Instructions for Server Usage

Host `naf.cs.hut.fi` is our course server for completing assignments 2 and 3. It runs a standard Debian Linux with some basic tools installed. If the server lacks some tools that you need, please contact the course staff. If you run into problems, please send us an email or reserve a reception time for yourself.

Accessing Server

You may login with SSH from the university network. If you are outside the university network, connect with your favourite SSH client first to `kosh.aalto.fi` (using Aalto user account) and then `ssh` to `naf.cs.hut.fi`. Remember to specify your username, e.g. `ssh mmeikal@naf.cs.hut.fi`.

Username: u + your student number (letter in lowercase); e.g. "u12345a"

Password: the same as with the subversion repository, you may change the server password with `passwd` (change won't affect SVN)

URLs

- `http://groupXX.naf.cs.hut.fi/`

HTTPS is also available, but the signature is self-signed.

Home Directories

Every student group in the course has only one user account, but with a different login name and password for each student group member. The home directory has permissions that prevent anyone else but you from seeing the files in it. It also contains symbolic links to the HTTP log files of your virtual host.

Software Versions

- PHP 5.3
- Python 2.6
- Ruby 1.9.2
- Django 1.3
- Rails 3.0.5

Ways to Serve Your Website

One Apache process group serves everyone. All your applications with the same privileges as you otherwise have.

Web Content

Apache serves `public_html` in the home directory. For every empty directory under `public_html`, Apache searches by default for `index.html`, `index.fcgi`, `index.php` and `index.cgi` (in this order).

CGI and FastCGI

With CGI and FCGI, you can run almost anything. CGI is a simple way to direct the HTTP request to a newly-starting process (note that some headers in the program output are necessary), but it is rather slow due to overhead from running this process. FCGI keeps the process running after first request but requires a specially crafted program that would handle the requests.

The files with following suffixes are handled as CGI or FCGI programs. You will probably use a readily-available script that initialises your favourite framework: this is the case with the following Django example. However, you may run your own programs as well. If they are interpretable, define the interpreter on the first line of the entry-point script. For instance, with Python you would use:
`#!/usr/bin/python`

The following filename suffixes are identified by default.

- CGI: `.cgi` files
- FCGI `.fcgi` files; idle timeout is set to 10 minutes but you can just kill your processes when you want to reload them.

Whenever you use CGI or FCGI, remember to set execute bit on for your program file!

```
# Setting execute bit for a file
chmod a+x filename

# Killing a process
# 1) find the id of the process you want to kill
# 2) send SIGTERM the process;
```

```
#    if the process is really stuck,
#    you may want to try out flag "-9" (SIGKILL)
ps a
kill process_id
```

Apache Rewrite as Proxy

You may use mod_rewrite as a proxy to run your own server processes. With the following code clip, you can forward all the HTTP requests that have a certain path to your own HTTP server that is running on the same machine. The remaining tail of the address will be requested from your local server.

`.htaccess`:

```
RewriteEngine on
# replace PORTNUMBER with a port of your choice
RewriteRule ^(.*) http://localhost:PORTNUMBER/$1 [P]
```

Perl

FastCGI library is installed for Perl, but we will not debug your Perl code.

PHP

PHP works out of the box with help of mod_suphp. Just put your code to .php files into `public_html`. Of course, you may also run PHP application as a (F)CGI script.

Python / Django

Python and Django are ready for use. In addition, you can use any Python software with appropriate (F)CGI scripts. For Django, we also describe here a way to use Django's internal development server. Here are the minimalistic tutorials for usage, but for more information please refer to Django documentation.

Creating Django Project

```
# This should not be done in your public_html!
# Create a project. This will create a directory "project_name"
django-admin.py startproject project_name
```

Method 1: Serving Django Project with Development Server

1. Make a new directory to public_html. You can decide the name.
2. Follow the instructions in the section about rewriting for proxy to create a *.htaccess* to the new directory.
3. Go to Django project directory.
4. Type the following command:

   ```
   python manage.py runserver PORTNUMBER
   ```

5. Navigate with your browser to the URL corresponding the directory you created.

Method 2: Serving Django Project with FastCGI

1. Make a new directory to `public_html`. You can decide the name.
2. Create a new file `index.fcgi` with the following contents.
3. Navigate with your browser to the URL corresponding the directory you created.

`index.fcgi`:

```
#!/usr/bin/python
import sys, os
sys.path.insert(0, "absolute path to Django project")

# Set the DJANGO_SETTINGS_MODULE environment variable.
os.environ['DJANGO_SETTINGS_MODULE'] = "project_name.settings"

from django.core.servers.fastcgi import runfastcgi
runfastcgi(method="threaded", daemonize="false")
```

Ruby / RoR

This section will be completed later, if needed.

Databases

MySQL and PostgreSQL databases are available on request. As database performance is at very low priority, we however recommend using SQLite for some extra convenience.

Other Services

Cron

If you need some scheduled scripts, you might want to use crontab. With “crontab -e” you can edit your own crontab.