IGLib IGShell NeurApp

A How-to Guide on Preparation of Bootable Media and Images

This is work in progress.

By Igor Grešovnik , April 2016

This guide provides you with basic knowledge needed to convert bootable media from one form to another (e.g. from an ISO image to a USB flash drive or from a disk to an ISO image), create custom bootable media for different operating systems (with added custom tools and data by your choice), and test bootable media. This is a practical guide that will concisely illustrate how you can perform various tasks.

You will probably want to skip directly to the topics of your interest, therefore a table of contents is displayed below. If you have questions or comments, please post them on my blog site:

https://ajgorhoe.wordpress.com/2016/04/19/a-how-to-guide-on-preparation-of-bootable-media-and-images/

Contents:

1 Why Might You Need to Create Bootable Media

There are numerous reasons why you might need to create bootable media by your own. One common situation includes troubleshooting your computer when the system does not boot – either due to a virus infection or disk and other hardware errors. With suitable bootable media containing the necessary tools, you might boot computer from a USB flash drive or a DVD, try to diagnose the problems and check if at least some data can be saved, and restore the operating system, installed software and user files from backups.

Another reason may be testing, distribution or presentation of software. You may have developed some specialized software that depends on a complex system configuration and a number of other softwares installed. With custom bootable media you can produce a self-contained environment that contains everything our software needs to execute properly. By packing everything to a custom bootable media, you can use this to boot any computer and run software on it, regardless of what is currently installed on the computer (regardless even of the operating system) because all the environment and configurations are provided on your media and you may be relatively sure that your software will run as expected. This is ideal for demonstrating the software at your client site where you have no influence on existing software environment, or for distribution of demonstrative versions of the software. You can also test the software on different environments by using virtualization software such as VirtualBox, for which you may want to prepare a number of pre-compiled environments, each contained on its own bootable media.

There are a number of issues you may encounter when preparing such media, especially in relation of the proper formatting and preparation of contents. This guide provides a how-to approach to solving such issues and contains information about:

- How to properly prepare various bootable media (ISO image files, USB sticks, CDs and DVDs) in such a way that they will actually work on the intended system.
- How to convert existent bootable media from one form to another.
- How to add your stuff to the media.
- How you can test the prepared media.
- Where to obtain some bootable media that you can use as basis to create your own.

1.1.1 Remark on Software Downloads and Security

This guide makes use of many free software. One should always be cautious about downloading software from the internet. There are many download sites for free software that bundle downloaded software (or installers these sites force you to use) with adware, other various crapware or even malicious software. It is instructive to read this article about freeware download sites, and this article provides some additional information about download sites you should be aware of.

When you download freeware, it is advised to **download** it **from the original developers' web sites**. Some software is not even available from original developer's site and is only distributed through various download sites like SoftPedia, Softonic or download.com. When it is, search results on Google may be obscured with links to these sites and it is difficult to locate the original software's web site. You should also beware of spurious sites that mimic certain software site and trick you into downloading their malicious software. You can increase safety by **reaching the original** software's **sites from some trusted source** such as Wikipedia or review articles published on trusted sources. Of course, this is also not 100% assurance of safety. Wikipedia can be edited by anyone and if somebody publishes a false external link on some page, this may not be discovered immediately by other, well-intentioned Wikipedia editors. Authors of software reviews can themselves be tricked into publishing false links, some sites seize to exist and their domain names may be taken over by ill-intentioned people, etc.

Therefore, additional advice is, **whenever you download software from the internet, scan it for malware before installing or running it**. Use antivirus software, and check it with more than one. You may already have some antivirus software installed, but it doesn't hurt downloading an additional one. You can use e.g. MalwareBytes and ClamWin, which usually don't interfere with other antivirus software and you can use them in conjunction with the software you already have installed.

VirusTotal is a tool that is **very convenient for checking downloaded software**. It is an online service that analyzes web sites and files and checks them for viruses, trojans and other kinds of malware. Before downloading software from a site, you can check the site itself by copying its address to the "URL" tab. After downloading a file to your computer, you can check the file under the "File" tab. Clicking "Choose File" lets you choose the file and clicking "Scan it!" performs the scan. The good thing with VirusTotal is that files are analyzed with more than 50 antivirus programs, which reduces the possibility that a real threat is missed. In case of false positives, this gives you the opportunity to suspect that false detection was in place. If only one or two scanners report a threat and other fifty do not, you can make a judgment that the file is safe in spite of the one or two tools that reported a threat, especially if there are other hints that indicate so. To be sure, you can further investigate (e.g. search Google about particular threats reported, etc.).

Tools for detection of malware and other threats are never 100% reliable. If you experiment a lot with different software, it is a good idea to do it **on a snapshot of a virtual machine** (Section 4.2.4). After testing is done, you can simply discard that snapshot and no damage is done if it is infected by malware. Eventually you will still install some software on your real computer, and there is a chance it gets infected. Many times infections can be efficiently treated with antivirus software, but in some cases the best solution is to delete and reinstall computer disks. For such cases, an image of your system disk at hand is very useful. Instead of tedious reinstallation of the system and all the data and software you need for your work, you can do it in a single swing by **restoring the system** (and other) disks to a state that is known to be without problems. For this, you will need a suitable backup software, a disk image you've created previously by this software, and a bootable medium to start a provisional operating system to be able to run the backup software that will restore the disk image (Section 4.1.1).

2 Creating and Using Bootable Disks, USB Flash Drives and Disks

Creating a bootable disk is not just copying files from some source to where you want to create the media. This is because in order for a disk (or other media) to be bootable, it must be physically prepared in some standard form such that computer's hardware knows how to run the system files on it. In particular, there must be a machine code at a particular standardized hardware address (in the "boot sector") that is loaded to RAM and executed by the built-in firmware (traditionally contained in ROM **BIOS**), and this code further loads runs the operating system.

Different media have different arrangement of data in blocks (e.g. hard disks usually have sectors of 512 bytes, some newer have sectors of 4096-bytes while CDs and DVDs have sector length of 2048 bytes).

Further complication is that that there are two main standards for how the boot are is organized and where the loader for the operating system is found . Traditionally, desktop computers and laptops (or IBM PC-compatible in general) used Master Boot Record (**MBR**), a special boot sector at the beginning of the storage device. MBR holds information on how partitions are organized and code (the boot loader) that loads the operation system by passing control to the second stage. On newer computers, the MBR-based partition scheme is being replaced by the GUID Partition Table (**GPT**) scheme, which can coexist with MBR (e.g. for backward compatibility). BIOS is being succeeded by the Unified Extensible Firmware Interface (**UEFI**) that addresses its technical limitations (MBR is limited to 4 primary partitions or 3 primary and one extended partition, and to 2 TB storage capacity). New PCs predominantly ship UEFI firmware. Legacy (BIOS) boot is still supported on many systems but not all of them (e.g. Microsoft Surface Pro-s don't support legacy boot), therefore you may need to create UEFI-compliant bootable media for your system.

You can check which booting modes are supported by entering BIOS or UEFI user interface at startup, usually by pressing the F2 key (on MS Surface, hold the volume up button while pressing the power button, don't release the volume up button until UEFI menu appears). You can check and change various options under *boot menu*, such as legacy/UEFI mode and the order in which devices are checked for eligible bootable media.

Typical tasks you will want to do are converting bootable media from one form to another (e.g. create a bootable USB disk, USB stick or DVD from bootable ISO images, or vice versa) and to add your own files do bootable media (to create custom self-contained system independent of any pre-existing software or operating system installed on computer). For example, many operating systems come packed in forms of ISO images (including Windows and Linux). New computers may be sold without bootable media and it is wise to create a rescue DVD or USB stick to use in case of troubles. Subsequent sections will provide step by step instructions instruction for performing various combinations of such tasks. Shortly, you may add the following conversions:

- Bootable ISO -> Bootable CD/DVD:
    - With the same **boot** (either UEFI or Legacy with MBR): directly burn an ISO image to CD or DVD using a program such as **ImgBurn**.
    - With a **different boot mode**: It is necessary to reformat the ISO before burning it to a CD or DVD. If you target **legacy boot**, this is easily done by **ImgBurn**, otherwise it is advisable to use some other tool such as a special variant of **cdrtools** (Section 2.2).
- Bootable ISO -> Bootable USB drive:
    - This can be done by software called **Rufus**, which works well for the same or different boot mode (**legacy as well as UEFI**). In case you are creating an USB with a different boot mode than the original, ISO image then the image must contain the appropriate boot loaders. See Section 2.3.
- Bootable USB -> bootable ISO or
- Arbitrary directory containing a boot image -> bootable ISO:
    - For **Legacy boot** you can uses **ImgBurn** (see Section 2.2).
    - For **UEFI boot**: **IMGBurn** should be able to do that, there may be problems with it. Zou can use a special variant of **cdrtools**, more specifically the **mkisofs** program that is included in the package (end of Section 2.2).

2.1 Manipulating ISO Image Files

ISO files are special archive files that contain image of optical drives (CDs or DVDs). They contain a sector by sector copy of the data on an optical disk, therefore they also contain a binary image of the file system of an optical media (usually ISO 9660 or UDF). It is therefore easy to burn an ISO file to CD or DVD by preserving everything, including hardware addresses of every bit of data.

If an ISO file contains contents of a bootable disk with specified boot scheme (legacy or UEFI or both), burning such ISO to CD or DVD will create a bootable optical disk under the same scheme.

There are many programs that are able to **burn ISO images to CDs and DVDs.** More recent versions of **Windows** come with built-in tools for burning ISOs to optical disks. You can also burn an ISO by running **ImgBurn** (freeware), clicking "Write Image file to disc", selecting the image file and the destination drive and clicking the action button below. Another freeware you can use for this task is Daemon tools lite.

Things get more complicated if you have an ISO image prepared for one type of boot, but need to **create a CD or DVD for another type of boot**. In this case, you must get the appropriate boot image corresponding to the targeting boot mode and system. Typically you will unpack the ISO image to a directory, add the necessary boot image, and create a new re-formatted ISO image from it, see Section 2.2. As you will see in this section, it is still easier to find easy to use and well working tools for this in case you need a legacy boot than UEFI boot.

