http://groups.google.com/d/topic/comp.dcom.modems/oQxiZ9m8M9c/discussion

comp.dcom.modems ›

checkblazer.c -- get spectrum data from last connection

1 post by 1 author

Steve Murphy

4/21/88

Here's something (checkblazer.c) maybe someone might find useful. Enjoy.
  --murf

```
--------------------------------------------------------------------------
#include <sys/types.h>
#include <signal.h>
#include <setjmp.h>
#include <stdio.h>
#ifdef        USG
#include <termio.h>
#endif
#ifndef        USG
#include <sgtty.h>
#endif
#include <sys/stat.h>
#define OUTPUTDIR "/usr/spool/uucp"
#define DEFSTARTSPEED EXTA
        /* A program to grab Telebit TrailBlazer Status Data and dump to ascii files */
        /* by Steve Murphy */

        /* Some say the globals, etc. seem similar to some used in a particular
           version of uucp. This is, of course, purely accidental, but those with
           discernment may be able to take advantage of this accident.
           People with discernment usually take advantage of "accidents".

           Wouldn't it be nice if, while closing the modem in uucp, you generated
           data files that showed the status of the connection?

           */

        /* For the rest of us: this little standalone will,

           IF the proper command line args are supplied,

           IF the modem is set up with just the right defaults in its regs,
           (I have E0 F1 M1 Q6 T V1 X3, and s51=5, for example)

           extract all the S70 type registers revealing the status of the last
           connection made, fast or slow (depending on the regs) */

        /* This data is dumped out to files in x,y pairs suitable for progs like
           gnuplot, or just echoed back to the user if non-array oriented. */

        /* I developed my own hp7475A interface to GNUPLOT and use it to generate
           nice-looking and informative pen-plots of connections. This info is really
           informative at times about what kind of problems I'm seeing. */

        /* here's an example GNUPLOT.INI.telebit file to display the files generated
           on an hp pen plot:

           set term hp74
           set samples 2048
           set output "hp.noise.telebit"
           plot [0:4000] [-80:10] "blazer_noise.telebit" with lines,
                                      "blazer_rbits.telebit" with impulses,
                                   "blazer_tbits.telebit" with lines
           quit

           This will display the noise profile on the bottom and the two bitrates
overlay
           each other above.

           Because This generates several files if run with several sys names,
           I generated a pile of these GNUPLOT.INI files, AND
           I have a generic GNUPLOT.INI submitter:
```

```
#
echo $argv[1]
cp GNUPLOT.INI.$argv[1] .gnuplot
gnuplot

And, of course, a lpr -Php hp.noise.telebit (in this case) would
be the next thing (go stick paper in the plotter first!)

Now, some clever person could do this much more neatly with sed scripts,
I'm sure. Have at it. Have fun. Go for it.
           */

        /* As usual, This will most likely require modification to run on your modems.
           I set up my registers like few others on the net, because of constraints
beyond
           my control.

           I doubt that this will work on anything but sun 3's.

           Releasing even this peice of junk is charity on my part; Don't bother me with
           anything but something better -- I didn't generate this for money, really,
           and so I really can't support it either. It may be a little work for you to
make
           it run, but hey, I think I did more than you, OK?

           Purpose: For those who want to grab this kind of data bad enough to make this
           work, but not bad enough, obviously, to write their own from scratch

           Encouragement: This code actually runs on my machine (sun3/160) with my OS
           (SunOS3.2) with my modems (Telebit Trailblazer Plus, rev 3 roms. */

jmp_buf Sjbuf;
jmp_buf Pipebuf;

#define STBNULL 0
#define CF_DIAL 5
#define BAND (4000.0/512.0)

int blazer_slow=0, blazer_fast = 0;
int onesys = 0;
int turntime = 30 * 60;         /* 30 minutes expressed in seconds */
char *ttyn = NULL;
extern int errno;
char devSel[200],Rmtname[200];

#ifdef        USG
struct termio Savettyb;
#endif
#ifndef        USG
struct sgttyb Savettyb;
#endif

alarmtr()
{
        signal(SIGALRM, alarmtr);
        longjmp(Sjbuf, 1);
}

expect(str,fd)
char *str;
{
        char *p = str;
        char bug[10];

        if (setjmp(Sjbuf))
        {
                fprintf(stderr, "TIMEOUT\n");
                return CF_DIAL;
        }
        signal(SIGALRM, alarmtr);
        alarm(10);
 again:
        while( *p &&  read( fd,bug,1) == 1 && *p == bug[0] )
                p++;
        if ( !*p )
        {
                alarm(0);
                return 0;
        else
        {
                p = str;
                goto again;
        }
}

logent( what, arg1,arg2,arg3,arg4)
char *what,*arg1,*arg2,*arg3,*arg4;
{
        FILE *lo;
        char buff[300];

/*         lo = fopen("/usr/spool/uucp/LOGFILE","a"); For those who enjoy permanent
records
        if( !lo )
                return; */
        sprintf(buff,what,arg1,arg2,arg3,arg4);
        printf(buff);
/*        fprintf(lo,"checkblazer: %s",buff); to go a log file
        fclose( lo ); */
        return;
}

main(argc, argv)
int argc;
register char *argv[];
{
        int seq,localbits,Ifn;
        char *q, tty_port[100];

