目次

socat を使う (シリアル-TCP変換, etc.) 1
設定 1
シリアルポート → SSH 1
仮想**シリアルポート** 2
Java **からシリアルポートへ**簡単**にアクセスする** 4
設定事例集 4
その他情報 13

socat を使う (シリアル-TCP変換, etc.)

socat を使って、様々なものを相互接続してみます。
たとえば、

- シリアルポート ↔ SSH
- TCP ↔ シリアルポート

などが簡単に接続できます。

※ LAN のように信頼できるネットワーク環境ならよいですが、3G/4G 経由ですと網側の都合で切断されますので、あまりオススメできません。

設定

シリアルポート → SSH

MA-E3xx のシリアルポート “PORT 0” を、server.example.com の SSH へ中継します。
事前に下記を済ませておきます。

- /dev/ttyO1 を一般ユーザ権限で使用するため、user1 を tty グループに所属させる(# usermod -a -G tty user1)
- SSH の秘密鍵・公開鍵ペアを作成する (ssh-keygen)
- SSH の公開鍵を、server.example.com の user アカウントの authorized_keys ファイルに登録しておく

```
user1@plum:~$ socat /dev/ttyO1,echo=0,raw exec:'ssh -l user
server.example.com',pty,setsid,ctty
```

シリアルポートからログインできますが、シェルを exit で抜けると socat も終了してしまうため、
while ループなどで再度実行するようにしたほうが実用上は良いと思います。

SSH でリモート側でシリアルポートのデータを処理するプログラムを実行することで、
セキュアなシリアルポートの Forwarding が実現できます。

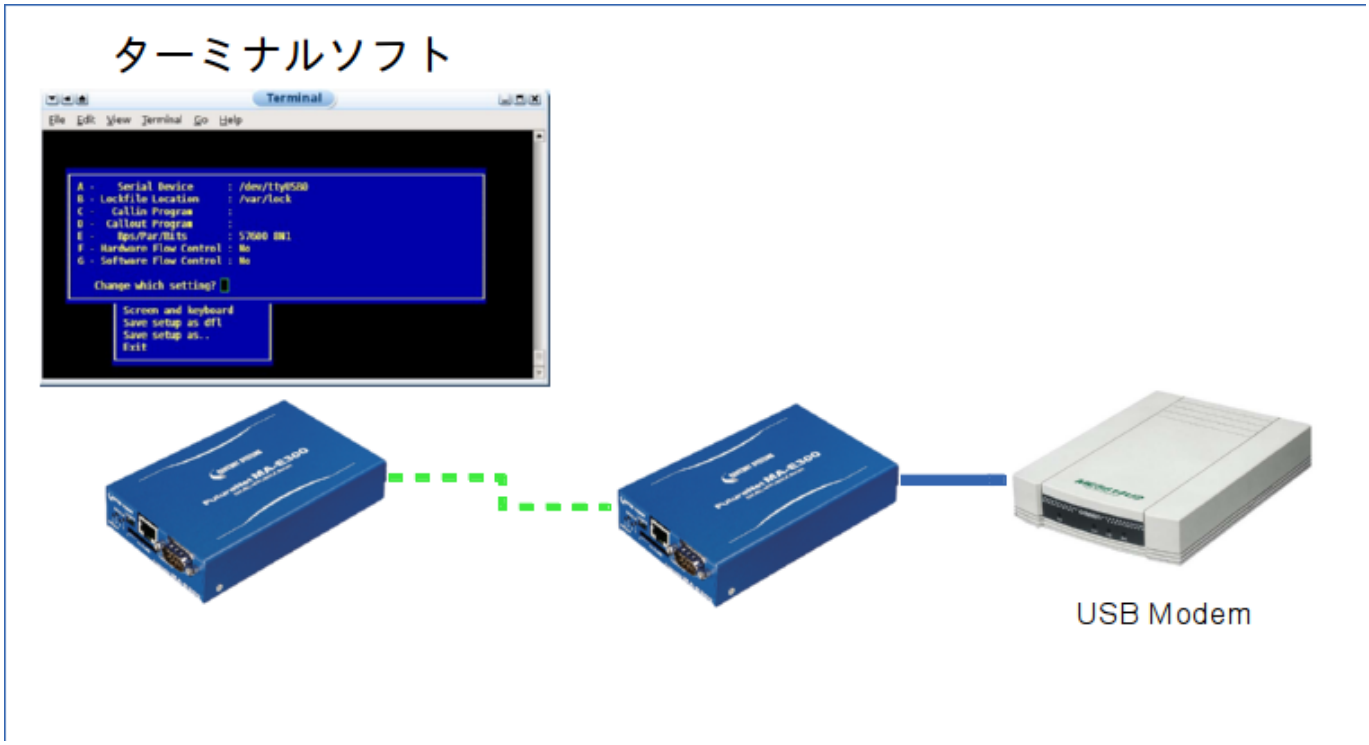
syslog

```
2014/09/04 12:51:55 socat[2549] N opening character device "/dev/ttyO1"
for reading and writing
2014/09/04 12:51:55 socat[2549] N forking off child, using pty for
reading and writing
2014/09/04 12:51:55 socat[2549] N forked off child process 2550
2014/09/04 12:51:55 socat[2549] N forked off child process 2550
2014/09/04 12:51:55 socat[2549] N starting data transfer loop with FDs
[3,3] and [4,4]
2014/09/04 12:51:55 socat[2550] W open("/dev/tty", O_NOCTTY, 0640): No
```

```
such device or address
2014/09/04 12:51:55 socat[2550] N execvp'ing "ssh"
```

