DICT MINIX

Secure version of Minix3 without using Swap Space

Amit Jain	amit.jain@iiitb.net
Nesha Rani	nesharani.m@iiitb.net
Samrat V	samrat.vooradi@iiitb.net
Sandeep S	sandeep.s@iiitb.net
V SatyaSai Kishore D	vsatyasaikishore.d@iiitb.net

Technical Report IIITB-OS-2010-01C

April 2010

ABSTRACT

The disk accesses required by common operating systems are a potential security threat when a device (and therefore the data on it) crosses a security barrier. One of the solutions to the problem can be to use an operating system which does not write any data to the secondary storage at all.

We have demonstrated the solution to this problem by creating a special operating system based on Minix 3, which can be loaded from a live CD and which does not use any swap space. We have also removed the Hard-Disk Drivers, so that there is no way the operating system can interact with the secondary storage. A web-server runs on this live CD of Minix, which serves word requests from clients and returns dictionary meanings from the Websters Second Dictionary.

This document gives an overview of the stripped down version of MINIX from a technical perspective. As such, it can serve as a brief documentation to run a web-server with a secure, reliable operating system. Also, it contains a detailed report on the issues encountered while removing hard disk drivers, swap space and running a web-server, so that others may avoid the pitfalls we faced.

Project URL: http://sourceforge.net/projects/miniminix3

Contents

1. Introduction

1.1 Problem Statement 4

1.2 Technical Specification 4

2. Existing Systems

2.1 Gap Analysis 5

3. Architecture Overview

3.1 Architecture of the System 6

3.2 User Perspective of the System 8

4. Client and Server Implementation

4.1 Challenges for DICT MINIX 8

4.2 Approach 8

5. Testing and Results

5.1 Web stress load and performance testing 11

6. Product Overview

6.1 Features of DICT MINIX 12

6.2 Obtaining DICT MINIX 12

6.3 Screen shots 12

7. Conclusion

7.1 Future Enhancements 14

Bibliography

1. Introduction

MINIX 3 is a new open-source operating system designed to be highly reliable, flexible, and secure [1]. MINIX 3 is an operating system on resource-limited and embedded computers and for applications requiring high reliability.

DICT MINIX is a secure operating system built on MINIX 3 with only the Dictionary Server in it. A web-based Dictionary Server with reference to Webster's Second, is provided for the users of DICT MINIX which responds to the client requests and retrieves the meaning. NCSA HTTPD Web Server is provided with DICT MINIX to establish client-server interaction. CGI script handles the dictionary requests from the client and replies the corresponding meaning. An external client has been packaged to run on any operating system.

1.1 Problem Statement

Security threats to operating systems today largely come from the network. The disk accesses required by common operating systems are potential security threat when a device or the data on it cross a security barrier. A far-end preventive approach to this problem would be to develop an Operating System which can be booted from a Live CD and run entirely on RAM without having to use any swap space. Thus we foreclose all access to the hard disk.

Minix3 OS being extremely small, the stripped down version of this OS can run entirely on RAM. We equip it with the resources to host a web-based dictionary server and a client.

1.2 Technical Specifications:

Operating System: MINIX3.1.6
Dictionary: Webster's Second
Server: NCSA HTTPD Web Server
Server Scripting: CGI Script.
Client: JAVA based MINIX Dictionary Client

2. Existing Systems

During the course of this project, we came across various efforts and strategies done in similar directions. Similar efforts and their strategies are illustrated below.

- ✓ Operating Systems like Puppy Linux, Morphix have the ability to run only on RAM, without using swap space.

Puppy Linux (of size 85MB), being so small, usually loads completely into RAM, which accounts for incredible speed. Morphix is also a light-weight OS which runs on a Live CD.

- ✓ Operating Systems with the feature of Live CD

Damn Small Linux is one of the distros which can run on a Live CD and it uses minimal resources. It can fit inside a 50MB Live CD.

