Design of the Light Weight Linux O.S on the DOM Memory

Seung-Ju Jang

Dept. of Computer Engineering, Dongeui Univ.

Summary

Many people are studying the embedded system. The embedded system becomes a small size device. The DOM memory is using in the mobile device and small size devices. This paper designs the light-weighted Linux O.S that is running onto the DOM flash memory. The embedded system with the DOM must have a light-weigthed O.S due to the memory space restriction. This paper designs a light-weigthed Linux O.S for the DOM memory. The new designed LILO boot loader boots the new designed light-weigthed Linux O.S as a normal Linux O.S. This paper experiments to compare the designed new light-weigthed Linux O.S with a Linux PC.

Key words:
DOM, Light Weight Linux

Introduction

The DOM(Disk On Module) storage type, which is using FDM(Flash Disk Module) semiconductor is a advanced device substution for E-IDE type. The DOM storage device has no seek error, but previous storage hard disk has a seek error. The DOM storage device has no seek error even shaking this device. The DOM storage device size is smaller than previous storage hard disk. This device can be used in the small size devices which are embedded system. The storage device is a kind of flash memory [1, 12, 13, 14].

The flash memory is an advanced non-volatile storage memory substution for the pervious hard disk. The recent flash memory has improved density and I/O performance for the device. It is available not only as a secondary device but also as a storage device for a computer system. The reading performance of flash memory is the same as that of a D-RAM. The writing performance of the flash memory is faster than that of the hard disk.

The size of this storage is less than 30%. And the stability and error rate is more efficient that it is possible to make a mass storage device. The flash memory doesn't need to read data from IO device. The necessary is located into itself. By these features, the flash memory can be used as high performance device [1, 9, 10, 11].

This paper suggests that the Linux O.S is available for DOM(Disk On Module) storage instead of E-IDE. The suggesting idea of this paper can be applied to a small embedded system. The developed system environments are Intel Pentium III 850MHz CPU, 256Mbyte RAM, RedHat Linux 9.0 - Kernel Version 2.4.28. The primary Linux O.S is ported for the embedded system in this paper. The lightweight Linux O.S is running on the DOM memory. The system can just use DOM memory. The DOM memory don't have enough memory, therefore, this paper makes a lightweight Linux O.S. In order to make a lightweight Linux O.S, we analyze Linux directory structure and the role of each directory.

After analyzing directory, the unnecessary directories are deleted, the necessary directories are inserted, and make a Linux kernel image. This working is minimum works to make a small size Linux kernel that is running on the DOM memory. After making a small size Linux kernel, kernel image must be copied onto the DOM memory.

Next we should install LILO boot loader on the DOM memory to make this system as running PC. We experiment and test a performance of the developed lightweight Linux O.S running well.

This paper is composed of followings. Section 2 is a related studies of this paper. Section 3 designs the Linux O.S and developing contents. Section 4 is an experimental result. And finally, Section 5 concludes this paper.

2. Related Research

The section shows the related research of this paper. The embedded Operating System is bootable on the DOM memory. The embedded O.S is applied into several areas. The common features of embedded O.S are real time O.S. Several embedded O.S is being developed for lightweight and real time feature. The real time system classifies with hard and soft. In order to execute task within deadline time, the real time system supports asynchronous signal processing and preemptive scheduling [1,2,3,4,5,15,16,17].

The lightweight embedded O.S is needed for the small size HW device system. Some embedded system has also low battery consumption algorithm. Moreover, the internal architecture supports minimal operation for some results. The commercial products of the embedded O.S are the followings [6,7,8].

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

Table 1. Embedded Operating System

Products	vendors
VxWorks, pSoS	Wind River
VRTX	Ready Systems
MicroC/OS-Ⅱ	Micrium INC.
RTLinux	FSM Labs
Windows CE.NET, XP Embedded, Mobile 2003	Microsoft

The embedded O.S that is stored into the secondary memory is running on the main memory like PC. The advanced embedded system architecture has a DOM memory. Therefore, the classic architecture of the embedded system should be changed. This paper proposes to overcome this problem.

3. Design of the Lightweight O.S

This section explains the design concept of this paper's proposal. The Fig. 1 shows the lightweight Linux O.S system architecture that is designed in this paper.

