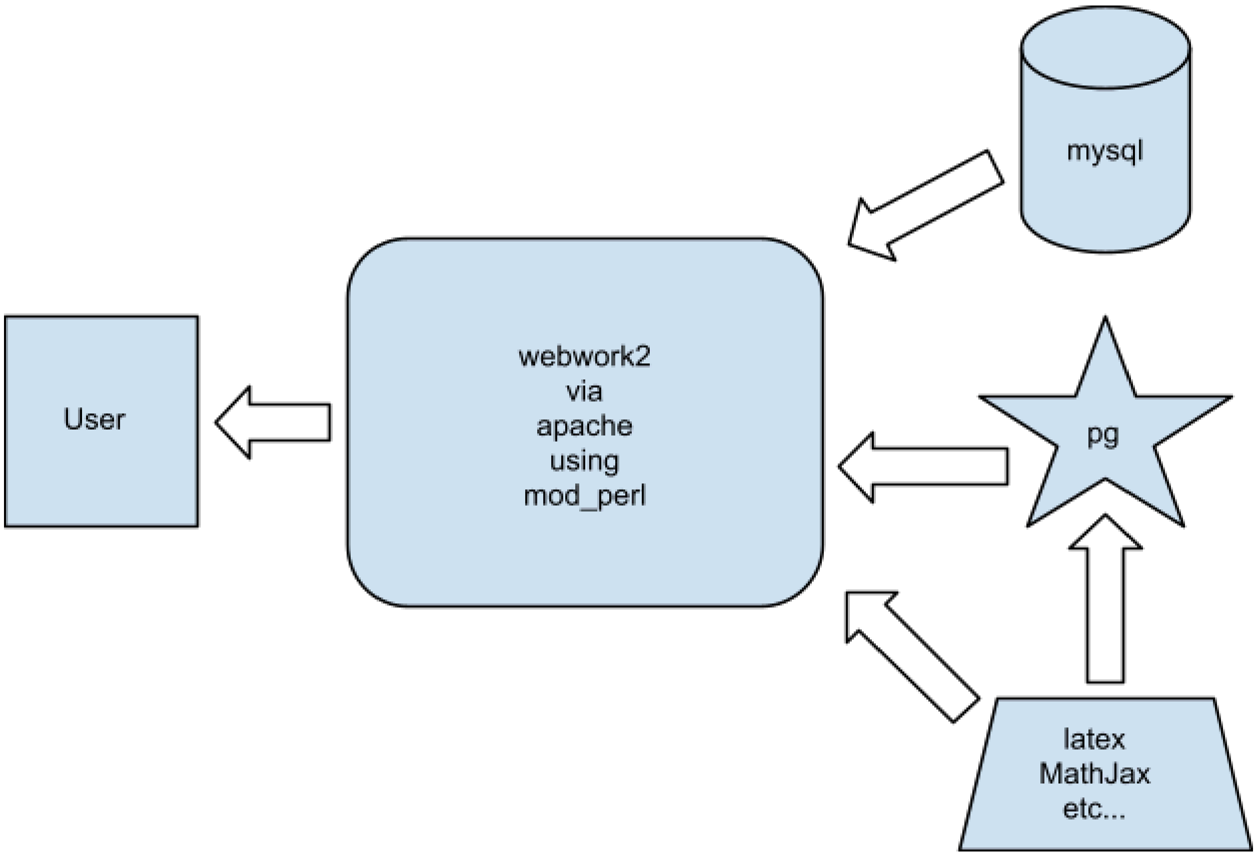
Manual Installation Outline

Rough outline for Session 1 PM for a manual installation of WeBWorK.
Note: This was done on a specially prepared system with a lot of the prerequisite packages installed and is not, by itself, a good set of installation instructions.



To install WeBWorK manually we have to do the following:

- Set up the user and directory structure for WeBWorK.
- Get the files for WeBWorK, pg and other git based components.
- Install apache, mysql, and other necessary software.
- Set up apache and mysql.
- Set folder permissions and make other miscellaneous preparations.
- Set up all of the various configuration files.
- Build the admin course, generate the OPL database and finalize installation.
- Test installation.

