INSTALLATION AND CONFIGURATION ISSUES ABOUT FREEBSD OPERATING SYSTEM

Lorentz JÄNTSCHI and Sorana BOLBOACĂ

Technical University of Cluj-Napoca, Romania, http://lori.academicdirect.ro
and
"Iuliu Haţieganu" Medicine and Pharmacy University, Cluj-Napoca, Romania

Abstract

The paper is based on the experience of the authors with the FreeBSD server operating system administration on three servers in use under academicdirect.ro domain. The paper describes a set of installation, preparation, and administration aspects of a FreeBSD server.

Keywords

Server operating systems, Operating system configuration, Server services, Client-server applications, Dial-in server, System testing.

1. INTRODUCTION

UNIX is an interactive time-sharing operating system invented in 1969 by Ken Thompson after Bell Labs left the Multics project, originally so he could play games on his scavenged PDP-7. The time-sharing is an operating system feature allowing several users to run several tasks concurrently on one processor, or in parallel on many processors, usually providing each user with his own terminal for input and output; time-sharing is multitasking for multiple users. By 1991, UNIX had become the most widely used multi-user general-purpose operating system in the world. UNIX is now offered by many manufacturers and is the subject of an international standardization effort. Unix-like operating systems include Debian, Linux and LinwowsOS, AIX, GNU, HP-UX, OSF and Solaris, BSD/OS, NetBSD, OpenBSD and FreeBSD (with TrustedBSD and PicoBSD project variations) [1].

FreeBSD (FreeBSD is a registered trademark of Wind River Systems, Inc. and this is expected to change soon) is an advanced operating system for x86 compatible, AMD64, Alpha, IA-64, PC-98 and UltraSPARC architectures. The FreeBSD operating system is developed and maintained by a large team of individuals. While you might expect an operating system with these features to sell for a high price, FreeBSD is available free of charge and comes with full source code.

The most important feature of a server system is system services. Most of the services in a server system are provided through a program or process that sits idly in the background until it is invoked to perform its task, called daemons [2]. The daemon word come from the mythological meaning, later rationalized as the acronym "Disk And

Execution MONitor" [3]. A daemon is program that is not invoked explicitly, but lays dormant waiting for some condition(s) to occur. The idea is that the perpetrator of the condition need not be aware that a daemon is lurking (though often a program will commit an action only because it knows that it will implicitly invoke a daemon). Daemons are usually spawned automatically by the system, and may either live forever or be regenerated at intervals. The discussed services are Internet domain name server (*named*, [4?query=named]), Internet super-server (*inetd*, [4?query=inetd]), OpenSSH SSH daemon (sshd, [4?query=sshd]), Internet file transfer protocol server (*ftpd*, [4?query=ftpd]), Apache hypertext transfer protocol server (*httpd*, [4?query=httpd]), proxy caching server (*squid*, [4?query=squid), the MySQL server demon (*mysqld*, [4?query=mysqld]) and PHP sub-service (post processed hypertext [5]).

2. OPERATING SYSTEM INSTALLING PROCEDURE

First step in FreeBSD operating system installation is to create a boot disk set, depending on machine type. If we are using a Personal Computer, based on i386 computer architecture, a disk boot set can be found at:

ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/5.2-RELEASE/floppies/

For 1.44Mb floppies, all that we have to do is to download at least *kern.flp* and *mfsroot.flp* files. If the planned computer to be a server has exotic or old components, is possible to need also the *drivers.flp* file. If we use a DOS/Windows operating system type, to create the boot disks is necessary to download and use an image file installation program, which can be found at the address: *ftp://ftp.freebsd.org/pub/FreeBSD/tools/*. It can be used any of *fdimage.exe* or *rawrite.exe* to create the disks. For *fdimage.exe* the commands (DOS commands) are (assuming that we use a: drive): *fdimage –f 1.44M kern.flp a:* (and similarly for *mfsroot.flp* and *drivers.flp* files).

If we use a UNIX operating system type, we can use dd program for disks creation: *dd if=kern.flp of=/dev/floppy* (and similarly for *mfsroot.flp* and *drivers.flp* files)

After the boot disks creation, we must boot from "*kern.flp*" floppy and "*mfsroot.flp*" floppy the FreeBSD operating system. Kernel and SysInstall utility are automatically loaded and after that, we have two consoles (alt+F1 and alt+F2 respectively). The second console is for *DEBUG* messages. In the *DEBUG* console we can watch how modules are loaded. In this moment, a good idea is to look at the *DEBUG* console to assure that our network card is proper used. At this point, *SysInstall* utility load *FDISK partion editor* and we must create a FreeBSD partition [6].