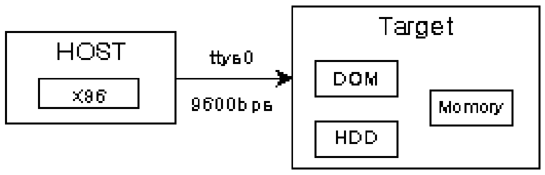


Fig. 1 System Architecture of the Lightweight Linux O.S

The Fig. 1 shows Redhat 9.0 O.S system architecture using DOM memory. The system architecture are divided into HOST PC and Target. The DOM are installed into Target system. The connection between HOST and Target are serial ports with minicom emulator. The design procedure of the lightweight Linux O.S is the following subsection.

3.1 Linux kernel directory structure for Lightweight Linux O.S

The directory structure of the Lightweight Linux O.S is designed in this section. The below Fig. 2 is directory structure for the light-weighted Linux O.S.

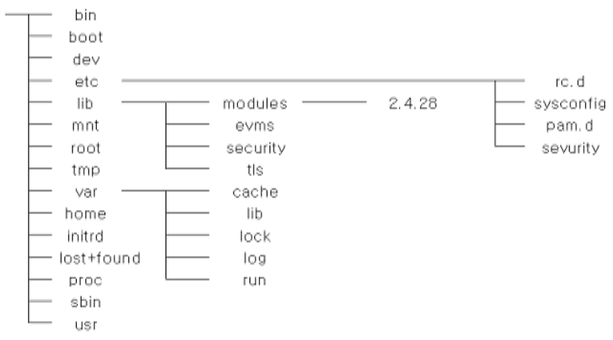


Fig. 2 Directory structure for Lightweight Linux O.S

Each directory has a role as followings.:

/bin folder – The system related command is stored here. The boot related command and general command is stored /bin folder

/boot folder – boot related files are stored boot folder

/dev folder – The device related files are stored /dev folder

/etc folder – The system environments files and directories are stored /etc folder

/lib folder – Each libraries are stored /lib folder

/root folder – home directory of the system administrator

/proc folder – this directory manages processor information, program information, and HW information.

/sbin folder – modules related environments files are located this directory

/usr folder – All Linux's application program and system files are located this directories

3.2 Design of the Lightweight Kernel

When copying a file to a lightweight Linux kernel directory, we use "cp –a" Linux command without changing file permission information.

The needed library is checked by the ldd command that makes Linux kernel image easy building. By using ldd command, we check the needed library for each object, and then we copy that library to a target O.S directory. The copyed library is necessary to execute each object.

We make virtual file system to make a lightweight Linux O.S kernel image. The made virtual file system becomes new root directory by chroot(1M) command. By this mechanism, we can handle command and library easily.

The following is the procedure for making lightweight Linux O.S kernel image.

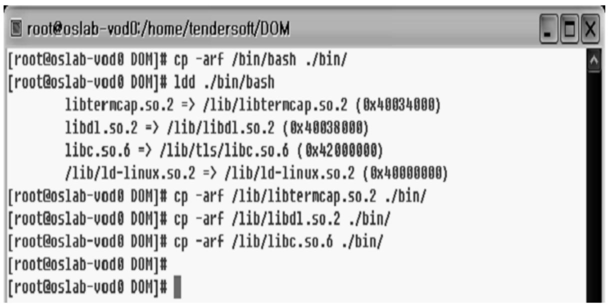
(1) making a VFS File System

(2) copying necessary files or library to the target directories
(3) compiling necessary source files
(4) making a kernel image

3.2.1 Making VFS(Virtual File System)

The virtual file system makes a lightweight Linux O.S kernel image easy by copying necessary commands and libraries to the virtual root directory.