- ✓ Operating Systems with a web-based server

The Live CD versions for KnoppixQuake, LAMPPIX, and Devil Linux solely run servers with a size less than 200MB. Dictionary Server Dictd is a TCP based server that allows a client to access dictionary definitions from a set of natural language dictionary databases.

2.1 Gap Analysis

There is no version of MINIX Live CD which runs entirely only on RAM and which don't interact with the hard disk. This customized version of MINIX runs a stand-alone web server with minimal resources. The modified version of MINIX is a small operating system that fits into the RAM also running a light-weight dictionary server on it. Our system not only provides a dedicated dictionary client but also a web interface to query the server.

3. Architecture Overview

During the course of this project, we had several tradeoffs in the design of DICT MINIX system. The following are the components present in DICT MINIX operating system.

- ✓ Kernel Space**:** Kernel defined as in MINIX3.1.6
- ✓ User Space: It consists of device drivers (except for hard disk driver), server processes and user processes.

As can be seen, absence of hard disk drivers is evidently clear in device drivers' layer. The server processes is similar to MINIX.1.6 and the User process consists of a dictionary server.

- ✓ Dictionary Server as a User Process

It is NCSA HTTPD web-server which runs a CGI script to handle http request.

- ✓ External Client: This component represents the client which sends the http request to the dictionary server.

3.1 Architecture of the System

The architecture [5] of DICT MINIX is shown in the Figure (1). It is represented as kernel space and user space. It has only a dictionary web-server running as user process with an external client as shown in Figure (1). This version of Minix does not use any swap space, hence it runs entirely in RAM as hard disk drivers are removed.

The OS has a working kernel, a web-server and device drivers for Ethernet, RAM Disk, display, etc., but not hard disk.

Dictionary Server is highlighted in the architecture diagram Figure (1)

