# Distributed Systems, Spring 2013
# Assignment 1 (a1): Setup and Getting Started
# Due: January 31th before class.

## Introduction

The goal of this lab is to get us started on developing a framework for our games. Our goal is to setup a basic build environment and construct a simple client and server.

## Hand Out Instructions

THIS LAB IS TO BE DONE COLLABORATIVELY WITH YOUR GAME DEVELOPMENT TEAM.

This task of this lab are broken up into several steps:

1. Come up with a team name
2. Git and github
   - Watch the video at `http://learn.github.com/p/intro.html`
   - Do the exercise at `http://try.github.com`. As part of this exercise you will need to sign up for a free github account if you do not have one.
   - When you are done as a team send me an email with your team name, members and github id's. I will then create a github repo for your team and send you info about it.
3. Setup a basic directory structure for your development:
   (a) Create a clone of your repository on curzon to work in:

   ```
   ssh curzon
   cd ~/<somediryoulike>
   git clone git@github.com:BU-CS451-DS/<teamname>.git
   ```

   (b) The prior step will create a subdirectory called <teamname>. Within your team directory create the following three directories: lib, client, server and associated empty READMEs:

   ```
   mkdir lib client server
   touch lib/README client/README server/README
   ```

   (c) Edit the top level README and replace the contents with:

```
Distributed Systems Spring 2013
TEAM: <TEAM NUM> <YOUR TEAM NAME>
MEMBERS:
<MEMBER 0 FULL NAME, EMAIL ADDRESS>
<MEMBER 1 FULL NAME, EMAIL ADDRESS>
<MEMBER 2 FULL NAME, EMAIL ADDRESS>
```

(d) Within your team directory create a LOG file. In this file you will put dated entries that document your team meetings. The format for the entries should be as follows:

```
<output from date command>
Attendees: <list of members>
Questions:
<enumerate list of questions that came up>
Decisions:
<enumerated list of any decisions made>
Todo:
<enumerated list of things that are to be done
 each item should have an owner>
Comments/Notes:
<general summary and notes of meet>
```

(e) commit changes and push to shared repository

```
git add client server lib LOG
git commit -a -m "A descriptive commit message"
git push
```

4. Each team member should create a clone of your team repository in some subdirectory of your home dir.

5. The next stage of this assignment is to create a dummy client, server and library

(a) Create dummy library

```
cd ~/<pathtoyourclone>/<teamname>
```

Create a Makefile in the lib directory with the following contents (don't forget that indented lines must start with a tab):

```
CFLAGS := -g
targets = libdagame.a
src  = $(wildcard *.c)
objs = $(patsubst %.c,%.o,$(src))

all: $(targets)
.PHONY : all

libdagame.a: $(objs)
```

```
        ar -c -r libdagame.a $(objs)

$(objs) : $(src)

clean:
        rm $(objs) $(targets)
```

Create a dummy.h and dummy.c in the lib dir with the following contents respectively:

```
---- contents of dummy.h ----
#ifndef __<TEAMNAME>_DUMMY_H__
#define __<TEAMNAME>_DUMMY_H__

void dummy_hello(void);
#endif

---- contents of dummy.c ----
#include <stdio.h>

void
dummy_hello(void)
{
   printf("dummy_hello(): Hello from lib\n");
}
```

You should now run make in the lib directory and you should end up with a libdagame.a. If not figure out what went wrong.

(b) Create a dummy server. In the server directory create a Makefile and server.c with following contents

```
---- Makefile contents -----
CFLAGS := -g
targets=server
src  = server.c
objs = $(patsubst %.c,%.o,$(src))
libs = -L../lib -ldagame

all: $(targets)
.PHONY: all

server: $(objs) ../lib/libdagame.a
        gcc $(CFLAGS) -o server $(objs) $(libs)

$(objs) : $(src)

clean:
```

```
        rm $(objs) $(targets)

---- server.c contents ----

#include <stdio.h>
#include "../lib/dummy.h"

int
main(int argc, char **argv)
{
   printf("server main: HELLO\n");
   dummy_hello();
   printf("server main: GOODBYE\n");
   return 0;
}
```

You should now be able to run make in the server directory and produce a server executable that does what you expect it too. If not figure out what went wrong.

(c) Create a Makefile and client.c in the client directory based on what you did above for the server. Running make in the client directory should produce a client executable that prints out the following:

```
client main: HELLO
dummy_hello(): Hello from lib
client main: GOODBYE
```

(d) When you are done the above make sure your master repository has a version of all functioning files.

```
cd <to your team directory>
git commit -a -m "first working dummy version of server, client and lib"
git push
```

(e) Go and learn a little more about git and make and have a meeting in which you discuss how you will use it. Don't forget to update your log file and commit the change to document the meeting.

6. Ok now for some real work.

(a) Some team member should read Chapter 3 from UNP

(b) Some team member should read Chapter 4 from UNP

(c) Some team member should read Chapter 5 from UNP

(d) Everyone should read See `http://en.wikipedia.org/wiki/Capture_the_flag`

(e) Have a meeting in which you discuss what you have discovered

(f) Look at the contents of `http://www.cs.bu.edu/~jappavoo/Resources/451/a1/Resources`. You will find all the necessary support files for the rest of the assignment here.

(g) Using the types.h and net.h provided by the instructor create a simple client and server as follows:

**client:** The client (see provided skeleton) should not take any command line arguments. Print a prompt to the user that is:

`<teamname>$`

Provide support for two commands using the functions of net.h which you will need to implement in a net.c :

i. connect <ip:port> : this command should initate the connection to the server. When the client starts it should not be connected to any server. Once a connect command is issued and a successfull connection is made the connect command should not do anything. You only need to support one connection for the life time of the client.

ii. send <string> : this command should send the string to the server. If no connection is made this command does nothing otherwise it should wait for a response string from the server and print that response string out.

**server:** Construct a server that acts as a simple echo server for the clients above (see the skeleton provided). Again you will use the functions defined in net.h to implement the networking functionality. You can use UNP 5.1-5.5 to help guide you. But again implement and use the functions defined in net.c (see the skeleton provided) for the networking code. Your server should take as a command line argument what port it should listen on. At start up it should print the port number that it is listening on to the screen.

(h) Don't forget to regularly commit and push all your code changes to the master repository. Committing often and regularly is a big secret to your success as a "real" programmer. Ensure that your LOG file is up to date documenting all your meetings and non-coding related work too.

(i) Your commit messages should be descriptive as I will be looking at your log messages.

(j) When you are done with this lab create a branch for it called 'a1-done'. I will checkout a version of this branch. Note all actions are time stamped so I will mark based on then contents of your a1-done branch as of the due date and time.

(k) That you should then checkout your master or some other branch to continue the work of this class on.

Most of your work will be to complete the skeletons provided for net.c, client.c and server.c files. Some of it asks you to write explanations for the code in your LOG File you should do so!

7. Please see the README file and associated files to test your client and server. I will be using these to grade your assignment.

**WARNING: DON'T JUST BLINDLY USE THE SKELETONS READ THEM AND UNDERSTAND...ASK QUESTIONS AS NECESSARY**