(1) pre-work to make a virtual root file system

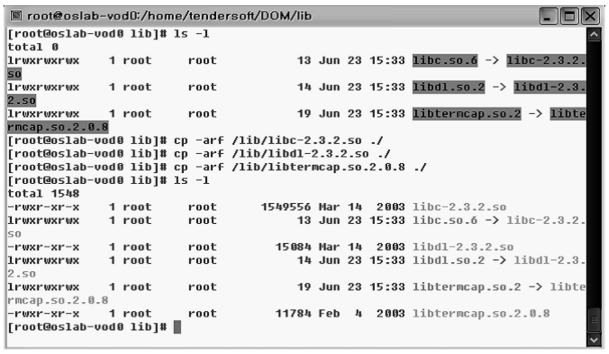


```
root@oslab-vod0:/home/tendersoft/DOM
[root@oslab-vod0 DOM]# cp -arf /bin/bash ./bin/
[root@oslab-vod0 DOM]# ldd ./bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40034000)
        libdl.so.2 => /lib/libdl.so.2 (0x40038000)
        libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
[root@oslab-vod0 DOM]# cp -arf /lib/libtermcap.so.2 ./bin/
[root@oslab-vod0 DOM]# cp -arf /lib/libdl.so.2 ./bin/
[root@oslab-vod0 DOM]# cp -arf /lib/libc.so.6 ./bin/
[root@oslab-vod0 DOM]#
[root@oslab-vod0 DOM]#
```

Fig. 3 copying procedure for bash object files

Fig. 3 is a copying procedure for bash object files. After copying object files to the virtual root directory, it copies library files to the virtual root directory.

A notice point of this case, if the library has a symbolic link, we must copy this symbolic link, too.



```
root@oslab-vod0:/home/tendersoft/DOM/lib
[root@oslab-vod0 lib]# ls -l
total 0
lrwxrwxrwx    1 root     root           13 Jun 23 15:33 libc.so.6 -> libc-2.3.2.
so
lrwxrwxrwx    1 root     root           14 Jun 23 15:33 libdl.so.2 -> libdl-2.3.
2.so
lrwxrwxrwx    1 root     root           19 Jun 23 15:33 libtermcap.so.2 -> libte
rmcap.so.2.0.8
[root@oslab-vod0 lib]# cp -arf /lib/libc-2.3.2.so ./
[root@oslab-vod0 lib]# cp -arf /lib/libdl-2.3.2.so ./
[root@oslab-vod0 lib]# cp -arf /lib/libtermcap.so.2.0.8 ./
[root@oslab-vod0 lib]# ls -l
total 1548
-rwxr-xr-x    1 root     root      1549556 Mar 14  2003 libc-2.3.2.so
lrwxrwxrwx    1 root     root           13 Jun 23 15:33 libc.so.6 -> libc-2.3.2.
so
-rwxr-xr-x    1 root     root        15084 Mar 14  2003 libdl-2.3.2.so
lrwxrwxrwx    1 root     root           14 Jun 23 15:33 libdl.so.2 -> libdl-2.3.
2.so
lrwxrwxrwx    1 root     root           19 Jun 23 15:33 libtermcap.so.2 -> libte
rmcap.so.2.0.8
-rwxr-xr-x    1 root     root        11784 Feb  4  2003 libtermcap.so.2.0.8
[root@oslab-vod0 lib]#
```

Fig. 4 Copy Procedure of Symbolic Link Library

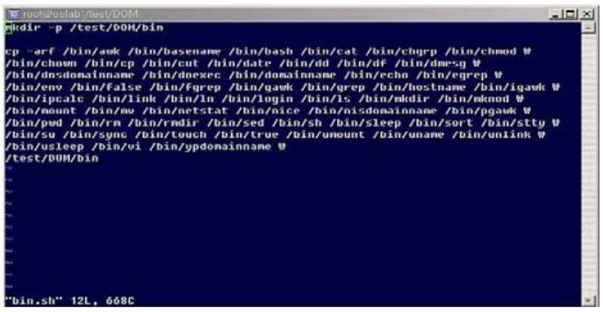
(2) copy necessary files and library to the target directories

The necessary files and library are copied to the target directory. A notice point of this case is that the related files' information should not be changed by using "cp –a" option. In addition, the necessary library files are checked by ldd command. The copying files from /bin directory are the following.

awk, basename, bash, cat, chgrp, chmod, chown, cp, cut, date, dd, df, dmesg, dnsdomainname, doexec, domainname, echo, egrep, env, false, fgrep, gawk, grep, hostname, igawk, ipcalc, link, ln, login, ls, mkdir, mknod, mount, mv, netstat, nice, nisdomainname, pgawk, pwd, rm, rmdir, sed, sh, sleep, sort, stty, su, sync, touch, true, umount, uname, unlink, usleep, vi, ypdomainname

