

Building Socket Libraries for TCP/UDP Programs in Minix

Written by Ronghua Wang and Bandan Das
Syracuse University

Copyright © 2006 Wenliang Du, Syracuse University.
The development of this document is funded by the National Science Foundation's Course, Curriculum, and Laboratory Improvement (CCLI) program under Award No. 0618680 and 0231122. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation. A copy of the license can be found at <http://www.gnu.org/licenses/fdl.html>.

Overview

In this document, we give step-by-step instructions on how to create a socket library (libsocket.a), compile/link/build TCP/UDP demo programs using the newly-built library, and run the demo programs. To run the TC/UDP demo programs, we need at least two Minix machines. One machine will serve as the server, while the other will be the client. In my configuration, the IP addresses of the two machines are 192.168.88.129 and 192.168.88.130 respectively. I will use the 192.168.88.129 machine as the server, and the 192.168.88.130 machine as the client. We need to do the same thing for both machines following the procedures in step 1, 2 and 3.

Step 1: Get the files needed

1. Download the `libtcpudp.tar` to your host machine from <http://www.cis.syr.edu/~wedu/seed/Labs/IPSec/files/libtcpudp.tar>.
2. Upload the `libtcpudp.tar` file to your Minix machine, and put it in the directory of `/usr/tmp`. You can use `ftp` to upload the `libtcpudp.tar` file.
3. Login to your Minix machine and do the following:

```
# cd /usr/tmp
# tar xvf libtcpudp.tar
```

You will see three sub-directories: `libsocket`, `tcp` and `udp`.

- In the `libsocket/` directory:
 - `socket.c`: function implementation of the socket communication
 - `socket.h`: header file for the socket program
 - `Makefile`: the makefile used to build the library
- In the `tcp/` directory:
 - `server.c`: server program, which will keep listening on TCP port
 - `client.c`: client program, which tries to connect to the server
- In the `udp/` directory:

- *listener.c*: server program, which will keep listening on UDP port
- *talker.c*: client program, which will send message to the server

In the following steps, we assume our current directory is `/usr/tmp`.

Step 2: Create the socket library of our own

We name the socket library that we will create as `libsocket.a`. Follow the procedures below:

1. Copy the header files to the `/usr/include` directory, using the following command:

```
# cp libsocket/*.h /usr/src/include
```

2. Create a sub-directory under `/usr/src/lib` called `socket`:

```
# mkdir /usr/src/lib/socket
```

3. Copy the function implementation files to `/usr/src/lib/socket` directory:

```
# cp libsocket/*.c /usr/src/lib/socket
```

4. We need to create a file called `Makefile.in` that will generate the required `Makefile`. Follow the instructions below:

```
# cd /usr/src/lib/socket
```

Edit `Makefile.in` (You can use `elle`, `vi`, `emacs`, or `mired` to do this), and type in the following contents:

```
# Makefile for lib/socket.  
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"  
LIBRARIES=libsocket  
libsocket_FILES="socket.c"  
TYPE=both
```

5. Issue the following command to generate the `Makefile`:

```
# make Makefile
```

6. Compile the newly installed library by the following commands:

```
# cd /usr/src/  
# make includes  
# make libraries (Typing just "make" gives you a list of  
make options)
```

Step 3: Compile and link the demo programs

1. Compile the `server.c` and `client.c` programs in `tcp` directory

```
# cd /usr/tmp/tcp
# cc server.c -o server -l socket
# cc client.c -o client -l socket
```

Note: In `cc server.c -o server -l socket`, the letter after “-” in “-l” is the lower-case *L*, not ONE. Please refer to the manual page of `cc` in Minix.

2. Compile the `talker.c` and `listener.c` programs in the `udp` directory

```
# cd /usr/tmp/udp
# cc talker.c -o talker
# cc listener.c -o listener
```

Running TCP client/server programs.

1. Login to the server machine (in our case, `192.168.88.129`):

```
# cd /usr/tmp/tcp
# ./server
```

2. Login to the client machine (in our case, `192.168.88.130`):

```
# cd /usr/tmp/tcp
# ./client 192.168.88.129
```

3. On the server side, you will see the following message:

```
Server: got connection from 192.168.88.130.
```

4. On the client side (`192.168.88.130`), you will see a message:

```
Received: Hello, World!
```

Please note the following: `server.c` and `client.c` are simple TCP communication programs. The server is always listening on a `tcp` port. If get a new connection request from client, the message “Hello, World!” is sent to client. The usage of `client` is “`client ip/hostname`”.

Running UDP talker/listener programs.

1. Login to the Minix machine that is used as the listener (in our case, `192.168.88.129`):

```
# cd /usr/tmp/udp
# ./listener
```

2. Login to the Minix machine that is used as the talker (in our case, 192.168.88.130):

```
# cd /usr/src/tmp
# ./talker 192.168.88.129 {} ``can you hear me now ?``
```

3. On the server (listener) side, you will see a message:

```
listener: from 192.168.88.130, 49153
message: can you hear me now?
```

Please note the following: `talker.c` and `listener.c` are simple communication programs using UDP. The listener listens at a UDP port and displays the incoming messages. The talker basically send messages to the listener using UDP. The usage of the talker is "`talker ip msg`".