Copying contents of an ISO image to a directory is straightforward. On recent versions of Windows (since Windows 7) you can mount an ISO image as a virtual read-only drive by double clicking on the image file in File Explorer. You can also mount ISO image as a virtual drive by using Daemon Tools Lite. Then, you can copy files from the virtual drive in the same way as from any other disk. 7-zip is another program that can be used to extract contents of an ISO file (right-click the ISO file and go to **Extract to** "*xxxx*" (or whatever the original ISO file name was originally), then you can normally access files in that directory.

Changing contents of an ISO image (i.e. adding or deleting files) is more difficult. In some commercial software you can seemingly edit contents of ISO images in a direct way. Most free versions of such software are limited in size of the generated target image. Usual procedure is to **extract contents of an ISO to a directory** as described above (you can skip the parts you don't need and want do "delete"), **change contents of that directory**, and **create a new ISO image from a directory**.

The last step – **creating an ISO image from contents of a directory** - is described in detail in Section 2.2. It is easy to find free software to perform this step for non-bootable disks (e.g. in ImgBurn, click "Crete image file from files/folders", then select a source directory by clicking the folder icon next to "Source", select the file where image is stored by clicking the folder icon next to "Destination" and click the action button below). This is more difficult if ISO image must be bootable because it is not enough to copy files in the logical directory structure of the file system, but part of the code that enables bootstrapping (loading of the operating system or another boot software) must be copied to the specified hardware address of the represented disk (brief explanation is at the beginning of Section 2). Especially for creating UEFI-compliant bootable image it is less straightforward to find reliable and easy-to-use tools (see the end of Section 2.2).

2.2 Modifying Contents of a Bootable ISO Image

As mentioned in Section 2.1, this task is normally composed of:

1. **Extracting contents** (or partial contents) of the ISO image to a directory
2. **Modifying** contents of that **directory** as required (adding, deleting or modifying its files and subdirectories
3. **Creating** a new, modified ISO image from the modified **directory contents**.

The first step is pretty straightforward and was covered in Section 2.1, and the second one consists of everyday tasks.

The last step is what somewhat more tricky because it is not just about archiving a directory structure to the ISO file, but must also prepare the image properly according to the targeted boot mode (legacy BIOS-based/MBR or UEFI/GPT or eventually both).

You can perform this step by the ImgBurn program. I downloaded the program from Softpedia because that download was not bundled with suspicious software (but this may change, see Section 1.1.1 for advice on precaution). In installer, don't select installation of any additional software.

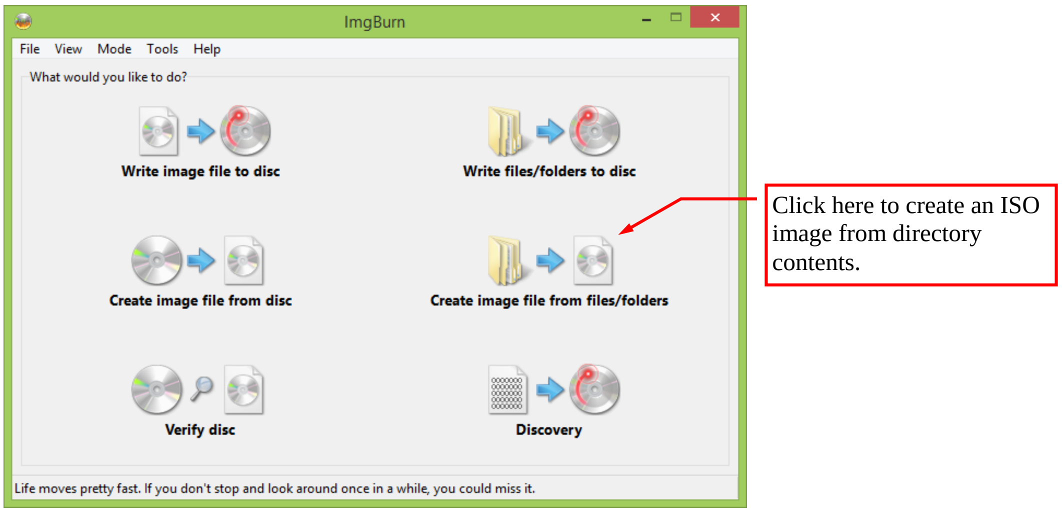

ImgBurn
File View Mode Tools Help
What would you like to do?
Write image file to disc
Write files/folders to disc
Create image file from disc
Create image file from files/folders
Verify disc
Discovery
Life moves pretty fast. If you don't stop and look around once in a while, you could miss it.
Click here to create an ISO image from directory contents.

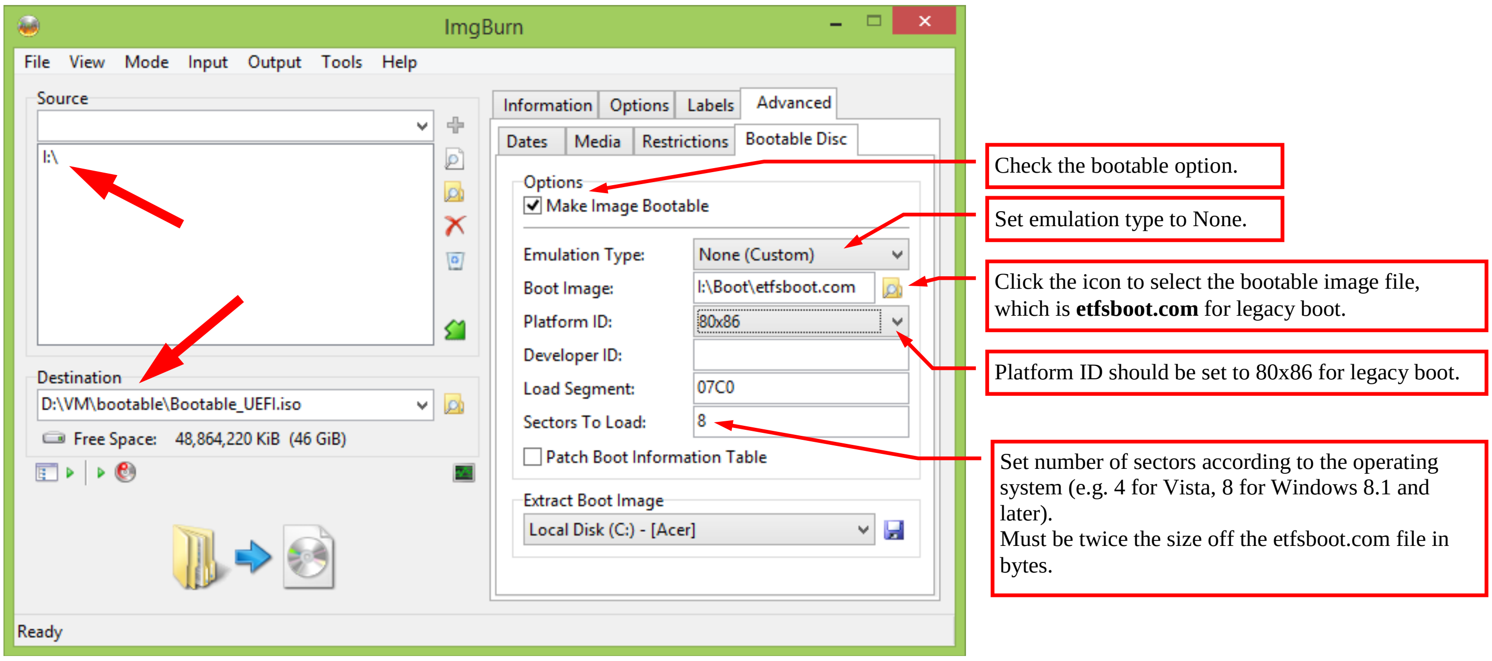
Figure 1:.Run ImgBurn and select "Create image file from files/folders".

Below are the steps to **create a bootable ISO image** (**for legacy boot**) image from a (possibly modified) directory extracted from a bootable:

- As additional advise, avoid adding files in the root directory, especially those with .exe and .com extensions.

- Run **ImgBurn** and click “**Create image file from files/folders**“.
- Select the directory from which ISO image will be created by clicking on the folder icon beside “**Source**”
- Insert the file path of the created ISO image by clicking the folder icon beside “**Destination**”.
- On the right-hand side of the window, select the “**Advanced**“and “**Botable**“ tabs. Then do the following in the “Options” region that appears:
    - Check the “**Make Image Bootable** “check box.
    - Beside “**Boot Image**“, click the folder icon and select the .../boot/**etfsboot.com** file within the directory containing ISO file contents.
    - Beside **Sectors To Load**, insert **8** (4 in the case of Windows Vista or previous Windows versions). More precisely*, enter 4 beside ‘Sectors To Load’ if your etfsboot.com file is of 2 kB size, or enter 8 if it’s 4 kB. Generally*, num = (size of *etfsboot.com* in bytes) / (512 *bytes)*
    - **Click the action button** on the lower left-hand side of the window to start creating the ISO file.
    - If necessary, change the volume label of the generated ISO file in the box that is launched.
    - Click “Yes” or “OK” in the subsequent dialog boxes that are launched.
    - **Wait until creation is finished.** This can take a while, dependent on image size and other factors.

Procedure is illustrated in Figure 2 below.


ImgBurn
File View Mode Input Output Tools Help
Source
I:\
Destination
D:\VM\bootable\Bootable_UEFI.iso
Free Space: 48,864,220 KiB (46 GiB)
Information Options Labels Advanced
Dates Media Restrictions Bootable Disc
Options
Make Image Bootable
Emulation Type: None (Custom)
Boot Image: I:\Boot\etfsboot.com
Platform ID: 80x86
Developer ID:
Load Segment: 07C0
Sectors To Load: 8
Patch Boot Information Table
Extract Boot Image
Local Disk (C:) - [Acer]
Ready

Check the bootable option.

Set emulation type to None.

Click the icon to select the bootable image file, which is **etfsboot.com** for legacy boot.

Platform ID should be set to 80x86 for legacy boot.

Set number of sectors according to the operating system (e.g. 4 for Vista, 8 for Windows 8.1 and later).
Must be twice the size off the etfsboot.com file in bytes.

Figure 2: Settings when creating a legacy boot ISO image.