Fig. 5. Copying Files from /bin

The Fig. 5 shows the copying procedure /bin directory.



```
root@oslab:/test/DOM
mkdir -p /test/DOM/bin

cp -arf /bin/awk /bin/basename /bin/bash /bin/cat /bin/chgrp /bin/chmod ₩
/bin/chown /bin/cp /bin/cut /bin/date /bin/dd /bin/df /bin/dmesg ₩
/bin/dnsdomainname /bin/doexec /bin/domainname /bin/echo /bin/egrep ₩
/bin/env /bin/false /bin/fgrep /bin/gawk /bin/grep /bin/hostname /bin/igawk ₩
/bin/ipcalc /bin/link /bin/ln /bin/login /bin/ls /bin/mkdir /bin/mknod ₩
/bin/mount /bin/mv /bin/netstat /bin/nice /bin/nisdomainname /bin/pgawk ₩
/bin/pwd /bin/rm /bin/rmdir /bin/sed /bin/sh /bin/sleep /bin/sort /bin/stty ₩
/bin/su /bin/sync /bin/touch /bin/true /bin/umount /bin/uname /bin/unlink ₩
/bin/usleep /bin/vi /bin/ypdomainname ₩
/test/DOM/bin
~
~
~
~
~
~
~
~
~
~
~
"bin.sh" 12L, 668C
```

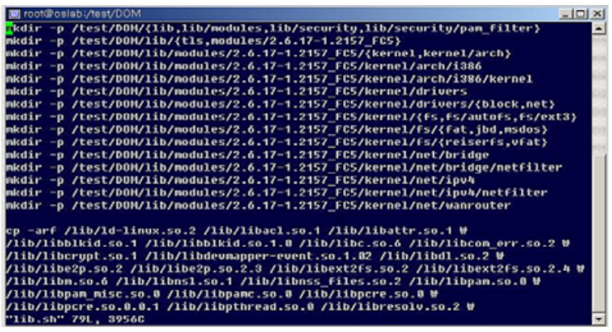
Fig. 6 Copying Procedure Screen for /bin Directory Files

The Fig. 6 shows the copying procedure for /bin directory files.

grub*, System.map-2.6.17-1.2157_FC5, initrd-2.6.17-1.2157_FC5.img, config-2.6.17-1.2157_FC5, vmlinuz-2.6.17-1.2157_FC5

Fig. 7 Copy Files to the /boot Directory

The Fig. 7 shows the copy files to the /boot directory.



```
root@oslab:/test/DOM
mkdir -p /test/DOM/{lib,lib/modules,lib/security,lib/security/pam_filter}
mkdir -p /test/DOM/lib/{tls,modules/2.6.17-1.2157_FC5}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/{kernel,kernel/arch}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/arch/i386
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/arch/i386/kernel
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/drivers
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/drivers/{block,net}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/{fs,fs/autofs,fs/ext3}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/fs/{fat,jbd,msdos}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/fs/{reiserfs,vfat}
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/net/bridge
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/net/bridge/netfilter
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/net/ipv4
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/net/ipv4/netfilter
mkdir -p /test/DOM/lib/modules/2.6.17-1.2157_FC5/kernel/net/wanrouter

cp -arf /lib/ld-linux.so.2 /lib/libacl.so.1 /lib/libattr.so.1 ₩
/lib/libblkid.so.1 /lib/libblkid.so.1.0 /lib/libc.so.6 /lib/libcom_err.so.2 ₩
/lib/libcrypt.so.1 /lib/libdevmapper-event.so.1.02 /lib/libdl.so.2 ₩
/lib/libe2p.so.2 /lib/libe2p.so.2.3 /lib/libext2fs.so.2 /lib/libext2fs.so.2.4 ₩
/lib/libm.so.6 /lib/libnsl.so.1 /lib/libnss_files.so.2 /lib/libpam.so.0 ₩
/lib/libpam_misc.so.0 /lib/libpamc.so.0 /lib/libpcre.so.0 ₩
/lib/libpcre.so.0.0.1 /lib/libpthread.so.0 /lib/libresolv.so.2 ₩
"lib.sh" 79L, 3956C
```

