Fall 2003 CS 450 (Computer Networks)  
Programming Project 2  

Due: November 18th 2003 (Tuesday), at 11:59PM

1 What To Do

For this project you will have to test the performance limits of a concurrent network server. In order to study the server limitations, you will have to implement three programs: server, client and monitor.

- The server is a concurrent file server similar to the one from the first project. This time however, no real files from the server’s filesystem are sent – virtual files are used instead. The server program should allow clients to connect over TCP to the specified port and then start sending a virtual file via the connection. For every established connection, the server should print to the standard output:
  - the total number of clients connected so far,
  - the IP address that the client is connecting from,
  - the port number that the client is connecting from.

Server implementation should not impose any limits on the number of concurrently serviced clients. The limit on the number of connections that the server can handle should depend only on the amount of resources available (e.g. network bandwidth, CPU cycles, number of forked processes, number of used file descriptors). Note that the type of the limiting resource and the limiting size depend on the way in which the server implements concurrent connections. Also, the server should not crash when the concurrency limit is reached and the connections that are already in progress should be serviced without interruption.

Your server program should be invoked by typing:

> server port

where port is the IP port number on which the server will be accepting connections.

- Client programs use TCP to access the virtual files from the server. See section 2 on virtual files for more details. Also, every client program uses UDP to send throughput reports to the monitor process. Clients should measure the throughput and send the reports every 1 second. The report should contain the unique id of the client (an integer number) and the throughput measured in bytes over the last 1 second.
Your client program should be invoked by typing:

```
client server server_port monitor monitor_port id
```

- `server` and `server_port` are the address and port of the server process,
- `monitor` and `monitor_port` are the address and port of the monitor process and
- `id` is an integer number that uniquely identifies the client and is used by it when communicating with the monitor.

- The monitor process receives reports sent by clients over UDP and every 5 seconds prints to the standard output their summary. For a given 5 seconds period, the summary consists of three items: total number of clients that sent the reports during that period, total throughput reported over that period and average throughput per client over that period.

Your monitor program should be invoked by typing:

```
monitor port
```

where `port` is the IP port number on which the monitor will receive reports from clients.

## 2 Virtual Files

For the purpose of performance testing, virtual files are used instead of real ones. Such an approach is chosen because virtual files do not require OS file descriptors and have infinite length (once a client gets connected, it stays connected, since the virtual file transferred never ends). After the server accepts a connection, it should perform a call to `openToReadVirtual()` to obtain a virtual file descriptor. That descriptor is then used in subsequent calls to `readVirtual()` that return the data that should be sent to the client. On the other end of the connection, the client should call `openToWriteVirtual()` to obtain the virtual descriptor and then use it when calling `writeVirtual()` to store data received from the server. Those four functions are provided to you. See the following files that are accessible from CS Sun machines.

```
/home4/instruct/i450/pub/virtualFile.h
/home4/instruct/i450/pub/virtualFile.c
```

## 3 Performance Testing

The goals of the project are two-fold:

1. You should try to reach the limits of the server by overloading it with concurrent connections from clients.

2. You should study how the throughput (both overall and per client) changes as the number of simultaneously connected clients increases.
When doing your tests, the server, the monitor, and the clients should be run on separate machines. Run the server and the monitor and increasingly run more and more clients. Observe the output that the server and monitor produce:

- the number of connected clients reported by the server
- the number of clients that sent the reports to the monitor
- total throughput reported over the 5 seconds periods
- average throughput reported over the 5 seconds periods

You should use only Sun workstations in 2260 SEL.

Do not run the performance tests on CS servers (bert, ernie, oscar, grover)!

4 Deliverables

For this project you should turnin the code as well as a written report and both are equally important. You should turnin the code for the server, the client and the monitor together with a suitable makefile. Your code should compile without errors using /usr/local/bin/gcc and run on CS Solaris machines. Executables’ names are to be server, client and monitor. Code should be written in C/C++, follow good programming practices and be well commented. Clearly state any assumptions you make.

In the report, you should describe the results of your performance testing. Explain how the tests were performed, how the statistics were gathered and analyzed. State what are the limiting resources for your server implementation (number of processes, number of file descriptors etc.) and what is the maximum load (number of concurrently serviced clients and the corresponding throughputs) reached. Describe what could be done to overcome the resource limits and enable even more clients to be serviced. If you were unable to reach the server performance limit you still should provide an explanation of why your approach failed. Also show how the overall and per-process throughput depends on the number of connected clients. Try to answer the following questions

- What is the underlying cause of the throughput distribution?
- When is the maximal throughput achieved (both overall and per-client)?
- What is the optimal number of concurrent clients?
- Is there a difference between the number of concurrent clients reported by the server and the monitor? If so, what is the cause of this difference?

When writing your report be concise, but provide all the necessary information.
5 Things To Remember

- You are allowed to work in teams of up to three people on this project. The work should be distributed evenly among all members of the team as any team member can be asked to present the results in class.

- Take a look at the shell script at:
  `/home4/instruct/i450/pub/spawn.sh`
  You can use it to run a large numbers of clients automatically like this:
  `> spawn.sh 1 10 "nice client s_addr s_port m_addr m_port "`

- You can create a number of processes that do not make extensive use of use system resources by invoking the `sleep` command multiple timea. Also, you can use the `spawn.sh` script to do this, for example:
  `> spawn.sh 6000 6500 sleep`
  will create 500 `sleep` processes, first of them sleeping for 100 minutes, second sleeping for 100 minutes and 1 second, and so on.

- When running large number of processes on one machine, use `nice`. Be considerate of other users of the system and the network. If you want to kill all the processes named `process_name` you can use the following command:
  `> ps | grep process_name | awk {print $1} | xargs kill -9`

- The following commands provide useful information using the Unix manual system:
  `> man socket`
  `> man -s 3HEAD socket`
  `> man bind`
  `> man -s 3SOCKET listen`
  `> man -s 3SOCKET accept`
  `> man connect`
  `> man -s 2 write`
  `> man -s 2 read`
  `> man recv`
  `> man sendto`
  `> man tcp`
  `> man udp`
  `> man nice`
  `> man sleep`

- You should submit your project files using the `turnin` command and a project name of `p2`. One submission/team.

- Besides the project report, you should submit a printout of all your project files. Put the report and the printouts into TA's (Bartłomiej Sieka) mailbox in 905 SEO no later than the next day after the project is due. Do not forget to put your name in each of your files.