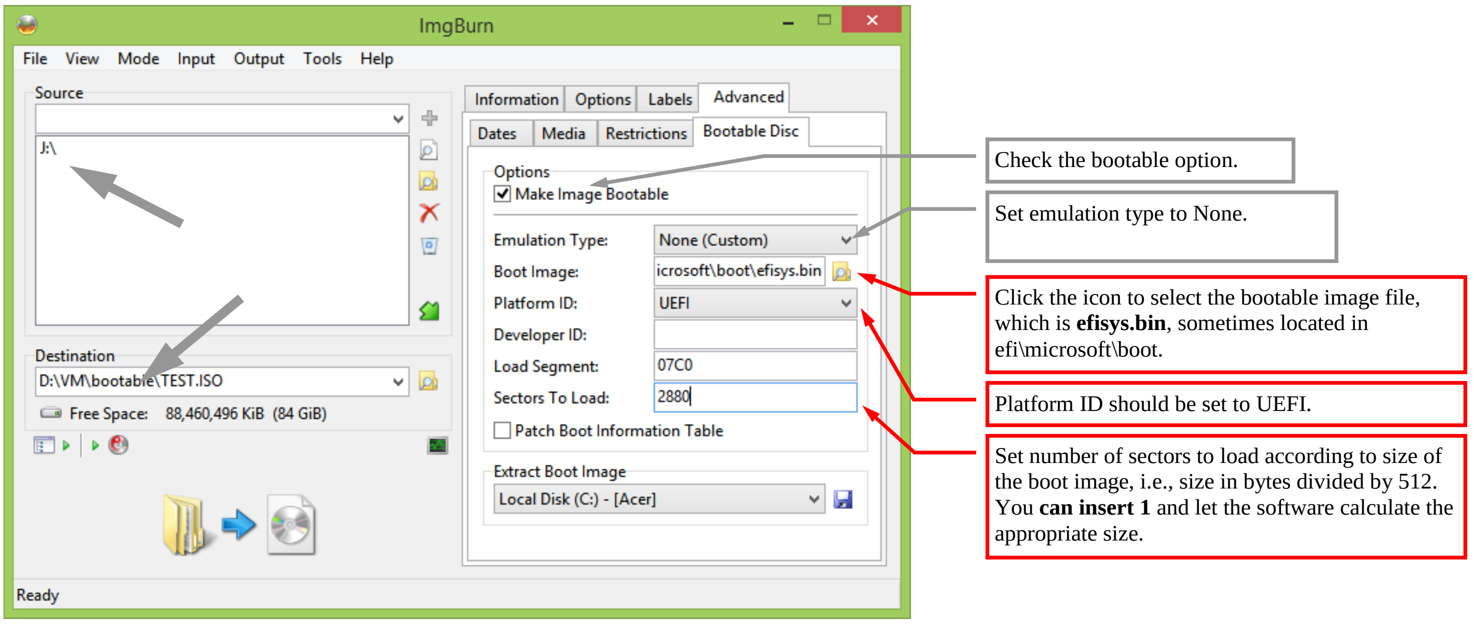
Remark:

In a similar way you can create a system ISO from a system USB drive, hard drive, DVD or simply from a directory containing all the necessary files. You should just be careful to include all the system files and files that participate in the booting process (e.g. select the **etfsboot.com** as Boot Image image; location of the boot image may vary between different distributions and operating systems). You can conveniently use this procedure on a system USB that was created from a system ISO (Section 2.3), where you can simply add files or install programs when necessary (for the later you can also boot from the USB disk). After that, archive the extended contents from the disk to a bootable ISO file.

ISO image created by the above procedure is prepared for legacy boot. For **UEFI boot**, you will have to change a number of parameters. According to my experience, UEFI-bootable images generated by ImgBurn don't always work as expected and it is sometimes difficult to figure out what went wrong. In order to spare unnecessary headaches, I will list settings that worked most reliably for me. For example, I suggest using the UDF file system instead of ISO 9660 when possible. Alternatively, you can perform the same task by the *mkisofs*, which is described later.

Below I list the **parameters** that you need to change in order **to create a UEFI-bootable image** (marhed in Figure 3 - Figure 7):

- Under **Advanced/Bootable Disc** (Figure 3):
    - Next to "**Boot Image**", point to **efisys.bin** (usually at the location efi\microsoft\boot\efisys.bin relative to the cource directory). Click the folder icon to select the location using a file dialog box.
    - For "**Platform ID**", select **UEFI**.
    - At "**Sectors to load**", insert the appropriate number according to the file size of the boot image (a typical value is *2880*). Divide the size of efisys.bin by 512 to obtain the correct number. Alternatively, you *can set value to 1*, in which case software will calculate the appropriate value.
- Under "**Options**" tab, change the following settings (Figure 4):
    - Set "**File System**" to **UDF**.
    - Check "**Include Hidden Files**".
    - Check "**Include System Files**".
- Under the "**Advanced/Restrictions/UDF**" tab, check "**Disable Unicode Support**".


ImgBurn
File View Mode Input Output Tools Help
Source
J:\
Destination
D:\VM\bootable\TEST.ISO
Free Space: 88,460,496 KiB (84 GiB)
Information Options Labels Advanced
Dates Media Restrictions Bootable Disc
Ready

Options
☑ Make Image Bootable

Emulation Type: None (Custom)
Boot Image: icrosoft\boot\efisys.bin
Platform ID: UEFI
Developer ID:
Load Segment: 07C0
Sectors To Load: 2880
☐ Patch Boot Information Table

Extract Boot Image
Local Disk (C:) - [Acer]

Check the bootable option.

Set emulation type to None.

Click the icon to select the bootable image file, which is **efisys.bin**, sometimes located in efi\microsoft\boot.

Platform ID should be set to UEFI.

Set number of sectors to load according to size of the boot image, i.e., size in bytes divided by 512. You **can insert 1** and let the software calculate the appropriate size.

Figure 3: *Bootable Disc* settings when creating ISO image for UEFI boot.

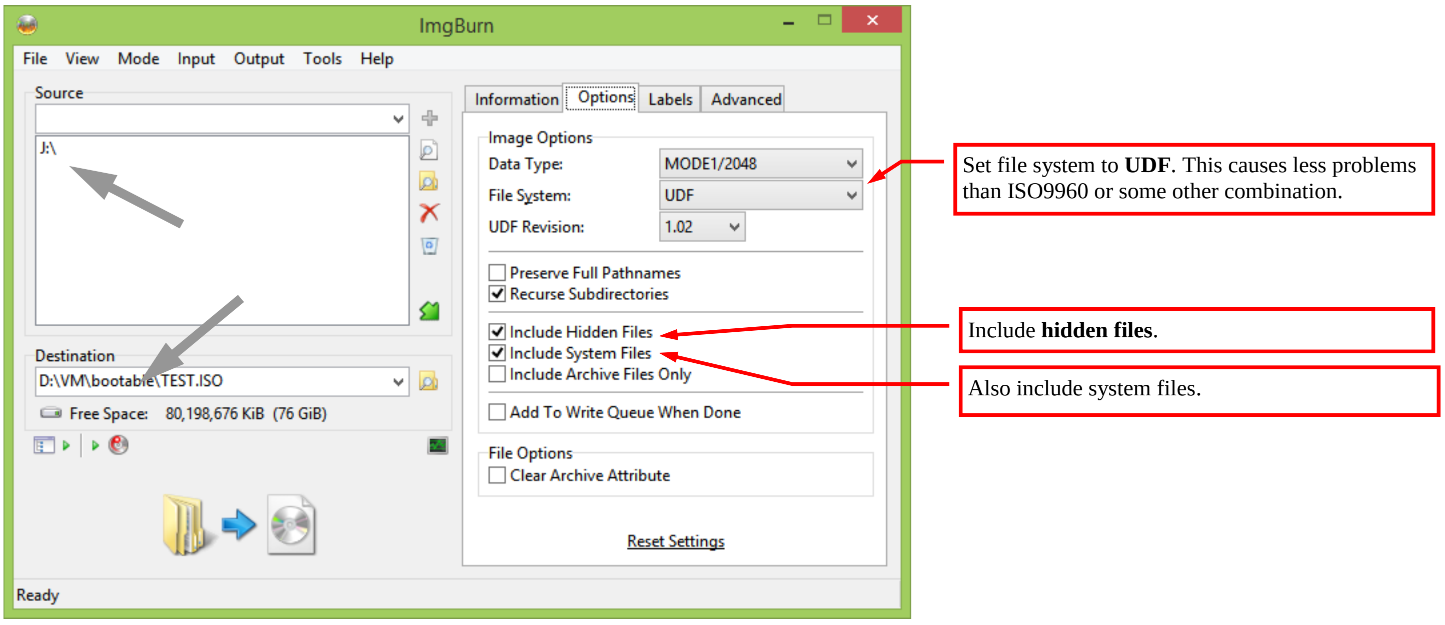

ImgBurn
File View Mode Input Output Tools Help
Source
J:\
Destination
D:\VM\bootable\TEST.ISO
Free Space: 80,198,676 KiB (76 GiB)
Information Options Labels Advanced
Reset Settings
Ready