Fig. 8 Copying Procedure to the /boot Directory Files

The Fig. 8 shows the copy files to the /lib directory.

modules*, security*, tls*, ld-linux.so.2, libacl.so.1, libattr.so.1, libblkid.so.1, libblkid.so.1.0, libc.so.6, libcom_err.so.2, libcrypt.so.1, libdevmapper-event.so.1.02, libdl.so.2, libe2p.so.2, libe2p.so.2.3, libext2fs.so.2, libext2fs.so.2.4, libm.so.6, libnsl.so.1, libnss_files.so.2, libpam.so.0, libpam_misc.so.0, libpamc.so.0, libpcre.so.0, libpcre.so.0.0.1, libpthread.so.0, libresolv.so.2

Fig. 9 Copy Files to the /lib Directory

The Fig. 9 shows the copy files to the /sbin directory.

agetty, arp, arping, badblocks, blockdev, consoletype, debugfs, depmod, dumpe2fs, e2fsck, e2image, e2label, ether-wake, findfs, fsck, fsck.ext2, fsck.ext3, getkey, grub, grub-install, grub-md5-crypt, grub-terminfo, grubby, halt, hwclock, ifconfig, ifdown, ifup, init, initlog, insmod, insmod.static, ip, ipmaddr, iptunnel, killall5, klogd, kudzu, ldconfig, lsmod, mii-tool, mingetty, mke2fs, mkfs.ext2, mkfs.ext3, modinfo, modprobe, nameif, netreport, pam_console_apply, pam_tally, pam_timestamp_check, pidof, plipconfig, poweroff, ppp-watch, reboot, resize2fs, rmmod, route, runlevel, service, setsysfont, shutdown, slattach, sulogin, sysctl, syslogd, telinit, tune2fs, unix_chkpwd

Fig. 10 Copy Files to the /sbin Directory

Following above procedure, the lightweight O.S is made and implemented. After making basic running environments, we can make a lightweight Linux image. The making procedure of making kernel image is the following.

3.2.2 Making 64MByte Linux Kernel Image

We are using dd(1) Linux command to make a lightweight Linux kernel image. The usage of dd(1) command is the following.

```
dd if=/dev/zero of=szips.img bs=1k
count=[size]
```

The bs option should be 1k and multiply count option [size] so that we can make 64Mbytes a lightweight Linux kernel image. The minimized Linux kernel is working minimum kernel function.

3.2.3 Making File System for Linux Kernel Image

We make ext3 file system and copy the related files using “ mkfs.ext3 -N 23000” command. This file system is enable to use Linux files in the lightweight Linux system. The -N 23000 option indicates inode number in the Linux file system. The related files should be copied into ext3 file system by using “cp –a” command. We are making final lightweight Linux image.The created ext3 file system can be mounted on a certain folder using “ mount [image name] [mount point] -o loop” command.

We can make a light weight Linux O.S kernel image by this procedure.

The made light weight Linux O.S kernel image is loaded into DOM memory.

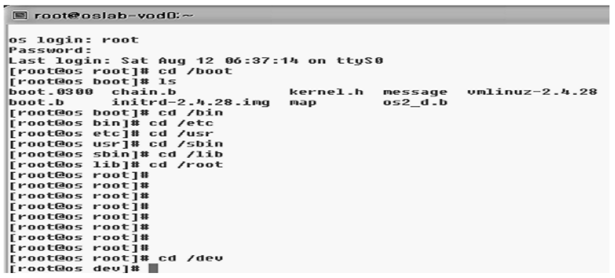
4. Experiments

We use the “minicom” to experiment a light-weight Linux operating system environment in which the experiment is implemented.

4.1 Set the Minicom

We connect a com port of the target system and com1 port of the host system. After that we construct the environment for an o'clock real communication. We use the s - option of a minicom order and set the minicom. We do here the setting up about an o'clock real communication.