3. OPERATING SYSTEM CONFIGURATION

After the system installation, we can configure it. Many configurations can be done. We can start to download now all system sources. A utility called *cvsup* can be used for this task. *CVSup* is a software package for distributing and updating source trees from a master CVS repository on a remote server host. The FreeBSD sources are maintained in a CVS repository on a central development machine in California. With CVSup, FreeBSD users can easily keep their own source trees up to date. Using *SysInstall* utility, we can fetch the *cvsup* program in the same way as we installed the system, from internet via FTP protocol (*sysinstall/Configure/Packages/…logging… /devel/cvsup-without-gui-16.1h*). After the *cvsup* installation, a *configuration file* (let us call it *configuration_file)* must be created (or edited from /usr/share/examples/cvsup/) and must contain the *host* (this specifies the server host which will supply the file updates), the *base* (this specifies the root where CVSup will store information about the collections you have transferred to our system), the prefix (this specifies where to place the requested files), and the desired release (version). Other options are also benefit:

**default host=cvsup.FreeBSD.org, *default base=/usr, *default prefix=/usr, *default release=cvs, *default delete use-rel-suffix, *default compress, src-all tag=., ports-all tag=., doc-all tag=., cvsroot-all tag=.*

Sources can be fetched separately (such as *src-base*) or entirely (such as *src-all*). Tag option is used to fetch one specific version of the sources (when "." means CURRENT versions). In addition, the date option can be used (as example: src-all tag=RELENG_4 date=2000.08.27.10.00.00). Fetching procedure can be done now from a text console, using a simple command: *cvsup -g -L 2 configuration_file* or from a graphical console (X-based) using the command: *cvsup configuration_file*.

4. RECOMPILATION AND SYSTEM OPTIMIZATION

The kernel is the core of the FreeBSD operating system. Building a custom kernel is one of the most important rites of passage nearly every UNIX user must endure. This process, while time consuming, will provide many benefits to your FreeBSD system. Unlike the *GENERIC* kernel, preinstalled in our system, which must support a wide range of hardware, a custom kernel only contains support for your PC's hardware. This has a number of benefits, such as faster boot time, less memory usage, and additional hardware support; a custom kernel allows you to add in support for devices such as sound cards, which are not present in the *GENERIC* kernel.

If we follow the acquiring procedure of the sources exactly, we can found for the kernel configuration a set of predefined configuration files at the location: */usr/src/sys/i386/conf/*. If the sources version fit with our system then the *GENERIC* file using must produce same kernel and modules with the existent ones. The idea is to optimize the kernel at compilation time. The kernel can be configured in a configuration file using the prescriptions that can be found in following files: *GENERIC*, *Makefile*, *NOTES* (/usr/src/sys/i386/conf/), *NOTES* from /usr/src/sys/conf/ and *README* and *UPDATING* from /usr/src/. Additionally, we can create the *LINT* file which contain additional kernel configuration options from *NOTES* files with make utility (*cd /usr/src/sys/i386/conf/ && make LINT*). In the optimizing process of the kernel, a good idea is to look at the system characteristics detected by the *GENERIC* kernel using the *dmesg* utility. Most of the essential options are well documented and we cannot miss. Anyway, a large set of network devices can be excluded from the kernel. To find which device driver is using in the system for network adapter management we can look again at the boot messages (dmesg | grep Ethernet). Supposing that we have finished our kernel configuration, the next step is to configure-it according with the new configuration file:

```
cd /usr/src/sys/i386/conf/ && config VL
```

The next three steps can be emerged in one composed command:

```
cd ../compile/VL && make depend && make && make install
```

5. THE SYSTEM SERVICES

The kernel configuration process allowed us to define console behavior (to disable cltr+alt+del reboot sequence), to increase the amount of free memory available for processes and increase the system speed. Now can begin to install and configure the server services. *The named service.* Name servers usually come in two forms: an *authoritative name server*, and a *caching name server*. An *authoritative name server* is needed when one wants to serve DNS information to the world, replying authoritatively to queries, a domain, such as academicdirect.ro, is registered (to RNC, [7]) and IP addresses need to be assigned to hostnames under it; an IP address block requires reverse DNS entries (IP to hostname) and/or a backup name server, called a slave, must reply to queries when the primary is down or inaccessible. *A named configuration file resides* in /etc/namedb/ directory, and to start automatically at boot, the /etc/rc.conf file must contain named_enable="YES".

