Step-By-Step-Guide

for installing

Solaris 7

on a SPARC system
and configuring it as a full web server

Written by Bernd Nigmann for AWWM.COM
nigmann@future-vision.net

Revised Version 2.2

Created on December 20, 2001

Contents

This document helps you installing Solaris 7 on a SPARC platform and describes the necessary installation steps for setting up that SPARC as a full web server with the following features:

- Apache Web server with SSL support via mod_ssl and suEXEC functions
- PHP4 support with mySQL access and additional gd library for dynamic png image creation
- MySQL database and easy administration using phpMyAdmin
- Mail support
- Easy system administration with Webmin

Please make sure to follow the step-by-step-instructions in the given order! Otherwise it's not guaranteed that everything works like it should work!

1 Before you start

Make sure that all files on the hard drive of the SPARC are backed up before you start the installation process. After re-partitioning and formatting, none of the old files will be available anymore… ☺

1.1 About this manual

Note: Text with a '**$** ' at the beginning of the line indicate, that this is a command, typed in at a command prompt. Normally the command prompt uses '**#** ', but this may be confused with comments in a configuration file.

To not getting confused about input and output, I'll use the following format for those:

Input will have a double line at the beginning of the line. This is a command you type in.

```
$ cd /usr/local/bin
```

Output will have a single line on the left. This also indicates the content of a configuration file for example. The following example is the content of **/etc/shells**:

```
/bin/sh
/usr/local/bin/bash
```

1.2 About the included CD

I created a CD that contains all the necessary files to follow the installation steps described in this manual. There may be newer versions available of some of the files. Just check out the online sources.

The CD contains just one huge tar archive, a small one and two binary packages. In the large archive, '**install.tgz**', is the directory **/usr/local/archive**: All the installation files and sources that you may need during installation are in there. The second archive, '**manual.tgz**', contains this documentation in PDF format and will extract to **/root/_DOC**. The two binary packages are **gzip-1.3-sol7.pkg** and **tar-1.13-sol7.pkg**.

In chapter 3.5 you will extract the files from the CD. When I refer to any installation package in this manual, I always assume, that it was copied to **/archive** on the SPARC's hard drive.

2 Installation of Solaris 7

Turn on the SPARC, insert the Solaris 7 CD and wait until you can see the memory self test on the screen. If you have a SUN keyboard connected to the SPARC, hold the 'Stop" key on the right side of the keyboard and press "A" to get an **`ok`** prompt. With a normal PS/2 keyboard connected to the SPARC, hold the "Ctrl" key and press "Pause/Break" on the upper right side of the keyboard to get this **`ok`** prompt.

On the **`ok`** prompt type '**`boot cdrom`**' and hit enter. The SPARC reboots and loads OpenWin and the installation software. The installation process of Solaris 7 is quite user-friendly and hardly needs any description. Anyway, I'm going to guide you through the setup process.

First you have to select the language and local settings. Then you must identify the hostname. Please type in the full hostname here (e.g. sun1.awwm.com). Answer 'yes' when it asks if this is a networked system. Then type in the IP address.

When it asks you what name service you want to use, choose 'other' (not NIS+, not NIS/yp,…!). At the question if this system is part of a subnet, say 'yes' and enter the subnet mask and the time zone.

It will then say something like 'System Identification Complete'.

Then it asks for the installation type. Select 'Initial' here, because we want to re-install everything from scratch and wipe out all the old stuff on that machine.

We don't want 'Allocate Client Services', so just click on 'continue' here. After that you can select additional languages that will be supported by Solaris.

The next question is about the software group. Do you want to install the whole nine yards or just the basic stuff? I always selected the 'complete distribution with OEM support'.

Note: It is also fine to just install the 'developer system' package, but **never** go with only the 'End-user system', because then Solaris 7 won't install important programs for compiling software packages. For example the assembler 'as' will not be there and you won't be able to create executables.

In the next dialog box ('Select disks'), click on **`c0t0d0`** and click on 'continue'. We want to wipe this disk, so click on 'continue' when it asks about preserving data. When it comes to setting up the hard drive partitions you can either set them manually or have the installation wizard set up everything automatically. Although the installation wizard normally does a good job, **I wouldn't trust it here**. When I had it set up everything, it created six small system partitions and one huge user partition. The user partition was large enough of course, but some of the system areas ran out of disk space during installation of many software packages. Every UNIX administrator has his own preferred partition settings; some of them are reasonable, others aren't. It makes certainly sense to split up the system into several partitions, but making them too small can be worse than having most of the system files in just one partition.

My suggestion doesn't claim to be the ultimate one – not at all. If you've got a better suggestion, feel free to mail it to me (nigmann@future-vision.net). I make it quite simple here and we'll create just three partitions (on a 17GB hard drive):

- / /dev/dsk/c0t0d0s0 with about 3 GB
- swap /dev/dsk/c0t0d0s3 with about 1 GB (same size as system RAM)
- /home /dev/dsk/c0t0d0s1 using the rest (about 13 GB)

As you can see in the following printout (done after this installation), the selected size of the system partition was quite good (modified the look of the output a little bit):

```
$ df -k
Filesystem            kbytes      used       avail       capacity   Mounted on
/dev/dsk/c0t0d0s0     3,096,856   1,213,705  1,821,214   40%        /
/dev/dsk/c0t0d0s1     13,527,865  848,171    12,544,416  7%         /home
swap                  1,791,192   296        1,790,896   1%         /tmp
```

Next, you'll be asked if you want to mount remote file systems. Click on 'Continue'. Then it asks you whether you want to have it re-boot automatically after installation or not. Select 'Automatic' here. This was the last question. The installation program will then start to install the Solaris 7 operating system and copy all the files to the hard drive, which takes about 40 Minutes (depending on the selected software package and the SPARC of course). A big progress bar shows the progress.

After the automatic reboot, you are asked to type in the root password. *Use* one and use a *good* one!

Then it asks you annoying questions about a power saving shutdown mode. We don't want this machine to shutdown by itself, nor do we want to be asked about that again. Type in 'n' on both questions. It then loads the OpenWindows user interface. On the first login as root, you have to choose the desktop environment (I prefer the Common Desktop Environment CDE) – and you're all set! Solaris is installed now. This was the *easy* part… ☺

3 Basic system setups

Before any software package is installed, I recommend the following modifications in several configuration files. Some of them are not necessarily needed, but nice to have…

3.1 Set up root's home directory

On a UNIX system, the default home directory of the super user "root" is just the root directory of the complete directory structure. As it is done in most Linux distributions, I recommend creating a new directory named "**/root**" and let this one be root's home directory. Make root the owner of that directory and let nobody else have access to it. Therefore the following steps are needed:

```
$ cd /
$ mkdir /root
$ chown root:root /root
$ chmod 770 /root
```

Then modify root's entry in the **/etc/passwd** so that his entry there looks like this:

```
root:x:0:1:Super-User:/root:/sbin/sh
```

3.2 Verify the system's IP address and netmask

The host's IP address is stored in **/etc/inet/hosts** and the netmask is stored in **/etc/inet/netmasks**.

3.3 Set the default router and the default gateway for the host

This is set in the files **/etc/defaultrouter** and **/etc/defaultgateway**. Just type in the IP address of the desired gateway in the first line of each of these configuration files.

