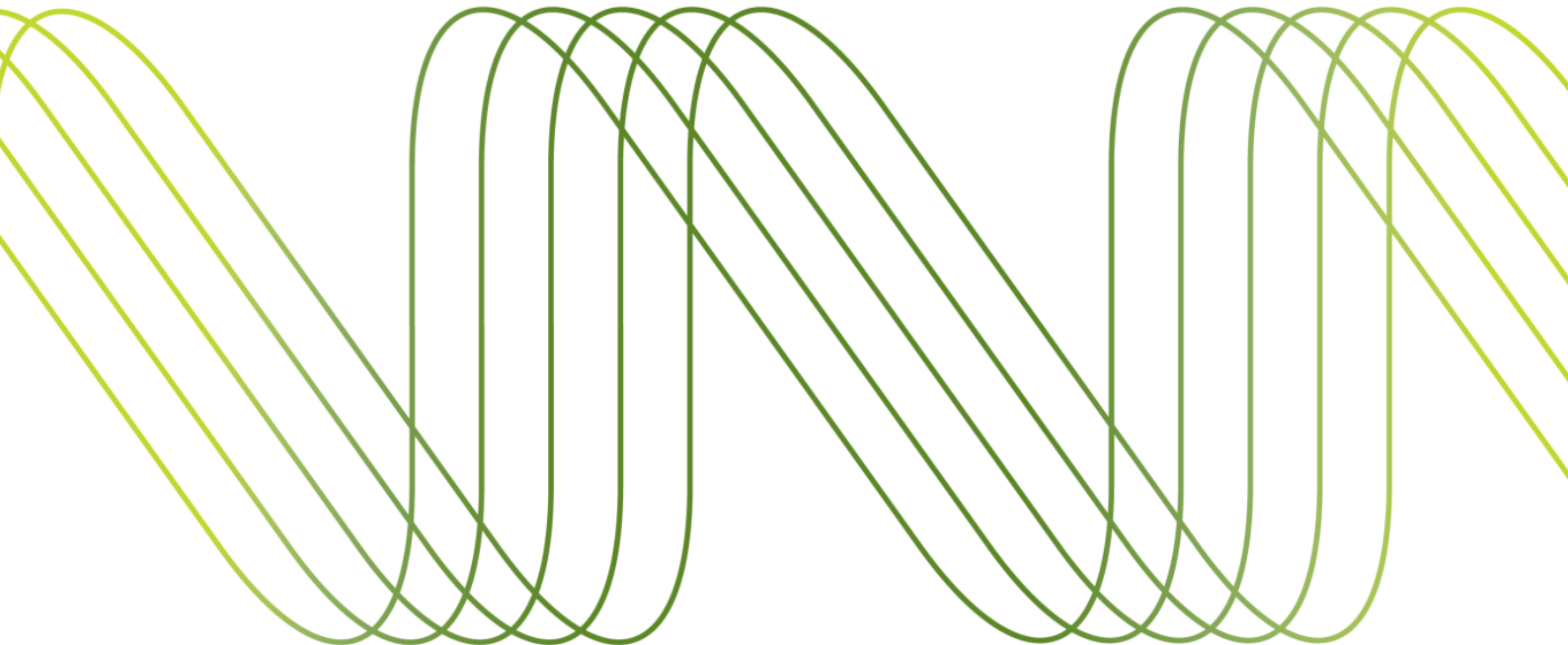

Apache Quick Build

CSC3504 Assignment 1



Getting the Source Code

Important

These materials assume you are using the SSH application found on the Newcastle University common desktop, however it is only the means by which you access your virtual machine (VM). **They also assume you have read the exercises contained in the *Linux Orientation*.**

Once you are in and at the command prompt, the materials become somewhat generic, so if you prefer to use something other than SSH (if you do, I'd recommend PuTTY) then feel free.

Method 1: Direct download

This is the method described in the Apache documentation and can be done entirely from the command line. It downloads the files directly to your VM, so is the most direct – but might not feel like it when you first try it as it uses the text-only browser, Lynx. This is also the only method that will work reliably from off-campus.

First, login into your VM and from the login prompt, enter your **build** directory.

```
cd build
```

By running Lynx from this location, **build** will be the default location for downloading to (which makes your life much easier!). So fire up Lynx and tell it to access the Apache download site:

```
lynx http://httpd.apache.org/download.cgi
```

You should find yourself looking at a text-only rendering of the webpage.



Figure 1. Lynx view of <http://httpd.apache.org/download.cgi>

Press the **<spacebar>** to scroll down the first part of the page, then switch to the **<down>** arrow key to scroll down until you see the section starting:

Apache HTTP Server (httpd) 2.2.NN is the best available version.

NN will be the current version e.g. **2.2.21** ...(or **2.2.16** as it was when the screen shots were made!)