仮想シリアルポート

他のマシンのシリアルポートに、ネットワーク経由で接続する。
ネタ元はこちらです。



サーバ側

```
user1@plum:~$ sudo socat -d -d tcp-l:54321,reuseaddr,fork
/dev/ttyACM7,raw,b115200,nonblock,waitlock=/var/run/ttyACM7.lock
```

TCP:54321 で待ち受け、接続してきたクライアントのデータを /dev/ttyACM7 (Omron USB modem) に中継します。

クライアント側

```
user1@plum:~$ sudo socat -d -d pty,link=/dev/vmodem0,waitslave
tcp:192.168.253.49:54321
```

/dev/vmodem0 という仮想シリアルデバイスを作成しています。

接続

クライアント側から /dev/vmodem0 に ターミナルソフトで接続

```
Welcome to minicom 2.6.2

OPTIONS: I18n
Compiled on Feb  8 2013, 07:03:03.
Port /dev/vmodem0, 14:31:44
Press CTRL-A Z for help on special keys
AT
OK
AT
OK
ATI
56000
OK
AT+GMI
+GMI: CONEXANT
OK
AT+GMM
+GMM: V90

OK
```

TCP で接続していますが、インターネット経由の場合 SSL を使ったほうが良いかと思います。

接続 (Telnet)

上の接続例では作成した仮想シリアルポート経由で接続しましたが Telnet で TCP:54321 に直接接続することもできます。

※ MA-E3xx には telnet client が入っていませんので PC から直接接続しています。

```
% telnet 192.168.253.49 54321
Trying 192.168.253.49...
Connected to 192.168.253.49.
Escape character is '^]'.
ATE0

OK
AT
```

```
OK
AT+GMM

+GMM: V90

OK
AT+GMI

+GMI: CONEXANT

OK
AT+GMS

ERROR
^]

telnet> quit
Connection closed.
```

Java からシリアルポートへ簡単にアクセスする

シリアルポート通信のプログラミング (Java) では□RXTX というライブラリを使用する方法を紹介しましたが、
socat を使用してソケットプログラミングにより簡単にアクセスする方法もあるようです。

- Accessing serial ports the easy way - www.rtc1149.net

このような問題があるから、という理由だそうです。

- The Java Communications 3.0 API looks awfully old and unmaintained. It is available for Solaris SPARC, Solaris x86, and Linux x86.
- RXTX is a mix between Java code and C code accessed through the Java native interface. It is hosted on a CVS server and it looks like the 2.2 release will never go out since it got stuck on version 2.2pre2 released in 2009. The last stable version is 2.1.7 from 2006.
- PureJavaComm is a drop-in replacement for those two libraries, written in Java and using JNA to interface with the system. It is simpler to setup than RXTX, is hosted on GitHub and is actively maintained.