        if( argc < 4)
        {
                fprintf(stderr,"usage: checkblazer SLOW|FAST device label\n");
                exit(0);
        }
        Ifn = setreuid(geteuid(),geteuid());
        if( Ifn )
        {
                logent("Couldn't setuid to %d\n",geteuid());
        }
        Ifn = setregid(getegid(),getegid());
        if( Ifn )
        {
                logent("Couldn't setgid to %d\n",getegid());
        }

        ttyn = tty_port;
        if( !strcmp(argv[1],"SLOW") || !strcmp(argv[1],"slow") )
                blazer_slow = 1;
        else
                blazer_fast =1;

        strcpy(tty_port,argv[2]);
        strcpy(Rmtname,argv[3]);

        if( ttyn )
                strcpy(devSel, ttyn);
        else
                strcpy(devSel,"");

        cleanup(0);
}

cleanup(code)
{
        int dh = -1;
        char dcname[40];

        sleep(2);
        strcpy(dcname,devSel);
        if (setjmp(Sjbuf))
        {
                logent( "TIMEOUT\n");
                return CF_DIAL;
        }
        signal(SIGALRM, alarmtr);
        alarm(0);

        /* modem is open */
        if (dh >= 0)
        {
                char spdy[30];
#ifdef        USG
                ioctl(dh, TCGETA, &Savettyb);
                Savettyb.c_cflag = (Savettyb.c_cflag & ~CS8) | CS7;
                Savettyb.c_oflag |= OPOST;
                Savettyb.c_lflag |= (ISIG|ICANON|ECHO);
#else !USG
                ioctl(dh, TIOCGETP, &Savettyb);
                Savettyb.sg_flags |= ECHO;
                Savettyb.sg_flags &= ~RAW;
                Savettyb.sg_ispeed = Savettyb.sg_ospeed = DEFSTARTSPEED;
                (void) ioctl(dh, TIOCHPCL, STBNULL);
#endif !USG
#ifdef        USG
                ioctl(dh, TCSETA, &Savettyb);
#else        !USG
                ioctl(dh, TIOCSETP, &Savettyb);
#endif !USG
                sleep(1);
                if( blazer_slow )
                        blazer_status_slow(dh);

                if( blazer_fast )
                        blazer_status_fast(dh);
                close(dh);
                if( !strcmp("/dev/cua",devSel) )
                        strcpy(spdy,"/dev/ttya");
                else
                        strcpy(spdy,"/dev/ttyb");
        }
        else
        {
                logent("CAN'T OPEN %s\n",dcname);
                exit(dh);
        }
}

blazer_status_slow(fd)
{

        char *cp,sss[300],qqq[100],*cq;
        char freq_offset[40],line_qual[40];
        int i;

        write(fd, "ATS77?\r", 7);
        /* freq offset */
        cp = freq_offset;
        while (read(fd, cp ,1) == 1)
        {
                if (*cp > 0x20 )
                        break;
        }
        while (++cp < &freq_offset[39] && read(fd, cp, 1) == 1
                   && *cp != '\n')
                ;
        cq = qqq -1;
        while (++cq < &qqq[99] && read(fd, cq, 1) == 1 /* collect OK */
                   && *cq != '\n')
                ;
        *cp-- = '\0';
        if (*cp == '\r')
                *cp = '\0';

        /* line quality */
        write(fd, "ATS78?\r", 7);
        cp = line_qual;
        while (read(fd, cp ,1) == 1)
        {
                if (*cp > ' ')
                        break;
        }
        while (++cp < &line_qual[39] && read(fd, cp, 1) == 1
        cq = qqq;
        while (++cq < &qqq[99] && read(fd, cq, 1) == 1 /* collect OK */
                    && *cq != '\n')
                ;
        *cp-- = '\0';
        if (*cp == '\r')
                *cp = '\0';
        sprintf(sss,"Modem reports freq offset=%s, Line Quality=%s\n",
                        freq_offset,line_qual);
        logent( sss );
}

blazer_status_fast(fd)
{
        char freq_offset[40],line_qual[40],tdr[20],rdr[20],retrans[20],
        packacc[20];
        char *cp,sss[300],*cq,qqq[100];
        double noise[512];
        struct sgttyb mode1;
        int i,bitpersec[512];

        /* freq offset */
        write(fd, "ATS77?\r", 7);
 syncup:
        cp = freq_offset;
        while (read(fd, cp ,1) == 1)
        {
                if (*cp > ' ')
                        break;
        }
        while (++cp < &freq_offset[39] && read(fd, cp, 1) == 1
                    && *cp != '\n')
                ;
        if( freq_offset[0] != '+' && freq_offset[0] != '-')
                goto syncup;
 sync_1:
        cq = qqq-1;
        while (++cq < &qqq[99] && read(fd, cq, 1) == 1 /* collect OK */
                    && *cq != '\n')
                ;
        if( strcmp(qqq,"OK\n") )
                goto sync_1;

        *cp-- = '\0';
        if (*cp == '\r')
                *cp = '\0';

        /* transmit data rate */
        write(fd, "ATS70?\r", 7);
        cp = tdr;
        while (read(fd, cp ,1) == 1)
        {
                if (*cp > ' ')
                        break;
        }
        while (++cp < &tdr[19] && read(fd, cp, 1) == 1
                    && *cp != '\n')
                ;
 sync_2:
        cq = qqq-1;
        while (++cq < &qqq[99] && read(fd, cq, 1) == 1 /* collect OK */
                    && *cq != '\n')
                ;
        if( strcmp(qqq,"OK\n") )
                goto sync_2;
        *cp-- = '\0';
        if (*cp == '\r')
                *cp = '\0';

        /* receive data rate */
        write(fd, "ATS72?\r", 7);
        cp = rdr;
        while (read(fd, cp ,1) == 1)
        {
                if (*cp > ' ')
                        break;
        }
        while (++cp < &rdr[19] && read(fd, cp, 1) == 1
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.