The Modern Mail Handler

master's thesis by

Markus Schnalke

Advised by

Prof. Dr. Franz Schweiggert
and Dr. Andreas Borchert

Ulm University

2012

At the heart of the UNIX philosophy is the idea that the power of a system comes more from the relationship among programs than from the programs themselves.

— Brian W. Kernighan and Rob Pike —

PREFACE

I have discovered the mail client *nmh* in September 2009. At that time I used to use *mutt*, as many advanced Unix users do. When I read about nmh, its concepts had convinced me at once. The transition from mutt to nmh was similar to managing files in the Unix shell when being used to the *midnight commander*, or like editing with vi when being used to modeless editors. Such a change is not trivial, but in being convinced by the concepts and by having done similar transitions for file management and editing already, it was not too difficult. In contrast, setting up nmh to a convenient state became a tedious task that took several months. Once having nmh arranged to a convenient state, I enjoyed using it because of its conceptional elegance and its scripting capabilities. On the other hand, nevertheless, it still was inconvenient for handling attachments, non-ASCII character encodings, and similar features of modern emailing. My setup demanded more and more additional configuration and helper scripts to get nmh behave the way I wanted, although my expectations were rather common for modern emailing. In being a computer scientist and programmer, I wanted to improve the situation.

In Spring 2010, I asked on the *nmh-workers* mailing list for the possibility to offer a Google Summer of Code project for me. Participating in the development of nmh this way appeared attractive to me, because I would have been able to work full time on nmh as the project could have been part of my official studies at university. Although the nmh community had been generally positive on the suggestion, the administrative work for a GSoC project had been to much to have it realized. Nontheless, my proposal had activated the nmh community. In the following weeks, goals for nmh's future were discussed. In these discussions, I became involved in the question whether nmh should include mail transfer facilities [ML:MTA-MUA]. In this central point, my opinion differed from the opinion of most others. I argued for the MTA facility of nmh to be removed. Besides the discussions, hardly any real work was done. Being unable to work on nmh in a way that would be accepted as part of my official studies, I needed to choose another project.

Half a year later, starting in August 2010, I took one semester off to travel through Latin America. During my time in Argentina, I planned to work on Free Software. This brought me back to nmh. Richard Sandelman, an active nmh user, cared for the official basis. Juan Granda, an argentine Free Software developer, provided a computer with Internet connection for my work. Thanks to them, I was able to work on nmh during my three-month stay in Santiago del Estero in Argentina. Quickly it became obvious that I wouldn't succeed with my main goal: improving the character encoding handling within the project. One of its ramifications is the missing transfer decoding of quoted text in replies. As this is one of the most intricate parts of the system, the goal was simply set too high. Instead, I improved the code base as I read through it. I found minor bugs for which I proposed fixes to the community. In the same go, I improved the documentation in minor ways. When I started with larger code changes, I had to discover that the community was reluctant to change. Its wish for compatibility was much stronger than its wish for convenient out-of-the-box setups – in contrast to my opinion. This led to long discussions, again. I came to understand their point of view, but it was different to mine. At the end of my three-month project, I had become

familiar with nmh's code base and community. I had improved the project in minor ways, and I still was convinced that I wanted to go on to do so.

Another half year later, the end of my studies came within reach. I needed a topic for my master's thesis. No question, I wanted to work on nmh. But well, not exactly on nmh, because I had accepted that the nmh community has different goals than I have. This would result in much discussion and thus little progress. After careful thought, I decided to start an experimental version of nmh. I wanted to implement my own ideas of how an MH-like system should look like. I wanted to create a usable alternative version to be compared with the present state of nmh. Eventually, my work would be proven successful or not. In any case, the nmh project would profit from my experiences.

Focus of this Document

This document explains the design goals and implementation decisions for mmh. It discusses technical, historical, social and philosophical considerations. On the technical side, this document explains how an existing project was stream-lined by removing rough edges and exploiting the central concepts better. On the historical side, changes through time in the use cases and the email features, as well as the reactions to them, are discussed. Socially, this document describes the effects and experiences of a newcomer with revolutionary aims entering an old and matured software project. Philosophical thoughts on style, mainly based to the Unix philosophy, are present throughout the discussions. The document describes the changes to nmh, but as well, it clarifies my personal perception of the concepts of MH and Unix, and explain my therefrom resulting point of view.

This document is written for the community around MH-like mail systems, including developers and users. Despite the focus on MH-like systems, this document is may be precious to anyone interested in the Unix philosophy and anyone in contact to old software projects, be it code or community-related.

The reader is expected to be well familiar with Unix, C and emailing. Good Unix shell knowledge is required, because MH relies fundamentally on the shell. Without the power of the shell, MH becomes a motorbike without winding roads: boring. Introductions to Unix and its shell can be found in ''The UNIX Programming Environment'' by Kernighan and Pike [Kernighan84] or ''The UNIX System'' by Bourne [Bourne82]. The reader is assumed to be a C programmer, but the document should be understandable otherwise, too. The definitive guide to C is Kernighan and Ritchie's ''The C Programming Language'' [Kernighan88]. Some book about system-level C programming can be helpful additional literature. Rochkind and Curry have written such books [Rochkind85, Curry96]. As large parts of the source code are old, old books are likely more helpful for understanding. The reader is expected to know the format of email messages and the structure of email transfer systems, at least on a basic level. It's advisable to have cross-read the RFCs 821 and 822. Further more, basic understanding of MIME is good to have. The Wikipedia provides good introduction-level information to email.