Image Options
Data Type: MODE1/2048
File System: UDF
UDF Revision: 1.02

☐ Preserve Full Pathnames
☑ Recurse Subdirectories

☑ Include Hidden Files
☑ Include System Files
☐ Include Archive Files Only

☐ Add To Write Queue When Done

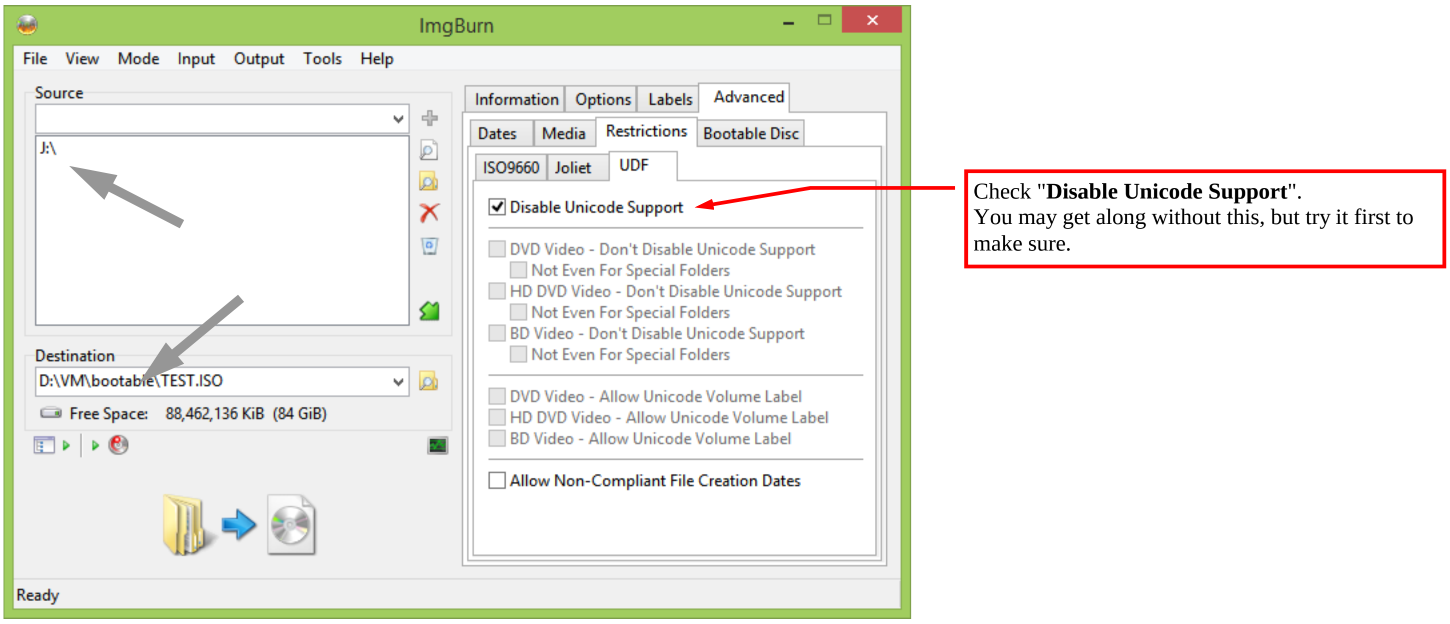
File Options
☐ Clear Archive Attribute

Set file system to **UDF**. This causes less problems than ISO9960 or some other combination.

Include **hidden files**.

Also include system files.

Figure 4: Changes under **Options** when creating ISO image for UEFI boot.


ImgBurn
File View Mode Input Output Tools Help
Source
Information Options Labels Advanced
Dates Media Restrictions Bootable Disc
ISO9660 Joliet UDF
Ready

J:\

Destination: D:\VM\bootable\TEST.ISO

Free Space: 88,462,136 KiB (84 GiB)

- [✓] Disable Unicode Support
- [ ] DVD Video - Don't Disable Unicode Support
  - [ ] Not Even For Special Folders
- [ ] HD DVD Video - Don't Disable Unicode Support
  - [ ] Not Even For Special Folders
- [ ] BD Video - Don't Disable Unicode Support
  - [ ] Not Even For Special Folders
- [ ] DVD Video - Allow Unicode Volume Label
- [ ] HD DVD Video - Allow Unicode Volume Label
- [ ] BD Video - Allow Unicode Volume Label
- [ ] Allow Non-Compliant File Creation Dates

Check "**Disable Unicode Support**".
You may get along without this, but try it first to make sure.

Figure 5: Changes under **Restrictions/UDF** when creating ISO image for UEFI boot.

Creating UEFI bootable images does not always work well (see e.g. this post and its continuation;, referring to ImgBurn version 2.5.8.0). Procedures to create UEFI bootable disks are also not so well documented, as opposed to creating media for legacy BIOS mode. UEFI is intensively being replacing the old BIOS interface, but it seems that software developers have not yet quite caught up.

If you have problems with the above method, read the paragraphs below that describe creation of bootable optical disks by using the **mkisofs** program, a command-line tool that is part of an extended distribution of the popular cdrtools. This tool behaves more predictably but is less easy to use for users without some expert knowledge of booting procedures and the related standards.

Instructions below will guide you through the procedure of creating a bootable optical disk by using this tool. The created discs will be **double bootable** both under **legacy BIOS/MBR and UEFI/GPT** schemes. As before, you can burn the created ISO file to a CD or DVD (Section 2.1).

You will first need to download tool **mkisofs**, which you can also do from this location (I used version 23). **Unpack contents** of the downloaded archive to desired location by using **7-Zip**.

Extracted directory contains two versions of **mkisofs.exe**, one created with Cygwin and one created with MinGW. I used the second one, located in the **mingw/** subdirectory of the extracted archive. You should **add** the directory containing mkisofs.exe to the **PATH** (on Windows, type "Environment Variables" in the search box after pressing the Windows key, then in the upper box, find the PATH variable (or click "New...") to add it), then double-click the PATH variable and append Variable value with ";xxx\mkisofs.exe", where "*xxx*" is complete path to the directory containing the executable). Alternatively, you can copy the executable (mkisofs.exe) into a directory that is already included in the path (If you use the executable contained in the *cygwin/* subdirectory, you should copy the *cygwyn.dll* with it, so the executable can find and load it).

Bootable CD or DVD is then **created by running mkisofs.exe with the appropriate parameters** in Windows command prompt. To make things easier and more transparent, I created a Windows command script named **uefi_iso.bat** to do the job. Its contents are shown below and the script can be run with three parameters:

```
uefi_iso input_directory iso_file_path [volume_label]
```

where:

- *input_directory* is path to the directory that contains the extracted contents of the original ISO file, eventually modified according to your needs (see Section 2.1 on how to extract contents of a ISO image to a directory)
- *iso_file_path* must be path to the file (preferably non-existent, otherwise it will be overwritten) where the generated ISO image is stored.
- *volume_label* is an optional parameter you can use to assign the desired volume label to the generated ISO image.

The script must be put into a directory that is contained in the PATH variable (or called by referencing it through absolute path or relative path with respect to the current directory). When running the script, it will call **mkisofs** (provided that the executable is located in a directory contained in the PATH variable) with the appropriate parameters to create a ISO image bootable both under UEFI and legacy scheme. Contents of the script are listed below:

Script file **uefi_iso.bat**:

```
echo Creating double bootable ISO file (legacy & uefi...)

echo.

set inputdir=I:\

set outputiso=d:\VM\bootable\TEST.ISO

set label="UEFI_BIOS_BOOT"
set biosboot=Boot/etfsboot.com
set efiboot=efi/microsoft/boot/efisys.bin

IF NOT [%1]==[] (set inputdir=%1)
IF NOT [%2]==[] (set outputiso=%2)
IF NOT [%3]==[] (set label=%3)

echo.
echo PARAMETERS:
echo inputdir = %inputdir%
echo outputiso = %outputiso%
echo label = %label%
echo biosboot = %biosboot%
echo efiboot = %efiboot%

echo.
echo Creating adouble bootable iso...
echo.

rem Command below must be put in single line!
mkisofs -iso-level 4 -l -R -UDF -D -volid %label% -b %biosboot% -no-emul-boot -
            boot-load-size 8 -hide boot.catalog -eltorito-alt-boot -eltorito-
            platform efi -no-emul-boot -b %efiboot%  -o %outputiso% %inputdir%
```

By removing some of the options you can create a ISO image that is bootable only under legacy or UEFI mode.

If you often repeat the operation with the same values of parameters, you can change initial values of the appropriate variables in the script (`inputdir`, `outputiso`, or `label`) to meet your situation and then omit these parameters when running the script.

The **mkisofs** program used is an extended version of the program from cdrtools software, created by Alex Kopylov. Original distribution of the popular cdrtools was created by Jörg Schilling et al.

2.3 Creating Bootable USB Disk or Flash Drive from an ISO image

Burning bootable ISO images to CDs/DVDs is straightforward because ISO is a verbatim, sector by sector representation of a CD or DVD (thus not only preserving the file system but also address space). Addressing and retrieval of data is different on USB drives, and creating a bootable USB drive from an ISO file requires special preparation of the USB drive (if you don't need a bootable USB, copy contents of ISO file according to instructions in Section 2.1).

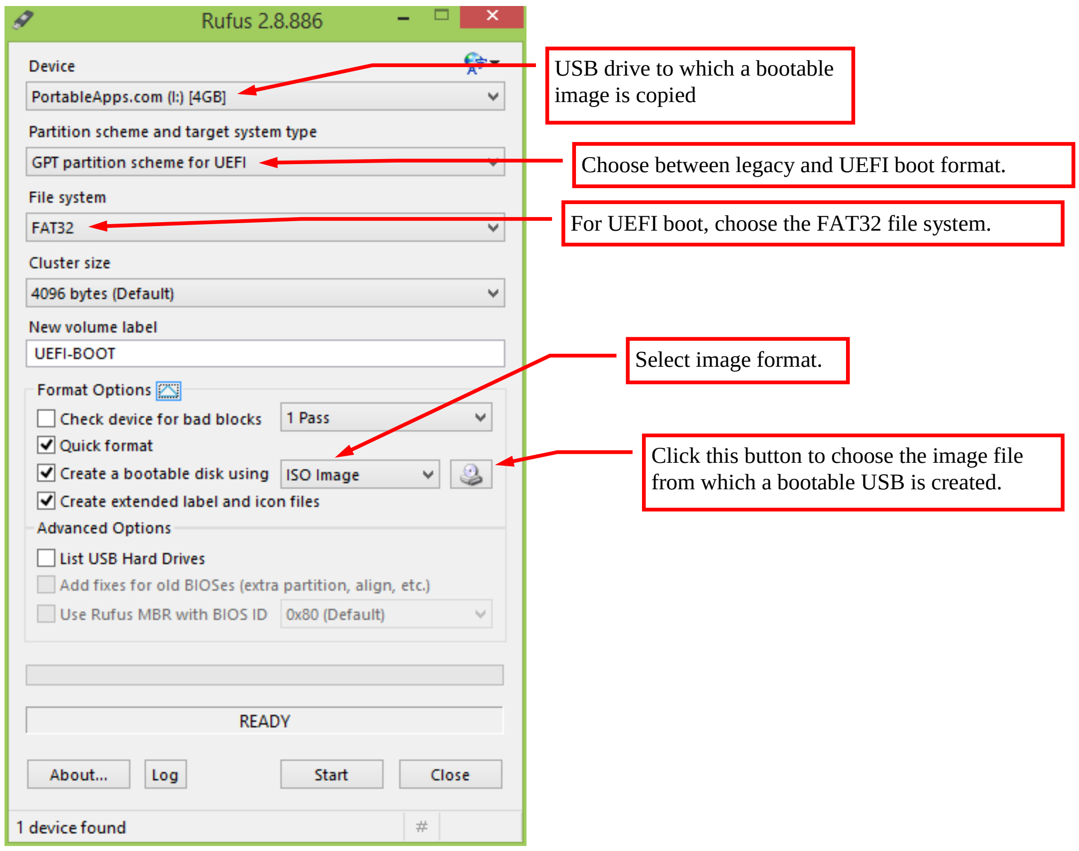
In order to **create a bootable USB** disk or flash drive **from a bootable ISO** file, you can use **Rufus**, a program that will perform the necessary steps to make the created USB copy bootable under the desired scheme. Creating a bootable USB will **erase everything on the target drive**, so be sure to copy any files you want to keep somewhere else before performing the operation.