The transmission chooses the setup and sets the back of Serial port which is the third environment. An initialization screen of the target system is equal to Fig. 11. if the setting up is completed normally. A minicom setting up is right to become if this screen appears.



```
root@oslab-vod0:~

os login: root
Password:
Last login: Sat Aug 12 06:37:14 on ttyS0
[root@os root]# cd /boot
[root@os boot]# ls
boot.0300  chain.b            kernel.h  message  vmlinuz-2.4.28
boot.b     initrd-2.4.28.img  map       os2_d.b
[root@os boot]# cd /bin
[root@os bin]# cd /etc
[root@os etc]# cd /usr
[root@os usr]# cd /sbin
[root@os sbin]# cd /lib
[root@os lib]# cd /root
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]#
[root@os root]# cd /dev
[root@os dev]#
```

Fig. 11 Screen of connection to the Target System using “minicom”

4.2 Memory Performance Measurement Program Creation and Test.

The experiment about a light weight Linux operating system to propose was performed in this paper. The creation tested the program for a memory performance measurement besides of the experiment. We used a performance benchmark program and also experimented a light weight Linux operating system. A memory performance measurement program is equal to a next Fig. 12.

```
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <stdlib.h>
#define BUFSIZE 2048
#define MAX 999999
int main()  {
  clock_t start;
  int i, j;
  float diff;
  char testTmp[BUFSIZE], buf[BUFSIZE];
  start = clock();
  for(i=1; i <= MAX; i++)  {
    for(j=0; j < BUFSIZE; j++)    {
      testTmp[j] = 'a';
    }
  }
    ..........
  start = clock();
  for(i=1; i <= MAX; i++)  {
    for(j=0; j < BUFSIZE; j++)    {
      buf[j] = testTmp[j];
    }
  }
  diff = diff + (float)((clock()-start)/((double)1000000));
  printf(" Total %4d th execution %.3f sec \n", MAX, diff);
  return 0;
}
```

Fig. 12 Test Program for a Light-Weight Linux Operating System Performance Measurement.

Fig. 12 is a performance measurement test program. This program puts a fixed value in a buffer space. Continuously this value is a movement, deletion, and copy.

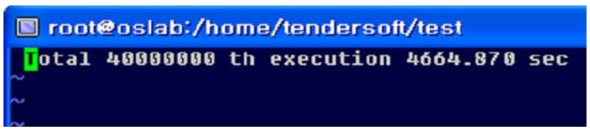
This task is achieved by 4 for loop statement. Each of loop statement measures particularly. We base the time which the recursion measures to carry.

We make 4 for loop statements accomplishing each as Fig. 12. We experimented the format the cumulative makes the price and to rescue a processing hour ultimately.

The Fig. 12 is the procedure which gets the result of the experiment to fill in the program. This program accomplishes 4 for loop statements each for a performance measurement. The execution result of this program is equal to a Fig. 13, Fig. 14. We installed Linux to accomplish this test program in E-IDE system. We also installed Linux operating system in DOM memory.

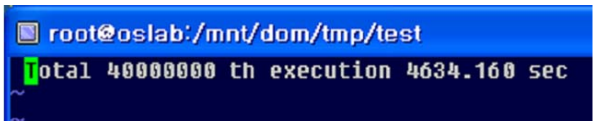
The result of a performance measurement is equal to a Fig. 13., Fig. 14. in two systems. A Fig. 14 is the result of a performance measurement in E-IDE system. The result of a performance measurement came out the 4664 seconds.

Fig. 14. is the result of a performance measurement in the DOM memory. The result of a performance measurement came out the 4634 seconds.



```
root@oslab:/home/tendersoft/test
Total 40000000 th execution 4664.870 sec
~
~
```

Fig. 13 Test Screen on the E-IDE Linux System



```
root@oslab:/mnt/dom/tmp/test
Total 40000000 th execution 4634.160 sec
~
```

Fig. 14 Test Screen on the DOM Memory System

The conclusion of the result of a performance measurement in Fig. 13, Fig. 14 is as follows. The performance of DOM memory with light weight O.S is better than that of the general IDE system. Table 2. shows the result of a performance measurement. This shows the result of a performance measurement in a memory E-IDE system and DOM system.

Table 2 Result of Performance Measurement in Two systems

Experiment methods	E-IDE (Hard Disk)	DOM(Disk On Module)
Write a data in BUFSIZE	28.850	28.855
Write, copy, delete in BUFSIZE	115.760	115.750
For loop statements	4664.870	4634.160

A performance measurement program is for the first time to record simply at BUFSIZE as Table 2. In this case, we can know the DOM Linux operating system performance is some bad. The next, when we eliminate, reproduce, and copy the record data at the buffer, we know a performance difference among the two systems.