For a real name server, at least following lines (from /etc/namedb/named.conf file) must fit with our system (academicdirect.ro):

zone "academicdirect.ro" {\r\n type master;\r\n file "academicdirect.ro";\r\n};

Therefore, in *academicdirect.ro* file we must specify the zone. At least following lines must fit (see also [7]):

$TTL 3600\r\n academicdirect.ro. IN SOA ns.academicdirect.ro. root.academicdirect.ro. (\r\n 2004020902;Serial\r\n 3600; Refresh\r\n 1800; Retry\r\n 604800; Expire\r\n 86400);Minimum TTL\r\n @ IN NS ns.academicdirect.ro. ; DNS Server\r\n @ IN NS hercule.utcluj.ro. ; DNS Server\r\n localhost IN A 127.0.0.1; Machine Name\r\n ns IN A 193.226.7.211; Machine Name\r\n mail IN A 193.226.7.211; Machine Name\r\n @ IN A 193.226.7.211; Machine Name

To properly create the local reverse DNS zone file, following command are necessary: cd /etc/namedb && sh make-localhost.

The *inetd* service manages (start, restart, and stop) a set of services (according with Internet server configuration database /etc/inetd.conf), for both IPv4 and IPv6 protocols, such as:

ftp stream tcp46 nowait root /usr/libexec/ftpd ftpd –l # ftp IPv4 and IPV6 service
ssh stream tcp46 nowait root /usr/sbin/sshd sshd -i -46 # ssh IPv4 and IPV6 service
finger stream tcp46 nowait/3/10 nobody /usr/libexec/fingerd fingerd –s # finger IPv4
ntalk dgram udp wait tty:tty /usr/libexec/ntalkd ntalkd # talk
pop3 stream tcp46 nowait root /usr/local/libexec/popper popper # pop3 IPv4 and IPV6 service

In some cases, is possible that *ined* service do not start. A solution is manual starting of a specific service (/usr/libexec/ftpd -46Dh) or creating of an executable shell script and place-it in an rc.d directory:

-r-xr-xr-x 1 root wheel 60 Feb 12 12:34 /usr/local/etc/rc.d/ftpd.sh (ls –al)
/usr/libexec/ftpd -46Dh && echo -n ' ftpd' (ftpd.sh file content)

The *Hypertext Transfer Protocol Server* can be provided also by many applications such as *httpd* (apache@apache.org), *bozohttpd* (Janos.Mohacsi@bsd.hu), *dhttpd* (gslin@ccca.nctu.edu.tw), *fhttpd* (ports@FreeBSD.org), *micro_httpd* (user@unknown.nu), *mini_httpd* (se@FreeBSD.org), *tclhttpd* (mi@aldan.algebra.com), *thttpd* (anders@FreeBSD.org), *w3c-httpd* (ports@FreeBSD.org), but full featured and multiplatform capable remains *httpd* from Apache [8]. The most important feature of Apache web server is PHP language modules support, which transform our web server into a real client-server interactive application. For *httpd* service (/usr/local/etc/apache2/httpd.conf): Listen 80 (httpd port), <IfModule mod_php5.c>\r\n AddType application/x-httpd-php .php\r\n AddType application/x-httpd-php-source .phps\r\n </IfModule> # not included by the default but required to work

For PHP module (/usr/local/etc/php.ini):
precision = 14 \r\n expose_php = On \r\n max_execution_time = 3000 \r\n max_input_time = 600 \r\n memory_limit = 128M \r\n post_max_size = 8M \r\n file_uploads = On \r\n upload_max_filesize = 8M \r\n display_errors = On (for production web sites, turn this feature Off).

For *squid* service (/usr/local/etc/squid/squid.conf) the most important section is for proxyed IP's: acl network src 172.27.211.1 172.27.211.2 193.226.7.200 192.168.211.2\r\n http_access allow network\r\n acl ppp src 192.168.211.0/24

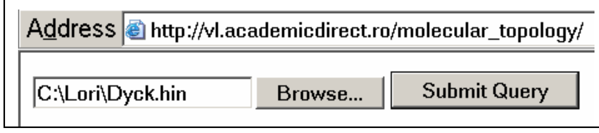
6. PHP LANGUAGE CAPABILITIES

The PHP language has a rich strong functions library, which can significantly shorten the algorithm design and implementation. In the following, some of them (already tested ones) are presented, using sequences of our first program for system information: `$b=preg_split("/[\n]/",$a,-1,PREG_SPLIT_NO_EMPTY);` (split string into an array using a perl-style regular expression as a delimiter), `$c=explode(" ",$b[$i]);` (splits a string on string separator and return array of components), `$a=`uptime`;` (PHP supports one execution operator: backticks - ``). A set of shell execution applications can be used with execution operator to collect information: `$a=`cat /etc/fstab | grep swap`;`

The output data download procedure to the client is achieved via a header function: `header("Content-type: application/octet-stream");\r\n header('Content-Disposition: attachment; filename="'.1_max_cycle.'".txt'");`

7. A PERFORMANCE COUNTER APPLICATION

An application was used as performance counter. It has a web interface:


Address http://vl.academicdirect.ro/molecular_topology/
C:\Lori\Dyck.hin

```
<form method='post' action='hin.php'
enctype='multipart/form-data'>
<input type='file' name='file'>
<input type='submit'>
</form>
```

Fig. 5. Submit form for the hin.php *application*

About application exploiting experience: there is a bug in Microsoft Internet Explorer 4.01 that does not allow header incomings for downloading of the output file. There is no paper devoted to this subject, in our best knowledge. There is also a bug in Microsoft Internet Explorer 5.5 that interferes with this, which can be solved by upgrading to Service Pack 2 or later. Anyway, this problem does not affect our program.

