Annales UMCS Informatica AI 5 (2006) 29-36

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Linux based Live CD on optical disks

Michał Chromiak[1], Andrzej Góźdź[2*]

[1]*Institute of Mathematics, Maria Curie-Skłodowska University,*
Pl. M.Curie-Skłodowskiej 1, 20-031 Lublin, Poland
[2]*Institute of Physics, Maria Curie-Skłodowska University,*
Pl. M.Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

We present an introduction to LiveCD system remastering based on a prospective UnionFS filesystem.

1. Introduction

Regarding increasing popularity of the LiveCD distributions[1] of a Linux system we compile the essential knowledge of how the LiveCD works (based on many rather scattered documents [1-8]) to facilitate preparation of the specialized distributions of systems based on this idea. We describe one of the most efficient ways of system architecture managing using UnionFS. All steps of the procedure were tested using laptops which have often more exotic hardware than desktop computers.

The possibility of booting a PC from a CD/DVD ROM has raised several possibilities, including the most important one, namely, the possibility of running the required system or programs independently of the actual operational system, configuration, and resources of the PC. The self-configuring CD/DVD ROMs can be also useful for administrative purposes and as the rescue disks. The LiveCD distributions allow for cheap and effective use of the PCs as the special servers, routers and other active elements over the network. This idea of the LiveCD systems is based on the possibility opened by the El Torito extension to the standard ISO 9660 system and the Linux kernel capabilities.

[*]Corresponding author: *e-mail address*: gozdz@tytan.umcs.lublin.pl

[1]The self-starting CD or DVD distributions of the Linux system which do not use the operation system which is actually installed on the computer are referred to as the LiveCD systems.

2. El Torito

"ISO 9660" is the standard that defines a file system for CD/DVD ROM media. There are three levels of the ISO 9660 system specification:

- Level 1: The file names have the 8.3 (like in DOS) format typed in the upper-case. The numbers and underscores are also allowed. Maximum depth of the directories is 8.
- Level 2: The file names can have up to 31 characters.
- Level 3: It the multisession mode is allowed (the possibility to add new sessions to the existing ones) on optical disks.

The "El Torito", which is totally compatible with the ISO 9660 specification, was created by the Phoenix Technologies and IBM. It was designed as a standard for creating bootable CDs and DVDs on the IBM compatible computers. Thanks to the BIOS procedures the "El Torito" standard allows the direct access to the CD/DVD ROM drive from the BIOS procedures. This concept enables the possibility of placing the floppy image at the CD/DVD filesystem. The pointer to this image is hidden inside the ISO 9660 filesystem header. Hence the BIOS has the access to this image which, in turn, allows to proceed with the boot process as if the floppy disk would be used.

To accomplish this procedure, the installable Boot CD/DVD ROM feature (using one of CD/DVD ROM configuration types described in Figure 1) makes available to the BIOS one of the two INT 19 functions:

1. Single-Image INT 19. This feature, if instructed to boot from the CD/DVD ROM, does the following:
   - It accesses the Booting Catalogue in the CD/DVD ROM header.
   - It verifies the existence of a boot image on the CD/DVD ROM.
   - It reads the Initial/Default Entry and boots from the disk image specified in this entry.

   The Single-Image INT 19 knows nothing about multiple-images. It does not also know about their possible entries listed in the Booting Catalogue.
2. Multiple-Image INT 19. This feature, if instructed to boot from the CD/DVD ROM, does the following:
   - It accesses the Booting Catalogue in the CD/DVD ROM header.
   - It verifies the existence of a boot image on the CD/DVD ROM.
   - It boots from either the image specified in the Initial/Default Entry or from one of the other images listed in the Section Headers and the Section Entries that follow the Initial/Default entry. The selection of which image to boot from depends on the Selection Criteria.

As it can be seen from the above considerations the Live CD/DVD systems start using only the basic properties of the hardware of the IBM PC like computers. It allows to use them on any computer with this architecture (there is also possibility to generalize this idea to other architectures).