Stop the cursor on the link to: `httpd-2.2.NN.tar.gz`

```

Download - The Apache HTTP Server Project (p3 of 6)
For details see the official Announcement and the CHANGES_2.2 or condensed CHANGES_2.2.16 lists
add-in modules for Apache 1.3 or 2.0 are not compatible with Apache 2.2, if you are running
third party add-in modules, you must obtain modules compiled or updated for Apache 2.2 from
that third party, before you attempt to upgrade from these previous versions, modules compiled
for Apache 2.2 should continue to work for all 2.2.x releases.
* unix source: httpd-2.2.16.tar.gz2 [PGP] [MD5] [SHA1]
* win32 source: httpd-2.2.16-win32-src-1.6 [PGP] [MD5] [SHA1]
* win32 binary without crypto (no mod_ssl) (MSI installer): httpd-2.2.16-win32-x86-no_ssl.msi
[PGP] [MD5] [SHA1]
* win32 binary including openssl 0.9.8o (MSI installer): httpd-2.2.16-win32-x86-openssl-0.9.8o.msi [PGP] [MD5] [SHA1]
* other files

```

Figure 2. Download link selected

Now hit <return> to follow the download link. The status at the bottom of the window should update to offer you the choice to download the file.

```

For details see the Official Announcement and the CHANGES_2.0 and CHANGES_2.0.63 lists.
application/x-gzip D)ownload, or C)ancel
Arrow keys: Up and down to move. Right to follow a link; Left to go back.
h)elp O)ptions P)rint G)o M)ain screen Q)uit /-search [delete]=history list

```

Figure 3. Download or cancel

Type d to confirm the download and it should begin.

```

For details see the Official Announcement and the CHANGES_2.0 and CHANGES_2.0.63 lists.
Read 5.61 of 6219 Kib of data, 9083 Kib/sec
Arrow keys: Up and down to move. Right to follow a link; Left to go back.
h)elp O)ptions P)rint G)o M)ain screen Q)uit /-search [delete]=history list

```

Figure 4. Download in progress

Eventually you will be asked what to do with the file.

The **Save to disk** option should already be selected, so hit <return> and once more to confirm the filename to save as (`httpd-2.2.NN.tar.gz`).

```

Download options (Lynx version 2.6.5rel1.1), help
downloaded link: http://mirror.lividpenguin.com/pub/apache/httpd/httpd-2.2.16.tar.gz
Suggested file name: httpd-2.2.16.tar.gz
standard download options:
Save to disk
Local additions:
view with less

Enter a filename: httpd-2.2.16.tar.gz
Arrow keys: up and down to move. Right to follow a link; Left to go back.
h)elp O)ptions P)rint G)o M)ain screen Q)uit /-search [delete]=history list

```

Figure 5. Save to disk and confirm filename

When you've finished hit q to quit Lynx.

Finally, view a listing of your **build** directory to check the source file is there!

```

[ncy2@v-eliot-01 build]$ ls -l
total 6232
-rw-rw-r-- 1 ncy2 ncy2 6369022 oct 10 16:25 httpd-2.2.16.tar.gz
[ncy2@v-eliot-01 build]$

```

Figure 6. Confirming a successful download with `ls -l`

Method 2: Download and copy to VM

This method requires you to download the source file to your local PC and then copy it over to your VM using `sftp`.

You will (probably) only be able to use this method, as described here, *on campus* and *using the SSH client* installed as part of the common desktop.

First fire up a web browser (IE, Firefox, Chrome etc.) on your local machine and go to:

<http://httpd.apache.org/download.cgi>

Scroll down to the section starting:

Apache HTTP Server (httpd) 2.2.NN is the best available version

Find the link to:

`httpd-2.2.NN.tar.gz`

(Where NN is the current version)

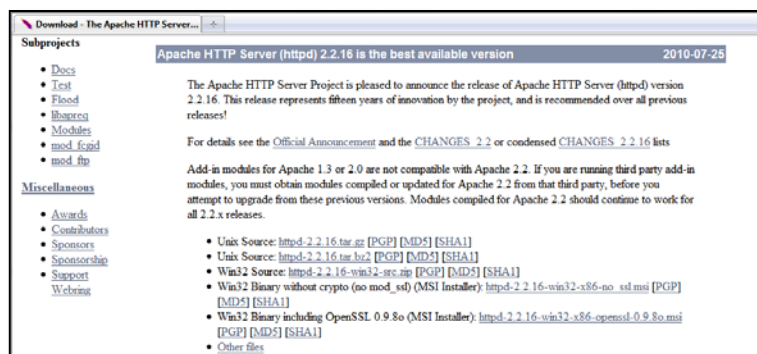


Figure 7. Figure 1 Apache current version download

Click on the link and follow your browser's instructions to save it to your local file space.

Now switch to the SSH client and login to your VM. Once you have logged in look for the New File transfer Window button on the toolbar.



This will open a new window enabling you to transfer files from your desktop PC to your home directory on your VM.

In the left-hand side panel, navigate to the location of your downloaded file and in the right-hand panel, navigate to your **build** directory.

