

# MyGobby

Joseph Cordina

April 12, 2009

This is the description for Assignment 1 of unit CSA 2205, System Programming, for the year 2008/2009. This assignment is worth 30% of the total mark for this part of the unit. The deadline for this project is 10th May 2009. You may do this assignment in a group of maximum 2 people, yet you may do this assignment individually also. A single mark will be allocated to the whole group, so choose your partner carefully and make sure that the work gets evenly distributed. Under no circumstances should code be shared outside your group. You will submit your assignment making use of the ASS system<sup>1</sup>. Those of you without login access, please contact me on joseph.cordina at um.edu.mt. Each team might be asked to present their implementation of your assignment during which your program will be executed and the design explained. Please be reminded that you cannot copy and plagiarize to ease your way to a final submission. While you may discuss ideas with others, do not steal. You can find more information at

[http://www.um.edu.mt/\\_\\_data/assets/pdf\\_file/  
0007/19879/PlagiarismDeclarationForm.pdf](http://www.um.edu.mt/__data/assets/pdf_file/0007/19879/PlagiarismDeclarationForm.pdf)

Read this information well since after-wards no excuses will save you. Anyone resorting to such methods will be considered as trying to cheat and will even risk the removal from the degree program.

## 1 Submission Contents

You will submit your C files together with any accompanying header files (C language has to be used obviously) separately. Please do not use any tar or zip file formats. Additionally include any libraries you might have used. The compilation can assume the presence of 'gcc' on which your system will be compiled. Your code has to work on kronos.cs.um.edu.mt. If your code works only on a Linux machine, you will be penalized (and you have to explicitly mention that it only works on Linux in your report). Include with your submission a *makefile* that can compile your system<sup>2</sup>. You will need a makefile for the server and another for the client code. With your submission, you will provide a file called report.txt which will be a very **brief** description of how your system works and provides the required functionality. Make sure you mention any special techniques or tools you made use of and also any considerations you have made to improve efficiency. Additionally, please note down any bugs your system might have. Listing bugs is common practice, and will benefit the marking process. This file should also contain references to resources you have made use of in your program. Finally, you will submit a file 'architecture.jpg' containing an image that describes in the most concise and clear manner possible your system's architecture (including any important communication paths and structures used).

---

<sup>1</sup>[www.cs.um.edu.mt/~jcord/ass](http://www.cs.um.edu.mt/~jcord/ass)

<sup>2</sup>You can find plenty of tutorials on the net that teach you how to set one up

Before submitting, please do clean up your code. Remove any dead code and useless comments. Every system calls output should be validated for errors and when appropriate, an error message reported to standard output. Also do place some comments to describe your structures in the code, at least to show that you understand how your own program works. Remember C is only unreadable if you make it unreadable. I cannot emphasize enough to be careful with your naming conventions and visual formatting.

## 2 And this is MyGobby

Have you ever wanted to work on a document with someone else, yet would have preferred or could not use the same keyboard and monitor. Wouldn't it be great to be able to see what someone else is writing in realtime, and change things on the fly? *Gobby*<sup>3</sup> is a very popular editor that allows multiple users on a system to collaborate when writing a document. It does this synchronously. So what user1 writes is visible to user2, and vice versa. You are expected to write an editor that behaves in the same manner. Since your time is limited, you will not be asked to write an editor as complex in its UI as *Gobby*, yet the basic functionality has to be there. At the very least your editor is expected to have the following functionality:

- Edit text with all standard keys (including arrow keys, backspace and delete)
- Show clearly where cursor is located
- Save File by pressing CTRL-S (program gives notification of success)
- Exit and Save Program by pressing CTRL-C
- Scrolling of text (up and down)

And in terms of operation your editor should behave as follows:

1. Your program is invoked by passing it a filename.
2. If the filename does not exist, a new file gets created.
3. If file exists, you get a full screen view of the file with the cursor at the beginning of the file.
4. When you type, you get to see what you are typing and *anyone else that is editing the same file gets to see what you type*, with changes being global to all collaborating writers. Note there will only be one instance of the file in disk. If another user is not currently looking at the piece of text that you are editing, he will not be directed to your changes but will see them when he moves to that section of text.
5. The number of users collaborating on a single document is not bounded (at worst will be bounded by system resources)

Sounds easy enough no? Well unless you are feeling very very heroic, I suggest you use the *ncurses* library which will help you a lot with the text editing UI. That is all. It will be up to you to decide on the appropriate data structures and synchronization primitives to use. If necessary you can expect (by specifying it clearly in your documentation) a particular program to be run first that might set up the data structures appropriately and you may expect this program to be terminated last.

---

<sup>3</sup><http://gobby.0x539.de/trac/>

### 3 What might matter

Most of the marks will be allocated to a good design, resulting in a solution free of race conditions, deadlocks and still considerably efficient. For your own good, first design your system before starting the coding. It will result in a better system and a quicker turn around time. For this assignment you cannot make use of the socket API, but you may use any other IPC structure. The processes involved can be assumed to run on the same system, and will be run under the same or different users (having non-root privileges). You cannot assume that the communicating processes are related to parent-child relationships. You can also assume that no deadline extensions will be given. I will compile your code using your provided make files and then play against your implementation a standardized test run which will dictate your marks.

### 4 And To Conclude

Feel free to make use of any optimisations you can think of. Optimisations you should consider are implementation optimisations (better data structures, less indirect references, etc) and also algorithmic optimisations. Keep the allocated time you spend on this project in perspective with the allocated marks. As a suggestion, first sit down and design your solution well before implementing it since this will aid you a lot in your programming.

Good luck and if you need anything, contact me on joseph dot cordina at um.edu.mt. Even better, make use of the discussion group to discuss ideas and ask questions. You can find a link to the discussion group on my website. Other students might help you out in the solution. Just make sure you do not post any code or given concrete solutions.

Note that you will be doing a lot of design work and scrapping lots of ideas, so do not expect to do this assignment in the last few days before the deadline. Do it slowly and the ideas will mature. Remember this is mainly a design assignment (like all good system tools) and you only do the coding after you are sure your design is correct.

I might post on the website a list of designs that impressed me (lets hope there are a couple :) ) !