SCADA ReadMe

LPRDS-BMS-2010

Chris Dionisio
Bradley B. Helland
Andrew Kotsopey
Aaron Lampley
5/15/2010
Contents
Introduction ................................................................................................................................. 3
Goal ........................................................................................................................................... 3
Basic Ubuntu Commands ........................................................................................................ 4
  ssh –Y address ..................................................................................................................... 4
  ls ......................................................................................................................................... 4
  ps aux | grep searchterm .................................................................................................. 5
  mv fromLoc toLoc ............................................................................................................... 5
  sudo command .................................................................................................................. 5
  tail filename ................................................................................................................... 6
  apt-get install packagename ......................................................................................... 6
Starting, Stopping, Compiling the Software ........................................................................... 7
  Starting and Stopping Kernel and the Operational State Manager ..................................... 7
  Starting and Stopping Applications .................................................................................. 7
  Compiling the Software ................................................................................................... 8
Troubleshooting .................................................................................................................... 8
Using the Software .................................................................................................................. 10
  Software Block Diagram ............................................................................................... 10
Pipes .................................................................................................................................... 11
Database ............................................................................................................................... 11
SQL Commands ................................................................................................................... 12
Installing MySQL and MySQL++ ......................................................................................... 13
Sunny Boy .............................................................................................................................. 14
  Using the yasdi library .................................................................................................. 14
Pico-LCD ............................................................................................................................... 14
  Viewing the configuration file and the contents: ......................................................... 14
  Starting and Stopping the configuration file: ............................................................. 15
Package List .......................................................................................................................... 16
**Introduction**

**Goal**

The purpose of the Supervisory Control and Data Acquisition (SCADA) software was to allow monitoring and control of the Lafayette Photovoltaic Research and Development System (LPRDS). The software would run on a computer called the FIT PC which is located in the LPRDS Tower. One general requirement was that all monitoring and control interactions with the system from the applications would be through a defined Application Programming Interface (API). For monitoring, the software had to provide the ability to poll Voltage, Current, and Temperature readings and store them in a Database, which could be accessed by applications as well as by a website. For control, the software had to provide the ability to control the switches and relays in the system from applications. Here is a simple block diagram of our initial view of the software to control the system:
Basic Ubuntu Commands

The software is running on the FIT PC, which has Ubuntu 9.04 installed. For interacting with the software, there are various useful terminal commands listed below.

**ssh –Y address**

This command allows a user to connect to another computer using the Secure Shell protocol. For example, if the user wanted to connect to the FIT PC, the command would be `ssh –Y fit@lprds.aec.lafayette.edu`. This is essentially the same as opening a terminal on the FIT PC itself, which can be useful for interacting with the software and the system from outside the project room. The password for the FIT PC is 111111. An example is shown below.

![SSH Command Example](image)

**ls**

This command lists all of the files in a directory. Adding `–l` switches it to a “long listing format”, essentially putting in a list format.
This command allows a user to search the list of all running processes for a specific term. For example, if the user wanted to see if any process with “lprds” in the name was running, the command would be `ps aux | grep lprds`. The command to view all running process is `ps aux`, and the command to search for a specific term is `grep searchterm`, and the `|` “pipes” the results of `ps aux` into `grep searchterm`. An example is shown below.

```
mv fromLoc toLoc
```

This command moves a file or files from the `fromLoc` to the `toLoc`. For example, if you were in the `/Desktop/scada` directory, and typed `mv kern /usr/local/lprds/bin/lprdskernd`, it would move the file “kern” in the `/Desktop/scada` directory to the `/usr/local/lprds/bin/` directory, and would rename it to lprdskernd. An example is shown in the `sudo` section below.

```
sudo command
```