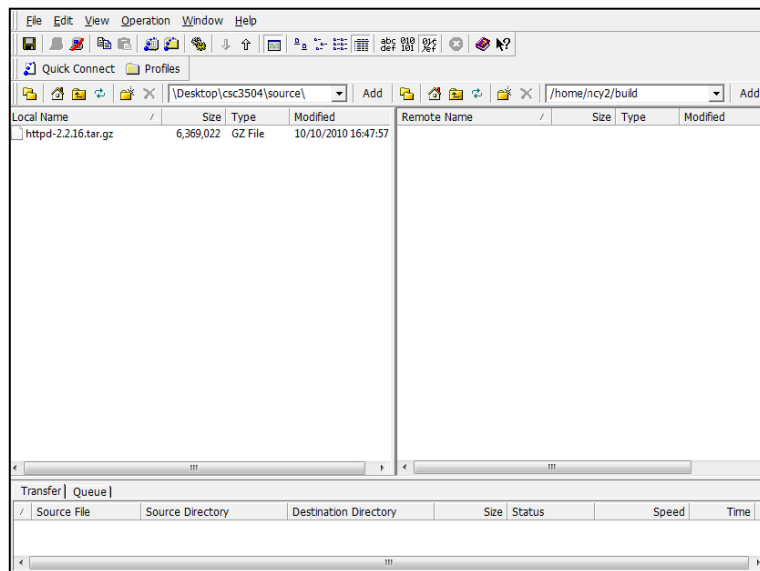


Figure 8. Local and remote sftp panels

Drag your downloaded source file from left to right to transfer it to your VM.

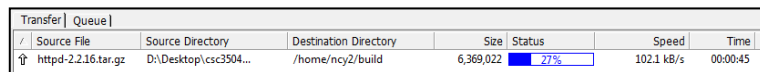


Figure 9. Transfer in progress

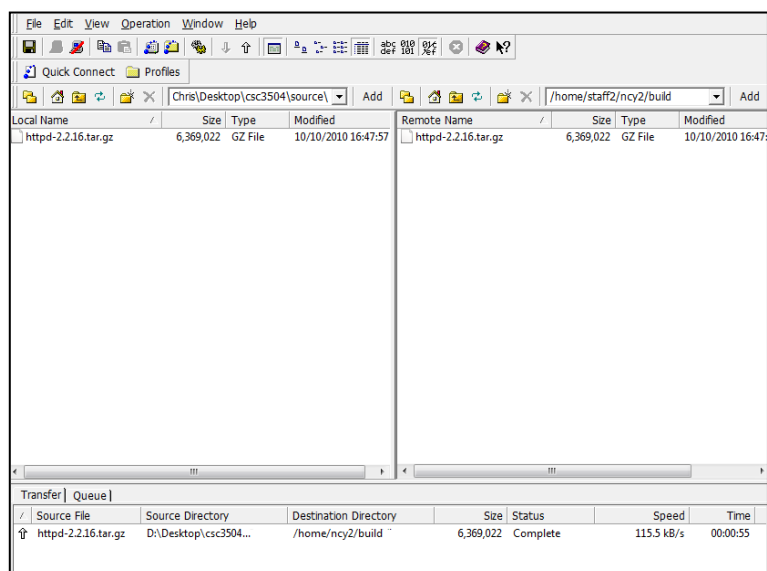


Figure 10. Transfer complete

When you have finished, close the file transfer window.

Unpacking

The source code you have downloaded is a compressed archive. This is a packed up file structure, which has been further compressed to make it smaller and easier to move around. Getting hold of the files it contains is a simple two-step process.

Decompress

The first step is to decompress the archive. This is similar to the way zip files are extracted and is handled by the Linux application **gzip**.

Switch to your **build** directory and type in the command: **gzip -d httpd-2.2.NN.tar.gz**

(Where NN is the version you downloaded)

This runs **gzip** over the file with the **-d** (decompress) option set.

You should be returned to the command prompt and a quick **ls -l** should confirm that the file has now lost its **.gz** extension.

```
[ncy2@v-eliot-01 build]$ gzip -d httpd-2.2.16.tar.gz
[ncy2@v-eliot-01 build]$ ls -l
total 32032
-rw-rw-r-- 1 ncy2 ncy2 32761344 Oct 10 16:25 httpd-2.2.16.tar
[ncy2@v-eliot-01 build]$
```

Figure 11. Decompressed archive

Unpack

The next task is to unpack the archive, which has been originally made using the Linux **tar** application.

Now type: **tar -xvf httpd-2.2.NN.tar**

This runs **tar** over the newly decompressed archive with the options set to:

- x (extract files and directories)
- v (verbosely list the files as they are extracted)
- f **httpd-2.2.NN.tar** (the file archive to open).

You should see files scroll by as they unpack until you return to the command prompt.

```
httpd-2.2.16/build/NWGNmakefile
httpd-2.2.16/build/NWGNscripts.inc
httpd-2.2.16/build/NWGNutail.inc
httpd-2.2.16/build/pkg/
httpd-2.2.16/build/PrintPath
httpd-2.2.16/build/program.mk
httpd-2.2.16/build/rpm/
httpd-2.2.16/build/rules.mk.in
httpd-2.2.16/build/special.mk
httpd-2.2.16/build/sysv_makefile
httpd-2.2.16/build/win32/
httpd-2.2.16/build/win32/apache.ico
httpd-2.2.16/build/win32/httpd.rc
httpd-2.2.16/build/win32/win32ver.awk
httpd-2.2.16/build/rpm/htcacheclean.init
httpd-2.2.16/build/rpm/httpd.init
httpd-2.2.16/build/rpm/httpd.logrotate
httpd-2.2.16/build/rpm/httpd.spec.in
httpd-2.2.16/build/pkg/buildpkg.sh
httpd-2.2.16/build/pkg/pkginfo.in
httpd-2.2.16/build/pkg/README
[ncy2@v-eliot-01 build]$
```

