RFC For Fingering Server And Client

This project covering the making and usage of a fingering **client** and **server** is nearly finished but not yet done. Some missing aspects on the work would be the possibility to ping any machine in the **crémi** building, which is not implemented in the code yet. However the main functions being, fingering the server with someone's name on your own machine, will result in the server sending back the person surname as asked.

Abstract

The purpose of this memo is to describe the **finger** protocol, used to give a user an interface to a remote user and let them communicate. In this we will cover the functionalities as well as the different purposes of our program.

Table of Contents

Introduction

Intent

As mentioned earlier, this memo will cover the different functionalities and usages of the fingering protocol as well as our program based on the **finger** protocol. This protocol main purpose is to provide an interface for a user, to be able to communicate and exchange with another user remotely.

History

This **Finger** protocol was a program at **SAIL** written by Les Earnest, which was one of the inspirations for the **NAME** program. Ken Harrenstien is the author of the RFC 742 named **"Name/Finger**.

Updates

Multiple errors on the original program as of today are now fixed on the memo, such as the optional **/w** switch in the Finger query specifications or the **BNF** int the Finger query specification which was very clear. The flow of events was also improved on the topic of the close of the Finger connection.

Use Of Protocol

Flow Of Events

The Finger protocol is based on the **Transmission Control Protocol**, which consists of one host opening a connection on a specific socket on his side sending a connection request, and a second remote host then proceeds to accept the request, send a message back with a socket closing message, and the connection ends with the first host receiving the message and closing the connection.

Data Format

The data shared between both hosts need to be in a specific format, this format being **ASCII**, no parity and with lines ending according to **CRLF**. Any other format will result in the messages being dropped.

Query specification

The Finger query is defined as follows :

- ::= [|]
- ::= [ ] [ ]
- ::= username
- ::= @host name | @host name
- ::= /W
- ::= < SP > | < SP >

- ::= < C R L F >

The program is recursive, which means that there is no limit on the number of **@hostname** tokens for the query. Another information to have in consideration is that for **Q1** and **Q2** refer to a line that an RUIP receives in reality.

Behavior

As explained earlier, the first host H1 will first open a connection with a second host H2. He will then send this same host a query of his choice, and after that, H2 will have to return the information he wants to send back, as well as close the connection on his side, and H1 will then receive the information and close the connection on his side as well.

Response

Any response respecting the specifications mentioned above can be expected, as the program is meant to be read by people and not a program, it will mainly remain conversations and chats.

Security

Security in this program is as important as in any other, since we are opening direct connections between different hosts, it is highly recommended to protect and ensure the safety of the users. The program can be vulnerable to many attacks, one of them could be code injection using one of the queries with can make, the program can be used maliciously in a **Man in the middle** scenario. So we recommend to limit what a user can access, ask and do with different roles such as administrator etc... It can also be a solution to allow some administrators to keep track of potentially harmful queries with log files.

Examples

- Server side :

  On the server side we simply run the program and wait for connection requests from a client user.

- Client side :

On the client site you can try commands like **< ./client Joao localhost >**, this command will let you open a connection on the server and passing **Joao** as a query. After requesting Joao's surname, the server will then reply to you with corresponding response and close the connection. And finally you will receive the server's response on the terminal.

Thanks

Thanks to the **LinuxHowtos.org** team who made the adequate server and client explanations and who made this program possible. And thanks to the classmates who helped a lot with resourceful discussions on the topic.

Security considerations

Discussed in the Section 3.

Author

Fernandes Lopes Joao Manuel

Université de Bordeaux

M2 CS ASPIC

33000 Bordeaux

Phone : XXX - XXX - XXX

Email : XXX@gmail.com