Physics 555b Assignment #1 (version 1.0)

Martin Snajdr

Spring 2006

1 Description of the client-server application

The task is to write a client-server application. The client communicates with the server through a *request pipe*. The client sends a request for a service through the request pipe realized as a named (FIFO) pipe. As a part of the request each client provides a unique base name identifying a pair of *communication pipes* through which the client and server communicate.

For the purpose of this assignment the server simply sorts the data provided by the client and sends them back. In a real application the server can perform something more interesting.

The whole client server communication goes like this:

1. The client creates a pair of pipes and sends a request for service to the request pipe.
2. When the server is ready to receive the data it sends a READY message to the client through the private communication pipe. If the server can not allocate the appropriate buffer space for the data it sends back a FAIL message. In this case the client can try to resubmit the request into the request queue.
3. After receiving the READY message the client sends a stream of data to the server.
4. The server receives the data, sorts them and sends them back to the client.
5. The client receives the data. At this point it can delete the communication pipes or reuse them for later communication.

2 Implementation details

The server applicaton uses a team of threads to handle requests. It creates $NTHREADS$ threads at the beginning and assignes each request to a free thread.

The request has the following format

bytes	description
1-2	request_length — the total number of bytes of the request
3-6	data_size — the size of data to sort
7- request_length	base_name — the base name of the communication pipe pair

The base name of the communication pipe pair should contain a full path. The pipe into which the client writes the data is then created by adding the suffix '_in' at the end of the base name. Similarly, the sorted data (and the READY/FAIL message) are written into the pipe created from the base name by adding the suffix '_out'. You can create a unique file name by using the function tempnam(). Read the man pages for the description of tempnam().

We pass the size of the data to be sorted so the server can allocate the appropriate buffer space. If there is not enough memory the server sends a FAIL message to the client.

The data are sent as a stream of double precision float numbers. The amount of memory needed for the server receive buffer is then size_data*sizeof(double) bytes.

The READ/FAIL messages are just a simple character strings READ/FAIL. All strings are \0 terminated.