Figure 12. Unpacked archive

Finally run `ls -l` over your `build` directory again to see you now have a new sub-directory called:

`httpd-2.2.NN`

```
[ncy2@v-eliot-01 build]$ ls -l
total 32036
drwxr-xr-x 11 ncy2 ncy2      4096 Jul 22 10:07 httpd-2.2.16
-rw-rw-r-- 1 ncy2 ncy2 32761344 Oct 10 16:25 httpd-2.2.16.tar
[ncy2@v-eliot-01 build]$
```

Figure 13. Figure 2 Confirmation of unpacked archive

This is your source code...

Installation

Now you have downloaded and extracted your source code you can build it into a working web server. There are three steps to this process and the instructions below should leave you with a working, but simple and unsecured server.

Configure the source

Change to the directory containing your source code:

```
cd /home/yourusername/build/httpd-2.2.NN
```

Run `ls -l` over this directory to view its contents.

If you look down this listing you should find a file called **configure**. This is the script you will need to run in order to prepare *your* source code to be built on *your* VM.

```
-rw-r--r-- 1 ncy2 ncy2 10943 Nov 21 2004 config.layout
-rwxr-xr-x 1 ncy2 ncy2 567939 Jul 22 10:07 configure
-rw-r--r-- 1 ncy2 ncy2 23953 Dec 1 2008 configure.in
drwxr-xr-x 9 ncy2 ncy2 4096 Oct 10 20:03 docs
-rw-r--r-- 1 ncy2 ncy2 403 Nov 21 2004 emacs-style
```

Figure 14. Apache configure script

Even without any extra options, **configure** will build you a basic, "default" Apache web server, so at the command prompt type:

```
./configure
```

The `./` at the start is very important as it is an instruction *to "run the **configure** script found in the current directory"* – as opposed to any others the system may know about.

Straight away you will see the script running, with output scrolling up the screen.

```
checking target system type... x86_64-unknown-linux-gnu
Configuring Apache Portable Runtime library ...
checking for APR... reconfig
checking for apr in src/lib/apr now
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
Configuring APR library
Platform: x86_64-unknown-linux-gnu
checking for working mkdir -p... yes
APR version: 1.4.2
checking for chosen layout... apr
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
Applying APR hints file rules for x86_64-unknown-linux-gnu
setting CPPFLAGS to "-DLINUX=2"
adding "-D_REENTRANT" to CPPFLAGS
adding "-D_GNU_SOURCE" to CPPFLAGS
(Default will be unix)
checking whether make sets $(MAKE)... yes
checking how to run the C preprocessor... gcc -E
checking for gawk... gawk
checking whether ln -s works... yes
checking for ranlib... ranlib
checking for a BSD-compatible install... /usr/bin/install -c
checking for rm... rm
checking for as... as
checking for cpp... cpp
checking for ar... ar
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... 
```

Figure 15. Output to screen from configure

This is perfectly normal and the good news is you can ignore it – unless something goes wrong!

Should an error occur (usually because the system checks find that an important component is missing), then the script will stop and you should see the error recorded towards the end of the output.

When **configure** has finished its work, you will be returned to the command prompt. You can scroll back up the output if you are interested in what it has been up to.

Notice how most of the output is from component checks and the creation of some of the installation files and documentation.

Make the installation files

Now your source has been *configured*, using the script that came with it, the next step is to let your VM *make* that source code into files that will run on this Linux system by *compiling* them.

The command **make** will fire up the compiler software and run it over the source code found in the location that **make** was called from (which is why you are inside your source directory already!).

So at the command prompt, simply type:

make

Once more the screen should fill with output, which again you can largely ignore unless an error occurs.



```
waking all in src/lib
make[1]: Entering directory '/home/ncv2/build/httpd-2.2.16/src/lib'
make[1]: Entering directory '/home/ncv2/build/httpd-2.2.16/src/lib/apr'
make[2]: Entering directory '/home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool'
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o passwd/apr_passwd.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_cpystrn.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_fmatch.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_sprintf.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_strings.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_strnatcmp.o gcc -g -O2 -pthread
/bin/sh /home/ncv2/build/httpd-2.2.16/src/lib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread
-DHAVE_CONFIG_H -DLINUX -D_REENTRANT -D_GNU_SOURCE -I./include -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include/arch/unix -I/home/ncv2/build/httpd-2.2.16/src/lib/apr/include -o strings/apr_strtok.o gcc -g -O2 -pthread
touch strings/apr_passwd.o strings/apr_cpystrn.o strings/apr_fmatch.o strings/apr_sprintf.o strings/apr_strings.o strings/apr_strnatcmp.o strings/apr_strtok.o
```

Figure 16. Screen output from make

When **make** has finished you will be dropped back to the command prompt again.

How long will make take?