3.4 Activate DNS for name resolution

In the Solaris setup we deactivated NIS and NIS+ and selected DNS for name resolution. But this is not automatically turned on then. Some manual work needs to be done first:

First create a file named **/etc/resolv.conf** that looks like this (or use different IPs for the nameservers):

```
nameserver 208.213.130.2
nameserver 208.213.130.4
nameserver 208.213.130.5
```

Then modify the "**hosts**" line in **/etc/nsswitch.conf** so that the system, i.e. the function gethostbyname(), actually uses DNS and the line looks like this:

```
hosts:      files dns
```

3.5 Prepare the directory for all packages: `/usr/local`

On a default UNIX system, this directory does not exist. However, most of the binary packages I got from **`http://www.sunfreeware.com`** or other sources, used **`/usr/local`** as their destination directory. I also started to collect all these packages and tar-ed source files in one central directory: **`/usr/local/archive`**. This is very helpful, when you later need a package on a different machine; you don't have get everything together over and over again, just copy the complete directory and start installing… ☺ Just copy the content of the CD into **`/usr/local/archive`**. For faster access, I recommend creating a symbolic link to that directory in **`/`**. Also, most of these packages use **`/usr/local/man`** for their man pages (manual or help pages).

```
$ mkdir /usr/local
$ mkdir /usr/local/man
$ mkdir /usr/local/bin
$ mkdir /usr/local/src
$ mkdir /usr/local/archive
$ ln –s /usr/local/archive /archive
```

Now extract the **`archive`** directory from the CD to **`/usr/local/archive`**. To do this, you first need to install **`gzip-1.3-sol7.pkg`** and **`tar-1.13-sol7.pkg`** from the CD:
(It may give you an error message about conflicting files. Just type 'y' and install it anyway)

```
$ cd /cdrom/install
$ pkgadd –d gzip-1.3-sol7.pkg
$ pkgadd –d tar-1.13-sol7.pkg
$ cd /
$ /usr/local/bin/tar xvzf /cdrom/install/install.tgz
$ /usr/local/bin/tar xvzf /cdrom/install/manual.tgz
```

The version of **`tar`** we just installed is the **`gnutar`**. It is able to create zipped archives on the fly (with the gzip packe installed). The **`tar`** that came with Solaris can't do that. You can easily tell which **`tar`** you are actually using by typing

```
$ tar --version
```

The stupid Solaris **`tar`** doesn't know the **`--version`** option, but it produces the following output:

```
tar: s: unknown option
Usage: tar {txruc}[vfbFXhiBEelmopwnq[0-7]] [-k size] [tapefile] [blocksize] [exclude-
file] [-I include-file] files ...
```

The **`gnutar`** is very nice and tells you more about itself:

```
tar (GNU tar) 1.13

Copyright (C) 1988, 92,93,94,95,96,97,98, 1999 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Written by John Gilmore and Jay Fenlason.
```

To make sure that you use the **gnutar**, I will refer to it as **gnutar** in this manual. Since the actual name of both is **tar** and the only difference is their location (Solaris-**tar** is in **/bin** and GNU-**tar** is in **/usr/local/bin**), I recommend creating a symbolic link and only use the command **gnutar** from now on:

```
$ ln -s /usr/local/bin/tar /usr/local/bin/gnutar
```

3.6 Set up root's environment

Whenever a user logs on, the shell "executes" a file in the user's home directory: **$HOME/.profile**, with "**$HOME**" being replaced by the home directory of the user, for example **/root** or **/home/bernd**. This **.profile** file can be used to set up the search path and other environment variables for the user. I recommend it (**/root/.profile**, **/home/bernd/.profile**,…) to be at least like the following example:

```
MANPATH=/usr/local/man:$MANPATH
PATH=/usr/local/bin:/usr/ccs/bin:/usr/local/mysql/bin:$PATH
LD_LIBRARY_PATH=/usr/local/lib
EDITOR=vi
export MANPATH
export PATH
export LD_LIBRARY_PATH
export EDITOR
```

Note:
/usr/local/bin will contain most of the additional commands we'll install
/usr/ccs/bin contains commands like **ar** or the assembler **as** wich will be needed later
The **EDITOR** variable is used by **crontab** for example. You can edit crontabs in **vi** then rather than the simple **ed**.

Now edit the default **.profile** file located in **/etc/skel**. The contents of this directory will be copied to the home directory of every new user on this system. Append the lines that you just added to root's **.profile**. The **/etc/skel/.profile** file already exists, so don't overwrite it, but append the lines.

Note: If you log in as root from outside, the paths **/usr/sbin** and **/usr/local/sbin** are not in your environment. If you want to be able to run *all* programms (like **ftpwho**, **patchadd**, **halt**, **shutdown**,…) remotely, without the need to add **/usr/bin/…** in front of them, just add **/usr/bin** to root's **.profile** PATH variable.

4 Installation of some useful packages

The following chapter describes the installation of some packages that might be needed later for the "real" packages or just nice to have for the everyday use. The version numbers of the files you may differ from the ones I use in this guide… ☺

Note: A general word about **Solaris Binary Packages**. Most of the packages we'll install come in a binary packages. They can easily be installed by using the **`pkgadd`** command. I gave all the binary packages in my **`/usr/local/archive`** the file extension **`.pkg`** so that I know which of the files are binary packages, binary executables or just **`tar`** archives. When you download binary packages from **`http://www.sunfreeware.com`**, they do of course not have the **`.pkg`** extension.
You can uninstall a binary package with the command **`pkgrm`** and then select the package from the list.

4.1 Install GNUzip package 'gzip'

You already installed the gzip package in chapter 3.5. The GNU Zip is needed for unzipping most of the packages that come as C source file. It will put its binary executable in **`/usr/local/bin`**. That's why we added this path to our environment…

```
$ pkgadd –d gzip-1.2.4.a-sol7.pkg
```

4.2 Apply the Recommended Patch-Cluster

SUN publishes Patch-Cluster for updating the Solaris Operating System. Please check the SUN website frequently for new releases.

They can be found at:

`http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access`.

To apply the Patch-Cluster, just run the binary file and after that REBOOT THE SYSTEM!

```
$ cp /archive/7_Recommended.zip /tmp
$ cd /tmp
$ unzip 7_Recommended
$ cd 7_Recommended
$ ./install-cluster
```

This will take about 15 Minutes, so be patient, even if is seems that the system hangs. You can proceed with the next installation steps in a different terminal console. When I installed it, it was not able to apply the cluster **`107022-05`** (something with Calendar). However, as soon as it is finished installing the patches, reboot the system:

```
$ reboot
```

4.3 Install the advanced command shell 'bash'

The Bash is a very comfortable command shell and it comes with most Linux distribution by default. Once you got used to it, you'll never miss it again… ☺

```
$ pkgadd -d /tmp/bash-2.03-sol7.pkg
```

It will be installed in **/usr/local/bin**. If, later, any user has the bash as his default shell and wants to make an FTP connection to this machine, there must exist the file **/etc/shells**, containing all allowed user shells. So better create the file **/etc/shells** with the following content:

```
/bin/sh
/bin/ksh
/usr/local/bin/bash
```

4.4 Install the improved GNU tar package 'gnutar'

GnuTAR is the GNU version of UNIX tar. It has more functionality than the default UNIX tar and allows long filenames, whereas UNIX tar is restricted there somehow. GNU tar is also able to unzip archives while untaring them at the same time. It handles .tar.Z archives as well as .tar.gz.

You already installed and used it to extract the files on the CD in chapter 3.5.

4.5 Install Netscape Communicator V4.7

Netscape is quite useful when you are logged in locally and install stuff. Whenever you need to download a package or need some installation information, it's neat to just surf there. The TAR archive needs to be un-zipped, untared and installed:

```
$ cp /archive/communicator-v47.tar.gz /tmp
$ cd /tmp
$ gnutar xvzf communicator-v47.tar
```

Then go to the newly created directory and execute the install-script there:

```
$ ./ns-install
```

You'll be asked for the location where Netscape should be installed. Type in **/usr/local/netscape** there and after that, create the following link so that it will be found automatically:

```
$ ln –s /usr/local/netscape/netscape /usr/local/bin/netscape
```

After that you can start netscape from any command prompt by just typing:

```
$ netscape &
```

4.6 Install the GNU C++ Compiler 2.95.2

We'll need a C compiler quite often during the rest of the installation and I suggest installing the GNU C Compiler. It is available as binary package and therefore is very easy to install. You can also use any other ANSI C compiler, but then you're on your own… ☺

```
$ cd /archive
$ pkgadd -d gcc-2.95.2-sol7.pkg
```

Since this system is going to be a web server, I recommend removing the C++ compiler package again after everything is installed, because we don't want any other user to consume CPU power and slowing down the system.

