Linux Netboot

Petr Velička

March 23, 2022

1 Preface

The aim of this project is to create a bootable operating system image based on OpenSUSE Linux to be booted and fully operated from a network server.

There are two possible modes to boot an operating system from the network. One is to have a bootloader reside on the local hard drive which then loads the Linux kernel or a different operating system located on that hard drive (for example Microsoft Windows). This has the advantage that the configuration of the network doesn't have to be changed at all, the only requirement for it is to have an HTTP-capable server running anywhere on the network. Even remote servers located on the internet can be used for that purpose, although running the operating system from the network requires a steady network connection (with speed and latency of the internet not coming even close to those offered by local networks).

2 Kiwi

For the preparation of the system image, a tool maintained by the developers of SUSE Linux called Kiwi was chosen. Refer to the KIWI documentation (`https://osinside.github.io/kiwi/installation.html`) in order to get it installed and running on your machine.

All files used by kiwi reside in the `description` directory of the repository.

To build the system image, issue the following command in the terminal

```
sudo kiwi-ng --type kis system build \
      --description description/ \
      --target-dir /tmp/myimage
```

The resulting image is located in `/tmp/myimage`. This can be configured by altering the `--target-dir` parameter in the above mentioned command.

After copying it to the server (look for the kernel, initrd and squashfs images), make sure to also copy the server-side iPXE configuration which is a compulsory requirement for successfully booting the resulting system. Don't forget to change the server address in the file itself.

2.1 Image description

Main image description is located in `description/appliance.kiwi`. It contains meta information about the image as well as the packages which should be included in the resulting image.

For normal usage, only several parameters are needed, `<packages type="image">` section includes packages which will be installed onto the resulting system and `<repository type="rpm-md">` which lists repositories available during image bootstraping and later usage of the operating system itself.

The `root` subdirectory of the image contains files which are going to be copied into the system overwriting any files already existing there.

3 iPXE

Not every firmware supports direct booting of the Linux kernel from UEFI, therefore a separate program called bootloader is needed. For our case, iPXE is going to let the user choose either booting from network (downloading the script from the server and executing it) or just chainloading the system already installed on the locally present HDD. The exact process differs depending on the firmware of the machine (mainly whether (U)EFI is being used) but the overall principle is pretty much the same.

3.1 Building iPXE

The iPXE website (`https://ipxe.org/`) states that the git repository of the project is ought to be production-ready, therefore the process of building it consists merely of cloning the repository (`https://github.com/ipxe/ipxe.git`) and running `make` in the source directory.

Basic setup doesn't include anything useful to boot a system without user interaction though, therefore it is recommended to embed a script which automatically connects to a wired network and downloads a boot script located on a remote server. This is useful for changing boot parameters of the system without needing to rebuild and redeploy the bootloader to every client machine every time the configuration is changed.

For that reason, `client.ipxe` is included in the repo, which can be used to build a customized iPXE binary using `make EMBED=path/to/client.ipxe`. Make sure to edit the server address for it to point to the actual server where the image is located. Then, the resulting image for booting with GNU GRUB or other bootloaders capable of booting the Linux kernel (`https://www.gnu.org/software/grub/`) can be found in `bin/ipxe.lkrn`.

For EFI-capable machines, `ipxe.efi` in the `bin-x86_64-efi` subdirectory should be used which can be compiled using `make bin-x86_64-efi/ipxe.efi`, including the `EMDED` parameter if needed. Refer to your bootloader and/or motherboard documentation in order to figure out how to configure it to boot iPXE.

3.2 Diskless boot

It is possible to configure a client to boot completely diskless using iPXE. For that, `bin/ipxe.pxe` is used which is copied onto a TFTP server, to which the DHCP server points when giving an IP address to the machine. DHCP server can also be used to assign unique hostnames to the clients because the default image only has one hostname preconfigured.