In order to create a bootable USB drive from an ISO file using Rufus, do the following (see also Figure 6):

- **Back up** any files from the **target USB** drive because its contents will be deleted.
- Download and **run** **Rufus**. Program comes as portable application, therefore you can just save the program to a disk and run it without running any installer. You will need to confirm User Access Control diallog.
- **Insert** the target **USB** disk or flash drive. Make absolutely sure which is its address (drive letter) to avoid unintentionally deleting some other USB drive..
- In Rufus, do the following:
    - In the "***Device***" drop-down menu, **select the target USB** drive to be made bootable by applying contents of the bootable ISO image.
    - Under "**Partition scheme and target system type**", select the appropriate option according to the systems on which you will need to boot from USB. The default (MBR

for VIOS or UEFI) may work on many systems. If you have an UEFI-based system and want to boot in native mode, select GPT partition scheme for UEFI. Most modern computers use UEFI mode while many still support MBR partition scheme for BIOS, but not all (see Section 4.2.1 for how to check).

- You can usually leave the "***File System***" as it is. If you only need to create a legacy boot then you can use NTFS. In order to create a USB drive that is UEFI-compliant or bootable under both (legacy BIOS/MBR or UEFI/GPT) schemes, you should choose **FAT32**. UEFI -based system do not normally support NTFS boot partitions.
- You can leave *Partition scheme* and *Cluster* size as they are, and eventually insert your own *volume label*.
- Uder "***Format Options***", you can leave most of the things as they are, except:
- Beside "***Create a bootable disk using***", select "***ISO Image***" from the drop-down menu, then **click the icon to the right** and **locate the ISO file** that you want to apply (or "burn") to the USB drive. Wait a few moments for *Rufus* to check the file.
- **Click** the ***Start*** **button** below. Now contents of the ISO image are being applied to the USB drive, with progress bar showing the amount of work done. Just wait for the process to finish and close *Rufus*. You should now have a valid bootable USB drive that you can test.



Rufus 2.8.886

Device: PortableApps.com (I:) [4GB]

Partition scheme and target system type: GPT partition scheme for UEFI

File system: FAT32

Cluster size: 4096 bytes (Default)

New volume label: UEFI-BOOT

Format Options

- [ ] Check device for bad blocks — 1 Pass
- [x] Quick format
- [x] Create a bootable disk using — ISO Image
- [x] Create extended label and icon files

Advanced Options

- [ ] List USB Hard Drives
- [ ] Add fixes for old BIOSes (extra partition, align, etc.)
- [ ] Use Rufus MBR with BIOS ID — 0x80 (Default)

READY

About... Log Start Close

1 device found

USB drive to which a bootable image is copied

Choose between legacy and UEFI boot format.

For UEFI boot, choose the FAT32 file system.

Select image format.

Click this button to choose the image file from which a bootable USB is created.

Figure 6:. Creation of a bootable USB drive with Rufus. Shown settings are for creating an USB disk or flash drive for UEFI-based systems.

Figure 6 shows what you need to set to create a bootable USB from an ISO image. The displayed settings are for UEFI-based boot. If you want to create media for legacy boot:

- Under *partition scheme*, select *MBR partition scheme for BIOS or UEFI*.
- In this case, you can also select NTFS under *file system*.

By selecting the MBR partition scheme for BIOS or UEFI and the FAT32 file system, you should be able to boot on both types of computers, provided that the appropriate boot images are present in the ISO image.

On Windows, it is also possible to create a bootable USB drive without using any special software (see e.g. these instructions), although using Rufus is more convenient.

For some additional information, you can check the following articles:

How to Create Bootable USB Drives and SD Cards For Every Operating System (at HowTo Geek).

A Comprehensive Guide to Make a Bootable USB in Windows

3 USE OF BOOTABLE MEDIA

It has already been mentioned in Section 1 why you might need to create bootable media. This section provides additional tips about this.

3.1 Distribution, testing and demonstration of software

3.2 System Troubleshooting, Backup and Restore

4 TESTING BOOTABLE MEDIA AND DOWNLOAD SOURCES

4.1 Sources of Pre-compiled Bootable Media

There are numerous sources where you can obtain various bootable images, which you can convert to DVDs, USB drives and other physical media according to instructions on this site.

Producers of most **major operating systems** offer sites where you can **download images** of installation and repair media for their OSs (e.g. the latest Ubuntu Linux, versions, MS Windows 10, Windows 8.1 or Windows 7). As free OS, different flavours of Linux are especially easy to download and manipulate.

For some versions of Windows (prior to version 8) you will need to type your product key before being able to download ISO file. You install **Windows 10** without buying a license for a 90 day evaluation period. If you have a valid Windows 7 or 8 license, you can upgrade to Win 10 for free. You can also join the insider program to receive preview builds for major updates of the operating system. A small annoyance with Windows 10 is that you have to use the Microsoft's Media Creation Tool in order to download it, so you can not make use of your favorite third party downloading software that may enable pausing the download and adjusting bandwidth limits. If you are fine with that, you can use the Media creation tool to save the downloaded Windows installation as ISO file or to a USB. In any case it is recommended to save installation rather to run the setup process immediately from the download tool because setup can easily go wrong and you may need to repeat the time consuming download in this case. If you really want to avoid using the MS's download tool, you can avoid it by using the direct ISO download link or this link. The first link will redirect to the usual download method (through their download tool) if you run browser on Windows 7 or 8, and you can avoid this if you download from another operating system (conveniently, if you have a Linux or older Windows virtual machine); there are also other tricks mentioned here.

There are plenty of **unofficial web sites** offering full **installation media** for various operating systems, sometimes with keys for proprietary operating systems. Many of these downloads are illegal and are also risky for virus infections. If you download anything from such sites, I recommend to verify the sites with web scanning tools and the downloaded contents by antivirus programs. VirusTotal is a very convenient tool for that, to check a web site, just enter its address under the "URL" tab. You can also check individual downloaded files (or any files on your computer) under the File tab, although there is a size limit which prevents checking long files like OS ISOs. The advantage of VirusTotal is that contents are checked by many engines thus reducing the possibility to missing a threat. However be aware that detection of viruses and other threats is never 100%. Playing with stuff in isolated environments such as virtual machines can also contribute to your safety. See also Section 1.1.1 for remarks remarks about safety regarding software downloaded form the internet.

4.1.1 Windows PE

Windows PE (Widows Preinstallation Environnment) are **free lightweight version of windows**. Although with reduced set of features, WinPE can be used for troubleshooting an operating system while it is offline, or even as a working operating system for less demanding users.

Windows PE media can be prepared on windows by using the Windows Assessment and Deployment Kit. This software enables creation of create Windows recovery environment or Windows PE images where you can add a number of additional packages such as the .NET framework, file management and network tools, and others.

It may seem *cumbersome* to you *to create such images on your own*. In this case, there are a number of other options. A free version of AOMEI Backupper, for example, can create a very simple Windows PE -based ISO, USB or CD/DVD (under Utilities / Create Bootable Media). It can create either legacy or UEFI bootable Win PE (or a Linux disk) which runs the AOMEI Backupper on startup. You can use this to browse, check or restore disk images previously created by the software itself or to run a Windows command prompt.

4.1.1.1 Downloadable and Updateable Precompiled Windows PE Images

There are web sites where you can download **bootable Windows PE images with** a number of **added packages**. An example of such site is

http://windowsmatters.com/category/winpe/

with choices including well tuned Windows PE 8.1 or Windows PE 10. These images are packed with a number of useful additions such as customized desktop, the **.NET framework**, and plenty of free portable applications (such as file managers, disk managing software, browsers, media players, backup and disk imaging, office software, etc.). For less demanding users, the images may even be sufficient for everyday work. Because of the tools installed, they are also more comfortable for troubleshooting operation systems than more minimalistic images. You can also easily add your own applications and data to the downloaded images by using procedures described in this document.

Original Images from this site won't fit on a CD, but will fit on a DVD. They can be easily tested on virtual machines (see Section 4.2.2).

When the system starts from the image, it mounts a RAM disk so that you can temporarily install small amount of additional software (such as AOMEI backupper or Total Commander) even if you start the system from read only storage (e.g. DVD) and you don't have any additional disks mounted. Such installations may be useful when you use the image for troubleshooting your system. RAM disk size depends on the total RAM installed, so you will have larger RAM disk if you assign more RAM to the virtual machine where the image is booted (or if you boot from physical media on a machine with more RAM available). Programs that you install to RAM disk will disappear when the machine is rebooted.

4.1.1.2 Adding Things Permanently

To add your custom stuff permanently to the image, you will need to record the image to a writeable drive (such as USB flash or external disk - see Section 2.3) or change (edit) the ISO image (see Section 2.2).

To change ISO image, you need to unpack it to a directory, add your stuff to that directory, and re-create the image from directory contents in such a way that it remains bootable(Section 2.2).

You can **"burn" an ISO to an USB** flash drive or external USB disk according to instructions in **Section 2.3**. There are some specifics about mounting the USB on a virtual machine under VirtualBox, which are covered in Section 4.2.3.

When the system is **booted from a writeable USB drive** containing the image, you can add **portable programs** of your choice to the USB. You can later copy contents of the USB drive to a new bootable ISO image (Section 2.2.) and deploy it where necessary.

Rather than just using portable applications that you can simply copy to the USB, you can use **Windows installers to install new software** after booting from a Windows PE USB. However, **not all programs will work** (or some features will not work) **after you reboot the system** (or remove the USB drive and boot from it later). The issue and possible workarounds are explained below.

In Windows PE bootable drives, the operating system is booted from boot.wim file, a compressed image file in WIM format containing bootable version of Windows PE. When you install a new program via installer after booting from USB, some parts of installation (such as registry entries, user settings or drivers added) are **not persistent** because they are physically written to computer's RAM rather than to the drive. If you install the software in a directory on the USB drive (not on the RAM disk) then the installation directory will remain on the USB after the system is shut down and USB drive removed, and will be accessible when you boot from USB next time. The non-persistent parts will not, however, and this can affect how programs work.