We used four times FOR loop and experimented this task finally. This is the performance measurement result which is accomplished in the paper. Existing action of E-IDE on the Linux operating system is faster in a hardware than Linux operating system of DOM memory. But, in this paper, The reason is inside the DOM memory's performance of the light-weight Linux operating system is good because the kernel is made as light-weight.

5. Conclusion

This paper proposes the design of a light-weight Linux operating system. It is that the existing Linux operating system was modified to the center of an essential module in the DOM memory. The module of this a light-weight Linux operating system which is designed in the paper resides within the DOM memory and is operated to an embedded system privately. The proposing light-weight Linux operating system in the paper has the basic facility of the kernel such as process, file system, memory, and IO management.

We accomplish the light-weight commands and the library task which the user uses as the kernel. After finishing such design task finished, we construct a simulated file system. We accomplish a necessary operating system image setup task for actual system. We use a “dd” Linux command for a 64KB Linux image setup. After we go via such course, a light-weight Linux operating system image generation is finished. We use this image. We run a light-weight Linux operating system image which actual DOM is produced within the memory. We can do the booting course to the DOM memory. The booting after we install Lilo becomes the finish if it becomes this task. Here the task can use DOM at general PC.

To measure the performance of this a light-weight Linux operating system which proposes in the paper, we fill in a benchmark program for a performance measurement. We measured a light-weight Linux operating system performance which uses the proposing program. The result of the experiment. DOM memory's Linux O.S performance is good.

Acknowledgment

This work was supported by Dong-eui University Grant.(2007AA181)

References

[1] Linux Kernel Programming, ADDISON WESLEY, Beck, 2002.
[2] Korea Embedded Linux Project, http://www.kelp.or.kr
[3] Syed Mansoor Sarwar, "Unix - A Textbook 2nd Edition", RADDISON WESLEY, Aug. 2004
[4] Daniel P.Bovet, Marco Cesati, "Understanding the LINUX KERNEL", O'Reilly, 2001.
[5] Preston, W. Curtis, "Unix Backup & Recovery", Oreilly & Associates Inc, Nov. 1999.
[6] Karim Yaghmour, "Building Embedded Linux Systems", O'Reilly, 2003.
[7] Skawratananond, Chakarat, "Unix to Linux Porting", Prentice Hall, Apr. 2006.
[8] W. Richard Stevens, "Advanced Programming in the UNIX(R) Environment", Addison-Wesley, Jun. 1992.
[9] W. Richard Stevens, "Unix Network Programming: The Sockets Networking API(Updated)", Addison-Wesley Professional, Nov. 2003
[10] W. Richard Stevens, "UNIX NETWORK PROGRAMMING VOLUME 2,2/E", Prentice Hall, Aug. 1998
[11] Maurice J. Bach, "The Design of the UNIX Operating System", Prentice Hall, Feb. 2000.
[12] Worldwide Embedded Operating Environments Forecast, 2003-2007, IDC #29308,2003,5.
[13] Eric S. Raymond, "Art of UNIX Programming", Addison-Wesley, Sep. 2003
[14] Syed Mansoor Sarwar, "Unix - A Textbook 2nd Edition", RADDISON WESLEY, Aug. 2004
[15] Preston, W. Curtis, "Unix Backup & Recovery", Oreilly & Associates Inc, Nov. 1999.
[16] Worldwide Embedded Operating Environments Forecast, 2003-2007, IDC #29308,2003, 5.

Seung-Ju, Jang received a B.Sc. degree in Computer Science and Statistics, and M.Sc. degree, and his Ph.D. in Computer Engineering, all from Busan National University, in 1985, 1991, and 1996, respectively. He is a member of IEEE and ACM. He has been an associate Professor in the Department of Computer Engineering at Dongeui University since 1996. He was a member of ETRI(Electronic and Telecommunication Research Institute) in Daejon, Korea, from 1987 to 1996, and developed the National Administration Multiprocessor Minicomputer during those years. His current research interests include fault-tolerant computing systems, distributed systems in the UNIX Operating Systems, multimedia operating systems, security system, and parallel algorithms.