設定事例集

- http://www.dest-unreach.org/socat/doc/socat-ttyovertcp.txt
- https://stuff.mit.edu/afs/sipb/machine/penguin-lust/src/socat-1.7.1.2/EXAMPLES より

```
// Examples for using socat (and filan)
```

```
//"$" means normal user, "#" requires privileges, "//" starts a comment

////////////////////////////////////////////////////////////////////////////
///
// similar to netcat

// connect to 10.1.1.1 on port 80 and relay to and from stdio
$ socat - TCP:10.1.1.1:80   # similar to "netcat 10.1.1.1 80"

// listen on port 25, wait for an incoming connection, use CR+NL on this
// connection, relay data to and from stdio;
// then emulate a mailserver by hand :-)
# socat - TCP-LISTEN:25,crlf

// listen on port 25, wait for an incoming connection, use CR+NL on this
// connection, relay data to and from stdio, but have line editing and
history;
// then emulate a mailserver by hand :-)
# socat readline TCP-LISTEN:25,crlf

// provide a transient history enabled front end to stupid line based
// interactive programs
$ socat readline exec:"nslookup",pty,ctty,setsid,echo=0
// same works for ftp (but password is not hidden)

// you may also use a file based history list
$ socat readline,history=.nslookup_hist
exec:"nslookup",pty,ctty,setsid,echo=0
// using ~ as abbreviation for $HOME does not work!

// poor mans 'telnetd' replacement
# socat tcp-l:2023,reuseaddr,fork
exec:/bin/login,pty,setsid,setpgid,stderr,ctty
// and here an appropriate client:
$ socat -,raw,echo=0 tcp:172.16.181.130:2023
// use ssl with client and server certificate for improved security;
// replace /bin/login by /bin/bash when using SSL client authentication, can
be
// run without root then

// this is a cool trick, proposed by Christophe Lohr, to dump communications
to
// two files; it would also work for other manipulations (recode,
compress...)
// and it might also work with netcat ;-)
$ socat TCP-LISTEN:5555 SYSTEM:'tee l2r | socat - "TCP:remote:5555"  | tee
r2l'

////////////////////////////////////////////////////////////////////////////
///
// emergence solution because usleep(1) is not always available
```

```
// this will "sleep" for 0.1s
$ socat -T 0.1 pipe pipe

////////////////////////////////////////////////////////////////////////////
///
// a very primitive HTTP/1.0 echo server (problems: sends reply headers
before
// request; hangs if client does not shutdown - HTTP keep-alive)
// wait for a connection on port 8000; do not wait for request, but
immediately
// start a shell that sends reply headers and an empty line; then echo all
// incoming data back to client
$ socat TCP-LISTEN:8000,crlf SYSTEM:"echo HTTP/1.0 200; echo Content-Type\:
text/plain; echo; cat"

// a less primitive HTTP echo server that sends back not only the reqest but
// also server and client address and port. Might have portability issues
with
// echo
./socat -T 1 -d -d tcp-l:10081,reuseaddr,fork,crlf system:"echo -e
\"\\\"HTTP/1.0 200 OK\\\nDocumentType: text/html\\\n\\\n<html>date:
\$\(date\)<br>server:\$SOCAT_SOCKADDR:\$SOCAT_SOCKPORT<br>client:
\$SOCAT_PEERADDR:\$SOCAT_PEERPORT\\\n<pre>\\\"\"; cat; echo -e
\"\\\"\\\n</pre></html>\\\"\""

////////////////////////////////////////////////////////////////////////////
///
// for communicating with an attached modem, I had reasonable results with
// following command line. Required privileges depend on device mode.
// after leaving socat, type "sane".
// replace /dev/ttyS0 by the correct serial line or with /dev/modem
