

CSIS0234B Computer and Communication Networks (Class B)

Tutorial 4

Remote directory listing

In this tutorial, you will write a program for remote directory listing (`rls`) to list the files in a directory of your Unix account, showing the types, names and sizes of the files.

The computer contains a program `list_dir.c`, which lists a local directory. It has a function `list_dir()`, which (by using an auxiliary function `do_list_dir()` that you don't need to change at all) lists all files in a specified directory and place the found information in a linked list; and a `main()` function, which calls `list_dir()` for the command-line argument. Your task is to convert the former to an RPC service that lists a specific directory when requested, and convert the latter to an RPC client that contact the RPC server of another computer, obtain the file list of a remote computer (checking for errors), and print it to the screen.

Guidelines

1. Read the beginning of the program to see what data types are defined. Note also the function prototype of `listdir()`. You will need to convert them to the RPC language.
2. Write a protocol definition `rls.x` for the program. It only needs to contain the definition of a **single service**. However, you will need to **define a structure** for the server to return **either the directory information** or an **error string** to the client. When writing the structure, remember that in many points of the file you are not passing an “optional character”, but instead a “C-style string”.

The server need to pass variable number of directory entries back to the client. Since the number of directory entries is known only when the search for files completes, it can be quite troublesome to use a variable-sized array. The original program uses a linked list. Luckily, the RPC language allows the use of linked list, so the struct written in C can be used directly.¹

3. Read the generated `.h` file to find the definitions created. Repeat steps 1–2 if it is not as desired.
4. Use `rpcgen -Ss` to create a template for server routines, and modify it to add the functionalities of the original `list_dir()` function. Compile the server, as described in the readings. You might need to make a few modification for it to be successfully compiled. But remember that the function signature must not be modified.
5. Use `rpcgen -Sc` to create a template client program, and modify it to add the functionalities of the original `main()` function. Compile the client, as described in the readings.
6. Optionally, write a Makefile to automate the compilation process. A sample one is included in the `hello` directory of the source of the tutorial reading.

¹In the RPC language, a type written like a pointer is actually called “optional data”. In C it is implemented by a pointer, so you might not see the difference. But when transferring to bytes, the content of the pointer is converted, rather than the pointer value itself. This works for a tree and a list, but won't work for general graphs.