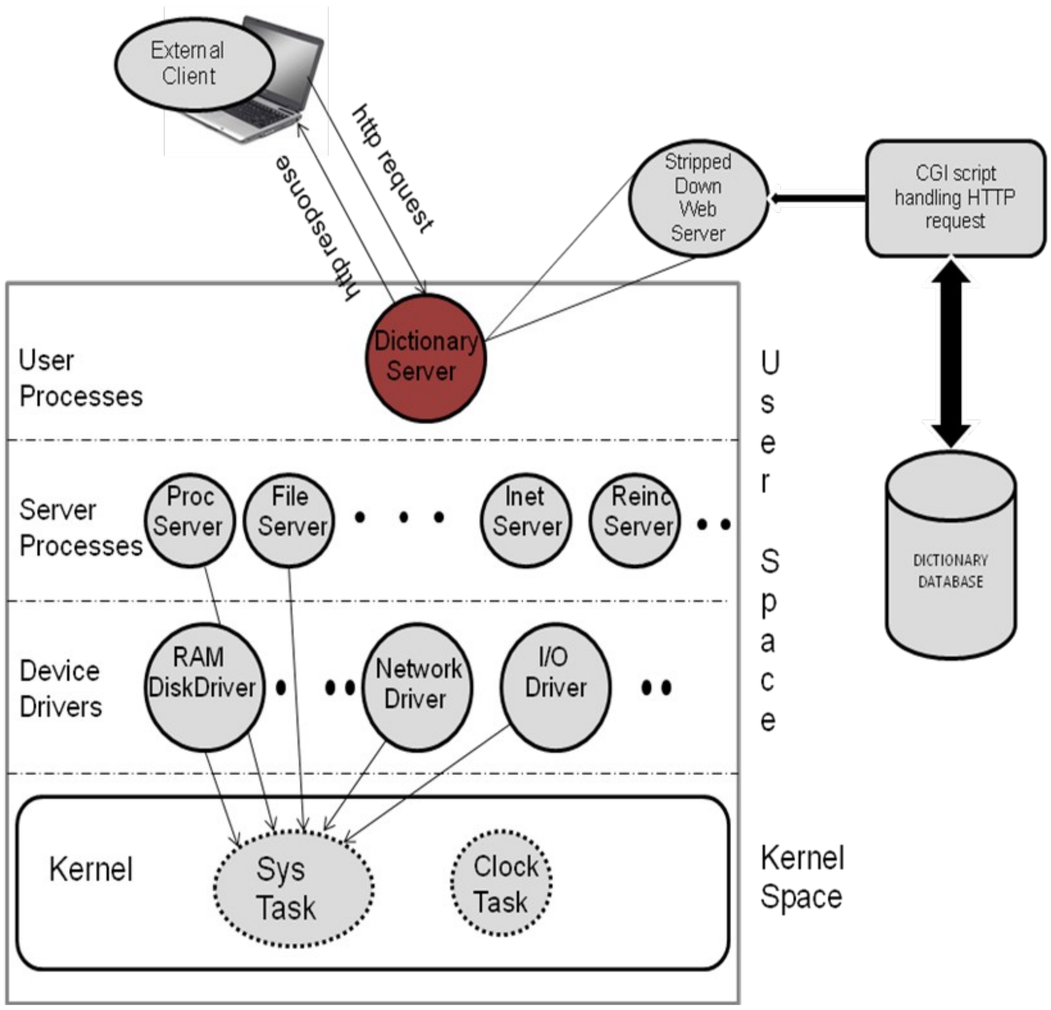


Figure (1): Architecture of DICT MINIX Operating System

The kernel space and user space identifies the processes, servers, drivers that are present in the Minix3 architecture. In addition, dictionary server running as a user process is shown in the figure. It holds a light weight stripped-down web server (NCSA HTTPD server) and a CGI script which interacts with the client and fetches the meaning from the database. The External client is also shown in the figure. The client server interaction via http is also depicted.

3.2 User Perspective of the System

The OS is loaded off a Live CD; it detects all device drivers except for any hard disk driver. With the Dictionary client or the web interface the user can place a request to the server that responds with the meaning.

4. Client and Server Implementation

The server runs on Minix Live CD and the external client is a cross-platform application which is built using Java.

4.1 Challenges for DICT MINIX

- ✓ Hard Disk Drivers Removal: The CD-ROM drives use the same physical connection as the hard disk drives, though the protocol differs - ATAPI & ATA, but since the hardware access are the same, the driver is common. Since CD support (read: ATAPI) is highly integrated with hard drive support (ATA), this task was a major challenge.
- ✓ Apache was the only web-server that was ported to MINIX. As it occupies more space and is a relatively a high-end web server, we had to port other light-weight web servers.
- ✓ Avoid swap space by maintaining all the processes in RAM [5].
- ✓ To run the dictionary server program on the server that accepts the client request and efficiently uses RAM to search from the one lakh plus words.
- ✓ Packaging the web-based dictionary server onto a Live CD.

4.2 Approach

Several approaches have been applied for this implementation. The following are some of the approaches taken to implement this OS.

- ✓ We found a dictionary database [7] from Gutenberg project. We parsed this document to a dictionary flat file (around 13 MB). It has about 1,10,000 words and meanings.

- ✓ Initially we implemented a hash table data structure to optimize the search time but it could not accommodate one lakh plus words but finally we programmed a script to retrieve the meaning for the given search word.
- ✓ Apache web server was initially chosen to cater to the client requests but due to its large size, the search for other light weight servers began. Mongoose, NCSA HTTPD servers were ported to Minix. We have finalized on the NCSA HTTPD server that runs the CGI-script to accept client request, search for the search word, retrieve and respond with the corresponding meaning.
- ✓ A Java based DICT MINIX client is packaged to run on any operating system. Running the Client, JAVA code, a pop-window appears to enter a *search word* and press the button '*search*'; it sends a request to the NCSA HTTPD Server already running in MINIX.
- ✓ For Hard disk driver removal, at_wini driver removal was a tedious task because it holds both the ATA driver (for the Hard Disk) and ATAPI driver (for the CD-ROM) [4]. Thus the part of the code was modified which ensures that only the CD-ROM driver(ATAPI) is being run without the hard disk driver (ATA) driver even being called or accessed.

