R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

4002-XXX
Software Development on Linux Systems
Lab 9 – MySQL, CouchDB, SQLite and D-bus

Name: _____ **Section:** _____

Note: This lab may be completed using Ubuntu or Fedora

Activity – Installing Libraries

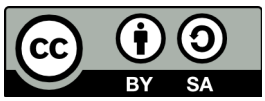
You will be using mysql, couchDB, SqlLite and D-bus in this lab.

NOTE: IF YOU DO NOT MAKE THE ROOT PASSWORD FOR MYSQL dblab , YOU WILL NEED TO MODIFY THE SOURCE CODE GIVEN TO YOU IN THIS LAB.

These are the package names:

Ubuntu: mysql-server [**MAKE THE ROOT PASSWORD FOR MYSQL: dblab]**
libmysql++-dev
python-dbus
curl
couchdb
python-couchdb
sqlite
python-sqlite
libsqlite3-dev
banshee

Fedora: mysql-server [**MAKE THE ROOT PASSWORD FOR MYSQL: dblab]**
mysql-devel
mysql++-devel
dbus-python
curl
couchdb
python-couchdb
sqlite
python-sqlite
sqlite-devel
banshee



R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Configuring MySQL

We will be setting up MySQL and creating a non-root user that can be used for querying data and inserting data into certain tables. **You will be using this database again later in the course.**

1) Open a terminal and type "mysql **-**u root **-**p" and hit enter
2) Type in your password.
3) Create a user named "user1" from anywhere with password "us3r"

   Type "CREATE USER **'**user1**'**@**'**%**'** identified by **'**us3r**'** ; "
4) Download the create_forums.sql script from mycourses
5) Run the script with " **\.** /full/path/to/file/create_forums.sql "

   /full/path/to/file/create_forums.sql is the actual full path of your file. It will probably be something like /home/*lastname*/Downloads/create_forums.sql
6) Now logout of the database with "quit"
7) Login as the new user with "mysql -u user1 -p"
8) Type the password of the use
9) Use the forums database "USE forums**;**"
10) Show the tables in the database with "SHOW TABLES**;**"
11) Select everything from the threads table with "SELECT ***** FROM threads**;**"
12) Now describe the messages table with "DESCRIBE messages**;**"

Instructor/TA sign off _____

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Using MySQL in development

You will be connecting to the mysql database using c++. On Linux it's just as easy to use almost any of the supported languages, but for this exercise we will use c++. Later in the lab we will use python also.

1) Download the file MysqlTest.cpp from Mycourses.

2) Place it in your desired location and cd to that location.

3) Compile the object file (assembler code for the executable to link to)

   " g++ **-W**all -pedantic **-W**no-long-long **-O**2 `mysql_config - -include **-I**/usr/include/mysql++ -c MysqlTest.cpp "

4) Now compile the executable with
   " g++ MysqlTest**.**o `mysql_config **--**libs` -lmysqlpp -o Mysql++_test "

5) Run the executable with " **./**Mysql++_test "

6) Now modify the cpp file (MysqlTest.cpp) to select all of the columns from the "messages" table.

7) Then modify the cpp file to only print out the "id", "messageText" and "poster" columns in the for loop. The actual database query should still collect all of the columns from the table. You just won't print all of them.

   Note: You may want to reference the "describe messages" command you ran in the previous activity.

8) Now recompile the object file and the executable

9) Run the executable and show the results of the "id", "messageText" and "poster" columns from the "messages" table.

Instructor/TA sign off _____

10) Download the MysqlTestInsert.cpp file from mycourses.

11) In terminal connect to mysql with " mysql -u root -p "

12) Choose the forums database with " use forums**;** "

13) See if there are any rows in the messages table by the user "dblab" with

SELECT * FROM messages WHERE poster = "dblab" **;**

NOTE: THERE SHOULD NOT BE ANY ROWS RETURNED.

14) Compile the object file (assembler code for the executable to link to)

" g++ **-W**all -pedantic **-W**no-long-long **-O**2 `mysql_config - -include **-I**/usr/include/mysql++ -c MysqlTestInsert.cpp "

15) Now compile the executable with
" g++ MysqlTestInsert**.**o `mysql_config --libs` -lmysqlpp -o Mysql++_test_insert "

16) Now execute Mysql++_test_insert

17) Rerun the messages table query from above

SELECT * FROM messages WHERE poster = "dblab" **;**

NOTE: THERE SHOULD BE ONE ROW RETURNED

18) Now using the editor of your choice you will modify the MysqlTestInsert.cpp file.