Putting `sudo` before a command allows the command to be run with root access (runs it as administrator). For example, if the directory you are trying to move a file to has restricted permissions, the `mv` command might give you a permissions error. However, if you put “`sudo`”
before it, it won’t give you an error about permissions. Sudo often asks for a password before executing the command, which, on the FIT PC, is 111111. An example is shown below.

```
fal filename
This command shows the last 10 lines of the file specified by filename. If you add the –f option, it continues to show the end of the file. An example is shown below.

apt-get install packagename
This command is used to install many packages in Ubuntu. It normally requires sudo before the command. For example, if you were to type sudo apt-get install randomPackage, it would install randomPackage.
Starting, Stopping, Compiling the Software

Starting and Stopping Kernel and the Operational State Manager

Both Kernel and the Operational State Manager are supposed to automatically start when the FIT PC is turned on. You can check if they are running using the `ps aux | grep lprds` command, and looking for `lprdskernd` and `lprdsoperd`.

Manually Starting and stopping the Kernel or the Operational State Manager is done using scripts located in the `/dev/init.d/` directory on the FIT PC. Always check to make sure they are not running before trying to start either Kernel or the Operational State Manager.

The command to start the kernel is `sudo /etc/init.d/lprdskernd start`, and the command to the start the Operational State Manager is `sudo /etc/init.d/lprdsoperd start`. Stopping the kernel or Operational State Manager is done through `sudo /etc/init.d/lprdskernd stop` and `sudo /etc/init.d/lprdsoperd stop`.

**Note:** Any application, including the Operational State Manager, must be started after Kernel, and must be stopped before Kernel for the software to work correctly.

Starting and Stopping Applications

All applications other than the Operational State Manager should be started from the `/Desktop/scada` directory on the FIT PC. To start them, you simply need to type `./AppName`. For example, the commands to start the existing applications are: `./maint`, `./mgmt`, `./clear_hv_unsafe`. Keep in mind, all applications require Kernel to be started in order for them to work properly.

Stopping the applications varies slightly between the applications. The maintenance application has an interface, so that the user can simply type “exit” to close the application. The battery management application has no interface, so to stop it the user would need to hold Ctrl and press C to stop it. The clear HV Unsafe application exits automatically after the user input request.
Compiling the Software

Compiling the software is done through a Makefile located in the /Desktop/scada directory. Before compiling an application, you should make sure that the application is not currently running. Once in the directory, simply type “make app” in order to compile the specific application. The make kern command will compile the kernel, and should be followed by sudo mv kern /usr/local/lprds/bin/lprdskernd. The make lprds command will compile the Operational State Manager, and should be followed by sudo mv /usr/local/lprds/bin/lprdsoperd. The make maint command will compile the maintenance application, and does not need to be followed by a mv command. The make mgmt command will compile the battery management application, and does not need to be followed by a mv command. Finally, the make clear_hvUnsafe command will compile the clear HV Unsafe application, and does not need to be followed by a mv command.

Troubleshooting

One possible problem with starting an application can arise from the application being shut down improperly. When this happens, the pipe which connects the application to the kernel will be left, and can interfere with the application starting up correctly. The pipes are located in the /usr/local/lprds/var/ directory, and the application specific pipes have the names “Results_AppName”. So, if the Maintenance application isn’t starting correctly, and typing ls –l /usr/local/lprds/var/ reveals that the Results_Maint pipe still exists, even though the Maintenance Application isn’t running, you should remove the pipe with sudo rm /usr/local/lprds/var/Results_Maint.

If the programs are running correctly, but the DAQ boards aren’t responding, one possible problem is that the SIB is using different USB ports than specified in the xcr.h file. For example, the software may be set to use /dev/ttyUSB0 (for the DAQ boards) and /dev/ttyUSB1 (for the Sunny Boy), while the SIB is set to use /dev/ttyUSB1 and /dev/ttyUSB2. To check the ports the SIB is set to use, type tail –f /var/log/messages, and then unplug the SIB from the USB port on the FIT PC, and then plug it back in. In the xcr.h file, the USB port should be set to the lower numbered USB port, and in the yasdi.ini file the Device should be set to the high numbered USB port. If these numbers have changed, the software will need to be stopped, recompiled, and started again.

If after using the scripts to stop the Operational State Manager or Kernel, the program is still running (you can still see lprdsoperd or lprdskernd using ps aux | grep lprds), you can still shutdown the process using the kill command. This can be used for any application which refuses to shut down. The command to shut it down would be sudo kill -9 pid, where pid is the Process ID. The Process ID can be found using ps aux | grep <name>, it is the number in the second
To fully shutdown and restart all LPRDS software, here are the steps to follow:

1. Stop any applications that are running using the methods listed under *Starting and Stopping Applications* and *Starting and Stopping Kernel and the Operational State Manager*. If these methods don’t work, then shutdown the processes using the `kill` command mentioned above.
2. Type `sudo rm /usr/local/lprds/var/*` to ensure that all pipes have been removed.
3. Start the Kernel, then the Operational State Manager, and then any applications that you want to run (assuming they are allowed to run: Note that the battery management application is only allowed to run in the Operational State).
Using the Software

For a description of the individual blocks, see the SCADA Software Architecture document.