Removing hard disk drivers

In the file /usr/src/drivers/at_wini/at_wini.c:

Line 637, w_io_test() ,The function w_do_open(dp,m_ptr) checks if a drive is a CD drive and if it is not a CD drive, checks if it is working by invoking the function w_io_test() in this line [3]:

if(!(wn->state & ATAPI) && w_io_test() !=OK){…}

We must remove the "w_io_test() != OK" part so that it just checks that the drive is a CD drive and the ignores the rest of the drives. The line is changed to:

if(!(wn->state & ATAPI)){…}

✓ We have used release.sh [2], and chrootmake.sh to build the Live CD image. The NCSA HTTP server holding the dictionary program and the database was also written into the Live CD.

Packaging the Web-Server and Dictionary Program in the CD

1. Add a few lines to the script /usr/src/tools/chrootmake.sh to make and make install ncsa_httpd. After the boot image gets installed and copied and changing the directory /usr/src/tools, add these lines:

   ```
   $ cd ncsa_httpd_1.4
   $ make
   $ make install
   $ cp –p dictprg.cgi/etc/www
   $ cp dict.txt /home
   ```
2. Copy the unzipped directory ncsa_httpd_1.4 to /usr/src/tools.
   First get the source tar ball of ncsa_httpd and extract it.
3. A few modifications has been made in the conf files of ncsa_http so as to necessitate changes to be made in the dictionary program, database and configuration files such that they are present in the read-write partitions of the CD_ROM.

To resolve the networking problem in Minix LiveCD, the configuration details in inet.conf was modified to call the AMD lance network driver.

Creating the Live CD from the source tree (/usr/src/)

```
$ cd /usr/src/tools
```

In /usr/src/tools/release.sh, set the USRMB variable to around 170, as 170MB will be enough for the CD without the packages

$ ./release.sh –c –p will create the Live CD image without the packages and from the source tree.

-p flag is for no packages

-c flag is used for building from the source tree, i.e., /usr/src/ on the installed system

Removing the setup script

Since, our CD will only be a Live CD and not an installable one; we have to remove the setup script which installs Minix 3 on the hard disk. For this, we do:

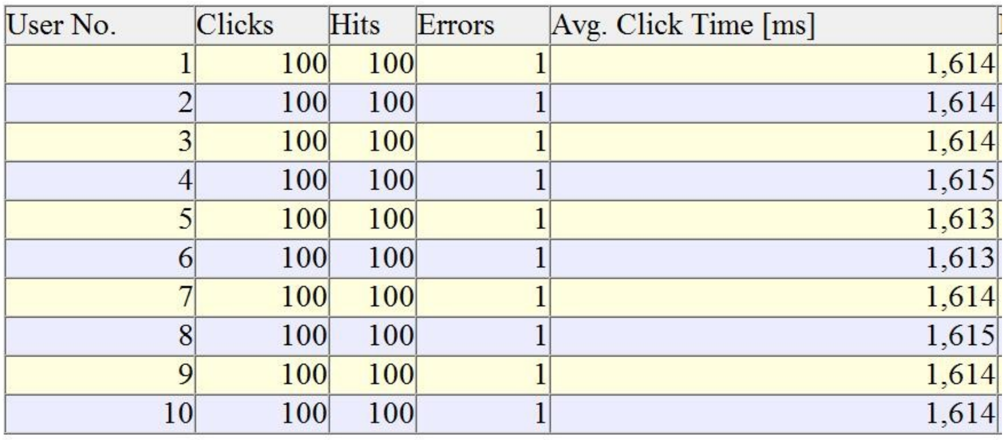
From the file, /usr/src/commands/scripts/Makefile, comment out the lines 42, 140 and 141, which installs the setup program in the new system [6].