4.7 Install the library 'libstdc++.2.8.1.1'

This package really freaked me out. Although with the GNU C Compiler we just installed, the new version 2.10.0 of this library was installed as well, some programs (like groff) still need that old library package. If anyone can explain to me what I need to do for not having to install this, please let me know…

```
$ cd /archive
$ pkgadd -d libstdc++-2.8.1.1-sol7.pkg
```

After analyzing the package it will tell you that there are conflicting files (attribute changes). Say 'n' for 'Install conflicting files?' and 'y' for 'Continue installation?'.

Now there are versions 2.8.1.1 and 2.10.0 stored in **/usr/local/lib**.

Note: In order for the system to actually find that library package, make sure you set the LD_LIBRARY_PATH environment variable in chapter '0 tar (GNU tar) 1.13

```
Copyright (C) 1988, 92,93,94,95,96,97,98, 1999 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Written by John Gilmore and Jay Fenlason.
```

To make sure that you use the **gnutar**, I will refer to it as **gnutar** in this manual. Since the actual name of both is **tar** and the only difference is their location (Solaris-**tar** is in **/bin** and GNU-**tar** is in **/usr/local/bin**), I recommend creating a symbolic link and only use the command **gnutar** from now on:

```
$ ln -s /usr/local/bin/tar /usr/local/bin/gnutar
```

Set up root's environment' on page 9! Otherwise you'll get strange error messages and some programs won't work.

4.8 Install the 'make' command

You need the **make** command for most of the installations later. If it's not already there, you need to install it. Type '**make**' and hit enter. If you get a 'not found' error message, you need to install it. Otherwise skip this chapter.

```
$ cd /archive
$ pkgadd -d make-3.78.1-sol7.pkg
```

4.9 Install the Berkeley database DB.3.0

Sendmail can use this module to generate usertables and mail relay tables. Also PHP4 supports dbm3, so why not installing this tiny little package..?

```
$ cd /archive
$ pkgadd -d db-3.0.55-sol7.pkg
```

4.10 Install the pager (simple text reader) 'less'

Like the built-in pager 'more' there is a more comfortable one called 'less'. It enables you to scroll up and down text output or files and it is just very convenient.

```
$ cd /archive
$ pkgadd -d less-340-sol7.pkg
```

4.11 Install the secure shell 'ssh'

Ssh ('secure shell') is very useful for remote administration. It allows you to log in the system from another host, similar to **telnet**, but the session is *encrypted*. Nobody will be able to read the data that is transferred between both sides. No passwords (root and sysop passwords) can be sniffed by a third person and you can directly log in as root.

The package includes both a ssh daemon and a ssh client. If the ssh daemon is running, ssh connections from outside are accepted. With the ssh client you can make secure connections to other machines. A freeware ssh client for Windows, written by Cedomir Igaly (e-mail: c.igaly@doc.ic.ac.uk) is available at various FTP sites. Just query your favourite FTP search engine for '**ssh3299054.zip**'.

```
$ cd /usr/local/src
$ tar xvfz /archive/ssh-2.3.0.tar.gz
$ cd ssh-2.3.0
$ ./configure --with-ssh-agent1-compat --without-x
$ make
$ make install
$ mkdir /var/run
$ cp sshd2.startup /etc/rc2.d/S75sshd
```

The 'make install' should also create hostkeys (otherwise use the **ssh-keygen** command). See ssh's documentation for further details.

4.12 Install the advanced process manager 'top'

'Top' is a very useful tool for monitoring the CPU and memory activity on the system. Just type in **top** after installation and you'll get a nice process list with many features (see man page).

```
$ pkgadd -d /archive/top-3.5beta12-sol7.pkg
```

4.13 Install the 'groff' command

Groff is needed for compiling sendmail for example.

```
$ pkgadd -d /archive/groff-1.11a-sol7.pkg
```

4.14 Install 'perl' and additional perl modules

Perl is needed for several tasks and is always "nice to have"... ☺

```
$ pkgadd –d /archive/perl-5.005_03-sol7.pkg
```

Since this package installs in **/usr/local/bin** and not in **/usr/bin**, you should also create a symbolic link to perl from the default location where some customers' scripts may look for it:

```
$ ln -s /usr/local/bin/perl /usr/bin/perl
```

Some additional Perl module may be needed by a customer. The way you install Perl modules is usually the same for all: Simply untar the package and run '**perl <packagename>**', '**make**', '**make test**' and '**make install**'. See the **README** file in the package for instructions. If the customer look for a Perl package you can search for it an download it from **http://search.cpan.org**.

Later, in chapter '10 Setting up WebminSetting up Webmin' you'll need a module called **Net::SSLeay**. One customer needed the **MIME::Base64** module for example:

```
$ cd /tmp
$ tar xvzf /archive/MIME-Base64-2.11.tar.gz
$ perl Makefile.PL
$ make
$ make test
$ make install
```

4.15 Install the simple mail reader 'pine'

Pine is a simple mail reader that allows reading the mailbox and composing mail messages. It is *not* necessary but enables you to read mails for root via a SSH connection for example. Sometimes the root-mailbox gets filled up with system messages. It is also quite useful for testing the sendmail configuration.

```
$ cd /archive
$ pkgadd –d pine-4.05-sol7.pkg
```

5 Compiling and installing sendmail

5.1 Back up the existing sendmail file and configuration

Before we install our own and newer sendmail, I'd suggest that you back up the existing sendmail program and configuration into a tar archive. First, however, you need to stop any running sendmail process.

```
$ /etc/rc2.d/S88sendmail stop
$ gnutar cvzf /archive/OriginalSendmail.tar.gz /usr/lib/sendmail /etc/mail
```

5.2 Make sure that the environment is set properly

For compiling sendmail, we'll need programs like **`ar`**, which are stored in **`/usr/ccs/bin`**. Make sure that you changed your environment like described in chapter '3.6 Set up root's environment' on page 10.

5.3 Compile and install sendmail

Untar the package and build the binaries.

```
$ cd /usr/local/src
$ tar xvfz sendmail.8.11.0.tar.gz
$ cd sendmail.8.11.0/sendmail
$ sh Build
```

Now copy the sendmail binary to its expected location:

```
$ cp obj.SunOS.5.7.sun4/sendmail/sendmail /usr/lib/sendmail
```

Finally create an empty file called **`local-host-names`**.

```
$ touch /etc/mail/local-host-names
```

Before **`sendmail`** works properly, it has to run as suid root. So it is important to set the Set-UID bit in the file permissions:

```
$ chmod u+s /usr/lib/sendmail
```

By default **`sendmail`** is run as suid root (see above). Therefore the mail queue needs to be owned by root. And also make sure, that the mail queue directory and the following temp directories exist and are set to the correct file permissions. These commands will do that:

```
$ chmod go-w / /etc /etc/mail /usr /var /var/spool /var/spool/mqueue
$ chown root / /etc /etc/mail /usr /var /var/spool /var/spool/mqueue
```

Note: **`.forward`** files in unsafe directory paths (directory paths which are group or world writable) will no longer be allowed. This would mean that if user joe's home directory was writable by group staff, sendmail would not use his **`.forward`** file.