Now modify the code to have the following string attributes.
string id = " **\"\"** " **;**
string messageID = "77777"**;**
string threadID = " **\"**Thread8**\"** ";
string dateCreated = " **\"**8-11-12**\"** "**;**
string threadName = " **\"** My First Thread **\"** "**;**

NOTES: \" IS AN ESCAPE CHARACTER FOR A DOUBLE QUOTE. TEXT INPUT NEEDS TO HAVE DOUBLE QUOTES AROUND IT IN MYSQL. NUMBERS DO NOT. ALSO TWO EMPTY DOUBLE QUOTES ON A PRIMARY KEY MEANS AUTO_INCREMENT.

19) Recompile the object file from the cpp file and make sure there are no errors in your attributes.

20) Now below the line mysqlpp::StoreQueryResult res = query.store()
paste the query string from above. It will look like this.

query << "INSERT INTO MESSAGES (id, messageText, poster, date, attachments) VALUES (" << id << ", " << messageText << ", " << poster << ", " << date << ", " << ", " << attachments << ");";

21) Now modify the new query line to reflect the threads table. In mysql use "describe threads" if you need to.

To do this task, you will need to update the table name, the table values and then change the variables you send in to your variables.

22) Now recompile the code and run the new executable.

23) In mysql run the query

" SELECT * FROM threads WHERE messageID = 77777; "

NOTE: THERE SHOULD BE ONE ROW RETURNED

Instructor/TA sign off _____

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Configuring and Using CouchDB

You will be setting up CouchDB and using queries through curl.

1) Make sure couchdb is running by typing this command in terminal
   " sudo service couchdb restart"

2) Test a connection to couchdb in terminal with
   " curl http://localhost:5984/ "

   It should respond with a welcome message and the software version

3) Open your couchdb configuration file with root privileges

   " sudo gedit /etc/couchdb/default.ini "

4) Find the lines for port number and bind_address

   Change the port to 9999 and comment out the bind address. Commenting out the bind address will allow this service to be used from other machines.
   port = 9999
   ; bind_address = 127.0.0.1

5) Restart couchdb as you did before

6) Now test a connection to the new port

   " curl http://localhost:9999/ "

7) Create a database called "testdb" using curl
   NOTE: DATABASE NAMES IN COUCHDB MUST BE LOWERCASE.
   TABLES/DOCUMENTS DO NOT NEED TO BE LOWERCASE.

   "curl -X PUT http://localhost:9999/testdb "

8) Now retrieve the information about your new database

   "curl -X GET http://localhost:9999/testdb "

   This should give you information, such as the database name and the number of documents stored in it.

9) Now create another database called “mydb” using curl

10) View all of the current databases with

“ curl -X GET http://localhost:9999/_all_dbs

11) Delete mydb with

“ curl -X DELETE http://localhost:9999/mydb “

12) View the current databases again to see that mydb has been deleted

13) Now add a document to the testdb database with

curl -X PUT http://localhost:9999/testdb/myDocument -H “Content-Type: application/json” -d '{ “id”:1, “name”:”me”, “email”:”me@rit.edu” }'

NOTE: IN JSON ONLY NUMBERS MAY BE OUTSIDE OF QUOTES

14) To retrieve your new document type

CURL -X GET http://localhost:9999/testdb/myDocument

15) Now open a web browser on the same machine and go to

http://127.0.0.1:9999/_utils/

This is Futon, the web front-end to couchdb

16) Choose testdb from the database list. You will see all of the documents within testdb.

17) Click on the myDocument document you just made. You will see you can edit all of the attributes, and add attachments or fields.

18) Change the name field to “dblab” and save the document

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Using Couchdb in development

In this activity we will be using python to connect to couchDB. Python could have been used for the section on MySQL, but c++ was used to give examples in other languages. Linux supports most languages for development, but c++ and python are two of the most used. They are also two of the languages that have the most development libraries available, thought you can probably find most libraries available for most major languages.