5. Testing and Results

Web stress tool has been used to find the load that the server can handle when bombarded with many requests.

5.1 Web stress load and performance testing

The hit ratio, the response times and detailed statistics can be viewed with the help of this tool. The load analysis for the server on simulating 10 clients with each of them placing 100 requests independently was performed. The following results were obtained.



User No.	Clicks	Hits	Errors	Avg. Click Time [ms]
1	100	100	1	1,614
2	100	100	1	1,614
3	100	100	1	1,614
4	100	100	1	1,615
5	100	100	1	1,613
6	100	100	1	1,613
7	100	100	1	1,614
8	100	100	1	1,615
9	100	100	1	1,614
10	100	100	1	1,614

Table (1): Testing results showing error-rate for simulated users

The table reads that on simulating 10 users who on an average place a request on every 1600 ms, tend to receive the response correctly 99% of the time. This simulation is performed for 100 requests from each user.

6. Product Overview

Dict Minix is a Live CD that establishes the fact that Minix is a powerful operating system in the Embedded Systems context. It runs a stand-alone web server entirely on the Live CD without using any swap space. It runs a dictionary server referencing the Webster's dictionary database. A dictionary client application has been developed for the User. It is platform independent, and it has been packaged as a JAR file.

6.1 Features of DICT MINIX

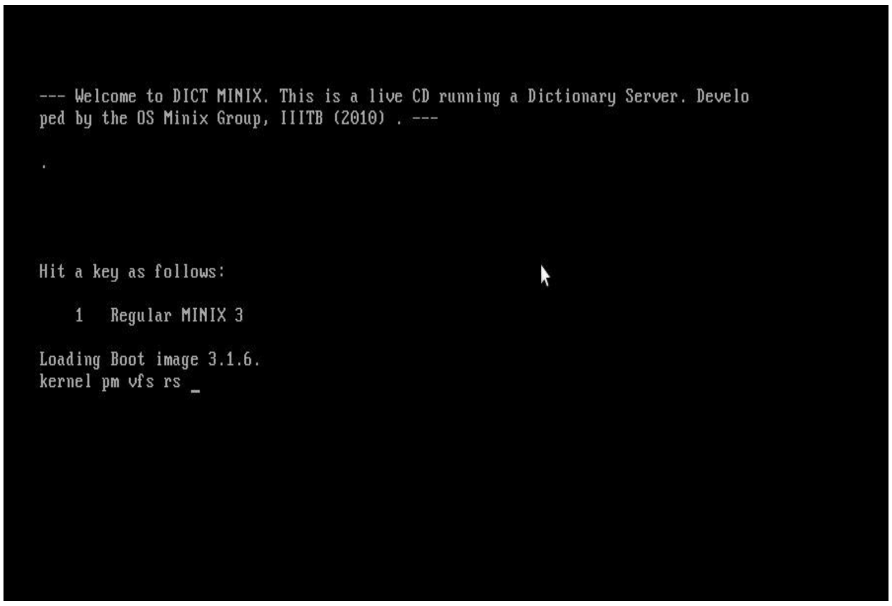
- ✓ POSIX Compliant
- ✓ No device driver for hard disk
- ✓ Networking with TCP/IP
- ✓ Dictionary Web Server implemented as User Process
- ✓ Webster's Second Dictionary database

6.2 Obtaining DICT MINIX

It is free software, can be found at the website http://sourceforge.net/projects/miniminix3 [8] which is an iso image for creating bootable CD version of DICT MINIX. The source code is available at https://miniminix3.svn.sourceforge.net/svnroot/miniminix3. This is the official code repository for SVN.