We start by upgrading and updating our system. Something to keep in mind is that this walkthrough done on an debian or apt based system (specifically Ubuntu). Most of the commands are the same for other systems, however there will be important differences. One big difference is that on a Red Hat or yum based system all of the apt commands will be replaced by corresponding yum commands. For example `apt-get upgrade` is `yum update`.

```
      sudo apt-get update / sudo yum check-update
      sudo apt-get upgrade / sudo yum update
```

Next we add the WeBWorK administrator user and the WeBWorK data group. Your machine may already have a wwadmin user if you created that user at installation time. The wwdata group is so that we can have files which are accessible by both the wwadmin user and the apache user. So we create wwadmin, and wwdata, and we add wwadmin to wwdata. We also add wwadmin to the sudo group so that wwadmin can install packages and restart the server. Note: in some systems, like Red Hat, the sudo group should be the wheel group. Finally we switch users and start acting as the wwadmin user.

```
      sudo adduser wwadmin
      sudo addgroup wwdata
      sudo adduser wwadmin wwdata
      sudo adduser wwadmin sudo
      su wwadmin
```

Now we install git, which is used to do version control for WeBWorK. We also create the /opt/webwork directory, change ownership to wwadmin, and use git to clone into the two main components of WeBWorK, webwork2 and pg.

```
      sudo apt-get install git
      cd /opt/
      sudo mkdir webwork
      sudo chown wwadmin:wwadmin webwork
      cd webwork/
      git clone http://github.com/openwebwork/webwork2.git
      git clone http://github.com/openwebwork/pg.git
```

We also have to use git to get a couple of other important components. We make a libraries directory and get the OPL and we get MathJax. We also create a courses directory. This completes the directory structure for WeBWorK.

```
      mkdir libraries
      cd libraries/
      git clone
      http://github.com/openwebwork/webwork-open-problem-library.git
      cd ..
      git clone http://github.com/mathjax/MathJax.git
      mkdir courses
```

Now we need to install two important components for WeBWorK, the apache webserver and the mysql database server. We do this using apt or yum, whichever is on the system. There are actually quite a few other packages we should install here, but for brevity most of them are preloaded onto this system for the demonstration.

```
      sudo apt-get install apache2-mpm-prefork
      sudo apt-get install mysql-server
      … and install lots and lots of other packages …
```

Just for reference a mostly complete is of packages we would need to install on a debian based system is:

- gcc
- make
- git
- subversion
- perl
- perl-modules
- dvipng
- netpbm
- unzip
- preview-latex-style
- texlive-latex-base
- texlive-latex-recommended
- mysql-server
- openssh-server
- apache2-mpm-prefork
- libapache2-request-perl
- libdatetime-perl
- libdbi-perl
- libdbd-mysql-perl
- libemail-address-perl
- libexception-class-perl
- libextutils-xsbuilder-perl
- libgd-gd2-perl
- libapache2-mod-fcgid
- liblocale-maketext-lexicon-perl
- libmime-tools-perl
- libnet-ip-perl
- libnet-ldap-perl
- libnet-oauth-perl
- libossp-uuid-perl
- libpadwalker-perl
- libyaml-perl libtemplate-perl
- libphp-serialization-perl
- libsoap-lite-perl
- libsql-abstract-perl
- libstring-shellquote-perl
- libtimedate-perl
- libuuid-tiny-perl
- libxml-parser-perl
- libxml-writer-perl
- libpod-wsdl-perl
- libjson-perl
- libtext-csv-perl
- libhtml-scrubber-perl
- texlive-generic-recommended
- texlive-fonts-recommended
- libfile-find-rule-perl

As an aside, if you are using CentOs or RedHat you will need to add the EPEL repositories to get all of the packages we need for WeBWorK. In general we will also need to install some things via the perl package manager CPAN. This works the same on any system. In this case, almost everything has been installed for us. However, just to demonstrate

```
    sudo cpan Iterator
```