$ socat readline /dev/ttyS0,raw,echo=0,crlf
// or
$ socat readline /dev/ttyS0,raw,echo=0,crlf,nonblock
// then enter "at$"

////////////////////////////////////////////////////////////////////////////
///
// relay TCP port 80 from everywhere (internet, intranet, dmz) through your
// firewall to your DMZ webserver (like plug-gw)
// listen on port 80; whenever a connection is made, fork a new process
(parent
// process keeps accepting connections), su to nobody, and connect to
// www.dmz.mydomain.org on port 80.
// attention: this is a substitute for a reverse proxy without providing
// application level security.
# socat TCP-LISTEN:80,reuseaddr,fork,su=nobody TCP:www.dmz.mydomain.org:80
// Note: parent process keeps running as root, su after forking

////////////////////////////////////////////////////////////////////////////
///
```

```
// relay mail from your DMZ server through your firewall.
// accept connections only on dmz interface and allow connections only from
// smtp.dmz.mydomain.org.
// the advantages over plug-gw and other relays are:
// * you can bind to an IP address (even an alias), therefore enhance
security
// * in your OS you can create several IP aliases and bind another socat
daemon
//   to each, making several application servers addressable
// * lots of options, like switching user, chroot, IP performance tuning
// * no need for inetd
# socat -lm -d -d TCP-
LISTEN:25,bind=fw.dmz.mydomain.org,fork,su=nobody,range=smtp.dmz.mydomain.or
g/32 TCP:smtp.intra.mydomain.org:25

////////////////////////////////////////////////////////////////////////////
///
// convert line terminator in ascii streams, stdin to stdout
// use unidirectional mode, convert nl to crnl
$ socat -u - -,crlf
// or cr to nl
$ socat -u -,cr -

// save piped data similar to 'tee':
// copies stdin to stdout, but writes everything to the file too
$ socat -,echo=0 open:/tmp/myfile,create,trunc,ignoreeof!!/tmp/myfile

////////////////////////////////////////////////////////////////////////////
///
// intrusion testing

// found an XWindow Server behind IP filters with FTP data hole? (you are
// lucky!)
// prepare your host:
# rm -f /tmp/.X11-unix/X1
// relay a pseudo display :1 on your machine to victim:0
# socat UNIX-LISTEN:/tmp/.X11-unix/X1,fork TCP:host.victim.org:6000,sp=20 &
// and try to take a screendump (must be very lucky - when server has not
even
// host based authentication!)
# xwd -root -display :1 -silent >victim.xwd

// you sit behind a socks firewall that has IP filters but lazily allows
socks
// connections to loopback and has only host based X11 security.
// like above, but from your inside client:
# socat UNIX-LISTEN:/tmp/.X11-unix/X1,fork SOCKS4:firewall:loopback:6000
// or for the HTTP proxy:
# socat UNIX-LISTEN:/tmp/.X11-unix/X1,fork PROXY:firewall:loopback:6000

////////////////////////////////////////////////////////////////////////////
```

```
///
// forms of stdin with stdout, all equivalent
$ socat echo -
$ socat echo STDIO
$ socat echo STDIN!!STDOUT
$ socat echo STDIO!!STDIO
$ socat echo -!!-
$ socat echo FD:0!!FD:1
$ socat echo 0!!1
$ socat echo /dev/stdin!!/dev/stdout    // if your OS provides these

////////////////////////////////////////////////////////////////////////////
///
// some echo address examples
$ socat - PIPE
$ socat - PIPE:/tmp/pipi        // other version of echo
$ socat - PIPE:/tmp/pipi,nonblock!!/tmp/pipi    // other version of echo
$ socat - EXEC:/bin/cat     // another echo
