

Contents

I	Setting up the Server	2
0.1	System Requirements	3
0.2	Installing the Base System	4
0.2.1	Acquire Program Files	4
0.2.2	Configure PostgreSQL	4
0.2.3	Edit Configuration Files	4
0.2.4	Initialize the Database	5
0.2.5	Email Notification Script	5
0.2.6	Final Tasks	7
0.3	Installing Optional Components	7
0.3.1	SSL Support	7
0.3.2	External Authentication	7

Part I

Setting up the Server

0.1 System Requirements

The calendaring system has very modest system requirements. It runs on top of any UNIX or UNIX type system such as FreeBSD and any Linux, and would probably run on more platforms but they have not been tested. The main tested platforms are FreeBSD, Gentoo Linux, and SuSE Linux. FreeBSD and Gentoo Linux were chosen for the production environment due to their configurability, flexibility, and lean nature. The main requirements are these:

- UNIX type system such as FreeBSD or Linux
- PostgreSQL Server (7.3.x or later)
- Apache Web Server (1.3.26 or later)
 - mod_php4 (4.3.x or later)
 - * blowfish encryption support
 - * gd
 - * postgres
 - * mcrypt
 - * mhash
 - * jpeg
 - * png
 - UNIX type Cron scheduler access

Requirements for optional features are the following:

- SSL support
 - mod_ssl (2.8.14 or later)
 - mod_php4 (4.3.x or later) with SSL support

0.2 Installing the Base System

0.2.1 Acquire Program Files

Download the latest release of Kronophobia from <http://kronophobia.sourceforge.net> and unpack the archive into either the root of your web server directory (if you are using a dedicated calendaring system) or into a subdirectory of your web root (if you are running on a web server that hosts other applications as well). Make sure to note the location you placed the files into. For the remainder of this guide we will be assuming that the folder name is a sub-folder called **calendar** and the full system path to this directory is `/www/webroot/calendar`.

0.2.2 Configure PostgreSQL

The following commands are a general template for creating a database and a user in PostgreSQL. We will assume that the username to create is **caluser** and that the database to create is called **calendar**. Also, you will need to obtain the username that PostgreSQL runs under in your system. This is usually **pgsql** or **postgres** and we will refer to it as **POSTGREUSER** in the commands below. If you do not know the username open up `/etc/passwd` in your favorite editor and look for a username that looks like `postgre` or something similar. Issue the following commands at a root prompt:

```
# psql template1 POSTGREUSER
template1=# CREATE USER caluser PASSWORD 'your_password' NOCREATEDB;
template1=# CREATE DATABASE calendar OWNER=caluser;
template1=# \q
```

0.2.3 Edit Configuration Files

The next step is to edit several of the Kronophobia configuration files and customize them with specific values that correspond to your environment. Inside the calendar folder you just extracted will be a folder called **config**. The following are the files in the config folder that we will be editing:

- `config.php`
- `db.php`
- `security.php`

config.php

This is the main Kronophobia configuration file and includes options for many things such as system paths, email servers, authentication methods, etc.

db.php

This file is where you enter the database name and user information for PostgreSQL that you created earlier.

security.php

This is the security file for Kronophobia. Here you can set the encryption code, SSL settings and authentication type for the system.

0.2.4 Initialize the Database

After editing the configuration files the database must be initialized for first use. Inside the Kronophobia folder there is a folder called **install**. This folder contains the initialization script. To run the script, open up a web browser and type in the URL to your Kronophobia installation followed by the commands. Here is an example using our imaginary installation:

<http://www.mydomain.com/calendar/install/install.php>

If everything is successful then you will get a series of Green success messages in your browser. If anything fails, go back to the configuration files and make sure that you entered everything in correctly. Make sure that your database name and username are correct in db.php. If you still cannot get it to work then save the HTML page from the output and mail it to the project leader along with as much detail as possible about your operating system, environment, etc. You can find the contact information at **<http://kronophobia.sourceforge.net>**.

0.2.5 Email Notification Script

Kronophobia features a script that sends out notification emails to parents/teachers/etc. that have chosen to track events when the events change.

In order to keep performance of the calendaring system very high the script that sends out the mails runs as its own process using Cron to schedule it. This script **kronophobia_utility.sh** is initially located in the scripts subfolder of Kronophobia but it should be moved to a more secure location and assigned appropriate permissions since it will run on a regular basis. Make sure that the permissions assigned to it allow it to read the Kronophobia configuration files located in your webroot.

We will be assuming that you placed **kronophobia_utility.sh** into `/root/bin/` on your server. Open this file in your favorite editor and on the first line put in the appropriate path to your PHP CLI interpreter. Example:

```
#!/usr/local/bin/php
```

Make sure to leave the `#!` characters in front of the path as it tells the shell what program to run. If you do not have a PHP CLI interpreter then you will have to install it. As of PHP 4.3 it comes with the CLI version built-in and installs by default. On FreeBSD you will have to install the full php port in order to get both the Apache module and CLI versions.

The most common mistake when editing the line mentioned in the previous paragraph is to hit enter after putting in the path to your php executable. Resist the temptation! If you put a carriage return in your file after the path and before the php start tag you will get an error and your script will not run.

The last step is to schedule the script to run from Cron. You should schedule this to run every 10 minutes (alarms are based on 10 minute intervals). Add the following line to the end of `/etc/crontab`:

```
*/10 * * * * root /root/bin/kronophobia_utility.sh
```

If you are using a cron implementation that doesn't support the above syntax you should check the documentation for how to do 10 minute intervals.

0.2.6 Final Tasks

If you will not be using external authentication such as an NT or NIS domain then delete the **auth** subfolder.

Make sure to DELETE the **install** and the **upgrade** folders after installation or anyone can mess up your database!

0.3 Installing Optional Components

0.3.1 SSL Support

Setting up SSL support is a snap, just configure Apache to use SSL and then make sure you enabled SSL in **c_security.php**. That's it.

0.3.2 External Authentication

Make sure that you have configured the **c_security.php** file to use the correct external authentication options. Next, go into the **auth** subfolder in the Kronophobia directory and create a symlink to **index.php** one level up by issuing the following command:

```
# ln -s ../index.php index.php
```

Configure a **.htaccess** file in the **auth** directory that requires a valid-user and uses an Apache module to authenticate with an external system. How to do that is beyond the scope of this document but you can consult the Apache documentation and other sources on the web if needed. To login to the system via the external authentication the users must click on the link above the normal login boxes that appears when you setup external authentication. It will say something like:

SCHOOL teachers & students click [HERE](#)

Normal users that use DB authentication or parents logging into the system after they have signed up for an account will continue to use the normal login boxes for authentication.