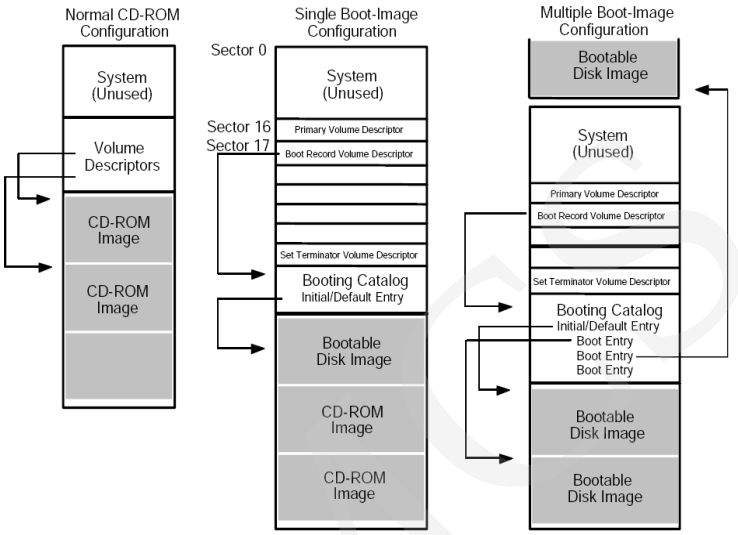

Fig. 1. Three types of the CD/DVD-ROM configurations

3. Live CD remastering[2]

The value of the LiveCD Linux systems is in their exibility which allows to prepare the specialized system on demand.

3.1. Hardware requirements regarding the workstation used to ISO 9660 CD/DVD preparation

In the case when the workstation running a Linux system, which we want to use for preparing the LiveCD, does not come up with the 1GB RAM+SWAP (this information can be found in /proc/meminfo) of free memory. The problem can be solved by creating an additional e.g. 750 MB swap file in the `/mnt/temp/knx` directory as follows:

```
mkdir /mnt/temp/knx
cd /mnt/temp/knx
dd if=/dev/zero of=swapfile bs=1M count=750
mkswap swapfile
swapon swapfile
```

Data stored in `/proc/swaps` provides the necessary information about the amount of the available swap space. Minimum 1GB of free RAM + SWAP memory and approximately 4GB of free disk space are needed.

[2]The procedure described in this section is based on one of the most popular and developed LiveCD distribution, called Knoppix, however the general idea is common to most of this type operating systems.

3.2. "Remastering"

The receipt described below uses the Knoppix distribution – version 4.0 as the pattern [9]. It is important to notice that all the commands required for creating of the operation system on the CD/DVD, known as the "remastering", have to be executed with the root privileges. In principle it is possible to use the fakeroot environment, however it requires much more preparations.

First we have to prepare the appropriate directories structure for the temporary and final files.

As the first step one needs to create the main directory called here knx typing:

```
mkdir /mnt/temp/knx
```

and the following two temporary directories:

```
mkdir -p /mnt/temp/knx/master/KNOPPIX
mkdir -p /mnt/temp/knx/source/KNOPPIX
```

Next, one has to mount the image of the Knoppix distribution to our system or to boot the computer from the LiveCD Knoppix system itself. The former can be done making use of the loop device:

```
mkdir /mnt/knoppix && mount -o loop -t iso9660 n
/path/to/_le/knoppix.iso /mnt/knoppix
```

and copy the directory /KNOPPIX to the directory /mnt/temp/knx/source/KNOPPIX:

```
cp -Rp /KNOPPIX/* /mnt/temp/knx/source/KNOPPIX
```

In the case when using natively different distribution to remaster the ISO 9660 image one needs to mount the LiveCD with the appropriate paths modification.

In addition, one needs to copy the directory /cdrom/boot, the index file and other files except the KNOPPIX file (approx. 700 MB), which will be essential during creation of the ISO 9660 image:

```
cp -ar /cdrom/boot /mnt/temp/knx/master/boot
cp /cdrom/index.html /mnt/temp/knx/master/
cd /cdrom && find . -size -10000k -type f \
-exec cp -p -parents '{}' /mnt/temp/knx/master/ \;
```

Then one has to change the root directory of our system using the chroot environment: chroot /mnt/temp/knx/source/KNOPPIX

In this stage one can make required changes adding new files, changing or removing unneeded programs and so on. The required portion of the software should be prepared somewhere on the hard disk or one can obtain it from the

internet either within the running Knoppix or using the native linux system of the computer.