Depending on the speed of your system and the load on it, this is usually a good point at which to put the kettle on and make a cup of tea ☺

Install the new Apache application

Now you have *compiled* your web server, the final stage is to actually *install* it.

You'll be running `make` again, but telling it to `install` the compiled files to their final destination.

The complication at this stage is that the default location for an Apache installation is:

`/usr/local/apache2`

That is in a location that only `root` has permission to write files too. This means you will need to use `sudo` to complete the job.

So type:

```
sudo make install
```

You'll be asked for *your* password. Then once more the screen will fill with output confirming that all the files are now being moved to the correct locations.

```
-----
/usr/bin/install -c -m 644 apr.exp /usr/local/apache2/lib/apr.exp
/usr/bin/install -c -m 644 apr.pc /usr/local/apache2/lib/pkgconfig/apr-1.pc
for f in libtool shlibtool; do
  if test -f $f; then /usr/bin/install -c -m 755 $f /usr/local/apache2/build; fi; \
done
/usr/bin/install -c -m 755 /home/ncy2/build/httpd-2.2.16/src/lib/apr/build/mkdir.sh /usr/local/apache2/build
for f in make_exports.awk make_var_exports.awk; do \
  /usr/bin/install -c -m 644 /home/ncy2/build/httpd-2.2.16/src/lib/apr/build/$f /usr/local/apache2/build; \
done
/usr/bin/install -c -m 644 build/apr_rules.out /usr/local/apache2/build/apr_rules.mk
/usr/bin/install -c -m 755 apr-config.out /usr/local/apache2/bin/apr-1-config
make[2]: Leaving directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr'
make[2]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util'
making install in apr-util
make[3]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat'
make[4]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat/lib'
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat/lib'
make[3]: Leaving directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat'
make[3]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util'
make[3]: Nothing to be done for 'local-all'.
make[2]: Leaving directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util'
/home/ncy2/build/httpd-2.2.16/src/lib/apr/build/mkdir.sh /usr/local/apache2/include /usr/local/apache2/lib/pkgconfig \
  /usr/local/apache2/lib /usr/local/apache2/bin
for f in /home/ncy2/build/httpd-2.2.16/src/lib/apr-util/include/*.h /home/ncy2/build/httpd-2.2.16/src/lib/apr-util/include/*.ht; do \
  /usr/bin/install -c -m 644 $f /usr/local/apache2/include; \
done
/usr/bin/install -c -m 644 apr-util.pc /usr/local/apache2/lib/pkgconfig/apr-util-1.pc
list='xml/expat'; for i in $list; do \
  { cd $i; make DESTDIR= install; }; \
done
make[3]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat'
make[4]: Entering directory '/home/ncy2/build/httpd-2.2.16/src/lib/apr-util/xml/expat/lib'
```

Figure 17. Screen output from `make install`

When you are finally returned to the command prompt again, the installation will be complete.

A quick `ls -l` on `/usr/local` will show you the new `apache2` directory.

```
[ncy2@v-eliot-01 httpd-2.2.16]$ ls -l /usr/local/
total 84
drwxr-xr-x 15 root root 4096 Oct 10 17:55 apache2
drwxr-xr-x  2 root root 4096 Jan 26 2010 bin
drwxr-xr-x  2 root root 4096 Jan 26 2010 etc
drwxr-xr-x  2 root root 4096 Jan 26 2010 games
drwxr-xr-x  2 root root 4096 Jan 26 2010 include
drwxr-xr-x  2 root root 4096 Jan 26 2010 lib
drwxr-xr-x  2 root root 4096 Jan 26 2010 lib64
drwxr-xr-x  2 root root 4096 Jan 26 2010 libexec
drwxr-xr-x  2 root root 4096 Jan 26 2010 sbin
drwxr-xr-x  4 root root 4096 Sep 29 17:18 share
drwxr-xr-x  2 root root 4096 Jan 26 2010 src
[ncy2@v-eliot-01 httpd-2.2.16]$
```

Figure 18. Checking for `/usr/local/apache2`

Running `ls -l` over `/usr/local/apache2` will show you your web server installation – a location you will soon become more familiar with!

```
[ncy2@v-eliot-01 httpd-2.2.16]$ ls -l /usr/local/apache2/
total 60
drwxr-xr-x  2 root root  4096 Oct 10 17:55 bin
drwxr-xr-x  2 root root  4096 Oct 10 17:55 build
drwxr-xr-x  2 root root  4096 Oct 10 17:55 cgi-bin
drwxr-xr-x  4 root root  4096 Oct 10 17:55 conf
drwxr-xr-x  3 root root  4096 Oct 10 17:55 error
drwxr-xr-x  2 root root  4096 Oct 10 17:08 htdocs
drwxr-xr-x  3 root root  4096 Oct 10 17:55 icons
drwxr-xr-x  2 root root  4096 Oct 10 17:55 include
drwxr-xr-x  3 root root  4096 Oct 10 17:55 lib
drwxr-xr-x  2 root root  4096 Oct 10 17:55 logs
drwxr-xr-x  4 root root  4096 Oct 10 17:55 man
drwxr-xr-x 14 root root 12288 Jul 22 10:07 manual
drwxr-xr-x  2 root root  4096 Oct 10 17:55 modules
[ncy2@v-eliot-01 httpd-2.2.16]$
```

Figure 19. Contents of the Apache installation

Testing

Now your web server is built, you'll be itching to test it out. You will need to start the server up first though; otherwise it won't be able to respond to any HTTP requests.