Just for reference a mostly complete list of perl modules we would need to install on a debian based system is:

- XML::Parser::EasyTree
- HTML::Template
- Iterator
- Iterator::Util
- Mail::Sender
- Dancer
- Dancer::Plugin::Database
- Plack::Runner Plack::Handler::FCGI
- Path::Class
- Array::Utils
- File::Find::Rule
- Path::Class
- File::Slurp

Now we start to configure a few things. For apache we first figure out the apache user, www-data in this case, httpd in other cases, and add it to the wwdata group. We also enable some necessary apache modules. In particular we need to make sure that apache is using the mpm_prefork module and not the incompatible mpm_event module. We also need to make sure the mod_perl modules are enabled. As a note for people using Red Hat based systems, apache2 tends to be referred to as httpd.

```
more /etc/apache2/apache2.conf
more /etc/apache2/envvars
sudo adduser www-data wwdata
sudo a2dismod mpm_event
sudo a2enmod mpm_prefork
sudo a2enmod perl
sudo a2enmod apreq2
```

Finally there are some apache configuration variables we should set to limit the number of simultaneous users that can connect to the server. This is because WeBWorK users are much more demanding on a system than someone viewing a static webpage. We want to set the Timeout to 1200, MaxRequestWorkers to 20 (or to approximately 5 per gb of system memory), and MaxConnectionsPerChild to 100.

```
cd /etc/apache2/
emacs apache2.conf
> Timeout 1200
emacs mods-enabled/mpm_prefork.conf
> MaxRequestWorkers 20
> MaxConnectionsPerChild 100
```

Now we configure the mysql database. We create the WeBWorK database and the webworkWrite mysql user. We also make a password for the webworkWrite user. This should be different from your wwadmin password.

```
mysql -u root -p
>CREATE DATABASE WeBWorK;
>GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP,
     LOCK TABLES
>ON WeBWorK.* TO webworkWrite@localhost IDENTIFIED
     BY 'password';
>exit
```

Next we export some variables to our wwadmin environment. This will make using WeBWorK script tools easier.

```
cd /home/wwadmin
echo 'export PATH=$PATH:/opt/webwork/webwork2/bin' >> .bashrc
echo 'export WEBWORK_ROOT=/opt/webwork/webwork2' >> .bashrc
echo 'export PG_ROOT=/opt/webwork/pg' >> .bashrc
source .bashrc
```

Now we need to check for missing modules and packages. We use the check_modules.pl script and look for packages with stars next to them. We would like to install packages using apt or yum if possible, so we look there first. However, there are a few perl packages which have to be installed using the perl package manager cpan.

```
    cd /opt/webwork/webwork2/
    check_modules.pl
    apt-cache search Text::CSV
    sudo apt-get install libtext-csv-perl
    check_modules.pl
    apt-cache search HTML::Scrubber
    sudo apt-get install libhtml-scrubber-perl
```

Note, even though in this case apt-cache search doesn't turn anything up for HTML::Scrubber, a google search shows that there is a package available.

```
    apt-cache search Array::Utils
    sudo cpan Array::Utils
```

For Array::Utils there isn't a good debian package available so we install using cpan. Using cpan will work as a fall back for almost any perl module that's missing. Missing binaries have to be installed using the package manager. We also run pdflatex on bin/check_latex.tex to look for missing latex packages. (Latex came preinstalled on this system so there isn't anything missing.)

```
    check_modules.pl
    apt-cache search dvipng
    sudo apt-get install dvipng
    check_modules.pl
    pdflatex bin/check_latex.tex
```

Now we create the coruses directory from the "distribution" courses directory. One option is to make modelCourse a symlink to the modelCourse directory in courses.dist. This ensures that your model course will be kept up to date with the distribution model course.

```
    cd /opt/webwork/courses
    cp ../webwork2/courses.dist/* .
    cp -R ../webwork2/courses.dist/* .
    rm -rf modelCourse/
    ln -s ../webwork2/courses.dist/modelCourse modelCourse
```