Before leaving the chroot environment (Ctrl+d or exit) it is mandatory to unmount /proc and remove the redundant files such as .bash_history, tmp, /root/* and the directory .rr moved:

```
rm -rf /mnt/temp/knx/source/KNOPPIX/.rr moved
```

Next step of the procedure is creating the image. In the case of lack of the kernel module *cloop* in the host system, it is necessary to install package *cloop_utils* or to use *create_compressed_fs* tool that can be found in the Knoppix /usr/bin directory.

Having all the required tools it is possible to create a compressed file *ibiza* which will be the desired ISO 9660 filesystem:

```
mkisofs -R -U -V "Conference IBIZA 2006" \
-publisher "www.mojknoppix.pl" -hide-rr-moved \
-cache-inodes -no-bak -pad \
/mnt/temp/knx/source/KNOPPIX – nice -5 \
/usr/bin/create_compressed_fs - 65536 > \
/mnt/temp/knx/master/KNOPPIX/KNOPPIX
```

In the end, to create the ISO 9660 image ready to be burned on the optical disk, we should apply the following command:

```
mkisofs -pad -l -r -J -v -V "IBIZA" -no-emul-boot \
-boot-load-size 4 -boot-info-table -b \
boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat \
-hide-rr-moved -o /mnt/temp/knx/ibiza.iso \
/mnt/temp/knx/master
```

This results with *ibiza.iso* file in the directory /mnt/temp/knx. It can be burned on the optical disk making use of any standard program like k3b.

4. UnionFS – adaptation with Live CD

UnionFS (stackable unification file system) is a filesystem (working as a kernel module) that combines the contents of several directories (*branches*) without changing their actual physical content. It allows any mix of read-only and read-write branches, as well as insertion and deletion of branches on the fly. For example, to join two ro directories into “union” we should write:

```
mkdir /mnt/union; modprobe unionfs
mount -t unionfs -o dirs=/mnt/cdrom1=ro:/mnt/cdrom2=ro unionfs \
/mnt/union
```

UnionFS can be managed using a tool created during UnionFS compilation called *unionctl*. It allows for:

- Listing the unions: `# unionctl /mnt/union -list`
- Adding new branches to the union: `# unionctl /mnt/union -add \`
  `-after /mnt/cdrom2 {mode ro /mnt/cdrom3`
- Changing the content of the branches: `uniondbf -g /mnt/union`
  In the case of joining the read-only directories the union itself stays in the read-only mode until the first read-write branch is added. In that case all the changes are held in the leftmost branch in the list using the *copy-up* method.
- The file deletion which is done by using one of the three available modes:
  WITHOUT – (the default) inserts a .wh file to mask out a real file.
  DELETE ALL – all instances of a file from all branches are deleted.
  DELETE FIRST – only the first instance of the file is deleted.

Another operation, so called copy-up method is a union file changes management method. It is clear that the files from the read-only branch, by definition, can not be modified. Therefore, they are copied to branches with the higher priority and with the read-write mode. This concept allows to modify files. The modified files stay in the read-write branch.

To add a read-write branch as a branch with the highest priority in previous example one needs to write:

```
# unionctl /mnt/union -add -before /mnt/cdrom1 -mode rw /mnt/changes
```

As a result all the changes will be found in `/mnt/changes` and the union will look like this:

```
# unionctl /mnt/union {list
/mnt/changes (rw)
/mnt/cdrom1 (r-)
/mnt/cdrom2 (r-)
/mnt/cdrom3 (r-)
```

The best documentation for the UnionFS can be found in [10].

5. UnionFS and the SLAX filesystem

To boot a Linux operating system from an optical disk there are three major things we should have.

First of all we need to have the kernel image. Usually it is the file *vmlinuz* placed either in the root or `/boot` directory of the Linux system.

Second, we should prepare the image file of the root system e.g., initrd.gz, containing among others, the initial program *linuxrc*.