$ socat - SYSTEM:/bin/cat       // another echo
$ socat - TCP:loopback:7    // if inetd echo/TCP service activated
$ socat - UDP:loopback:7    // if inetd echo/UDP service activated
$ socat - /tmp/hugo,trunc,ignoreeof!!/tmp/hugo  // with delay
$ socat - UDP:loopback:2000,bind=:2000  // self "connection"
$ socat - TCP:loopback:2000,bind=:2000  // Linux bug?
# socat - IP:loopback:222   // raw protocol, self "connected" (attention,
// Linux might drop packets with less than 8 bytes payload)

////////////////////////////////////////////////////////////////////////////
///
// unidirectional data transfer
$ socat -u - -
// like "tail -f", but start with showing all file contents
$ socat -u FILE:/var/log/syslog.debug,ignoreeof -
// like "tail -f", but do not show existing file contents
$ socat -u FILE:/var/log/syslog.debug,ignoreeof,seek-end -
// write to new file, create with given permission and group (must be
member) - race condition with group!!!
$ socat -u - CREATE:/tmp/outfile1,group=floppy,perm=0640
//
// for an existing file /tmp/outfile1
# socat -u - FILE:/tmp/outfile1,group=floppy,perm=0700,user=4321


////////////////////////////////////////////////////////////////////////////
///
// file handling
$ socat - FILE:/tmp/outfile1,ignoreeof!!FILE:/tmp/outfile1,append   //
prints outfile1, then echoes input and protocols into file (appends to old
data)

////////////////////////////////////////////////////////////////////////////
```

```
///
// unix socket handling

// create a listening unix socket
$ rm -f /tmp/mysocket; socat UNIX-LISTEN:/tmp/mysocket -
// from another terminal, connect to this socket
$ socat UNIX:/tmp/mysocket -
// then transfer data bidirectionally


////////////////////////////////////////////////////////////////////////////
///
// transport examples

// socks relay (externally socksify applications);
// your ssh client and OS are not socksified, but you want to pass a socks
// server with ssh:
$ socat TCP-LISTEN:10022,fork SOCKS4:socks.mydomain.org:ssh-serv:22
$ ssh -p 10022 loopback
// or better define a ProxyCommand in ~/.ssh/config:
ProxyCommand socat - SOCKS:socks.mydomain.org:%h:%p
// and with proxy:
ProxyCommand socat - PROXY:proxy.mydomain.org:%h:%p,proxyport=8000

////////////////////////////////////////////////////////////////////////////
///
// application examples

// run sendmail daemon with your favorite network options
# socat TCP-LISTEN:25,fork,ip-ttl=4,ip-tos=7,tcp-maxseg=576
EXEC:"/usr/sbin/sendmail -bs",nofork

// local mail delivery over UNIX socket - no SUID program required
# socat UNIX-LISTEN:/tmp/postoffice,fork,perm-early=0666
EXEC:"/usr/sbin/sendmail -bs"
$ socat - /tmp/postoffice

////////////////////////////////////////////////////////////////////////////
///
// uses of filan
// see what your operating system opens for you
$ filan
// or if that was too detailled
$ filan -s
// see what file descriptors are passed via exec function
$ socat - EXEC:filan,nofork
$ socat - EXEC:filan
$ socat - EXEC:filan,pipes,stderr
$ socat - EXEC:filan,pipes
$ socat - EXEC:filan,pty
// see what's done by your shell and with option "pipes"
```

```
$ socat - SYSTEM:filan,pipes
// see if gdb gives you an equivalent environment or opens some files for
your program
$ gdb ./filan
(gdb) r
(gdb) r -s

////////////////////////////////////////////////////////////////////////////
///
// want to use chat from the ppp package?
// note: some OS's do not need "-e" for echo to print control characters