6.3 Screen Shots

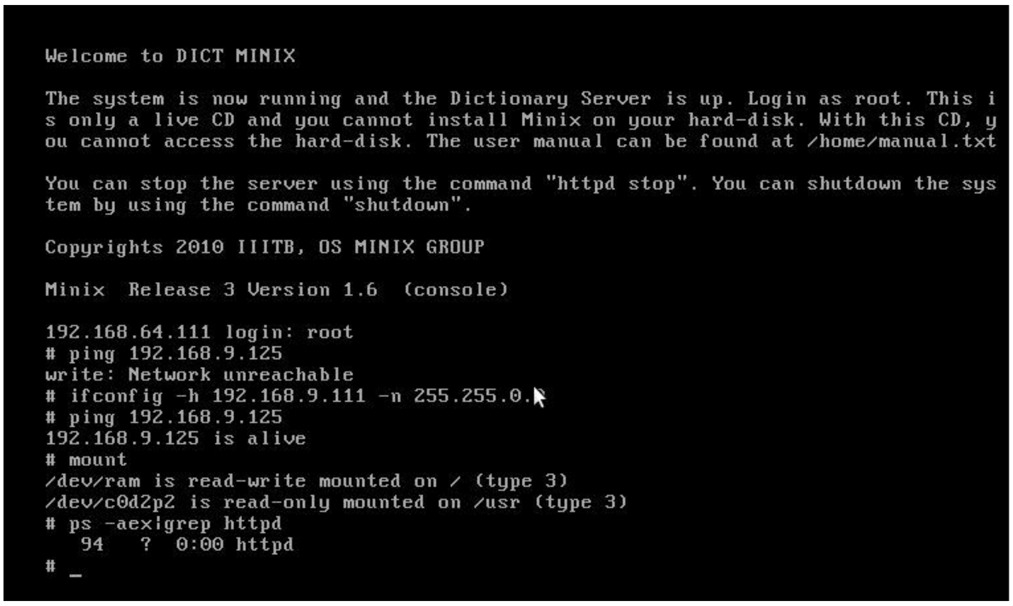
Figure (2) shows the setup script being removed, i.e. the OS cannot be installed from this Live CD.



```
--- Welcome to DICT MINIX. This is a live CD running a Dictionary Server. Develo
ped by the OS Minix Group, IIITB (2010) . ---

.



Hit a key as follows:

    1   Regular MINIX 3

Loading Boot image 3.1.6.
kernel pm vfs rs _
```

Figure (2): Dict Minix booting up

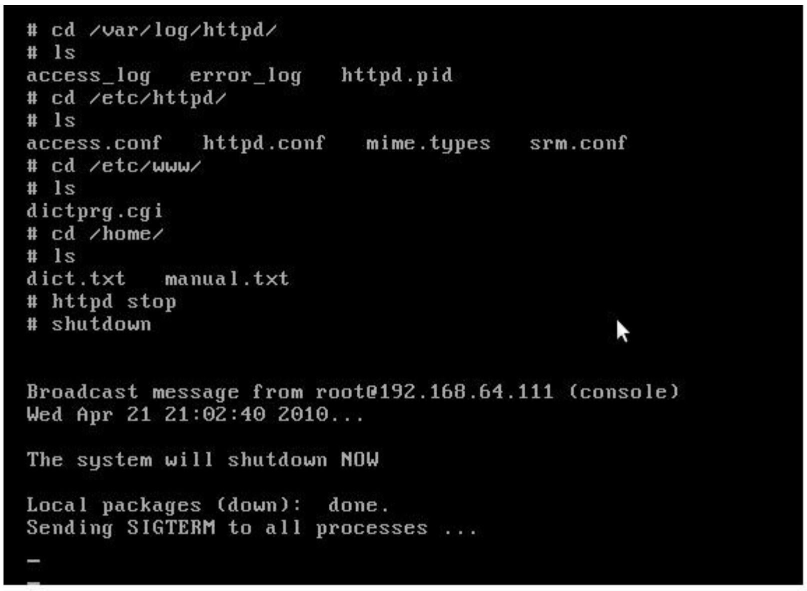
Figure (3) shows that the hard disk cannot be mounted, thereby proving that the Live CD does not use the swap space. The communication via the network is also depicted



```
Welcome to DICT MINIX

The system is now running and the Dictionary Server is up. Login as root. This i
s only a live CD and you cannot install Minix on your hard-disk. With this CD, y
ou cannot access the hard-disk. The user manual can be found at /home/manual.txt

You can stop the server using the command "httpd stop". You can shutdown the sys
tem by using the command "shutdown".

Copyrights 2010 IIITB, OS MINIX GROUP

Minix  Release 3 Version 1.6  (console)

192.168.64.111 login: root
# ping 192.168.9.125
write: Network unreachable
# ifconfig -h 192.168.9.111 -n 255.255.0.
# ping 192.168.9.125
192.168.9.125 is alive
# mount
/dev/ram is read-write mounted on / (type 3)
/dev/c0d2p2 is read-only mounted on /usr (type 3)
# ps -aex|grep httpd
   94   ?  0:00 httpd
# _
```