5.4 Create a configuration file for sendmail: sendmail.cf

Go to the template directory and copy the suitable template to **config.mc**:

```
$ cd cf/cf
$ cp generic-solaris2.mc config.mc
```

Modify this **config.mc** by adding the following lines after '**DOMAIN(generic)dnl**' (make sure you installed the Berkeley Database package in chapter 4.9, or dbm won't work!):

```
FEATURE(rbl)
FEATURE(`mailertable', `dbm -o /etc/mail/mailertable')
define(`DATABASE_MAP_TYPE', `dbm')
```

After that build **config.cf** from this template and copy it to **/etc/mail/sendmail.cf**:

```
$ cd cf/cf
$ sh Build config.cf
$ cp config.cf /etc/mail/sendmail.cf
```

5.5 Check if automatic startup of sendmail is enabled

When I installed sendmail on that Solaris 7 system, all necessary settings for automatic startup were already done. As the final binary executable program **sendmail** ends up in **/usr/lib/**, the same location of the previous sendmail, we don't need to change anything.

However, if you want to know, where and how the sendmail daemon is started at boot up, the name of the script is **/etc/rc2.d/S88sendmail** and it needs one of the parameters **start** or **stop**.

Note: We're not yet ready to start sendmail right now! More configuration is necessary!

5.6 Additional configuration files `aliases` and `mailertable`

5.6.1 /etc/mail/aliases

The file **/etc/mail/aliases** should already exist on the system, because it comes with the default installation of sendmail. It is just a list of email aliases. For example if all incoming mail for postmaster should be forwarded to admin@somehost.com, the corresponding line in **/etc/mail/aliases** would look like this:

```
Postmaster: admin@somehost.com
```

You can create any alias and also have the mail for that alias forwarded to several people, by just adding the other recipients separated by commas:

```
Admins:     root, sysop, nigmann@future-vision.net
```

The only thing that needs to be done is to create a database out of this **/etc/mail/aliases** file using the **newaliases** command. Every time you modify the **/etc/mail/aliases**, you need to do that. Example:

```
$ echo -e "Postmaster:\tBernd@awwm.com" >> /etc/mail/aliases
$ newaliases
```

5.6.2 /etc/mail/mailertable

This database file can be used to fine tune mail delivery depending on the recipient's address. Normally you won't need it, but it's better to have that feature prepared for future use. To avoid error messages by sendmail, however, I recommend to create an empty database for **mailertable**:

```
$ cd /etc/mail
$ touch mailertable
$ makemap dbm mailertable < mailertable
```

For more information on that feature, please refer to the sendmail book or various online sources... A sample **mailertable** could look like this:

```
first.domain.com:       smtp:route.to.this.domain.com
second.domain.com:      uucp:go.there.com
third.domain.com:       [123.45.34.23]
bad.domain.com:         error:nohost We don't deliver to this domain!
.                       smtp:everything.else.goes.there.com  #this is the "smarthost"
```

Note: Every time you make changes to the **mailertable**, you need to re-build the database with the **makemap** command!

5.7 Some notes on mail relaying

5.7.1 General information

As of sendmail version 8.9, forwarding of SMTP messages is not permitted by default. For example, if you are on site **A.COM**, you will not accept mail from **B.NET** destined for **C.ORG** without special arrangements.

The simplest approach is to list the domains you are willing to relay in the file **/etc/mail/relay-domains**. Anything listed in this file will be accepted for relaying. **N.B.**: Sendmail must be restarted after this file is modified.

For more precise tuning, several FEATUREs have been added to control relaying. Add these to the **config.mc** file you created in chapter 5.4 and re-build your **sendmail.cf** as described:

- **FEATURE(relay_hosts_only)**. Normally domains are listed in **/etc/mail/relay-domains**; any hosts in those domains match. With this feature, each host in a domain must be listed.
- **FEATURE(relay_entire_domain)**. Setting this feature allows relaying of all hosts within your domain. For example, on the host **gateway.A.COM**, this feature allows mail to or from any host in the **A.COM** domain. More precisely, this relays any host listed in the $=m class. This is equivalent to listing the name of the domain in **/etc/mail/relay-domains**.
- **FEATURE(`access_db', `dbm -o /etc/mail/access')**. This enables the hash database **/etc/mail/access** to enable or disable access from individual domains (or hosts, if **FEATURE(relay_hosts_only)** is set). The database format is described below.

- **FEATURE(blacklist_recipients)**. If set, this feature looks up recipients as well as senders in the access database.

5.7.2 The Access Database

The access database (normally in **/etc/mail/access**) allows a mail administrator to administratively allow access to the mail server by individual domains. Each database entry consists of a domain name or network number as the key and an action as the value.

Keys can be a fully or partly qualified host or domain name such as **host.subdomain.domain.com**, **subdomain.domain.com**, or **domain.com**. The last two forms match any host or subdomain under the specified domain. (If **FEATURE(relay_hosts_only)** is set, only the first form works.) Keys can also be a network address or subnetwork, e.g., **205.199.2.250**, **205.199.2**, or **205.199**. The latter two forms match any host in the indicated subnetwork. Lastly, keys can be **user@host.domain** to reject mail from a specific user.

Values can be **REJECT** to refuse connections from this host, **DISCARD** to accept the message but silently discard it (the sender will think it has been accepted), **OK** to allow access (overriding other built-in checks), **RELAY** to allow access including relaying SMTP through your machine, or an arbitrary message to reject the mail with the customized message.

For example, a database (**/etc/mail/access**) might contain:

```
cyberpromo.com    REJECT
sendmail.org      RELAY
spam@buyme.com    550 Spammers shan't see sunlight here
```

to reject all mail from any host in the **cyberpromo.com** domain, allow any relaying to or from any host in the **sendmail.org** domain, and reject mail from **spam@buyme.com** with a specific message.

Note that the **/etc/mail/access** database is a map (i.e. dbm database) and just as with all maps, the database must be generated using **makemap**. For example:

```
$ cd /etc/mail
$ makemap dbm /etc/mail/access < /etc/mail/access
```

After that, sendmail needs to be restarted.

5.8 Redirect all mail messages to a separate log file

By default, only severe error messages concerning the mail system will be listed in the system log file **/var/adm/messages**. Often it is interesting to see what kind of mail traffic took place on a system. This is very helpful to trace any abuse or illegal spammers. Just add the following line in **/etc/syslog.conf** to have all mail activity logged to **/var/log/mail**:

```
mail.info                       /var/log/mail
```

After that you need to create that log file, change permissions and reboot the server (just restarting the syslog daemon didn't work when I tried it…):

```
$ touch /var/log/mail
$ chmod 700 /var/log/mail
$ reboot
```

Note: Make sure the path **`/usr/sbin`** is in your search path. Otherwise the **`reboot`** command won't be found. You need to type **`/usr/sbin/reboot`** instead then.

5.9 Finally start sendmail and test it

Now everything is set up to run the sendmail daemon. Simply type in:

```
$ /etc/rc2.d/S88sendmail start
```

Try to send internal as well as external mail using either **`pine`**, Netscape or the program **`mail`**.

6 Install the database system 'mysql'

6.1 Install and set up the mysql binaries

You may find a newer version of the MySQL binaries for Solaris 7 at **http://www.mysql.com**. For unpacking the tar archive you **MUST** use **gnutar** because of a bug in UNIX tar! This is very important. I basically follow the installation steps in the file **manual.txt**, which comes with MySQL.

```
$ cd /usr/local
$ gnutar xvzf /archive/mysql-3.22.32-sun-solaris2.7-sparc.tar.gz
```

This will create a directory like **/usr/local/mysql-3.22.32-sun-solaris2.7-sparc**. Rename it to **/usr/local/mysql** and make sure, that **/usr/local/mysql/bin** is in your path environment (see chapter 3.6 on page 10).

```
$ mv /usr/local/mysql-3.22.32-sun-solaris2.7-sparc /usr/local/mysql
```

Next, you need to create the 'grant tables' by running a script that came with the distribution:

```
$ cd /usr/local/mysql
$ ./scripts/mysql_install_db
```

Then you need to make sure, that MySQL is automatically started on boot up, by copying the included startup script to **/etc/rc2.d**:

```
$ cp /usr/local/mysql/support-files/mysql.server /etc/rc2.d/S81mysql
$ chowm root:sys /etc/rc2.d/S81mysql
$ chmod 744 /etc/rc2.d/S81mysql
```

6.2 Add some security to MySQL

6.2.1 MySQL should not run as root

Make sure that the mysql server is not running (if you didn't start it yet, it's okay). Create a new user and a new group named mysql and change the ownership of the mysql directory to this new username:

```
$ groupadd mysql
$ useradd -g mysql -d /tmp -s /bin/false -c "MySQL Daemon" mysql
$ chown -R mysql:mysql /usr/local/mysql
$ chmod -R o-r-w-x /usr/local/mysql
```

You also need to modify the startup script for the mysql server in **/etc/rc2.d/S81mysql**, so that mysql is started as user mysql. Add the following line before the '**export PATH**' statement:

```
mysql_daemon_user=mysql         # Run mysqld as this user.
```

Note: MySQL databases are likely to get very large. By default, the data directory is in **/usr/local/mysql**, which is on the 3GB system partition. You may consider moving the complete **/usr/local/mysql** directory to the **/home** partition for example.