Access your web site

Try visiting your web site by opening any web browser and going to:

`http://vm-eliot-NNN.ncl.ac.uk/`

(where NNN is the number of your VM)

You should find you get an error as the site cannot be found.

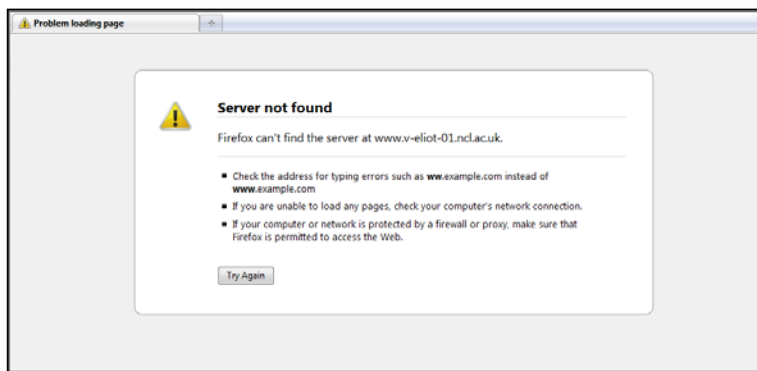


Figure 20. Unable to access web site over HTTP

Start your server

You will need to run the server control script `apachectl` and start Apache on your VM.

The script is in `/usr/local/apache2/bin`, along with the other executables that make up your web server.

Return to your VM and try:

```
/usr/local/apache2/bin/apachectl start
```

You should get a permissions error as you do not have the authority to start (or stop) the web server. However the `root` user does have appropriate permissions so instead try:

```
sudo /usr/local/apache2/bin/apachectl start
```

Provide your password and Apache will start. You may or may not get a warning about domain names – don't worry about that at this stage.

```
[ncy2@v-eliot-01 ~]$ sudo /usr/local/apache2/bin/apachectl start
[sudo] password for ncy2:
httpd: Could not reliably determine the server's fully qualified domain name, using 10.8.151.185 for
ServerName
[ncy2@v-eliot-01 ~]$
```

Figure 21. Apache start-up (with domain name error)

Try the web site again...

Now return to your browser and try accessing your site again.

You should get a (rather understated) response, but at least you'll know it is working!

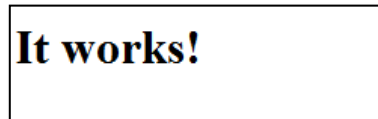


Figure 22. Apache... It works!

Basic Configuration

Even though you only have a simple server running at the minute, you can (and should) make some simple configuration changes. If nothing else this will be good practice for the more complex setup yet to come.

Important

These materials assume you have read and understood the section on editing files with VI in the *Linux Orientation* exercises.

Backup the current configuration

Return to your VM console and switch to the apache configuration directory.

```
cd /usr/local/apache2/conf
```

List the files inside and you will see all the default files provided with the installation.

```
[ncy2@v-eliot-01 conf]$ ls -l
total 88
drwxr-xr-x 2 root root 4096 Oct 10 17:55 extra
-rw-r--r-- 1 root root 13346 Oct 10 17:55 httpd.conf
-rw-r--r-- 1 root root 12958 Oct 10 17:55 magic
-rw-r--r-- 1 root root 45472 Oct 10 17:55 mime.types
drwxr-xr-x 3 root root 4096 Oct 10 17:55 original
[ncy2@v-eliot-01 conf]$
```

Figure 23. Listing of the Apache configuration directory

It is usually a good idea to back these up before making any major changes, so that if something goes very wrong, you can quickly return to the "last known good" configuration by restoring the backup files.

Create a backup location in your home directory.

```
mkdir -p /home/yourusername/backups/apache2
```

Now copy the `conf` directory and all its contents to the new location.

```
cp -R ./ /home/yourusername/backups/apache2/conf
```

The `./` used here means "*copy all the files from the current location*" (remember you are already in the directory you want to back up).

You can check the backup worked by running:

```
ls -l /home/yourusername/backups/apache2/conf
```

Editing the config file

Now you have a backup, you can edit the real configuration file using VI (remember you need to be **root** in order to make changes here).

```
sudo vi httpd.conf
```

Enter your password and take a look through the file using the arrow keys to navigate around.

Notice how the file is heavily commented to let you know what each section is for (commented lines begin with a hash #).

