Client Server Interaction and Simple Remote Control(Play songs clients chose)

Mingkun Yang 900506-T008 minyan09@student.hh.se
Hakan Yildirgan 790214-T113 hakyil09@student.hh.se
Vivekanandan Mahadevan 860901-T492 vivmah09@student.hh.se

March 14 2010

Abstract

Clients go to the web page and then make request for songs by providing the names. Server will store the names into a list and show the particular client what he or she chose. Server will pick up songs from the list and play them in a certain.

Introduction

Clients and server will interact with each other via a particular web page.

On this web page, there are two forms. One is a text-box and the other is a drop-down list. After receiving the name of a song, server appends it to a list(the list is achieved using queue data structure in the fashion of FIFO), and shows the client what he or she chose.

Every one minute, the server retrieves the first song from the list, plays it and removes it until it reaches the end of the list.

Analysis

The problem can be easily broken down into two sub problems.

1. Clients and server interaction
2. Remote control — play music

For the first sub problem, the following tasks have to be performed.

- Install a web server and make it work properly.
  Available web server: Apache2, Sun Java System Web Server
- An html page has to be designed such that the user can give the name of the song as the input and he or she can also select the song from the drop-down list.
- The CGI script will be designed such that it reads the input (name of the song), stores it in the list and displays the name.
  Available language for CGI: bash,PHP,Perl

For the second sub problem, the following tasks have to be finished:

- Server retireves the content of the list in FIFO fashion, plays it using a music player and removes it from the list for every minute.
  Available music players: mpg123,MPD&MPC
  Available scheduling tool: cron

Implementation

Clients and server interaction

1. According to the Linux-Apache-MySOL-PHP(LAMP) server environment, which many Internet Web servers use for hosting online stores, blogs, and applications, we use Apache2(the latest version).
2. Out of the same reason, we choose to use PHP.

Remote control — play music

1. We used MPD&MPC to play the songs because it is much more effective than mpg321. Its difficult to make use of the control operation (play, pause, forward) in the case of mpg321.

Solution

For sub problem 1:

0.0.1 Apache2

Installation

sudo apt-get install apache2

Configuration

It is reasonable to deal with all files in current user's home directory, so we need to tell Apache where we put our html files and script files. In this project, we put all the files related to the web page or sub problem 1 into /home/username/html.
We should open the file " /etc/apache2/sites-available/default " as a super user.

sudo gedit /etc/apache2/sites-available/default

We should perform the following changes to point the files present in the folder html.(/home/username/html/)

DocumentRoot /home/username/html/
<Directory /home/username/html>
ScriptAlias /bin/ /home/username/bin
<Directory "/home/username/bin">

0.0.2 PHP

sudo apt-get install php5

0.0.3 Scripts

Put "process1.php process2.php read.php project.html web.css queue [1]" in html folder.

[1] All necessary files and explanations will be found in Appendix

(create html folder if this folder does not exist using following commands:

cd
mkdir html

).

The following are for sub problem 2:

0.0.4 MPD&MPC

sudo apt-get install mpc mpd
sudo ln -s ~/Music /var/lib/mpd/music

The second command creates a soft symbolic link in /var/lib/mpd/music, pointing to Music folder in home directory.

0.0.5 Scripts&Music

Put"play [2]" in bin folder.
Put the three songs in Music folder.
(create bin folder if this folder does not exist using following commands:

cd
mkdir bin

).

0.0.6 Cron

We use crontab to execute the play script for every minute.

crontab -e

will edit crontab, and the following need to be written in the crontab.

* * * * * ~/bin/play

Here the "play"script which is present in the bin folder will be executed for every minute.

[2]All necessary files and explanations will be found in Appendix

Once we finished our project, we utilized our University wireless network to completely test our project.

We used two to three laptops to test our project in which we made one of our laptop as server and we copied all the necessary scripts to test our project to the server. Then with the help of other two laptops, we logged into our html page by typing the server's IP address and then we gave the songs of our wish as an input to the server.

we enabled crontab to exucute our scripts every minute and we tested whether the input in the form of songs were played sequentially like a queue for every minute. We also cross-checked whether the input from the queue was deleted once it was played successfully using the music player.

These are testing methodologies we used to test the efficiency of our project.

Conclusion

We implement a script which will take the input from the user either through keyboard or from the drop-down list, and the following can be the extensions.

1. A forum can be created using this project and the clients can send their profiles or requests to the server. All users' profiles and passwords(encrypted) will be stored in a database created in the server. Users of this forum could have easy access to their own account and customise what songs will appear in their drop down list.
2. This project gives one simple but non-trivial illustration of remote control. This could be extended to all kinds of tasks by adding corresponding task-listener in server. This will be much more convenient and could save administrators' time and all kinds of expenditure on transportation.