It depends on the software whether some or all features will work properly after rebooting. Total Commander, for example, would work more or less normally as it can run without relying on any system-located components. AOMEI Backupper Standard, on the other hand, can not browse its disk images without full installation that makes system changes. When you need to restore a system from a system disk image by AOMEI Backupper, you can **boot** from the Win PE disk, **install the software and run it** to do whatever you need. In this case the software is properly installed and will run with all its features, but complete installation will persist only until you reboot Windows PE. Such ***temporary full-feature installations*** may be sufficient for tasks such as system disk restore and system troubleshooting.

A number of popular applications (especially freeware) are also available in portable form, which does not require any installation and applications can be simply copied to the Windows PE disk. This includes the AOMEI Backupper discussed above. Editions of precompiled disks from this site include many such applications, but complete list of software available in portable form is by far larger. For any particular application you use, you should search the Internet to find out whether a portable version also exists. The PortableApps.com contains many such applications.

4.1.1.2.1 Use for IGLib

I have conveniently used this method for deployment of my IGLib-based software (Section 5). There are many advantages of packing your software with such self-contained bootable ISO or physical media. Such media contain complete software environment including the operating system. For software that heavily depends on many other software packages, you can make sure that the wherever the software is run, it will have the same software environment and will therefore behave more predictably. You can, for example, physically demonstrate a software that depends on Windows operating system, .NET and other packages on a site where they only have Linux computers, as long as hardware meets the requirements of the software. This is because you can boot your own prepared system and set up complete software environment form your USB or DVD, no matter what is contained on the computer where you plug in your drive and boot from it. Another advantage is that you can you can safely run, test and demonstrate your or third party software on a virtual machine by using platform virtualization software such as VirtualBox. This is very useful when testing the software inside security-sensitive environments or when testing unknown software that might contain security threats. Virtual machines provide high level of isolation if configured in that way, therefore the tested software can not affect other storage devices of the computer where VM is run, or infect other systems by viruses. Virtual machines can be directly booted from ISO images, so you don't need even to create physical bootable media in order to run the software on a virtual machine.

4.1.1.3 Other Options for Windows PE-based Boot Disks

Some other options to create or download Windows PE-based boot disks are listed on the following site:

https://www.raymond.cc/blog/5-system-rescue-boot-disc-based-on-windows-pe/

4.2 Testing Bootable Media

You should always test your bootable media after creation. If possible, you should **test the media on the same system**(s) where they will be used to boot from, or at least on as similar systems as possible. Booting procedures are standardized, but many systems are more or less tolerant for deviation from standards or follow the standard more or less strictly. Being able to boot certain media on one system does not always mean that you can boot on another, although its specification refers to the same booting scheme.

4.2.1 Testing on a computer

When creating bootable media for some particular computer (or more hardware systems), you should first know **which booting scheme** those **computers use** (se beginning of Section 2). There are two main possibilities, UEFI with GUID Partition Table (GPT) or legacy BIOS scheme with Master Boot Record (MBR).

To **find out which scheme is used**, you should enter the **firmware settings menu (BIOS or UEFI)** before loading of the operating system begins. Firmware is resident in a special chip that is built into computer's motherboard. It is the first software that is run when a computer is powered on and is responsible for loading the operating system (booting) from the specified bootable media.

To **enter firmware settings** on most computers, you should **switch off** (shut down) the computer, **press power button to switch it on** and then **continuously tap the appropriate key**, (usually **F2**, but also *Del*, *F10* or *F12* on some computers) **until you see the setup screen**. On *Microsoft Surface*, you will have to *press and hold the volume up* button, then *press the power button* and continue to *hold the volume up until the setup screen appears*.

In the outer **setup menu**, select the "**Boot**" option. There you will **see whether** the system is **BIOS or UEFI based**. If it mentions nothing about UEFI then your system uses a legacy BIOS boot scheme, and you should create legacy bootable (or double bootable) media to boot on the system. If the system is UEFI based, it is best to prepare UEFI-based bootable media, but you can also prepare double-bootable (see Section 2.3) to be eventually able to boot on older systems that only support legacy BIOS boot. Some UEFI-based systems still support legacy BIOS boot. In such a case, you will find an option to switch between UEFI and legacy mode in the Boot menu of the setup. In such a case, you can boot from both types of media, but you can set the appropriate boot mode (legacy or UEFI) in the setup before booting.

In firmware setup, follow the instructions (usually at the bottom or at the side of the screen) for how to *exit without saving* your changes (usually with the *Esc* key) or how to *save changes and restart* (usually the *F10* key).

After the bootable media are created, you should **test** (whenever possible) **whether you can actually boot** from the created media **on the target computer**(s). Usually, to boot from a removable drive (CD, USB, DVD, etc.) or another internal drive, you will need to **change the boot order** in firmware (BIOS or UEFI) setup. Enter the firmware setup according to instructions above, and change the boot order under the "*Boot*" menu of the setup. Follow the instructions (usually displayed in the setup) about how to bring the drive you want to boot from to the top of the boot order and how to save the changes (they will be permanent and you will have to manually change them back later, if necessary). Then insert the drive you want to boot from and restart the computer (this usually happens automatically when saving changes). If your media is properly created and consistent with the boot scheme used by the computer, the system or other bootable program should start normally. If it does not happen and you created media for troubleshooting system problems, you may try to boot on a similar computer (with the same boot scheme) to exclude hardware errors. On computers that support both legacy and UEFI boot, make sure to set the appropriate scheme in firmware setup's Boot menu beforehand (especially if you created bootable media for only one scheme or another).

4.2.2 Testing on Virtual Machines (VM)

Platform virtualization software such as VirtualBox, QEMU, VMware Player or Windows Virtual PC is ideal for testing bootable images and media that should work on different computers (e.g. to distribute self-contained software packages), to test and run bootable media in isolated environment (such that the software run from media can not affect other software and operating system installed on the computer) or to play with different options when still trying to figure out working and efficient ways to create the media.

However, you should treat results achieved on virtual machines (VM) with some **caution**. If you can boot from your media on a virtual machine, this is not a 100% guarantee that the media will run on the actual hardware with similar specifications as set up on the testing virtual machine. Hardware does not always strictly follow standards regarding booting on a particular schemes (such standards are quite complex) and some hardware is tolerant to non-standard features, and these tiny details may differ from one hardware to another and can prevent booting form the media that worked perfectly on a system with the same official specifications.

This is not better when it comes to virtualization software. No such software can support all possible hardware combinations. Typically, when support to some feature (such as UEFI firmware replacing legacy BIOS) is added, it usually takes quite some time when the feature is in the testing stage, and even afterwards some details may be missing or there are bugs for which it takes long to be discovered. Hardware virtualization is a complex business, and any complex software created by humans comes with bugs, period.

Taking the above into consideration, when you create bootable media intended for known computers you have access to, you should always **test the bootable media also on those computers the media are intended to be used with**.

As virtualization software, I like to use VirtualBox as a professional open source solution. It is easy so configure and supports excessive number of combinations of guest and host operating systems. Below (Section 4.2.3) there are some additional advices related to testing bootable media on VirtualBox.

It is easy to install VirtualBox and create virtual machines for it. See for example the following web tutorials:

- How to Install Ubuntu on VirtualBox - wikiHow

- Installing Ubuntu inside Windows using VirtualBox
- Create a Windows Virtual Machine in Linux / Windows , by Angelos Kyritsis.

4.2.3 Booting VM on VirtualBox with a Bootable USB Disk or Flash Drive

Setting of a system that **starts from a USB** drive is not direct in VirtualBox. The version I used (5.0.16) does not support performing this from GUI. The drive must first be **prepared as** "**raw disk**" and you need to prepare the disk by tools run from **command-line**. Virtual machines that use raw disks must be run with **administrator's privileges** because this is needed to see raw disks in VirtualBox.

Instructions for starting a virtual machine from USB in VirtualBox **are here**, and I am adding some remarks:

- It seems that this procedure will not always work. I had situations where I could boot from USB flash in UEFI scheme from my laptop but I could not boot from it in a virtual machine (with administrator privileges and with EFI enabled).
- When booting a VM from raw disk, VirtualBox must be run with *administrator privileges*, which sometimes is not what you want due to other VMs started from VirtualBox GUI. However, you can open more than one GUI frontends with different privileges and start different virtual machines from different front ends.
- Instructions suggest to make a new virtual machine to boot from a raw disk. You may stick with this to avoid problems.

There is another annoyance with starting a virtual machine from a *USB drive mounted as raw disk*. When a virtual machine is started, the **USB drive is locked by Windows**. Because of this, the drive is not writeable and all write operations are directed to a temporary copy of the disk, which can cause that the virtual machine can not bee booted completely (dependent on the operating system and maybe some other factors).

You can **avoid** many **problems related to booting virtual machines** from a USB drive by using the **Virtual Machine USB Boot (VMUB)** program. This program dismounts the boot USB drive from Windows before a virtual machine started, and mounts it back after virtual machine is stopped. In this way, the **USB drive** is not locked and is **writeable by the virtual machine** that boots from it.

VMUB is a portable application so you can just save it to an easily accessible place on your system (e.g. in the directory where you keep virtual machines) and run it. You also don't need to include the executable location in the path variable because you will not need to run it from a script.

In order to **run a virtual machine that boots from a USB drive**, do the following:

- **Set up configurations** that you will run.
- **Run** the "***Virtual Machine USB Boot.exe***" (the VMUB application).
- In the window that opens (Figure 7), **choose** the appropriate **configuration** and click **Start**.
    - **If an error is reported**, select the configuration, click **Edit**, check the values and correct what needs to be corrected. E.g. the drive may have changed (you have put USB to other port than last time) or you don't have a virtual machine registered in VirtualBox (e.g. you removed it to change configuration files manually) or you renamed the virtual machine.