Remember that because it is always safer to remove configuration changes by *commenting them out* rather than deleting them – you can't always be **absolutely sure** you won't need them again!

```
<IfModule !mpm_netware_module>
<IfModule !mpm_winnt_module>
#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# It is usually good practice to create a dedicated user and group for
# running httpd, as with most system services.
#
User daemon
Group daemon
</IfModule>
</IfModule>

# 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition. These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#
```

Figure 24. The default httpd.conf

Scroll through the file until you find the following section:

```
#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents. e.g. admin@your-domain.com
#
ServerAdmin you@example.com
#
# ServerName gives the name and port that the server uses to identify itself
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here
#
#ServerName www.example.com:80
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/apache2/htdocs"
#
```

Figure 25. ServerAdmin and ServerName directives

Enter *insert* mode and make the following changes:

1. Edit the `ServerAdmin` directive to:

```
ServerAdmin your.email@nc1.ac.uk (i.e. your real email address)
```

2. Now *add* a `ServerName` directive:

```
ServerName vm-eliot-NNN.nc1.ac.uk (where NNN is your VM)
```

3. Scroll down until you find the following line:

```
CustomLog "logs/access_log" common
```

Comment out this line and replace it with the following:

```
# CustomLog "logs/access_log" common
CustomLog "logs/access_log" combined
```

The significance of this last change will become apparent in future practicals.... 😊

Finally, return to *command* mode, save and quit.

Restart to reload

In order to load the new configuration you will need to *stop and start* your server.

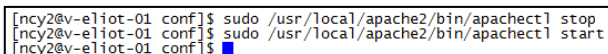
Stop the server first:

```
sudo /usr/local/apache2/bin/apachectl stop
```

Then start it up again:

```
sudo /usr/local/apache2/bin/apachectl start
```

This time you should notice there was no domain name warning (as you have explicitly set the `ServerName` in `httpd.conf`).



```
[ncy2@v-eliot-01 conf]$ sudo /usr/local/apache2/bin/apachectl stop
[ncy2@v-eliot-01 conf]$ sudo /usr/local/apache2/bin/apachectl start
[ncy2@v-eliot-01 conf]$
```

Figure 26. Stopping and starting Apache (no domain name error)

In practice you can simply *restart* your server between configuration changes using:

```
sudo /usr/local/apache2/bin/apachectl restart
```

Change the Document Root

One of the first changes you are likely to make will be to instruct the server to deliver its web content from a directory that sits *outside* the installation directory.

By default Apache is delivering pages from `/usr/local/apache2/htdocs`. In practice you will usually move this elsewhere because separating the content from the application is both more secure and also easier to manage backups etc.

Create a new web directory

Create a directory in `/usr/local` to be the new document root.

```
sudo mkdir /usr/local/www
```

Make your new home page

Use VI to create a new home page in `/usr/local/www` called `index.html`

```
sudo vi /usr/local/www/index.html
```

This will be a blank file, so enter *insert* mode (*i*) and add the following text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1"
/>
    <title>vm-eliot-NNN</title>
</head>
<body>
<h1>Hi from vm-eliot-NNN</h1>
</body>
</html>
```

Change the **NNNs** for your own VM number.

HINT You can copy and paste into VI when in *insert* mode!

Save `index.html` and quit.

Update the config

Your server is still delivering its pages from `/usr/local/apache2/htdocs` so you now need to update the configuration to point to the new location instead.

Open `httpd.conf` again:

```
sudo vi /usr/local/apache2/conf/httpd.conf
```

Scroll to the `DocumentRoot` directive (**not** the `ServerRoot` one!).

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/apache2/htdocs"
```

Figure 27. The `DocumentRoot` directive

Enter *insert* mode and edit the directive as shown below:

```
DocumentRoot "/usr/local/www"
```

Make sure you do **not** add a slash on the end.

Now scroll down the page and find the **second** place you need to update the `DocumentRoot` (HINT: read the comments as you go).

Save `httpd.conf` and quit VI again.

Test ...then restart and test again

Open a web browser and go to:

```
http://vm-eliot-NNN.nc1.ac.uk/ (NNN is the number of your VM)
```

You should still be seeing the original `It works!` page.

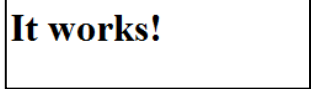


Figure 28. The original home page

Return to your VM and restart Apache (to load the new configuration) with:

```
sudo /usr/local/apache2/bin/apachectl restart
```

Now refresh your web browser and you should be seeing the new home page...

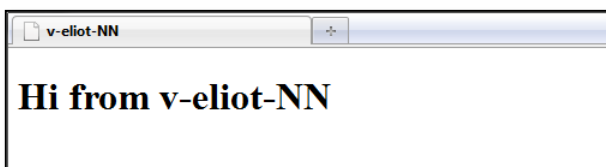


Figure 29. Updated configuration and home page

Adding Modules

A basic Apache set up will simply deliver web pages at a given URL. To add extra functionality you need to add and enable additional modules.

Some modules are included with the default installation. Others have to be specifically enabled and compiled into the server at build time (*static*), or assembled separately and loaded in to the server at start-up (*dynamic* or *shared*).

Understanding how to recompile Apache to add static modules is important, so you are going to add `mod_info` to your set up and get it to do some work.

Reconfigure your source code

The first step is to re-run the `configure` script you used to prepare your source code for compilation – and add the option to enable to module(s) you require.

You'll need to be in your `build` directory to do this.

```
cd ~/build (~ is a shortcut for "the current user's home directory")
```

The command options are added to the end of the call for `./configure` e.g.

```
./configure --enable-something --disable-something etc.
```

To find out what to add to enable the feature you require you can either...