6.2.2 Root's MySQL password

Root has full access and control over the complete MySQL server. This MySQL user is set up by default. You may want to change the password of root's access to MySQL. Make sure that the MySQL server is running before you do that. Change it to something like '**my$ql**' or something more cryptic:

```
$ /etc/rc2.d/S81mysql stop
$ /etc/rc2.d/S81mysql start
$ /usr/local/mysql/bin/mysqladmin -u root password 'my$ql'
```

6.3 Start the mysql server

```
$ /etc/rc2.d/S81mysql start
```

6.4 Modify grant tables

Now you need to modify the grant tables for mysql and may create new MySQL users and so on. Please refer to other MySQL documentation for those tasks, because this goes beyond the scope of this manual. Make sure you reload the grant tables after every modification of the grant tables:

```
$ mysqladmin reload
```

Note: Later (in chapter 9) you will install **`phpMyAdmin`**, a web based tool for administrating MySQL databases. Modifying grant tables and adding database users will be a piece of cake then.

7 Install the database system 'PostgreSQL'

7.1 Update the shared memory settings of Solaris

In Solaris 7, the default shared memory maximum segment size kernel parameter is set too low. In order to start up PostgreSQL this needs to be increased. The solution is to put something like the following into **/etc/system** and reboot the system:

```
set shmsys:shminfo_shmmax=0x7fffffff
```

More information on that can be found at:

http://www.sunworld.com/swol-09-1997/swol-09-insidesolaris.html.

7.2 Install 'readline'

Readline enables you to have much more command line editing functionality.

```
$ cd /usr/local/src/
$ tar xvzf /archive/readline-4.1.tgz
$ cd readline-4.1
$ ./configure
$ make
$ make install
```

7.3 Create the postgres superuser account

The PostgreSQL database will run with its own user and group account that you need to create first. Also add user postgres to whatever group the directory **/var/log** belongs to (probably **sys**). Give write permissions to group **sys** in **/var/log** as well.

```
$ groupadd postgres
$ useradd -g postgres -G sys -c "PostgreSQL Database" -s /usr/local/bin/bash -d
  /usr/local/pgsql postgres
$ mkdir /usr/local/pgsql
$ cp /root/.profile ~postgres
$ chown -R postgres:postgres ~postgres
$ chmod 770 /var/log
```

7.4 Compile the source code (don't 'make install' it yet)

```
$ cd /usr/local/src
$ tar xvzf /archive/postgresql-7.0.2.tar.gz
$ cd postgresql-7.0.2/src
$ ./configure --prefix=/usr/local/pgsql
$ make
```

7.5 Run the regression tests to see if it works

The regression checks make sure that the ownership settings are correct and the shared memory modification has been made to the system successfully. To run the regression checks, make sure a file named '**gmake**' exists. This is the GNUmake, that we installed in chapter 4.8. Just create a symbolic link from '**make**' to '**gmake**' so that the test program finds it:

```
$ ln –s /usr/local/bin/make /usr/local/bin/gmake
```

Then change ownership of the complete source tree to the postgres user. Become the postgres user to run the check:

```
$ chown –R postgres:postgres /usr/local/src/postgresql-7.0.2
$ su – postgres
$ cd /usr/local/src/postgresql-7.0.2/src/test/regress
$ make clean all
$ make runcheck
```

Note: The checks will run about four to five minutes. Even if eventually a test results '**FAILED**' on the screen, that's nothing to worry about. The PostgreSQL FAQ says failures in date and time related tests are acceptable...

7.6 Finally install the PostgreSQL package

Still be user postgres when doing the installation!

```
$ cd /usr/local/src/postgresql-7.0.2/src/
$ make install
```

7.7 Some post installation work

Logout and be root again. Then link the shared libraries in the system wide library path **/usr/local/lib** and the executables to **/usr/local/bin**:

```
$ logout
$ ln -s /usr/local/pgsql/lib/libpq.so /usr/local/lib/
$ ln -s /usr/local/pgsql/bin/psql /usr/local/bin/
$ ln -s /usr/local/pgsql/bin/createdb /usr/local/bin/
$ ln -s /usr/local/pgsql/bin/dropdb /usr/local/bin/
```

Create the data directory and initialize the database (**Be user postgres when doing this!**):

```
$ su - postgres
$ cd /usr/local/pgsql
$ mkdir data
$ bin/initdb -D /usr/local/pgsql/data
```

7.8 Start PostgreSQL at bootup automatically and produce logfile

Create a startup script (later with webmin) that looks like this:

```
#!/bin/sh
# Starts and stopps the PostgreSQL database

case "$1" in
'start')
      nohup su postgres -c "/usr/local/pgsql/bin/pg_ctl -D \
            /usr/local/pgsql/data start >/var/log/postgresql.log 2>&1"
      ;;
'stop')
      /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop
      ;;
*)
      echo "Usage: $0 { start | stop }"
      ;;
esac
exit 0
```

8 Install the apache web server and components

Installing Apache needs several steps, because you need to insatall some modules first on which other modules depend. See the dependencies in the following figure:

Apache
- **mod_ssl**
  - **OpenSSL**
  - **MM Shared Memory Library**
- **mod_php4**
  - **gd library (gdlib)**
    - **FreeType Library (libttf)**
    - **Zlib Library (zlib)**
    - **PNG Library (pnglib)**

8.1 Install the gd library 'gdlib'

The **gd** library can be used to generate images in the PNG format (Portable Network Graphics). This is a very nice feature for PHP4, because it allows a PHP script to directly output binaries of a picture to a web-browser. Also (with using the **libttf** library) it is possible to write any text of any size, rotation and color to that image using a TrueType font. See the PHP documentaion (**http://www.php.net**) about creating images with PHP…

8.1.1 Install the FreeType Library (libttf)

```
$ cd /usr/local/src
$ tar xvzf /archive/freetype-1.3.1.tar.gz
$ cd freetype-1.3.1
$ ./configure
$ make
$ make install
```

8.1.2 Install the Zlib Library (zlib)

```
$ cd /usr/local/src
$ tar xvzf /archive/zlib.tar.gz
$ cd zlib-1.1.3/
$ ./configure
$ make
$ make install
```

8.1.3 Install the PNG Library (libpng)

```
$ cd /usr/local/src
$ tar xvzf /archive/libpng-1.0.8.tar.gz
$ cd /libpng-1.0.8
$ mkdir /usr/local/libpng
$ cp scripts/makefile.solaris makefile
$ make
$ make install
```

Optionally you can run a '**`make test`**' before the '**`make install`**' to see, if the package works.