To **set up a new configuration or change existend one**, follow these steps:

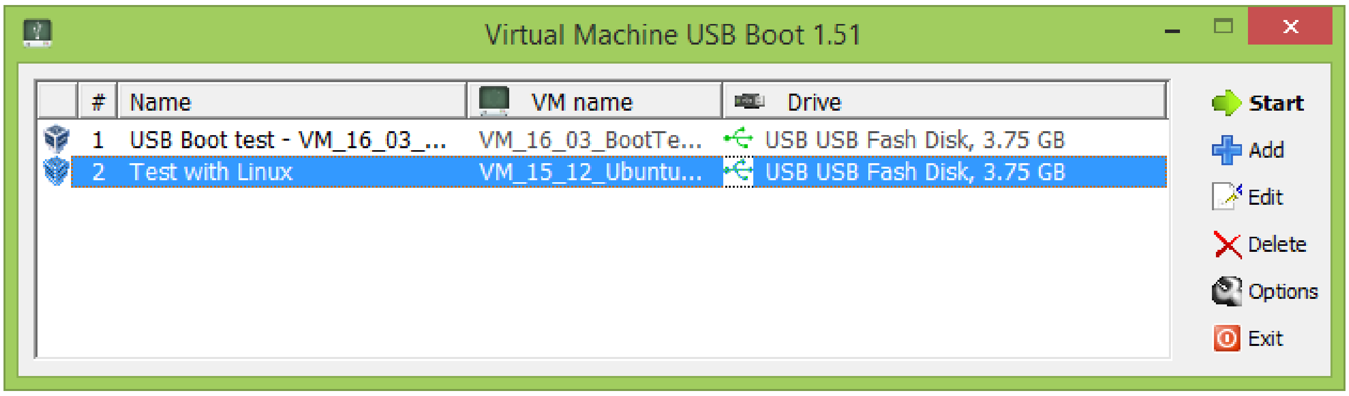
- **Run** the "***Virtual Machine USB Boot.exe***".
- Click **Add** and choose configuration name (**or select** an existent configuration **and click Edit**).
- Set what is necessary:
    - Under "**Mode to load the VM**" you can choose " **VM Name** ".
    - Under " **VM Name** ", **select the virtual machine** that you wand to start. If the mode was set by name (previous point) then the VM must be registered in VirtualBox. If you don't see the intended VM's name, run VirtualBox and add that name, then re-open settings.
    - Under " **Drive to add and boot** ", **select the USB drive** that you want your VM **to boot from**. The USB drive must be mounted on the system, so insert the flash or disk cord to the USB port first.

Important:

- For boot to succeed, the **USB drive must be mounted as the first disk** on the virtual machine. Therefore, the virtual machine must be set in such a way that the first disk (i.e. IDE Primary master or SATA 0) is empty, such that VMUB can mount the USB at this location.
- VMUB must be run with **administrator privileges**. On windows 8 or later this is done automatically, otherwise you can set under Properties/Compatibility menu for the executable (Virtual Machine USB Boot.exe) that the program is run as administrator.
- **If VirtualBox is runnning** when you start your VM (booted from USB) through VMUB, the VirtualBox **must be run with administrator privileges**.
- **Don's use the "Save state"** option with a virtual machine that is run through VMUB.

- When a virtual machine that boots from USB is run through VUMB, the **USB** flash or disk **is dismounted** from the host system by VMUB and is automatically **mounted back when the virtual machine is closed**. Therefore, the USB drive is not visible in the host system during the time when it is used by the virtual machine that booted from it.

In VMUB, you can configure a number of options by clicking **Settings**. Most likely you will not need to change any of these, but in case something behaves strangely, there are comprehensible descriptions on this web page.


Virtual Machine USB Boot 1.51

#	Name	VM name	Drive
1	USB Boot test - VM_16_03_...	VM_16_03_BootTe...	USB USB Fash Disk, 3.75 GB
2	Test with Linux	VM_15_12_Ubuntu...	USB USB Fash Disk, 3.75 GB

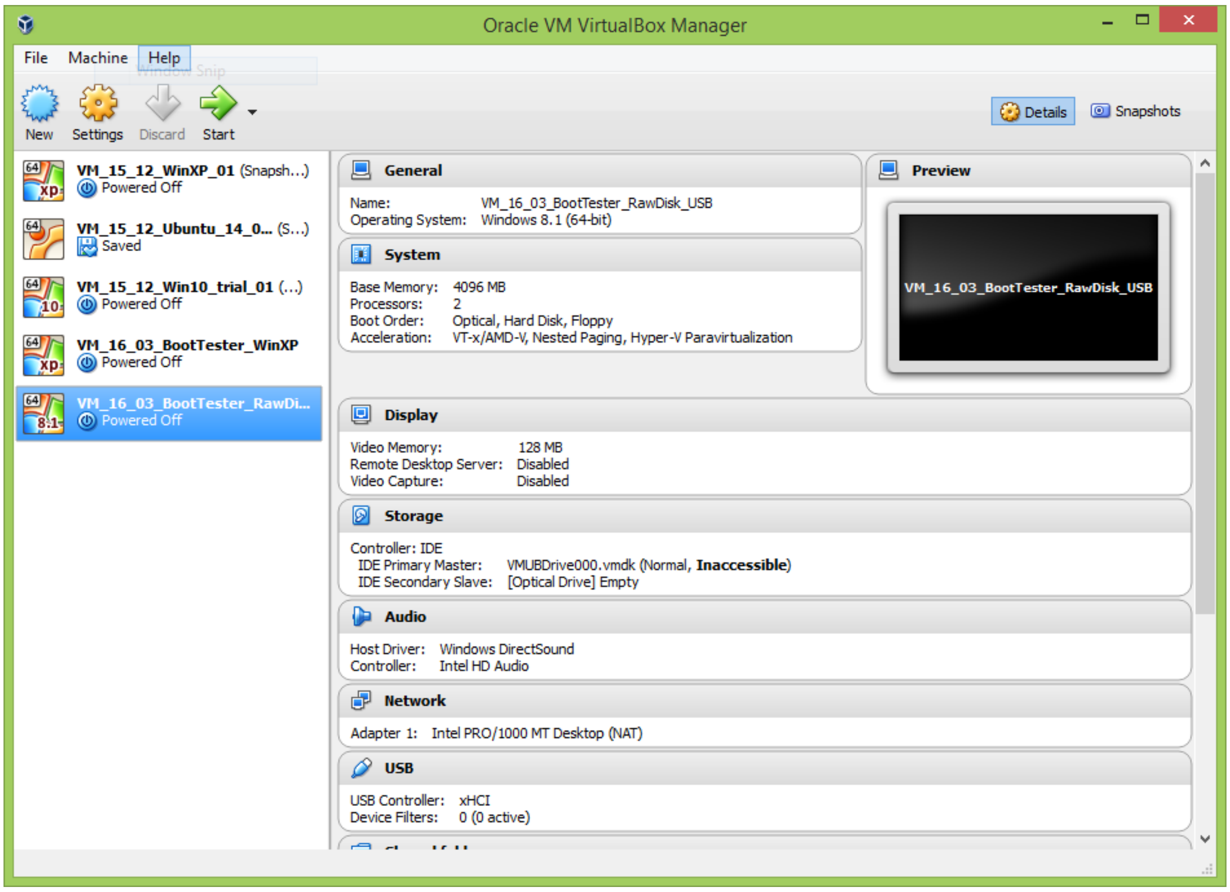
Figure 7:. VMUB when started (with two configurations available). Select configuration and click Start to start a virtual machine, or click Add to create a new configuration.

See also the following links:

Virtual Machine USB Boot 1.51 at reboot.pro

How to boot directly from a USB drive using an Emulator or VM under Windows

4.2.4 A Word about Virtual Machines



VM_15_12_WinXP_01 (Snapsh...)
Powered Off

VM_15_12_Ubuntu_14_0... (S...)
Saved

VM_15_12_Win10_trial_01 (...)
Powered Off

VM_16_03_BootTester_WinXP
Powered Off

VM_16_03_BootTester_RawDi...
Powered Off

General

Name: VM_16_03_BootTester_RawDisk_USB
Operating System: Windows 8.1 (64-bit)

System

Base Memory: 4096 MB
Processors: 2
Boot Order: Optical, Hard Disk, Floppy
Acceleration: VT-x/AMD-V, Nested Paging, Hyper-V Paravirtualization

Preview

VM_16_03_BootTester_RawDisk_USB

Display

Video Memory: 128 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: IDE
IDE Primary Master: VMUBDrive000.vmdk (Normal, **Inaccessible**)
IDE Secondary Slave: [Optical Drive] Empty

Audio

Host Driver: Windows DirectSound
Controller: Intel HD Audio

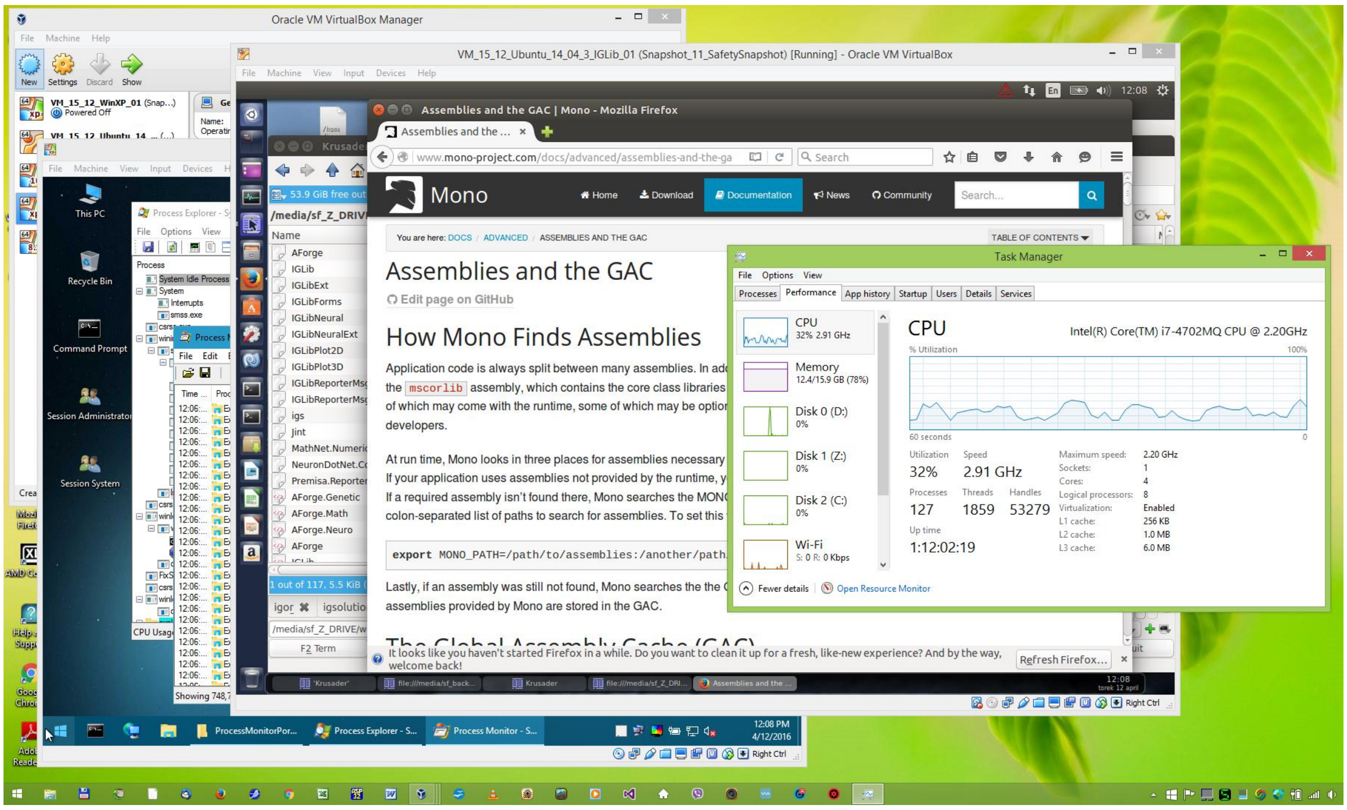
Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB

USB Controller: xHCI
Device Filters: 0 (0 active)

Figure 8:. VirtualBox front end. From here you create and configure virtual machines, start, stop or pause them, clone them, attach drives and other hardware, etc.