Look it up at:

```
http://httpd.apache.org/docs/2.0/programs/configure.html
```

...or simply type `./configure --help` and scroll down until you find the entry for your module or feature.

What am I looking for?

You are looking for the option to enable the module called `mod_info`. *

Is there a nice-er way to do this?

A cool feature of `configure` is that each time it runs successfully it creates a simple script in the source directory called `config.nice`.

This script can save you a lot of time when recompiling as it contains the whole of the *last successful configure command line*.

If you want to add a new module, you can simply run:

```
"Run configure with all the previous options..." → ./config.nice --enable-somethingelse ← "... and also do this"
```

Instead of:

```
./configure --enable-something --enable something  
--enable-something --enable-somethingelse
```

The `config.nice` script is just a text file, so when you have one, you can look at it using:

```
more config.nice
```

You can also edit in VI (when you are a bit more confident!)

Rebuild the binaries

Once you have reconfigured the source run `make` again to build the new server binaries.

This will not affect your running server - all you are doing is building the files for a new version, you haven't actually installed them yet!

Re-install the server

Assuming `make` ran OK then you are ready to re-install Apache.

Stop the current server if it is still running, then run:

```
sudo make install
```

This copies all the new binaries to `/usr/local/apache2`

Importantly this **does not** overwrite your existing configuration files. It only affects the server software itself.

Change the server configuration

Now you have the new module compiled and installed, you will need to adjust `httpd.conf` to make use of it.

Your task is to work out how to deliver the detailed server information provided by `mod_info` and make it accessible from:

```
http://vm-eliot-NNN.ncl.ac.uk/my-server-information
```

Submitting Your Work

Although you may have been submitting files as you go along, the *only* URLs and files that will be marked will be those included in the *last submission you make before the deadline*.

This means that you will need to submit the unique files and URLs for all the components together.

Below you will find a checklist for the whole assessment and the filenames and URLs you will need to include.

Please pay close attention as *some of your files may need to be renamed in order to submit successfully*.

Also not you only need to submit one copy of each file for the final submission. For example the final version of your `httpd.conf` should contain all the changes required for all of the sub-components.

URLs to be auto-marked

These URLs should be submitted to the **URL** submission area for this assignment:

`http://vm-eliot-NNN.ncl.ac.uk`

`http://vm-eliot-NNN.ncl.ac.uk/my-server-information`

A script will access these URLs, check they are delivering the required content, functionality or error message as per the assignment details and store a copy of the output.

You should ensure that they all deliver the correct content when accessed... and that your server is on and running!

Files to submit

Submit the files required for marking in *a single zip archive* to the **FILES** submission area for this assignment.

Name your zip file:

`a123456789-vm-eliot-NNN.zip`

Where `a123456789` is your student number and `NNN` is your VM number

File Checklist

Your file...	Renamed as...	x/✓
<code>httpd.conf</code>	<code>vm-eliot-NNN-httpd.conf.txt</code>	

Off Campus Access (Web Site)

As your server is not visible outside the campus network, you will need to use **RAS** in order to view the web front end at: <http://vm-eliot-NNN.ncl.ac.uk>

Accessing RAS In any web browser (IE might work best), visit:

<http://ras.ncl.ac.uk>

You may be prompted to download the latest version of the client plugin – do so. Once you have the plugin you will need to login using your normal campus ID



Figure 30. RAS login

Then navigate to the **Communication** and **Browsers** folder

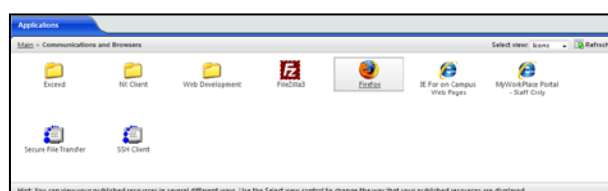


Figure 31. Browsers available via RAS

Select Firefox or Internet Explorer, which will then "deploy" to your desktop.

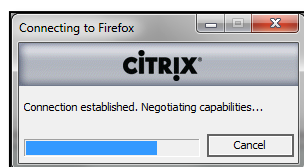


Figure 32. RAS application deploying

You will end up with a browser which should behave as if it was running on your local machine, but is being managed by a server on campus.

Use this browser to access your website.

What Now?

Now you know how to:

- Download and install Apache from source
- Configure Apache for compilation
- Compile Apache using `make`
- Install Apache using `make install`
- Start, stop and restart Apache
- Access your web site using a web browser
- Locate, backup and edit the main Apache configuration file, `httpd.conf`
- Instruct Apache to deliver web pages from a new location
- Re-compile Apache with support for additional modules
- Configure Apache to use a new module

You will need to build on these basic skills in order to complete subsequent assessments for the module and create a server with more functionality and more complex configuration.

In order to add more modules to your installation you will need to run through these processes again.

Familiarise yourself with the core bits of the Apache documentation (it is very good) at:

<http://httpd.apache.org/docs/2.2/>

You will find the compiling and installing section and the `configure` reference useful:

<http://httpd.apache.org/docs/2.2/install.html>

<http://httpd.apache.org/docs/2.2/programs/configure.html>