CS 603 Caching Socket Library Spring 1996

1 Introduction

The goal of this project is to build a socket library on top of the existing one that caches TCP connections. By using this library connections between clients and servers are seldom closed. Instead they are reused again and again reducing the connection overhead. Applications that make lots of connections such as *httpd* and *netscape* could improve their performance by using this library.

The project is divided in two parts. In the first part you should write a socket call tracer program *socktrace* that displays the function calls of the socket library that are invoked by an arbitrary program. In the second part you use the socket tracer program to implement the caching socket library.

2 Part 1: Socket Call Tracer

In this part you implement *socktrace*. The esqueleton for *socktrace* is in the directory /u/u3/603/96spring/project1/part1/socktrace. The core of *socktrace* is the shared library *socktracelib.so* that defines new socket calls (*socket(), connect(), etc.*) that print the function name, the arguments and then call the original implementation of these calls (*_socket(), _connect(), etc.*). Inside each new call the name, the arguments, and result of the call are printed to *stderr*.

To link *socktracelib.so* to an already compiled application the script *socktrace* defines the environment variable *LD_PRELOAD* to be *./socktracelib.so* . This environment variable defines the libraries that the run-time linker loads before starting the execution of a program. In this way the old socket calls are replaced by the new ones. Later in the script, *socktrace* executes the application that is passed as argument. This application is automatically linked to *./socktracelib.so* before it starts executing.

Copy /u/u3/603/96spring/project1/part1/socktrace to your directory and type *make*. The *Makefile* has everything needed to build the application.

To test *socktrace* you can run:

```
> socktrace finger russo@ector
> socktrace ftp ector
> socktrace netscape
```

As you see, not all the socket calls are traced in *socktracelib.cc* . You have to insert more socket calls to make the tool complete. You should trace the following socket calls: *socket(), socketpair(), listen(), shutdown(), setsockopt(), send(), getsockopt(), bind(), connect(), accept(), getpeername(), getsockname(), recv(), recvfrom(), recvmsg(), sendmsg(), and sendto().*

Also print the arguments and results in a more readable way. For example host addreses should be printed as names. Constants should be printed with a symbolic name instead of numbers etc. The information printed and its format are important for your grade. If you are creative you can make *socktrace* to print at the end the number of times each call is invoked or to trace specific calls.

2.1 What to turn in?

The deadline of Project-1 Part-1 is Tuesday Feb 6 before 6:00pm. You can turn in your projects during class or in my office (CS274).

You have to turn in a printout of the following:

1. All your source files.
2. The output of running the following commands

```
>socktrace finger russo@leonardo

>socktrace ftp <some-host>
        ls <some-directory>
        get <some-file>
        bye

>socktrace telnet <some-host>
        login: <your-account>
        password: <your-password>
        date
        exit
```

```
>socktrace Mosaic
        <follow one link>
        quit
```

3. Answer to the following questions:

   (a) Why the command "socktrace rlogin" does not work?

   (b) Why the following does not work?

   ```
   cd /
   cd <path-where-socktrace-is>/socktrace finger
   ```

   (c) What could you do to solve this problem?

Some manual pages that may be useful to answer the previous questions are ld(1) and ldd(1).

3 Part 2: Cached Socket Libray

In this section you are going to write a new socket library *libcachedsocket.so* that will cache TCP/IP connections. You are going to use the same technique used in the previous part. You will write your own socket functions to cache TCP/IP connections that then may/may not call the real socket calls.

Your library does not need to run for all the programs. You are going to concentrate on running your library with the non-fork version of the web server *httpd* and the web client *mosaic*. The sources of these programs are provided in /u/u3/603/96spring/project1/part2. The sources are only for reference. They should not be modified.

You will have to install and configure your own copy of httpd. To help you in this task follow these steps:

1. Copy the directory */u/u3/603/96spring/project1/part2/httpd-1.4.2* to your home directory, and change to the new *httpd-1.4.2/* directory.

   ```
   chs> cd
   csh> cp -r /u/u3/603/96spring/project1/part2/httpd-1.4.2 .
   csh> cd httpd-1.4.2
   ```

2. Modify the following entries of the *conf/httpd.conf* file:

```
old->   Port 8080
new->   Port <DAY-OF-BIRTH(YYMMDD)>

old-> ServerRoot /u/u3/603/96spring/project1/part2/httpd-1.4.2
new->   ServerRoot <ABSOLUTE-PATH-OF-YOUR-HOME-DIR>/httpd-1.4.2
```

3. Modify the following entries of the *conf/srm.conf* file.

```
old->   DocumentRoot .../project1/part2/httpd-1.4.2/what-i-want-to-export
new->   DocumentRoot <ABSOLUTE-PATH-OF-YOUR-HOME-DIR>/httpd-1.4.2/what-i-want-to-export
```

4. Now run the http deamon

```
csh> ./httpd -d .
```

5. To access your http server run *Mosaic* and open location: *http://MACHINE-WHERE-YOUR-HTTPD-IS-RUNNING:PORT*

6. You can modify the file *what-i-want-to-export/index.html* to export other files you want.

7. To kill the http program execute:

```
csh> kill -9 `cat logs/httpd.pid`
```

You can start by tracing which socket calls *Mosaic* and *httpd* make to establish connections and exchange information. These calls are the calls you have to reimplement to cache connections. You can measure the efficiency of your implementation by the number of real connections that are opened compared to the connections that the program thinks is opening. The html file *what-i-want-to-export/index.html* includes several graphics so you can use it as a test file.

Good Luck.