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.

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: the first letter of your first name, followed by at most six letters of your surname (possible non-ASCII characters converted to nearest ASCII equivalents, e.g. Matti Meikäläinen -> mmeikal)

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

URLs

You can use two different forms of an address. Both of them point to the same resource.

- `http://username.naf.cs.hut.fi/`
- `http://naf.cs.hut.fi/~username/`

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

Home Directories

Your home directory has permissions that prevent anyone else but you from seeing the files in it.

Working in Pairs

An additional shared user has been created for pairs. This user cannot be used for login and has no password. The only way you can use it is `sudo`. The username is "group" followed by your group number. In the following examples, the placeholder for this username is "groupuser". Note that the default umask has not been changed from 022, meaning that the files that you create will get only read group permissions. If you want to have it otherwise, please change your umask to 002 (read-and-write for group). You can even put a line "umask 002" to ".profile" in your home directory.

- Accessing files: just change directory
  `cd ~groupuser`
- A convenient way to ease accessing files, namely creating a symbolic link (writes a file "shared") Execute the following in your home directory:
  `ln -s ~groupuser shared`
  `cd shared`
- Gaining a shell (your own environment is used, e.g. command `cd` gets you to your own home directory)
  `sudo -u groupuser -s`
- Gaining a login shell (your environment variables are discarded and groupuser's environment is set up; almost like having an SSH connection as groupuser)
  `sudo -u groupuser -i`

Software Versions

- PHP 5.3
- Python 2.6
- Ruby 1.9.1
- Django 1.2
- Rails 3.0.5

Ways to Serve Your Website

One Apache process group serves everyone. All your applications are run under the user process with the same privileges as you would normally have. Apache logs will be available later.

Web Content

For web content, `public_html` in the home directory is available to Apache. For directories, Apache will search 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-started 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`

- 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!

```
# Setting execute bit for 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

If you want to run your own server processes, you can use mod_rewrite as proxy. With the following code clip, you can forward all the requests that have a certain path to a local server. The tail of the address will be sent to 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 local or (F)CGI scripts written in PHP.

Python / Django

Python and Django are ready for use. 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 consult 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 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")
```

Method 2: Serving Django Project with Development Server

1. Make a new directory to public_html. You can decide the name.
2. Follow the instructions in rewrite section 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.

Ruby / RoR

This section will be completed later, if needed.

Databases

MySQL is offered by default. PostgreSQL is available on request.

MySQL

User accounts have been created both for student users and group users.

- username: your username (or group username)
- password: look at db.pwd in your home directory
- privileges: you may create new databases starting with "username_"; pairs also have privileges to the databases of group user