SCHOOL OF ENGINEERING AND BUILT ENVIRONMENT
Department of Computer, Communications and Interactive Systems

BSc/BSc (Hons) Computing
BSc/BSc (Hons) Software Development for Business

SYSTEMS PROGRAMMING
module code M3I322922
Coursework Specification and Guidance
Session 2016/2017
First Diet COURSEWORK 2

Module Leader –Edwin Gray

This coursework is to be submitted to the Module Leader no later than **Timetabled labs in Week 12 (w/b 10/04/2017)**

March 2017

Issue 1
07/03/2017

Edwin M Gray, BA, MSc, MBCS, CITP, CPSSADM

Glasgow Caledonian University
Department of Computer, Communications and Interactive Systems

Systems Programming (M3I322922)
Diet 1 Coursework 2 16/17
A Remote File and System Information Service using Threads and Sockets

This coursework constitutes 50% of the total module mark. You must follow the submission instructions at the end of this specification or marks will be zeroed. This is an individual piece of coursework (i.e. not a group assignment/project).

Introduction

The purpose of this coursework is to allow you to demonstrate your understanding of the C programming and systems programming concepts covered in the module. The assignment involves the development and test of a simple **Remote File Service and System Information** client and server, coded in C on Linux which can exchange messages using TCP synchronous sockets. All development will be in **C**. The development environment is **Linux Mint 17.3 64 bit.**

Coursework Requirements

A Remote File and System Information Service is required that allows clients to get information about the server host and to work with file transfer i.e. list server files, copy files across etc. The service must employ TCP synchronous sockets to allow access by its clients. The server will be multi-threaded. Although more than one client can connect to the server at the one time, you do not have to consider any further concurrency issues in this coursework. The server is a text-only application. The client should also be a text-only application and be interactive with a simple menu.

A connected client can request that the server carries out one of several functions repeatedly, on demand and in any order:-

- Return to the client your hardcoded Student ID *from the server* prefixed with the server date and time (as a string).
- Get and display system information about the server.
- Obtain a list of file names for the ordinary files present *on the server* in its "transfer" directory and return these to the client. This can be sent as a single string formatted with a suitable separator.
- Copy the contents of a file specified by the client user (text or binary - of any length) from the server to the client.

The above functions should use suitable system calls e.g. system information can be obtained using uname().

To avoid message boundary problems you should use the rdwrn.c code as provided.
The expectation is that you will develop and run the client and server applications on the same PC.

Each of the client and server programs will have a sub-directory "transfer". This is the

directory to be used for transferring files etc. As it is the same architecture and version of C/Linux on both sides you can ignore any cross-platform complexities involved with serialization.

You should have your application(s) running either on the lab Linux Mint 17.3 64 bit VMWare image or on your own laptop running Linux. It cannot be guarantee that issued code will build or execute on other Linux distros or versions (it most likely will however) and **you develop your coursework on them at your own risk. A** copy of the VMWare Linux lab image is freely available for you to download and use.

The code is to be developed using a simple text editor (gedit/gvim) and built using make at the command line. Do not use Eclipse or another IDE.

The Remote File and System Information Service Client

A simple interactive (i.e. simple menu driven) client is to be implemented which will connect to the server on port 50021 on startup. (No extra marks will be given for implementing a GUI - you are strongly advised not to waste effort implementing one). The options supported through a simple menu interface will be as given in the *Coursework Requirements* section above with an additional option to quit the client (the server will keep going). Note that the client should be *interactive* and that any data values (i.e. file names) *should not be* hardcoded into the client code.

The Remote File Service and System Information Service Server

The server must listen for Socket connections on port 50021 and must start a new thread to process the request sent over each socket connection. This multithreaded approach allows the server to deal with many clients simultaneously. Better courseworks will handle error conditions. The code executed by each thread will be identical. You should use POSIX threads.

Additionally the server should implement a signal handler to gracefully exit, clean up and display the total server execution time. This should be executed on receipt of <ctrl> + c or SIGINT.

For the server only a shared library of the rdwrn code should be created and linked in for the build, rather than compiling the source for it.

Documentation:-

Your work must be submitted in printed, hardcopy form. Put it in a plastic wallet/folder properly labelled on the title page with your name, matriculation number, course and module. Start with an index page and number the pages.

Additionally you must provide the zipped directories and documentation (in Word format) on a properly labelled CD-R. Tape the CD-R securely to the front of the wallet/folder. "Complete code" means you must submit source code, compiled executables, built shared library, Makefiles and data files in the correct directories as when the client and server were last successfully built.

I will copy the solution directories "as is" to a hard disk. *If they do not compile or run* "as is" on the lab VMWare image *I will award zero marks* for this coursework. Therefore check all this before submitting.

*Include a **printed** annotated listing* (printed in ***landscape*** **mode**) of your programs in the documentation. The annotation should explain clearly how your program statements work. (**Legible** handwritten annotation in **black/blue biro** on the printed listing is acceptable.)

Include also:-

- a single page user guide stating simply how to run the client and server
- a testing table

In summary the *printed* documentation should contain:-

- Title Page
- Index
- User Guide
- Testing
- **Annotated Listings (please print these in *landscape* mode)**

Marking Scheme:- Server: 50%
Client: 20%
Annotation: 10%
Testing: 10%
Quality of documentation: 10%

Submission date: Timetabled labs in Week 12 (w/b 10/04/17)

All students must attend both their staffed *and unstaffed* lab hour that day. There may be too many of you to complete this process in the staffed hour.

Please submit your coursework, in a plastic wallet/folder and properly labelled with your name, student ID, course and module number. You are warned that any plagiarism or any failure to submit the coursework by the submission date will be dealt with according to sections 5.5.2 and 5.10 etc. of your Student Handbook which detail the relevant University Assessment Regulations (and it would be wise to re-read these sections now).

Please follow these instructions on what to submit exactly or I will not accept your work.

Coursework Demonstrations:-

These will take place in the timetabled labs in Week 12. You must demonstrate your code working to me on the lab Linux Mint 17.3 64 bit VMWare image or on your own laptop. Failure to attend or to undertake a demonstration means a zero mark will be entered for this coursework.