8.1.4 Install the actual GD Library (gdlib)

```
$ cd /usr/local/src
$ tar xvzf /archive/gd-1.8.3.tar.gz
$ cd gd-1.8.3
```

Modify the following lines in the **`Makefile`**:

```
CFLAGS=-O  -DHAVE_LIBTTF
LIBS=-lm -lgd -lpng -lz -lttf
INCLUDEDIRS=-I. -I/usr/local/include -I/usr/local/include/freetype
LIBDIRS=-L. -L/usr/local/lib
```

Then build the library (will be installed to **`/usr/local/lib/libgd.a`**):

```
$ make
$ make install
```

8.2 Install the MM Shared Memory library

```
$ cd /usr/local/src
$ tar xvzf /archive/mm-1.1.3.tar.gz
$ cd mm-1.1.3
$ ./configure --disable-shared
$ make
```

Optionally you can run a '**`make test`**', but notice that we are **not** installing the package. Apache just needs some header files from the source directory!

8.3 Install OpenSSL:

```
$ cd /usr/local/src
$ tar xvfz /archive/openssl-0.9.5a.tar.gz
$ cd openssl-0.9.5a
$ ./config --prefix=/usr/local/ssl -fPIC
$ make
$ make test
```

Now edit the **`Makefile`** and change **`MANDIR`** to **`/usr/local/man`** there. Then install the package:

```
$ make install
```

Create some random data in root's home directory for OpenSSL by repeatedly writing the current process status into a file called .rnd in the root directory:

```
$ ps -ea >> $HOME/.rnd
$ ps -ea >> $HOME/.rnd
$ ps -ea >> $HOME/.rnd
```

Then create a symbolic link for the OpenSSL binary in /usr/local/bin:

```
$ ln -s /usr/local/ssl/bin/openssl /usr/local/bin/openssl
```

8.4 Pre-Configure 'mod_ssl'

The mod_ssl package comes with an excellent manual (a file called **INSTALL**) about installing all the necessary libraries and programs required for mod_ssl. It is really a great documentation that helps to understand how Apache ant its modules work together.

```
$ cd /usr/local/src
$ tar xvzf /archive/mod_ssl-2.6.6-1.3.12.tar.gz
$ cd /usr/local/src/mod_ssl-2.6.6-1.3.12/
$ ./configure --with-apache=../apache_1.3.12/
```

8.5 Compile and install Apache

```
$ cd /usr/local/src
$ tar xvzf /archive/apache_1.3.12.tar.gz
$ cd /usr/local/src/apache_1.3.12
$ SSL_BASE=../openssl-0.9.5a EAPI_MM=../mm-1.1.3 ./configure \
        --prefix=/usr/local/httpd --enable-module=ssl --enable-module=so \
        --enable-shared=ssl --enable-shared=max --enable-suexec --suexec-caller=httpd \
        --suexec-docroot=/home/customers --suexec-userdir=/home/customers --show-layout
$ make
```

Warning: Read chapter 8.9 about suEXEC!!

If you don't have a SSL certificate yet, you should create one right now in order to test, if Apache's SSL features work or not. Just type in

```
$ make certificate TYPE=test
```

to create such a test certificate. Type in some stuff when you're asked for names, cities, states and so on. It automatically stores the certificates in the right position /usr/local/httpd/conf. A full list of possible certificate types you can create is shown when the make command finished.

I copied the output to the end of this sub-chapter on the next page.

When you created or copied an SSL certificate, install Apache:

```
$ make install
```

```
+--------------------------------------------------------------------+
| Before you install the package you now should prepare the SSL      |
| certificate system by running the 'make certificate' command.      |
| For different situations the following variants are provided:      |
|                                                                    |
| % make certificate TYPE=dummy    (dummy self-signed Snake Oil cert)|
| % make certificate TYPE=test     (test cert signed by Snake Oil CA)|
| % make certificate TYPE=custom   (custom cert signed by own CA)    |
| % make certificate TYPE=existing (existing cert)                   |
|        CRT=/path/to/your.crt [KEY=/path/to/your.key]               |
|                                                                    |
| Use TYPE=dummy    when you're a  vendor package maintainer,        |
| the TYPE=test     when you're an admin but want to do tests only,  |
| the TYPE=custom   when you're an admin willing to run a real server|
| and TYPE=existing when you're an admin who upgrades a server.      |
| (The default is TYPE=test)                                         |
|                                                                    |
| Use 'make certificate VIEW=1' to display the generated data.       |
|                                                                    |
+--------------------------------------------------------------------+
```

8.6 Install mod_php4

The mod_php4 module will be installed dynamically as a DAO module. It doesn't need to be compiled into Apache, but can be installed **after** Apache was compiled and installed.

```
$ cd /usr/local/src
$ tar xvzf /archive/php-4.0.2.tar.gz
$ cd php-4.0.2
$ ./configure --with-apxs=/usr/local/httpd/bin/apxs \
        --with-db3=/usr/local/BerkeleyDB.3.0/lib --with-zlib-dir=/usr/local/lib \
        --enable-ftp --with-gd=yes --with-ttf=/usr/local/lib \
        --with-mysql=/usr/local/mysql --with-png-dir=/usr/local/lib --with-zlib
$ make
$ make install
```

Finally, copy the default **`php.ini`** file to its location:

```
$ cp /usr/local/src/php-4.0.2/php.ini-dist /usr/local/lib/php.ini
```

8.7 Tune the Apache configuration file /usr/local/httpd/conf/httpd.conf

Apache's configuration file needs some fine tuning in order to activate PHP4 and Perl scripts for example. Here is a copy of the current httpd file without all the comments. The sections which are very important are highlighted in **bold**. The corresponding annotation is sticking on the right side of the page.

```
##
## httpd.conf -- Apache HTTP server configuration file
##

### Section 1: Global Environment ########################################
#
ServerType standalone
ServerRoot "/usr/local/httpd"
PidFile /usr/local/httpd/logs/httpd.pid
ScoreBoardFile /usr/local/httpd/logs/httpd.scoreboard
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
MinSpareServers 5
MaxSpareServers 10
StartServers 5
MaxClients 150
MaxRequestsPerChild 10000

LoadModule env_module          libexec/mod_env.so
LoadModule config_log_module   libexec/mod_log_config.so
LoadModule mime_module         libexec/mod_mime.so
LoadModule negotiation_module libexec/mod_negotiation.so
LoadModule status_module       libexec/mod_status.so
LoadModule includes_module     libexec/mod_include.so
LoadModule autoindex_module    libexec/mod_autoindex.so
LoadModule dir_module          libexec/mod_dir.so
LoadModule cgi_module          libexec/mod_cgi.so
LoadModule asis_module         libexec/mod_asis.so
LoadModule imap_module         libexec/mod_imap.so
LoadModule action_module       libexec/mod_actions.so
LoadModule userdir_module      libexec/mod_userdir.so
LoadModule alias_module        libexec/mod_alias.so
LoadModule access_module       libexec/mod_access.so
LoadModule auth_module         libexec/mod_auth.so
LoadModule setenvif_module     libexec/mod_setenvif.so
LoadModule php4_module         libexec/libphp4.so

<IfDefine SSL>
LoadModule ssl_module          libexec/libssl.so
</IfDefine>

ClearModuleList
AddModule mod_env.c
AddModule mod_log_config.c
AddModule mod_mime.c
AddModule mod_negotiation.c
AddModule mod_status.c
AddModule mod_include.c
AddModule mod_autoindex.c
AddModule mod_dir.c
AddModule mod_cgi.c
AddModule mod_asis.c
AddModule mod_imap.c
```

```
AddModule mod_actions.c
AddModule mod_userdir.c
AddModule mod_alias.c
AddModule mod_access.c
AddModule mod_auth.c
AddModule mod_so.c
AddModule mod_setenvif.c
AddModule mod_php4.c

<IfDefine SSL>
AddModule mod_ssl.c
</IfDefine>

ExtendedStatus On

### Section 2: 'Main' server configuration ###########################################
#
Port 80

##
##  SSL Support
##
<IfDefine SSL>
    Listen 80
    Listen 443
</IfDefine>

User httpd
Group httpd

ServerAdmin Bernd@awwm.com
#ServerName Sun1.awwm.com

DocumentRoot "/usr/local/httpd/htdocs"

<Directory />
    Options FollowSymLinks
    AllowOverride FileInfo AuthConfig Limit
</Directory>

<Directory "/usr/local/httpd/htdocs">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>

<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm default.html default.htm index.php index.php3
default.php default.php3
</IfModule>
```

(1) Gives an extended server status if you do a status query. See Annotation (7).

