HelenOS installer
Having low-level fun with HelenOS

Dominik Táborský
Department of Distributed and Dependable Systems,
Charles University in Prague
dominik.taborsky@zoho.com

Introduction

HelenOS is a modern operating system built from scratch. One of its missing features had been the ability to self-replicate, i.e., to install itself on a hard disk from its own running instance. This work aimed at solving this problem by tackling all the individual subproblems. Please note IA32 and AMD64 are the only processor architectures considered.

For self-hosted installation few things are needed:
1. Partition the disk.
2. Create filesystem(s).
3. Install a bootloader.
4. Install the rest of the system.

Filesystems have already been covered and installing the system itself is, fortunately, simple - it only requires file copying.

The rest is a problem, though. Partitioning and bootloader installation can only be performed from other operating system, e.g., a GNU/Linux distribution. Both can be worked around if HelenOS is not the only system to be installed on the host. But in the other case it is impossible to install HelenOS on its own. Therefore the main goal of this thesis is to provide tools sufficient for successful self-hosted HelenOS installation.

For partitioning Master Boot Record (MBR) and GUID Partition Table (GPT) labels were chosen. Master Boot Record is a legacy partition scheme designed in 1980s. It is probably the most widely used scheme to day. GUID Partition Table is the replacement for the old MBR. It has much cleaner design and offers more features, e.g., redundancy and checksums.

Therefore the main goal of this thesis is to provide tools sufficient for successful self-hosted HelenOS installation.

Methods

For partitioning Master Boot Record (MBR) and GUID Partition Table (GPT) labels were chosen. Master Boot Record is a legacy partition scheme designed in 1980s. It is probably the most widely used scheme to day. GUID Partition Table is the replacement for the old MBR. It has much cleaner design and offers more features, e.g., redundancy and checksums.

These are discussed individually below.

Objectives

These are our main objectives:
1. Develop a partition editor.
2. Port a bootloader.
3. Set up an installation procedure of the rest of the system.

These are discussed individually below.

Implementation

MBR and GPT were implemented as libraries libmbr and libgpt, respectively. They provide an interface that can be used in any other program. The partition editor is called hdisk.

Hdisk was designed to be only a user interface to both libmbr and libgpt. It is not, however, dependent on any of them. It is extensible by implementing a specific object and its methods, that is common to all label-manipulating libraries.

The user interface is simple, similar to that of fdisk known from GNU/Linux distributions. User inputs single letter to invoke each command which may request more input afterwards. Example of that is simply adding a partition. Some common commands are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Adds partition.</td>
</tr>
<tr>
<td>d</td>
<td>Deletes partition.</td>
</tr>
<tr>
<td>p</td>
<td>Prints current partition list.</td>
</tr>
<tr>
<td>s</td>
<td>Prints list of commands.</td>
</tr>
<tr>
<td>f</td>
<td>Re-reads partition records from the device.</td>
</tr>
<tr>
<td>w</td>
<td>Writes the partition list to the device.</td>
</tr>
</tbody>
</table>

Hdisk is also extensible beyond the basic command set. Any label extension can implement command of which may contain a new command set. Returning from this command resumes normal operation.

For porting GRUB Legacy port required many small changes in many different places. These were mostly simple, though, requiring including different header files and slightly different function calls, e.g., replacing strlen() with strnlen(). A major change was required to code simulating BIOS read/write functions. This had to be implemented using fdisk, the core service and IPC. Another change includes device name generation to allow looking up devices.

Finally, the easiest way to allow self-replication was chosen. The build procedure was adapted to output a RAM disk containing itself, the kernel and all the other modules. Although this change brought some disadvantages - doubled the RAM disk size and the loading time, it is simple and flexible enough for future improvements. When HelenOS no longer expects all root filesystem in RAM disk, or when device tree is stable enough so that all modules can be read from the install medium, then this adjustment is no longer needed and can be discarded.

Conclusion

All the objectives have been accomplished and the user now may install the system on a hard disk. Even though the solution is not optimal, HelenOS is expected to change substantially and therefore any complex solution might have come as a waste. However, hdisk and label-modifying libraries are well-written and useful in any case.

Possible improvements

Follow-up work may include porting GRUB 2, designing completely new partition scheme (similarly to, for example, BSD slices) or even implementing graphical partition editor.

HelenOS overview

HelenOS is an experimental, open source, general-purpose operating system built from scratch on top of its SPARTAN microkernel. It is a set of services, libraries and applications. One of its goals is portability so it runs on several different architectures. It also strives to be as modular as possible and eventually become a fully usable system.

Acknowledgements

I thank my family, my supervisor and anyone who supported me.