Figure (3): Drives mounted on Dict Minix

Figure (4) shows the directories where the config files, the dictionary program, and the database is present in the CD file system.



```
# cd /var/log/httpd/
# ls
access_log   error_log   httpd.pid
# cd /etc/httpd/
# ls
access.conf   httpd.conf   mime.types   srm.conf
# cd /etc/www/
# ls
dictprg.cgi
# cd /home/
# ls
dict.txt   manual.txt
# httpd stop
# shutdown


Broadcast message from root@192.168.64.111 (console)
Wed Apr 21 21:02:40 2010...

The system will shutdown NOW

Local packages (down):  done.
Sending SIGTERM to all processes ...
_
_
```

Figure (4): Location of NCSA HTTPD server and configuration files

7. Conclusion

Our goal was to run a web-based dictionary server on MINIX 3 Operating system, which runs on a Live CD without using swap space. The hard disk drivers have been removed. The OS can only mount from the CD, thus the security threat of hard disk being accessed by any malicious program is prevented.

7.1 Future enhancements

1. We are presently using a CGI script to handle the http requests at the server. But we can use a modern scripting language like python or perl.

2. If the database does not have a word, then we can suggest a few words which are close to the queried word. Also, we can search online for the word which doesn't exist in our database and return that result to the user.

3. The client can be improved in such a way that the user has more options like connecting to a different server and the GUI can be improved.

Bibliography

[1] “MINIX 3: A Highly Reliable, Self-Repairing Operating System”, July 2006. [Online]. Available: http://www.few.vu.nl/~jnherder/publications/osr-jul06.pdf. [Accessed: Jan. 26, 2010].

[2] “MINIX 3 Developers Guide”, Oct. 24, 2005. [Online]. Available: http://wiki.minix3.org/en/DevelopersGuide. [Accessed: Jan. 20, 2010].

[3] Jorrit N. Herder,”MINIX 3 Kernel API”, Oct. 20, 2005. [Online]. Available: http://www.minix3.org/doc/kernel-api.ps. [Accessed: Jan.23, 2010].

[4] Andrew S. Tanenbaum and Albert S. Woodhull, *Operating Systems: Design and Implementation*, 3rd ed. Prentice Hall, 2006.

[5] “ Modular System Programming in MINIX 3”,Apr. 2006. [Online]. Available: http://www.usenix.com/publications/login/2006-04/openpdfs/herder.pdf. [Accessed: Jan. 26, 2010].

[6] “ The Official Minix3 google Developers Google Group” Oct. 24, 2005. [Online]. Available: http://groups.google.com/group/minix3 [Accessed Feb23,2010].

[7] “ Gutenberg Project – Link to Webster’s Database” Oct. 01, 1996. [Online]. Available: http://www.gutenberg.org/ebooks/673 [Accessed: Mar. 20, 2010].

[8] “ SVN code repository for DICT MINIX OS”,Jan. 25, 2010.[Online]. Available: http://sourceforge.net/projects/miniminix3 [Accessed: April. 20. 2010].