |= CLOCAL;
options.c_cflag |= CREAD;
```

But of course it is just as easy to do this:

```
options.c_cflag = CS8 | CLOCAL | CREAD;
```

```
options.c_cflag |= CS8;
options.c_cflag &= ~PARENB;
options.c_cflag &= ~PARODD;
options.c_cflag &= ~CSTOPB;
options.c_cflag |= CLOCAL;
options.c_cflag |= CREAD;
options.c_cflag &= ~CRTSCTS;
options.c_cflag &= ~HUPCL;
```

```
// iflag options
options.c_iflag = IGNBRK;
options.c_iflag &= ~IXON;
options.c_iflag &= ~IXOFF;
options.c_iflag &= ~IXANY;
```

In this case, setting iflag equal to IGNBRK has cleared all other bits, hence the last three accomplished nothing.

Re: Serial Programming trouble

```
// lflag options
options.c_lflag = ~OPOST;
options.c_lflag &= ~OLCUC;
options.c_lflag &= ~OCRNL;
options.c_lflag &= ~ONLCR;
options.c_lflag &= ~ONOCR;
options.c_lflag &= ~ONLRET;
options.c_lflag &= ~OFILL;
options.c_lflag &= ~OFDEL;
options.c_lflag |= NL0;
options.c_lflag |= CR0;
options.c_lflag |= TAB0;
options.c_lflag |= BS0;
options.c_lflag |= VT0;
options.c_lflag |= FFO;
```

I can't tell what you actually want to do. You've cleared OPOST and in the process set everything else. Then you clear several bits and then set a few, but none of them have any effect if OPOST is cleared.

Of course, all of these bit mask apply to oflag, **not** to lflag!

```
// oflag options
options.c_oflag = 0;
```

Put the calls to `cfsetispeed()` and `cfsetospeed()` here.

```
// set port with current options
tcsetattr(fd, TCSANOW, &options);
}

/*
 * Actively waits for some milliseconds
 */
void active_wait(int milliseconds)
{
    timeval system_time;
    gettimeofday(&system_time, NULL);
    double start_time = (system_time.tv_sec*1000.0) +
        (system_time.tv_usec/1000.0);
    gettimeofday(&system_time, NULL);
    double current_time = (system_time.tv_sec*1000.0) +
        (system_time.tv_usec/1000.0);
    while (current_time - start_time < milliseconds)
    {
```

Re: Serial Programming trouble

```
gettimeofday(&system_time,NULL);
current_time = (system_time.tv_sec*1000.0) +
(system_time.tv_usec/1000.0);
}
}
```

Use nanosleep(2).

```
/*
 * Sends a character string with intervals of 100 milliseconds
 */
void send_string(char* sp, int* fd)
{
while(*sp != '\0')
{
write(*fd,sp,1);
active_wait(100);
sp++;
}
}

int main(int argc, char **argv) {
int fd;
open_port(&fd);
configure_communication(fd);
send_string("DM", &fd); // Start PC communication session
send_string("DM?", &fd); // Request software version and date

char buf[255];
int n;
while ((n = read(fd, buf, 255))>0)
{
printf("read %d bytes: ", n);
for(int i=0; i<n; i++)
{
printf("%c", buf[i]);
}
printf("\n");
}
close_port(&fd);
}
```

—
Floyd L. Davidson <http://www.apaflo.com/floyd_davidson>
Ukpeagvik (Barrow, Alaska) floyd@xxxxxxxxxx