The output data of execution for all three servers is presented in table 1:

Table 1. Statistics of the *hin.php* Program Execution

ip	name	CPU	RAM	mics (start)	s. (start)	mics (stop)	s. (stop)	time (s)
140	j	2P166MMX	128	0.565873	1077348916	0.213418	1077349252	335.6475
211	ns	P2/400	256	0.634248	1077349090	0.432039	1077349225	134.7978
200	vl	P3/800	512	0.623252	1077305562	0.632815	1077305616	54.00956

The data from table 1 allow us to put on a chart the compared results.

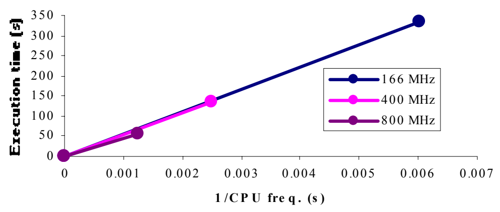


Fig. 6. Execution time vs. 1/CPU frequency

An observation is immediate: the time-time dependence from one generation to another one of Pentiums is almost linear, considering only the usage of pointer, string, and integer instructions (without any floating point instructions). In terms of performance, it means just a speed meter.

Also note that: the usage of dual processor system is not different from the single processor one. A possible explanation comes from the algorithm design, which is classical, not a parallel one. About deviation from linear dependence (Fig. 6 - all lines are draws from 0): it appears that the same real time for the processor is used more efficiently for instructions processing in P II processors in comparison to the P processors and the jump is more obviously at P III architectures.

7. CONCLUSIONS

The feature of boot floppies allows us to install FreeBSD even if we do not have a FreeBSD CD or CD-drive in the system. CVSup mechanism offers an efficient way to maintain and update the system. The kernel recompilation allows us to improve the performance of the system, in terms of speed and memory management.

Looking at hardware characteristics and including corresponding options in kernel are obtained a better exploiting of the hardware resources. More, some specific hardware are then detected and configured for use. Creating a backup copy of the kernel, we can undo any action of kernel reinstalling.

If some service does not start automatically, from unknown reasons (such as *ftpd* on j.academicdirect.ro server), we can try to start manually and after that we can create a script for auto start. Another exemplified situation show that not always the installation scripts puts all required data in configuration files (*AddType application/x-httpd-php .php*) and if a module does not start, a good idea is to look carefully at service configuration file.

The PHP language offers a very good interface with system utilities and an efficient way to develop client-server applications. The application which tests both PHP capabilities and system performance, proves that, even if the constant controlling the number of consecutive calls of a recursive function has big values (like 40 or 50), the program does not crash. The comparative study on the three Intel-based systems showed the qualitative difference among various Pentium processor architectures. Surprisingly (or not), using a dual processor system within an interactive time-sharing operating system does not mean that the system makes parallel processing.

REFERENCES

1. UNIX-like operating system official http sites: debian.org (Debian), www.linux.org (Linux), lindows.com (LinwowsOS), ibm.com/servers/aix/os (AIX), www.gnu.org (GNU), hp.com/products1/unix/operating (HP-UX), opengroup.org (OSF), sun.com/software/solaris (Solaris), www.bsd.org (BSD/OS), netbsd.org (NetBSD), openbsd.org (OpenBSD), freebsd.org (FreeBSD), www.trustedbsd.org (TrustedBSD), picobsd.org (PicoBSD)
2. http://www.bartleby.com/61, The American Heritage Dictionary of the English Language, Fourth Edition
3. http://www.delorie.com/gnu/docs/vera, Virtual Entity of Relevant Acronyms, The Free Software Found.
4. www.freebsd.org/cgi/man.cgi, FreeBSD Hypertext Man Pages
5. www.php.net, The PHP Group (Arntzen T. C., Bakken S., Caraveo S., Gutmans A., Lerdorf R., Ruby S., Schumann S., Suraski Z., Winstead J., Zmievski A.)
6. http://lejpt.academicdirect.ro/data/03/01_40.pdf, Jäntschi L., (2003) Installing and Testing a FreeBSD Server *Operating System, Leonardo Electronic Journal of Practices and Technologies,* 3, p. 1-30.
7. http://server.rotld.ro/cgi-bin/whois?whois=academicdirect.ro, Romanian National RD Computer Network
8. http://www.apache.org, The Apache Software Foundation