You are here: DOCS / ADVANCED / ASSEMBLIES AND THE GAC

Assemblies and the GAC

Edit page on GitHub

How Mono Finds Assemblies

Application code is always split between many assemblies. In ad the `mscorlib` assembly, which contains the core class libraries of which may come with the runtime, some of which may be optior developers.

At run time, Mono looks in three places for assemblies necessary If your application uses assemblies not provided by the runtime, y If a required assembly isn't found there, Mono searches the MONO colon-separated list of paths to search for assemblies. To set this

```
export MONO_PATH=/path/to/assemblies:/another/path
```

Lastly, if an assembly was still not found, Mono searches the the G assemblies provided by Mono are stored in the GAC.

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Utilization	Speed			Maximum speed:	2.20 GHz
32%	2.91 GHz			Sockets:	1
				Cores:	4
Processes	Threads	Handles		Logical processors:	8
127	1859	53279		Virtualization:	Enabled
Up time				L1 cache:	256 KB
1:12:02:19				L2 cache:	1.0 MB
				L3 cache:	6.0 MB

Figure 9:. VirtualBox in action: running two VMs - Windows 10 PE and Ubuntu Linux guests on a Windows 8.1 host.

5 About the Software Supported

Tools described in this blog were used to support distribution of AnnApp, a software developed by Igor Grešovnik as an easy to use and intuitive tool for exploring models based on artificial neural networks (ANNs). Igor Grešovnik and Tadej Kodelja have developed another GUI-based software (*NeuralParametric*) for performing parametric studies on ANN-based models while modelling influence of process parameters on material production in the Centre of Excellence for Biosensors, Instrumentation and Process Control (COBIK). In the scope of his work at COBIK, Tadej Kodelja was also working on his Ph.D. thesis under Gesovnik's guidance regarding algorithms and software.

in the scope of this work, modeling software *NeuralShell* was designed to tackle challenging ANN modeling problems in industry. It was based on scripting framework of the *Investigative Generic Library* (*IGLib*), therefore it took some degree of expertise and skills to use the software, with the benefit of being a powerful tool adaptable to solve a wide range of problems. *NeuralParametric* was designed as supplementary tool to enable more intuitive exploration of the generated models by industrial users not skilled in scripting. However, the software was designed in haste and was not a priority at that time because *NeuralShell* already provided all the necessary utilities to explore and present modeling results. It was less easy to use, but majority of model manipulation, interpretation and use in process optimization was performed by well trained members of development team anyway. When the funded period of the project ended, the promised funding for continuation of work was not provided, and *NeuralParametric* remained in a rather crude development stage.

Igor felt it would be nice to make the accomplished work on ANN-based modeling more accessible and understandable to wider audience through an intuitive GUI-based application. He was reluctant to start from *NeuralParametric*, which contained a number of awkward design flaws due to artificially induced nervousness and weird requirements imposed by some stakeholders when the software was created. A simpler and more logical design was set up, resulting in *AnnApp*.

Some References

Tools:

[1] *IMgBurn*, a software for recording CD, DVD and Blu-ray images to recordable media. Can be used to create bootable ISO images. http://www.imgburn.com/ .

[2] *Rufus*, a soffware for creating bootable USB drives. https://rufus.akeo.ie/ .

[3] *cdrtools*, collection of open source packages for writing optical disks (CDs, DVDs, Blu-Ray disks), written by Jörg Schilling, Eric Youngdale, Heiko Eißfeldt and James Pearson. http://cdrecord.org/ .

[4] *mkisofs*, a program originally from *cdrtools* with extended options, which enable creation of legacy and UEFI-bootable ISO files, written by Alex Kopylov (implementation of additional features w.r. *cdrtools*). http://cdob.reboot.pro/ .

[5] *Virtual Machine USB Boot* (*VMUB*), a program for booting virtual machines from USB drives. http://reboot.pro/files/file/339-virtual-machine-usb-boot/

[6]

[7]

[8]

Virtualization:

[9] VirtualBox, a free open source platform virtualization software created by Oracle. https://www.virtualbox.org/ .

[10] How to Install Ubuntu on VirtualBox. A wikiHow tutorial. http://www.wikihow.com/Install-Ubuntu-on-VirtualBox

[11] Create a Windows Virtual Machine in Linux / Windows. By Angelos Kyritsis. https://www.pcsteps.com/207-windows-virtual-machine-linux-windows/#Have_aWindows_CDDVD_or_ISO .

[12] Installing Ubuntu inside Windows using VirtualBox. Web tutorial. http://www.psychocats.net/ubuntu/virtualbox .

[13] Oracle VM VirtualBox. User manual, by Oracle Corporation. https://www.virtualbox.org/manual/

[14]

[15]

[16]

Download sites:

[17] Windows PE. Links to downloadable optimized ISO images of Windows PE with additional software. Musings of an IT Pro. http://windowsmatters.com/category/winpe/

[18] Gandalf's Windows 10PE x64 Threshold 2 build 10586. Windows 10 PE with usable software, December 2015. Musings of an IT Pro. http://windowsmatters.com/2015/12/29/gandalfs-win10pe-x64-version-12-29-2015/

[19] PortableApps.com. A source of numerous popular software in form of portable applications. http://portableapps.com/

[20]

[21]

Basic terms:

[22] Portable application. https://en.wikipedia.org/wiki/Portable_application

[23] PATH variable. https://en.wikipedia.org/wiki/PATH_%28variable%29

[24] Bootable. English word definition at Wiktionary. https://en.wiktionary.org/wiki/bootable

[25] BIOS (Basic Input/Output System). https://en.wikipedia.org/wiki/BIOS

[26] Master Boot Record (MBR). https://en.wikipedia.org/wiki/Master_boot_record

[27] Booting with MBR. https://en.wikipedia.org/wiki/Master_boot_record#System_bootstrapping

[28] UEFI (Unified Extensible Firmware Interface). https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

[29] GUID Partition Table (GPT). https://en.wikipedia.org/wiki/GUID_Partition_Table

[30] Booting. https://en.wikipedia.org/wiki/Booting

[31] Boot Loaders. https://en.wikipedia.org/wiki/Booting#Modern_boot_loaders

[32] GRUB, an example of boot loader. https://en.wikipedia.org/wiki/GNU_GRUB

[33] Boot Managers. https://en.wikipedia.org/w/index.php?title=Boot_manager&redirect=no

[34] Windows Vista startup process. Describes boot process in laterr Windows OS, applies also to later systems (Windows 8 and 10). https://en.wikipedia.org/wiki/Windows_Vista_startup_process

[35] Live CD. https://en.wikipedia.org/wiki/Live_CD

[36] El-Torito, a standard for creating bootable optical media. http://wiki.osdev.org/El-Torito

[37] mkisofs - Linux man page. http://linux.die.net/man/8/mkisofs

[38] mkisofs at OSDev.org - contains some additional information. http://wiki.osdev.org/Mkisofs

[39] SATA. https://en.wikipedia.org/wiki/Serial_ATA

[40] IDE. https://en.wikipedia.org/wiki/Parallel_ATA

[41]

[42]

[43]

Software supported by recipes from this tutorial:

[44] Optimization shell Inverse, http://www2.arnes.si/~ljc3m2/inverse/.

[45] *IOptLib* – library for solving inverse and optimization problems. Available at: http://www2.arnes.si/~ljc3m2/igor/ioptlib/

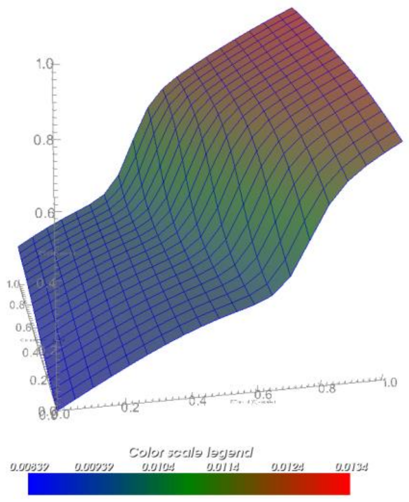
[46] *IGLib.NET* - investigative generic library, a library and framework for developing engineering, scientific and business applications in .NET. http://www2.arnes.si/~ljc3m2/igor/iglib/.

[47] *IGShell* - basic IGLib demonstrational and testing shell with some useful utilities. Available at: http://www2.arnes.si/~ljc3m2/igor/software/IGLibShellApp/ .

[48] *NeurApp* - a simple neural approximations application, educational software. Available at: http://www2.arnes.si/~ljc3m2/igor/software/neural_approximation/ .

[49] *AnnApp* - a software for exploring artificial neural networks-based modelling. Available at: http://www2.arnes.si/~ljc3m2/igor/software/AnnApp/index.html .

[50] *Aforge.Net*. (2012): Artificial intelligence library. Available at: http://www.aforgenet.com/.


Color scale legend
0.00839
0.00939
0.0104
0.0114
0.0124
0.0134