Finally, we need the bootloader like lilo or GRUB loading program, that reads the kernel to the memory and runs it.

A good example of a distribution that uses UnionFS is the Slax. Slax uses *isolinux.bin* as its loading program. It consists of drivers for the filesystem on the CD so it can send the Linux kernel (*vmlinuz*) and the root filesystem (*initrd.gz*) from CD/DVD to the memory. After that the Linux kernel creates inside a computer's RAM a virtual drive (called ramdisk), where the `initrd.gz` is unpacked and mounted as the root filesystem. Initial ramdisk used in SLAX is only 4.4 MB big and contains only basic software and drivers needed to handle SLAX startup.

The `/linuxrc` file, placed in the ramdisk, is started by the Linux kernel as the initial program that firstly creates an empty union in the `/union` directory. Next it mounts the temporal file system – *tempfs*[3] – in the `/memory` directory and it creates the directory `/memory/changes` (the new directory is in *tempfs* not in the ramdisk). The union is mounted to the `/union` directory and the directory `/memory/changes` is added to it as a read-write branch.

Next the linuxrc mounts all the CD/DVD drives and the hard disks and search for the file livecd.sgn (CD-ROM itself is mounted on the directory `/union/mnt/`). All the necessary images **.mo* from SLAX CD are mounted to the directories `/memory/images/*.mo` and all these mountpoints are, by using *unionctl*, added to the union as individual read-only branches. They are added after the first branch (using rw *changes*).

Now we are in the position that all the modules are added to the union and one `rw` branch is at the top of it. In the meantime the command `pivot_root` is executed.

In the next step of the procedure the original root is moved to the `/mnt/live` and the `/union` directory becomes a new *root* directory.

Finally, linuxrc executes `/sbin/init` which will start all the system services and display login screen as usual.

This results with the model described as shown in Figure 2.

In this way, we got the working, self-consistent system on the CD/DVD. It contains the required software for a specialized projects which can work on the PCs independently of their operational systems and installed resources. In this way, one can also construct the general/specialized system which can convert the PC into a special purpose device.

In general, one can try to construct the multiboot system which could be able to use the memory devices in different computer architectures, booting the appropriate Linux kernels. It means that, the LiveCD idea can serve as a prototype of the way of the device independent distribution of projects in informatics.

[3]A file system which keeps all files in virtual memory.

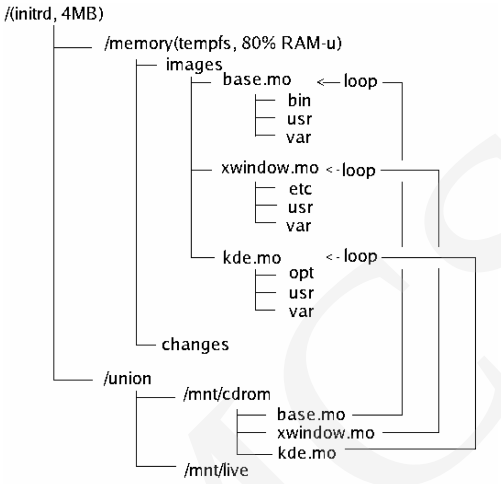

Fig. 2. Logical schematics of the UnionFS in Slax

References

[1] http://littlesvr.ca/isomaster/resources/JolietSpecification.html – Joliet specification.
[2] http://littlesvr.ca/isomaster/resources/RRIP-112.ps – Rock Ridge.
[3] http://littlesvr.ca/isomaster/resources/SUSP-112.ps – ISO 9660 "system use fields" guide line document.
[4] http://littlesvr.ca/isomaster/resources/ECMA-119.pdf – ECMA-119.
[5] http://littlesvr.ca/isomaster/eltoritosuppl.php - El Torito specification unofficial supplement.
[6] http://people.freebsd.org/~daichi/unionfs/ – implementation of the filesystem for FreeBSD.
[7] http://www.filesystems.org/project-unionfs.html
[8] http://aviostechnotalk.com/wordpress/index.php/2005/06/28/filesystemsnapshots-with-unionfs/
[9] http://www.knoppix.net
[10] Tomas Matejicek http://www.unionfs.org/