(2) See next chapter: we'll run Apache with its own user/group settings.

(3) These pages will be loaded automatically if present in a directory.

```
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

UseCanonicalName On

<IfModule mod_mime.c>
    TypesConfig /usr/local/httpd/conf/mime.types
</IfModule>

DefaultType text/plain

<IfModule mod_mime_magic.c>
    MIMEMagicFile /usr/local/httpd/conf/magic
</IfModule>

HostnameLookups Off
ErrorLog /usr/local/httpd/logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog /usr/local/httpd/logs/access_log common
#CustomLog /usr/local/httpd/logs/referer_log referer
#CustomLog /usr/local/httpd/logs/agent_log agent

ServerSignature EMail

<IfModule mod_alias.c>
    ## BND: Alias /icons/ "/usr/local/httpd/icons/"

    <Directory "/usr/local/httpd/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    ## Removed by Bernd:
    #ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"

    <Directory "/usr/local/httpd/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>

<IfModule mod_autoindex.c>
```

```
    IndexOptions FancyIndexing
    AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
    AddIconByType (TXT,/icons/text.gif) text/*
    AddIconByType (IMG,/icons/image2.gif) image/*
    AddIconByType (SND,/icons/sound2.gif) audio/*
    AddIconByType (VID,/icons/movie.gif) video/*

    AddIcon /icons/binary.gif .bin .exe
    [...]
    DefaultIcon /icons/unknown.gif

    ReadmeName README
    HeaderName HEADER
    IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
</IfModule>

<IfModule mod_mime.c>
    AddEncoding x-compress Z
    AddEncoding x-gzip gz tgz
    AddLanguage da .dk
    [...]
    <IfModule mod_negotiation.c>
        LanguagePriority en da nl et fr de el it ja pl pt pt-br ltz ca es sv
    </IfModule>

    AddType application/x-httpd-php .php .php3
    AddType application/x-httpd-php-source .phps

    AddType application/x-tar .tgz
    AddHandler cgi-script .cgi
    AddHandler cgi-script .pl
</IfModule>

## Added by Bernd

<Files *.pl>
  SetHandler cgi-script
  AddHandler cgi-script .pl
  Options +ExecCGI
</Files>

<IfModule mod_setenvif.c>
    BrowserMatch "Mozilla/2" nokeepalive
    BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
    BrowserMatch "RealPlayer 4\.0" force-response-1.0
    BrowserMatch "Java/1\.0" force-response-1.0
    BrowserMatch "JDK/1\.0" force-response-1.0
</IfModule>


# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
<Location /server-status>
    SetHandler server-status
```

(4) This is necessary so that PHP3 or PHP4 scripts are processed by the PHP engine.

(5) Assign the .pl extension to perl scripts

(6) All Files ending with .pl are exeuted as perl scripts

```
    Order deny,allow
    Deny from all
    Allow from .infoisintelligence.com
</Location>

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#    ServerAdmin webmaster@host.some_domain.com
#    DocumentRoot /www/docs/host.some_domain.com
#    ServerName host.some_domain.com
#    ErrorLog logs/host.some_domain.com-error_log
#    CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>
#<VirtualHost _default_:*>
#</VirtualHost>

##
##  SSL Global Context Virtual Host for SSL port
##
<IfDefine SSL>
    AddType application/x-x509-ca-cert .crt
    AddType application/x-pkcs7-crl    .crl
</IfDefine>

<IfModule mod_ssl.c>
    SSLPassPhraseDialog  builtin
    SSLSessionCache         dbm:/usr/local/httpd/logs/ssl_scache
    SSLSessionCacheTimeout  300
    SSLMutex  file:/usr/local/httpd/logs/ssl_mutex
    SSLRandomSeed startup builtin
    SSLRandomSeed connect builtin
    SSLLog      /usr/local/httpd/logs/ssl_engine_log
    SSLLogLevel info
</IfModule>

<IfDefine SSL>
## SSL Virtual Host Context
##
  <VirtualHost _default_:443>
    DocumentRoot "/usr/local/httpd/htdocs"
    ServerName Sun1.awwm.com
    ServerAdmin Bernd@awwm.com
    ErrorLog /usr/local/httpd/logs/error_log
    TransferLog /usr/local/httpd/logs/access_log

#   Enable/Disable SSL for this virtual host.
    SSLEngine on

#   SSL Cipher Suite:
    SSLCipherSuite ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
```

(7) If you enable this and set the right “AllowFrom” value, then you can query the Apache status by browsing to
`http://servername/server-status`
See annotation **Error! Reference source not found.**

```
#   Server Certificate:
    SSLCertificateFile /usr/local/httpd/conf/ssl.crt/server.crt
    SSLCertificateKeyFile /usr/local/httpd/conf/ssl.key/server.key

    <Files ~ "\.(cgi|shtml|phtml|php3?)$">
      SSLOptions +StdEnvVars
    </Files>
    <Directory "/usr/local/httpd/cgi-bin">
      SSLOptions +StdEnvVars
    </Directory>
    SetEnvIf User-Agent ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

    CustomLog /usr/local/httpd/logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

  </VirtualHost>
</IfDefine>


<VirtualHost 63.64.191.43>
      DocumentRoot /home/customers/digav/public_html
      User digav
      Group cust
      ServerName 63.64.191.43
      ServerAdmin support@digav.com
      ErrorLog /home/customers/digav/logs/error_log
      TransferLog /home/customers/digav/logs/access_log
</VirtualHost>
```

(8) This is just a sample VirtualHost definition.

8.8 Set up a new user/group for Apache

We'll run Apache with its own user and group settings. We'll create a user httpd and a group httpd for that. Make sure it's set in the **`httpd.conf`** file (see above).

```
$ groupadd httpd
$ useradd -g httpd -d /usr/local/httpd/ -s /bin/false -c "The Apache Webserver" httpd
$ chown -R httpd:httpd /usr/local/httpd
$ chmod -R 770 /usr/local/httpd
```

Warning: READ NEXT CHAPTER!!

8.9 Caution: SuEXEC!

You just installed Apache with the suEXEC module turned on! This feature enables users to run CGI and perl scripts with their own user ID rather than using the 'virtual' user 'httpd' you created in the previous chapter. To make sure this works, the scripts and CGI programs in the user's web directory need to be executable by the owner only! This gives a little more security to them. However, to do so, the program that changes the user ID of the script needs to have the 'setuid root' bit. With the commands in the previous

chapter, you changed the ownership of all files in the httpd directory to **-rwx------**. Now you **MUST** type the following commands to get suEXEC working again:

```
$ cd /usr/local/httpd/bin
$ chown root suexec
$ chmod u=rwxs suexec
$ chmod g+rwx suexec
$ ls -l suexec

-rws------   1 root     httpd      11512 Sep  7 16:10 suexec
```

8.10 Set up the startup script for Apache

copy the startup script to /etc/rc2.d and modify it:

```
$ cp /usr/local/httpd/bin/apachectl /etc/rc2.d/S75apache
$ chown root:sys /etc/rc2.d/S75apache
```

Attention: If you want Apache to run with the SSL module enabled, you **have to** modify that startup script: Find the line "**if $HTTPD ; then**" in the section "**start)**" and add **"-DSSL**", so that it looks like this:

```
 "if $HTTPD -DSSL; then"
```