1) Download couchTest.py from mycourses

   This file will open a connection to your couchdb server, create a database, create tables (documents), update tables and query the database

2) Open the file and examine each step of the code

3) Run the code with ./couchTest.py

4) Add another JSON object (dictionary) for character4. You will do this with the other dictionaries.

   The uid will be 4, the level will be 45 and the name will be 'x'

5) Now modify the query string to return the uid, level and name in a different order

   Rerun the code to make sure it works

6) Now modify the update to update character2's level by 2 instead of one.

7) Now create another update to modify character3's name to “four”

   You will need to use db.get('character3') and save the document when you are finished

8) Write another query at the of the program that will the doc.uid > 2

   Have the function emit doc._id and doc.uid

   Print your results

Turn this program in as part of your lab write up

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Using SQLite in Development

In this activity you will be configuring SQLite and creating tables to use for development. SQLite is a relational database that is designed to be embedded within an application. SQLite is not a client/server model, but an SQL database that runs within the client. This is often used in applications that need to store user and application data. SQLite is common in Android applications and is included in the Android environment.

1) Download the sqliteTest.py file from mycourses

2) Open the file and examine each step of the code

3) Execute the file

4) Now create a table called " videos "

```
CREATE TABLE videos (
        id INT primary key,
        name TEXT,
        type VARCHAR(5),
        duration INT,
        height INT,
        width INT
);
```

5) Add these entries to the "videos" table using the secure question marks to prevent sql injection. HINT: You may do it like the insert loop in the code given.

(1, "vid1", ".mkv", 1800, 1080, 1920),
(2, "vid2", ".mp4", 204, 640, 480),
(3, "vid3", ".avi", 1678, 1440, 900)
(4, "vid4", ".mp4", 439, 1024, 768)

6) Write a query to remove delete "vid3" from the video table

7) Write a query to select all attributes from videos where the video duration is greater than 400 seconds and print your results.

Turn this program in as part of your lab write up

R · I · T

Rochester Institute of Technology
Golisano College of Computing and Information Sciences
Department of Information Sciences and Technologies

Activity – Using D-Bus in Development

The desktop bus (D-Bus) is a communication system for desktop applications. You will be creating an a small desktop application that will use D-Bus to share its methods. You will also create another application to communicate with your first application and communicate with banshee media player.

1) Download the files dbusTest.py and dbusTestClient.py from mycourses

2) Open the file and examine each step of the code

3) Execute dbusTest.py which will run as a dbus process until it is killed

   Note: This process needs to be running for our next process to run

4) Execute dbusTestClient.py

   This execute the hello method and the nextMedia method from dbusTest.py class via dbus. This will show "Hello World" and a song name from the dbusTest.py class

5) Now execute the dbusTestclient.py again without restarting the dbusTest.py. You will see that it has gone to the song name.

   The nextMedia method that is getting called through dbus, goes to the dbusTest.py class and runs the nextMedia method to change the current song in our song list. This is how many of the processes within Linux communicate with each other under the hood. This allows your applications to leverage other application and library code on your machine so that you do not have to duplicate functionality of another library in your own code.

6) Modify the nextMedia method in dbusTest.py to check if the parameter mType was not "next". This can be done with an else statement. If mType was not "next" have it decrement the curPos variable by one. Make sure that this value does not go below 0 and goes back to 4 instead.

   Kill the dbusTest.py process if you have not already. Execute the modified dbusTest.py file

7) Now modify the call to nextMedia in dbusTestClient.py to send "previous" as a parameter instead of "next"

8) Execute your modified dbusTestClient.py file several times to ensure that your songs now iterate backwards and that there are no errors from the array being out of bounds.

9) Now in dbusTest.py create another dbus method called “toggleVisible” that only has the parameter “self”.

   This method will toggle the variable “visible” to “on” or “off” depending on its current state. It will then return the variable visible. Visible has already been defined for you as an attribute.

   Note: You will need to use “global visible” in your method so that it uses the attribute and does not create a local variable called visible.

10) In dbusTestClient.py modify the line *visible = “off”* to call your toggleVisible() method from the TestService object instead of setting it to “off”

11) Restart the dbusTest.py script

12) Run dbusTestClient.py several times and make sure toggles between making banshee visible and hiding banshee.

Turn this program in as part of your lab write up