When an application wants to get data from the Data Acquisition boards, it calls the appropriate C++ API function. This request goes through the Kernel pipe (explained in a later section) to the Kernel. Kernel then calls the appropriate function in IOMgr, which formats the packet that needs to be sent to the Data Acquisition (DAQ) boards. XCR is responsible for physically writing the packet to the USB port, which gets sent to the SIB. The SIB passes the packet along to the DAQ boards using RS-485, which have sensors that are controlled by a microcontroller. The microcontroller sends a response back with the requested value, which is passed through the SIB to the USB port. XCR reads the packet, and passes it to IOMgr which decodes the packet, sending only the requested value back to Kernel. Finally, Kernel returns the requested value through Results pipe to the API, and to the application.

The data flow for controlling the digital outputs on the DAQ boards is exactly the same as for requesting data. Sunny Boy requests are also done in a similar fashion, except that the Sunny Boy class is used instead of the IOMgr and XCR.
Every time that data is received from the DAQ boards or Sunny Boy, as well as when switches are set, the data or switch position is recorded in the database. This data is available to the website using PHP to access the database.

**Pipes**

The C++ API uses a UNIX feature called pipes to allow communication between the applications and the Kernel, which is constantly running. Pipes are essentially a FIFO between two running processes. One application can write to the pipe, and then another application can read from it. Writing to and reading from pipes is very similar to interacting with a file, as pipes are essentially special files that are optimized for having multiple processes write to them.

In the interactions between the Kernel and the applications, there are two kinds of pipes: the kernel pipe, and the Results pipes. There is only 1 kernel pipe, which is meant to be read by Kernel, and written to by applications. There is one Results pipe for each application, which is created when the application calls the Connect function in the API, and deleted when the application calls the Disconnect function in the API. All pipes are created in the /usr/local/lprds/var/ directory.

**Database**

The LPRDS software has the ability to automatically log system information to a MySQL database. This was accomplished by creating a MySQL database locally on the fitPC and using MySQL++, which is a commercial package that allows communication from C++ to MySQL. The current database can be viewed on the fitPC with a graphical user interface within firefox. A ‘database’ link is available in the bookmark bar, or you can type in the following address:

[http://localhost/phpmyadmin/](http://localhost/phpmyadmin/)

Username: root  
Password: 111111  
Database: SCADAdata  
Server = localhost

The database is arranged into subsections known as tables with each entry in the table known as a row. The SCADAdata database is arranged into 9 tables, each with an auto-incrementing index named ‘id’. Tables such as event_log, fault_log, state_log and data have additional fields ending in ‘_id’ (event_id, hw_id etc.) that correspond to the ‘id’ number of the table named in the prefix. For example, the ‘hw’ table stores detailed information about each sensor and assigns each a unique index ‘id’ number. When a sensor reading is stored in the ‘data’ table, it only needs to store the matching ‘id’ number from the ‘hw’ table in the ‘hw_id’ field to identify the sensor. This was done to minimize the amount of space used by the database.
SQL Commands

To manipulate data in the database we issue commands written in the Structured Query Language (SQL). In the LPRDS system, SQL commands are commonly used to retrieve data, write new data, and update previous entries. Below are a few SQL command examples:

\[
SE\text{LSELECT} * \text{ FROM } \text{hw WHERE daq } = 4 \text{ AND pin } = 7
\]

The ‘SELECT’ command is used to retrieve information back from the database. The ‘*’ means retrieve every field from the table. ‘FROM hw’ specifies the table to pull data from and ‘WHERE’ is used to filter information by finding entries that match the following specifications.

\[
\text{SELECT data.id, data.hw_id, hw.daq, hw.pin, hw.name, data.value, hw.units, hw.type, hw.active, data.time FROM hw, data WHERE data.hw_id = hw.id}
\]

In this ‘SELECT’ command, information is gathered from two separate tables. To do this, each desired field must be prefixed with its table name, (ex. data.id for the id field in the data table)
and ‘FROM’ is used to specify the tables. In this case, WHERE is used to match up each ‘data’ entry with its detailed sensor information in the ‘hw’ table by matching data.hw_id to hw.id.

\[
\text{INSERT INTO state_log (state_id, time) values ( 3, NOW() )}
\]

The ‘INSERT INTO’ command allows us to add a new entry to the database. It is best to specify the fields you intend to populate after the table name within parenthesis, leaving out ‘id’ to avoid problems with the automatically incremented index field in each table. Use NOW() for the timestamp value to write the current time. Any text must be enclosed in single quotes ‘’.

\[
\text{UPDATE hw SET active = 1 WHERE name = 'sensor_name'}
\]

To change the value of information already in the database, use the ‘UPDATE’ and ‘SET’ command. The values to be updated can be specified using ‘WHERE’ as seen before. Any text must be enclosed in single quotes ‘’.

\[
\text{ORDER BY event_log.time ASC}
\]

The ‘ORDER BY’ option can be added to the end of a ‘SELECT’ command to arrange the results by their timestamp field. Use ‘ASC’ for ascending time and ‘DESC’ for descending.

**Tip:** When formulating a new SQL command, it is convenient to use the phpmyadmin user interface in firefox. You will find a tab near the top of the page labeled ‘SQL’ where you can type in commands and get feedback if there is an error in the formatting.

**Installing MySQL and MySQL++**

MySQL and MySQL++ are already configured on the fitPC and right-most Ubuntu computer along the windows in AEC400. To install MySQL from scratch (as well as PHP and Apache for the website) follow this tutorial:

https://help.ubuntu.com/community/ApacheMySQLPHP

To install MySQL++, run the following commands in terminal:

```
wget http://www.tangentsoft.net/mysql++/releases/mysql++-3.0.9.tar.gz
 tar xvfz mysql++-3.0.9.tar.gz
 cd mysql++-3.0.9
 ./configure
 make
 make install
```
To use MySQL++ in a C++ program, you first will need to modify the Makefile with the following lines:

```
CFLAGS := -I /usr/include/mysql -I /usr/local/include/mysql++
LDFLAGS := -L /usr/local/lib -lmysqlpp -lmysqlclient -lnsl -lz -lm
$(CFLAGS) $(LDFLAGS)
```

Then within the .h file include…

```
#include <mysql++.h>
#include <stdlib.h>
using namespace mysqlpp;
```

**Sunny Boy**

**Using the yasdi library:**

The Sunny Boy communication protocol is implemented using the YASDI (yet another SMA data implementation) library which was provided by the Sunny Boy manufacturer. The Sunny Boy must be connected to the FIT PC in order to be properly communicated with via an RS-485 connection. The current Sunny Boy class utilizes only a few of the provided functions from the YASDI library; however, only a select few are required in order to poll data from it. These methods and their definitions are located within the sunnyBoy.cpp file. Since these methods draw upon the YASDI library, those libraries must be compiled along with the Sunny Boy code. Consult the Sunny Boy Communication Manual to see how to use this code to communicate with the Sunny Boy.

**Pico-LCD**

**Viewing the configuration file and the contents:**

The file that determines the layout for the Pico-LCD is located in /usr/local/lprds/etc/, the file is named lcd4linux-lprds.conf. By modifying this file, the displayed elements of the Pico-LCD screen can be changed. In the beginning of the file there are various widgets that are currently being displayed on the Pico-LCD screen. There are additional widgets available to add and a list of available widget is located at [http://ssl.bulix.org/projects/lcd4linux/wiki/Layout](http://ssl.bulix.org/projects/lcd4linux/wiki/Layout) along with their respective fields and examples. Near the bottom of the screen is the layout.
which describes where the widgets will be displayed on the Pico-LCD screen. Here is an example screenshot of how the widgets are put into the file:

![Screenshot of widget configuration file]

The exec expression executes a terminal command and prints the result on the LCD screen at a refresh rate specified by the second argument.

Starting and Stopping the configuration file:

The configuration file mentioned above is currently set to be run on startup so when the fit pc is booted up the layout described in the file will be displayed. The startup script, lcd4linux, located in the /etc/init.d directory on the FIT PC, is responsible for calling the correct commands to configure the layout. It has a start and stop command which can be called manually from the terminal by `sudo /etc/init.d/lcd4linux start` and `sudo /etc/init.d/lcd4linux stop`. These are useful if the configuration file has been changed and the changes are desired to be displayed immediately.
Package List

Here is a list of the packages that were installed on the FIT PC to run the SCADA software. Many packages can be installed using the Synaptic Package Manager found in the System>Administration menu.

<table>
<thead>
<tr>
<th>Package</th>
<th>MySQL++</th>
<th>G++</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNUPLUT</td>
<td>MySQL++</td>
<td>G++</td>
</tr>
<tr>
<td>php5-mysql</td>
<td>mysql-server</td>
<td>mysql-client-5.0</td>
</tr>
<tr>
<td>phpmyadmin</td>
<td>mysql-server-core-5.0</td>
<td>mysql-client</td>
</tr>
<tr>
<td>mysql-common</td>
<td>mysql-server-5.0</td>
<td>libmysqlclient15off</td>
</tr>
<tr>
<td>libqt4-sql-mysql</td>
<td>libmysqlclient15-off</td>
<td>libmysqlclient15-dev</td>
</tr>
</tbody>
</table>