Bibliography

[1] Richard Blum *Linux Command Line and Shell Scripting Bible* ISBN: 978-0-470-25128-7

[2] Bill Kennedy, Chuck Musciano *HTML & XHTML: The Definitive Guide, 5th Edition* ISBN: 0-596-00382-X

.1 Source Code

project.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>


<head>
<title>MUSIC</title>
<link rel="stylesheet" type="text/css" href="web.css" />
</head>


<body>


<h1><strong>Music World</strong></h1>


<h2>Input the name of a song</h2>
<form action="process1.php" method="post">
Enter name:<input name="1" type="text" />
<input type="submit" value="submit" />
</form>


<h2>Select the name of a song</h2>
<form action="process2.php" method="post">
<select name="2">
<option value="sunshine in the rain">sunshine in the rain</option>
```

```
<option value="take me to your heart">take me to your heart</option>
<option value="my heart will go on">my heart will go on</option>
</select>
<input type="submit" value="submit" />
</form>

</body>
</html>
```

read.php

We used "read.php" to read the input and to put the input in the queue.

```
<?php function read($data){
$queue = "queue";
$filehandle= fopen($queue, 'a') or die("can't open file");
fwrite($filehandle,"$data"."\n");
fclose($filehandle);}
?>
```

The queue file is opened in append mode and it is captured by the variable $filehandle.

```
$filehandle= fopen($queue, 'a') or die("can't open file");
```

Then the user's input is written into the file queue.

```
fwrite($filehandle,"$data"."\n");
```

Once all nput has been stored in the queue, the file has to be closed.

```
fclose($filehandle);
```

process1.php

We used the script process1 to get the input from the user through keyboard.

```
<?php
require("/home/albertnet/html/read.php");
$data=$_POST['1'];
switch($data){
case "sunshine in the rain":$flag=1;break;
case "take me to your heart":$flag=1;break;
case "my heart will go on":$flag=1;break;
```

```
default:
$flag=0;
echo “there is no such song”;}
if($flag==1){
echo $data<br />;
read($data);}
?>
```

If the input given by user is any of these 3, then the flag is set to 1. For all other input it will be 0, because there are only 3 songs which are present in the music folder. This is achieved by making use of switch case.

If the flag is set to 1 , then the corresponding input is taken as the input to the function read.

process2.php

```
<?php
require(“read.php”);
$data=$_POST[‘2’];
echo “$data<br />”;
read($data);
?>
```

As we see, we make use of the function read() to put the users input into the queue.

play

```
#!/bin/bash PLAY=` which mpc`
music=` head -n 1  /html/queue`
if [ -n “$music” ];then
tail -n +2  /html/queue > ~/html/temp
rm ~/html/queue
mv ~/html/temp ~/html/queue
chmod 666  /html/queue
if [ -z ` $PLAy playlist` ];then
$PLAY ls — $PLAY add
fi
n=`$PLAY playlist — sed -n “/$music/p” — gawk ´print $1´ —
sed -e ´s/>//;s/)//´`
$PLAY play $n
fi
```

Here the player which we selected to play the song is mpc . First song from the queue is selected by making use of the command

```
music=`head -n 1 ~/html/queue`
```

The first song is captured by the variable music and if there is a song, then the statements present inside the if loop will be executed.

```
tail -n +2 ~/html/queue > ~/html/temp
```

The command tail is used to capture the content from the second line to the end of the queue and it is redirected to temp.

```
rm ~/html/queue
mv ~/html/temp ~/html/queue
```

Once the rest of the content of the queue is moved to temp, the queue is removed and then the temp file is renamed to queue. We are doing this because once the song is played it has to be removed from the queue.

```
chmod 666 ~/html/queue
```

Then we are giving access right 666 to the file queue, which means we are making the file queue readable and writable by anyone.

```
n=`$PLAY playlist — sed -n "/$music/p" — gawk ´print $1´ —
sed -e ´s/>//;s/)//´`
$PLAY play $n
```

The variable music contains the name of the song. The music player takes only the number as its argument. Therefore, the songs corresponding number is matched and it is captured by the variable n. Then finally we played the song by passing the number through n as an argument.

Appendix A

Ethic discussion

Anyone, who find this project and want to do further development with it, should keep the original authors' name and contact remained.

Even thought this website is available from outside, it will not be used to make a profit without buying the songs' copyright.

Appendix B

Project Schedule

	Work
Week 1 (Dec 7-14)	Install Ubuntu, set up one web server and create one web page containing necessary elements
Week 2 (Dec 15-32)	Write scripts achieving the function that prints the user's text input and store it in one file in web server
Week 3 (Dec23-30)	Pick the first line of the file and play the corresponding music in interval of 1 minute
Week 4 (Dec 31-Jan 6)	Prepare the report