Now we need to set our permissions appropriately. First, by default all of the files in WeBWorK will be readable by everybody. This is necessary because the webserver has to be able to read all of these files. Furthermore, the webserver has to be able to write to some files, but it's dangerous for it to be able to write to all of them. So we make some directories have the wwdata group (which contains the webserver user) and have those be group writable with the sticky bit. This allows the webserver user to create things like course files, or temporary files or log files, etc…

```
    cd /opt/webwork/webwork2/
    ls -la
    chgrp -R wwdata DATA ../courses htdocs/tmp logs tmp
    chmod -R g+w DATA ../courses htdocs/tmp logs tmp
    find DATA/ ../courses/ htdocs/tmp/ logs/ tmp/ -type d -a
```

```
          -exec chmod g+s {} \;
```
Next we show off git. We can use git to pull the latest updates from the openwebwork repository. There isn't anything new now, but you should do this before the beginning of every semester. If you want to stay up to date. We can also checkout the develop branch here for access to extra, but sometimes unstable, features. Note, the webwork2 and pg folders operate independently.
```
git pull origin master
git checkout origin develop
git checkout origin master
cd ../pg
git pull origin master
git checkout origin develop
git checkout origin master
cd ../libraries/webwork-open-problem-library
git pull origin master
```
Now we need to set up the configuration files. First we copy all of the configuration files over from their distribution versions. The only one that really needs our attention is site.conf. Here we edit the file and add in a bunch of information including:

- The website address (!)
- Changing the database user password to be what we set it to earlier. (!)
- Double checking the WeBWorK server user.
- Adding an smtp mail server to enable the email features in WeBWorK. (You would get your smtp server information from your local IT person.)
- Checking the default timezone and locale.

As a side note about how WeBWorK config files function. WeBWorK has a cascading configuration setup where files that are evaluated "later" can override values set in files evaluated "earlier". In general the order of evaluation is

1. defaults.conf, site.conf
2. localOverrides.conf
3. course.conf, simple.conf.

The default values are set first. Then the global site overrides are evaluated in localOverrides.conf. Finally variables set in course.conf or simple.conf can override anything, but are course specific. Now, moving on, while we are looking through the config files we can also look at localOverrides to see what other options we want to mess with; enabling update notification for example. We need to check which version of apache we have (usually between 2 and 2.4 or above 2.4) and choose the appropriate apache config file to use.
```
cd ../webwork2/conf/
more database.conf.dist
cp site.conf.dist site.conf
emacs site.conf (set site and db info)
cp localOverrides.conf.dist localOverrides.conf
emacs localOverrides.conf (make other changes)
cp database.conf.dist database.conf
```

```
    sudo apachectl status
    cp webwork.apache2.4-config.dist webwork.apache2.4-config
```
Now we need to enable the site in apache. We can look at the sites-available portion of the apache2 config to see what is available. If you want to set up more standard webserver sites, or enable https, this is where you go. We put a link to our apache configuration file here and enable it. Finally we restart the server and cross our fingers.
```
    cd /etc/apache2/
    cd sites-available/
    sudo ln -s /opt/webwork/webwork2/conf/webwork.apache2.4-config
    webwork.conf
    sudo a2ensite webwork
    sudo apachectl restart
```
We have a couple of extra things to do. First we need to make the admin course. We switch over to the courses directory, and change our working group to wwdata. This means files we create will have the wwdata group and be accessible by the webserver. Then we add the admin course and make the file which will hide it from view on the main WeBWorK page. Next we compile the color c program, which is used by the chromatic component. Last but not least we need to update the OPL database.
```
    cd /opt/webwork/courses/
    newgrp wwdata
    addcourse admin --db-layout=sql_single
         --users=adminClasslist.lst --professors=admin
    cd admin
    touch hide_directory
    cd ../pg/lib/chromatic
    gcc -O3 -o color color.c
    OPL-update
```
Now with any luck we have a running WeBWorK server. We should go to the admin page and begin checking things out. Some things which we should test are:

- Register!
- Check that we can access the admin page and create a course.
- Log into the new course and check that we can create a student.
- Check that we can import the demo set.
- Check that we can check answers to a problem and that the answers are checked correctly.
- Go to the Library Browser and check that we can create a set, view problems, and add problems to the set.
- Change due dates and check that we can submit problems to a set. Check that the answers show up in Past Answers.
- Check that you can edit a problem and save a local version.
- Check that user settings are working, in particular check both images and MathJax display modes.
- Download a hard copy of a homework set to check latex system.

- If email was set up, check that we can send an email using the mailer.

Now that we have a working system, lets take a quick look at what the logs for the system look like. These logs are very important because if the server ever breaks, the log files are the first place you should look. In general there are three kinds of logs you should keep in mind.

- System logs, e.g. apache logs.
- Webwork logs, e.g. server logs, course logs.

Now that we have actually used our server we should have some data in the logs to look at. First, lets see the apache logs. These will contain any server errors. Note on Red Hat based systems the apache logs are stored in /var/log/httpd.

```
sudo ls /var/log/apache2/
tail /var/log/apache2/error.log
ls /opt/webwork/webwork2/logs
ls /opt/webwork/courses/TestCourse/logs
tail /opt/webwork/courses/TestCourse/logs/login.log
tail /opt/webwork/courses/TestCourse/logs/answer_log
```

The course logs are useful for telling who has been logging in and if they really submitted answers when they said they did.

Once we are satisfied that the installation is successful we can start thinking about additional modifications. Two common ones are enabling ssl for security, and linking WeBWorK authentication with your university authentication. For the first we will look at securing http. We begin by enabling the ssl site. Then we input our certificate information and redirect traffic to https. We *must* change our address in site.conf to use https as well. You will need to get in touch with your universities it people to get a ssl certificate, but it is the single biggest step you can make for improving your server security.

```
sudo a2enmod ssl
cd /etc/apache2/sites-enabled
sudo a2ensite default-ssl
emacs default-ssl.conf (set up cert)
emacs 000-default.conf (set up redirect)
> Redirect permanent / https://webwork.myuniv.edu/
cd /opt/webwork/webwork2/conf
emacs site.conf (change http to https)
```

Next we will look at setting up alternate authentication. For this situation we will be going through ldap as an example because it is the most common. There general process for setting up the other authentication modules (CAS or LTI) is similar but the specifics are very different. First we create the conf file from the distribution file. Then we edit the conf file with information we get from our it contacts concerning our university LDAP server. Next we enable the module in localOverrides.conf

```
cd /opt/webwork/webwork2/conf
cp authen_ldap.conf.dist authen_ldap.conf
emacs authen_LTI.conf (setting school specific options)
emacs localOverrides.conf (uncomment ldap line)
```

Break Questions: (command line only!)

- Easy
    - Install the package “emacs” (it has the same name for both yum and apt).
    - Install the module “Text::CSV“ from CPAN.
    - Look in the localOverrides file (https://github.com/openwebwork/webwork2/blob/master/conf/localOverrides.conf.dist) and find where you would change the default language for your installation.
    - Add a new user to your system and give them sudo permissions.
- Medium
    - Install the apache webserver using your package manager. (Yum and apt have different package names for this.)
    - Install the perl module “File::Slurp”, preferably using the package manager, but possibly using CPAN
    - Determine if the default installation of the apache webserver for your distribution is version 2 to 2.4, or version 2.4 or greater.
    - Look in the localOverrides file (https://github.com/openwebwork/webwork2/blob/master/conf/defaults.config) and figure out what variable controls whether students can change their passwords or not.
- Hard
    - Use your package manager to get the source code and build dependencies for the ‘vim’ package, and compile your own version of the package.
    - By changing where webwork stores its temporary files in localOverrides you can set WeBWorK up so that these files are served by the smaller, faster web server lighttpd. Figure out how to do this.