// note: chat might send bytes one by one
// with AIX, a similar program is available under the name "pppdial"
$ socat -d -d tcp:localhost:25,crlf,nodelay exec:'/usr/sbin/chat -v -s
"\"220 \"" "\"HELO loopback\"" "\"250 \"" "\"MAIL FROM: <hugo@localhost>\""
"\"250 \"" "\"RCPT TO: root\"" "\"250 \"" "\"DATA\"" "\"354 \""
"\"test1'$(echo -e "\r.")'\"" "\"250 \"" "\"QUIT\"" "\"221
\"'",pty,echo=0,cr

////////////////////////////////////////////////////////////////////////////
//
// IP6

# socat readline TCP6:[::1]:21  # if your inetd/ftp is listening on ip6


////////////////////////////////////////////////////////////////////////////
///
// application server solutions
// run a program (here: /bin/sh) chrooted, unprivileged;
// parent process stays in real / running as root
# socat -d -d - EXEC:/bin/sh,chroot=/home/sandbox,su=sandbox,pty

// make a program available on the network chrooted, unprivileged;
// parent process stays in / running as root
// script path is already chrooted
# ./socat -lm -d -d TCP-LISTEN:5555,fork
EXEC:/bin/myscript,chroot=/home/sandbox,su=sandbox,pty,stderr
// to avoid terminal problems, you might - instead of telnet - connect using
$ socat -,icanon=0,echo=0 tcp:target:5555; reset


// access local display from ssh server, when ssh port forwarding is
disabled
// socat must be installed on ssh server host
// might have to use xauth...
// this example is one-shot because ssh can handle only one channel
xterm1$ socat -d -d exec:"ssh www.dest-unreach.org rm -f /tmp/.X11-unix/X9;
~/bin/socat -d -d unix-l\:/tmp/.X11-unix/X9\,fork -" unix:/tmp/.X11-unix/X0
xterm2$ ssh target
target$ DISPLAY=:9 myxapplication
```

```
// touch with perms:
// no race condition for perms (applied with creat() call)
$ socat -u /dev/null creat:/tmp/tempfile,perm=0600

// touch with owner and perms:
// race condition before changing owner, but who cares - only root may
access
# socat -u /dev/null creat:/tmp/tempfile,user=user1,perm=0600

// invoke an interactive ssh with exec
// first example passes control chars (^C etc.) to remote server as usual
socat -,echo=0,raw exec:'ssh server',pty,setsid,ctty
// second example interprets control chars on local command line
socat -,echo=0,icanon=0 exec:'ssh server',pty,setsid,ctty
// afterwards, type "reset"!

// convince ssh to provide an "interactive" shell to your script
// three main versions for entering password:
// 1) from your TTY; have 10 seconds to enter password:
(sleep 10; echo "ls"; sleep 1) |socat - exec:'ssh server',pty
// 2) from XWindows (DISPLAY !); again 10 seconds
(sleep 10; echo "ls"; sleep 1) |socat - exec:'ssh server',pty,setsid
// 3) from script
(sleep 5; echo PASSWORD; echo ls; sleep 1) |./socat - exec:'ssh
server',pty,setsid,ctty



// download with proxy CONNECT
// use echo -e if required for \n
$ (echo -e "CONNECT 128.129.130.131:80 HTTP/1.0\n"; sleep 5; echo -e "GET
/download/file HTTP/1.0\n"; sleep 10) |socat -d -d -t 3600 -
tcp:proxy:8080,crlf

// retrieve a file from an sshd site with sourceforge style entry menu;
// fill in your personal values; cat lets you enter your password (will be
// visible on screen)
$ (sleep 10; read pass; echo $pass; sleep 10; echo M; sleep 5; echo cat
FILENAME; sleep 10) |./socat -d -d -ly - EXEC:'ssh -c 3des -l USER
cf.sourceforge.net',pty,setsid,ctty |tee FILENAME

// multicast community on local network: start the following command on all
// participating hosts; like a conference call:
# socat -d -d -d -d - udp-datagram:224.0.0.2:6666,bind=:6666,ip-add-
membership=224.0.0.2:eth0,bindtodevice=eth0
// or
$ socat -d -d -d -d - udp-datagram:224.0.0.2:6666,bind=:6666,ip-add-
membership=224.0.0.2:eth0
// possible reasons for failure:
// iptables or other filters (open your filters as required)
// packets leave via wrong interface (set route: ...)
// socket bound to specific address
```

```
//============================================================================
===
// GENERIC FUNCTION CALLS

// ioctl(): open CD drive (given value valid on Linux)
// on my Linux system I find in /usr/include/linux/cdrom.h the define:
// #define CDROMEJECT           0x5309 /* Ejects the cdrom media */
// the following command makes something like ioctl(fd, CDROMEJECT, NULL)
// (don't care about the read error):
$ socat /dev/cdrom,o-nonblock,ioctl-void=0x5309 -

// setsockopt(): SO_REUSEADDR
// the following command performs - beyond lots of overhead - something
like:
// myint=1; setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &myint, sizeof(myint))
$ socat -u udp-recv:7777,setsockopt-int=1:2:1 -
// setsockopt(): SO_BINDTODEVICE

// ways to apply SO_BINDTODEVICE without using the special socat address
option
// so-bindtodevice:
// with string argument:
$ sudo ./socat tcp-l:7777,setsockopt-string=1:25:eth0 pipe
// with binary argument:
$ sudo ./socat tcp-l:7777,setsockopt-bin=1:25:x6574683000 pipe

============================================================================
===

// not tested, just ideas, or have problems


// traverse firewall for making internal telnet server accessible for
outside
// telnet client, when only outbound traffic (syn-filter) is allowed:
//   on external client run "double server". this process waits for a
// connection from localhost on port 10023, and, when it is established,
waits
// for a connection from anywhere to port 20023:
ext$ socat -d TCP-LISTEN:10023,range=localhost TCP-LISTEN:20023
//   on internal server run double client:
int$ socat -d TCP:localhost:23 TCP:extclient:10023
//   or, with socks firewall:
int$ socat -d TCP:localhost:23 SOCKS:socksserver:extclient:10023
//   login with:
ext$ telnet localhost 20023

// you can make a double server capable of handling multiple instances:
ext$ socat -d TCP-LISTEN:10023,range=localhost,fork TCP-
LISTEN:20023,reuseaddr
```

```
// access remote display via ssh, when ssh port forwarding is disabled
$ socat -d -d EXEC:"ssh target socat - UNIX:/tmp/.X11-unix/X0" TCP-
LISTEN:6030
$ xclock -display localhost:30

// relay multiple webserver addresses through your firewall into your DMZ:
// make IP aliases on your firewall, and then:
# socat -d -d TCP-L:80,bind=fw-addr1,fork TCP:dmz-www1:80
# socat -d -d TCP-L:80,bind=fw-addr2,fork TCP:dmz-www2:80
// and for improved security:
# socat -d -d TCP-L:80,bind=fw-addr3,su=nobody,fork TCP:dmz-www3:80

// proxy an arbitrary IP protocol over your firewall (answers won't work)
# socat -d -d IP:0.0.0.0:150,bind=fwnonsec IP:sec-host:150,bind=fwsec

// proxy an unsupported IP protocol over your firewall, point to point
// end points see firewall interfaces as IP peers!
# socat -d -d IP:nonsec-host:150,bind=fwnonsec IP:sec-host:150,bind=fwsec
// note that, for IPsec, you might face problems that are known with NAT
```

その他情報

- RFC2217 - Telnet Com Port Control Option
- Make RS232 Serial Devices Accessible via Ethernet - Philipp Klaus's Computing Blog
- HW VSP3 - Virtual Serial Port (RFC2217 Windows driver (Free!))