8.11 Restricting web access to certain directories

To restrict the access to special web page directories you can use a **.htaccess** file. Just store it in the directory you want to control. This is just a primitive example, more detailed information about this can be found in the documentation at **http://www.apache.org**. Make the **.htaccess** file look like this:

```
AuthUserFile /etc/htpasswd
AuthGroupFile /dev/null
AuthName "MySQL Configuration"
AuthType Basic
<Limit GET POST>
  require user bernd sysop
</Limit>
```

For using this **.htaccess**, however, you need to create a **/etc/htpasswd** file, containing the allowed usernames and encrypted passwords for accessing the directory. This has nothing to do with the **/etc/passwd** and the users on this system, i.e. it is completely independent. Also, you can create several different **htpasswd** files. Create the **/etc/htpasswd** by using the command **htpasswd** in the Apache distribution:

```
$ cd /usr/local/apache/bin
$ ./htpasswd -h
Usage:
        htpasswd [-cmdps] passwordfile username
        htpasswd -b[cmdps] passwordfile username password

 -c  Create a new file.
 -m  Force MD5 encryption of the password.
```

```
 -d  Force CRYPT encryption of the password (default).
 -p  Do not encrypt the password (plaintext).
 -s  Force SHA encryption of the password.
 -b  Use the password from the command line rather than prompting for it.
$ ./htpasswd -bc /etc/htpasswd bernd GfgFgRgr
$ ./htpasswd -b /etc/htpasswd sysop aaDWqw!2
```

Note: Only for the *first* time (i.e. for *creating* the htpasswd file) use the `-bc` option. Once the file exists, just use the `-b` option!

8.12 Just for Bernd's reference: Create a SSL test certificate – Ignore this!!

Make sure you created the random file when installing OpenSSL – otherwise you'll get an error now.

```
$ cd /usr/local/ssl/ssl/certs
$ openssl req -new > Solaris71.csr
```

Remove the password from the key:

```
$ openssl  rsa -in privkey.pem -out Solaris71.key
```

Then sign the key:

```
$ openssl x509 -in Solaris71.csr -out Solaris71.cert -req \
          -signkey Solaris71.key -days 365
```

9 PhpMyAdmin for managing MySQL

PhpMyAdmin is a PHP3 based tool for managing any MySQL databases via any web browser. Just unpack the archive into a directory that is reachable by Apache, **/usr/local/apache/htdoct/phpMyAdmin** for example.

```
$ cd /usr/local/httpd/htdocs
$ tar xvzf /archive/phpMyAdmin_2.1.0php.tar.gz
```

You need to modify the config file **/usr/local/httpd/htdocs/phpMyAdmin/config.inc.php"**, if you have set a password for root's access to MySQL (see chapter 6.2.2). Otherwise phpMyAdmin will just return an error message, saying that it cannot access MySQL. Modify the following lines:

```
[...]
$cfgServers[1]['host'] = 'localhost';    // MySQL hostname
$cfgServers[1]['port'] = '';             // MySQL port - leave blank for default port
$cfgServers[1]['adv_auth'] = true;       // Use advanced authentication?
$cfgServers[1]['stduser'] = 'root';      // MySQL standard user (advanced auth)
$cfgServers[1]['stdpass'] = 'my$ql';     // MySQL standard password (advanced auth)
$cfgServers[1]['user'] = 'root';         // MySQL user (basic auth)
$cfgServers[1]['password'] = '';         // MySQL password (basic auth)
[...]
```

When you connect to **http://localhost/phpMyAdmin** (or the hostname instead of 'localhost' if done remotely), you are required to type in a username and a password. This username will be used to access the MySQL grant tables, so you should log on as '**root**' and password '**my$ql**' (or whatever you set it to). If mod_ssl works fine, you can also connect via secure connection: **https://...**

10 Setting up Webmin

New Webmin versions show up very frequently. So please check their website first for a new version before installing it: **http://www.webmin.com**.

10.1 Nice feature: SSL connectiom with webmin

Since you'll set up very sensitive settings with webmin, I highly recommend using only SSL connections to connect to webmin. Webmin-SSL is completely independent from any Apache SSL stuff and brings its own SSL certificate. The only thing you'll need is the **Net::SSLeay** module for Perl first:

```
$ cd /usr/local/src
$ tar xvzf /archive/Net_SSLeay.pm-1.05.tar.gz
$ cd Net_SSLeay.pm-1.05
$ ./Makefile.PL
$ make
$ make install
```

Ignore any error messages about random generators that could not be seeded… That's just because Solaris doesn't have the **/dev/random** device. The connections will be encrypted anyway (**make test**).

10.2 Install webmin

To install Webmin, just unpack it and run the **setup.sh** script. At most of the questions you can just press enter to accept the suggested defaults in '[' and ']'.

```
$ cd /usr/local/
$ tar xvzf /archive/webmin-0.80.tar.gz
$ cd /usr/local/webmin-0.80/
$ ./setup.sh
```

```
*************************************************************************
*            Welcome to the Webmin setup script, version 0.80           *
*************************************************************************
Webmin is a web-based interface that allows Unix-like operating
systems and common Unix services to be easily administered.

Installing Webmin in /usr/local/webmin-0.80 ...

*************************************************************************
Webmin uses separate directories for configuration files and log files.
Unless you want to run multiple versions of Webmin at the same time
you can just accept the defaults.

Config file directory [/etc/webmin]:
Log file directory [/var/webmin]:

*************************************************************************
Webmin is written entirely in Perl. Please enter the full path to the
```

```
Perl 5 interpreter on your system.

Full path to perl (default /usr/bin/perl): /usr/local/bin/perl

Testing Perl ...
Perl seems to be installed ok

******************************************************************************
For Webmin to work properly, it needs to know which operating system
type and version you are running. Please select your system type by
entering the number next to it from the list below
------------------------------------------------------------------------------
  1) Sun Solaris                          2) Caldera OpenLinux eServer
  3) Caldera OpenLinux                    4) Redhat Linux
  5) Slackware Linux                      6) Debian Linux
  7) SuSE Linux                           8) Corel Linux
  9) TurboLinux                          10) Cobalt Linux
 11) Mandrake Linux                      12) Delix DLD Linux
 13) MkLinux                             14) XLinux
 15) LinuxPL                             16) Linux From Scratch
 17) FreeBSD                             18) OpenBSD
 19) BSDI                                20) HP/UX
 21) SGI Irix                            22) DEC/Compaq OSF/1
 23) IBM AIX                             24) SCO UnixWare
 25) SCO OpenServer                      26) MacOS Server X
------------------------------------------------------------------------------
Operating system: 1

Please choose which version of Sun Solaris you are running, by entering
the number next to it from the list below
------------------------------------------------------------------------------
  1) Sun Solaris 2.5                      2) Sun Solaris 2.5.1
  3) Sun Solaris 2.6                      4) Sun Solaris 7
  5) Sun Solaris 8
------------------------------------------------------------------------------
Version: 4

Operating system name:    Sun Solaris
Operating system version: 7

******************************************************************************
Webmin uses its own password protected web server to provide access
to the administration programs. The setup script needs to know :
 - What port to run the web server on. There must not be another
   web server already using this port.
 - The login name required to access the web server.
 - The password required to access the web server.
 - The hostname of this system that the web server should use.
 - If the webserver should use SSL (if your system supports it).
 - Whether to start webmin at boot time.

Web server port (default 10000): 9999
Login name (default admin):
Login password: eoflfbmg
Password again: eoflfbmg
```

```
Web server hostname (default Sun1.awwm.com):
Use SSL (y/n): y
Start Webmin at boot time (y/n): y
*************************************************************************

... configuration ...

*************************************************************************
Webmin has been installed and started successfully. Use your web
browser to go to

  https://Sun1.awwm.com:9999/

and login with the name and password you entered previously.

Because Webmin uses SSL for encryption only, the certificate
it uses is not signed by one of the recognized CAs such as
Verisign. When you first connect to the Webmin server, your
browser will ask you if you want to accept the certificate
presented, as it does not recognize the CA. Say yes.
```