Frequent references to the Unix philosophy will be made. Gancarz has tried to sum it up in his book ''The UNIX Philosophy'' [Gancarz95]. Even better, though less concrete, are ''The UNIX Programming Environment'' [Kernighan84] and ''The Practice of Programming'' [Kernighan99] by Kernighan and Pike. The term paper ''Why the Unix Philosophy still matters'' [Schnalke10] by myself provides an overview on the philosophy, including a case study of MH.

Although a brief introduction to MH is provided in Chapter 1, the reader is encouraged to have a look at the *MH Book* ''MH & nmh: Email for Users & Programmers'' by Jerry Peek [Peek95]. The current version is available freely on the Internet. It is the definitive guide to MH and nmh.

This document is neither a user's tutorial to mmh nor an introduction to any of the topics covered. The technical discussions are on an advanced level. Nevertheless, as knowledge of the fundamental concepts is the most valuable information a user can acquire about some program or software system, this document may be worth a read for non-developers as well.

Organization

Which font for what use. Meaning of 'foo(1)'. RFCs.

This thesis is divided into XXX chapters, ...

Chapter 1 introduces ...

Chapter 2 describes ...

Chapter 3 covers ...

Acknowledgments

To be written at the very end.

Chapter 1
INTRODUCTION

MH is a set of mail handling tools with a common concept, similar to the Unix tool chest, which is a set of file handling tools with a common concept. *nmh* is the currently most popular implementation of an MH-like mail handling system. This thesis describes an experimental version of nmh, named *mmh*.

This chapter introduces MH, its history, concepts and how it is used. It describes nmh's code base and community to give the reader a better understanding of the state from which mmh started off. Further more, this chapter outlines the mmh project itself, describing the motivation for it and its goals.

1.1 MH – THE MAIL HANDLER

MH is a conceptual email system design and its concrete implementation. Notably, MH had started as a design proposal at RAND Corporation, where the first implementation followed later. In spirit, MH is similar to Unix, which influenced the world more in being a set of system design concepts than in being a specific software product. The ideas behind Unix are summarized in the *Unix philosophy*. MH follows this philosophy.

History

In 1977 at RAND Corporation, Norman Shapiro and Stockton Gaines had proposed the design of a new mail handling system, called ''Mail Handler'' (MH), to superseed RAND's old monolithic ''Mail System'' (MS). Two years later, in 1979, Bruce Borden took the proposal and implemented a prototype of MH. Before the prototype had been available, the concept was believed to be practically unusable. But the prototype had proven successful and replaced MS thereafter. In replacing MS, MH grew to an all-in-one mail system.

In the early Eighties, the University of California at Irvine (UCI) had started to use MH. Marshall T. Rose and John L. Romine became the driving force then. They took over the development and pushed MH forward. RAND had put the code into the public domain by then. MH was developed at UCI at the time when the Internet appeared, when UCB implemented the TCP/IP stack, and when Allman wrote Sendmail. MH was extended as emailing became more featured. The development of MH was closely related to the development of email RFCs. In the advent of MIME, MH was the first implementation of this new email standard.

In the Nineties, the Internet had become popular and in December 1996, Richard Coleman initiated the ''New Mail Handler'' (nmh) project. Nmh is a fork of MH 6.8.3 and bases strongly on the *LBL changes* by Van Jacobson, Mike Karels and Craig Leres. Colman intended to modernize MH and improve its portability and MIME handling capabilities. This should be

done openly within the Internet community. The development of MH at UCI stopped after the 6.8.4 release in February 1996, soon after the development of nmh had started. Today, nmh has almost completely replaced the original MH. Some systems might still provide old MH, but mainly for historical reasons.

In the last years, the work on nmh was mostly maintenance work. However, the development revived in December 2011 and stayed busy since then.

Concepts

MH consists of a set of tools, each covering a specific task of email handling, like composing a message, replying to a message, refiling a message to a different folder, listing the messages in a folder. All of the programs operate on a common mail storage.

The mail storage consists of *mail folders* (directories) and *messages* (regular files). Each message is stored in a separate file in the format it had been received (i.e. transfer format). The files are named with ascending numbers in each folder. The specific format of the mail storage characterizes MH in the same way like the format of the file system characterizes Unix.

MH tools maintain a *context*, which includes the current mail folder. Processes in Unix have a similar context, containing the current working directory, for instance. In contrast, the process context is maintained by the Unix kernel automatically, whereas MH tools need to maintain the MH context themselves. The user can have one MH context or multiple ones, he can even share it with other users.

Messages are named by their numeric filename, but they can have symbolic names, too. These are either automatically updated position names like being the next or the last message, or user-settable group names for arbitrary sets of messages. These names are called sequences. Sequences can be bound to the containing folder or to the context.

The user's *profile* is a file that contains his MH configuration. Default switches for the individual tools can be specified to adjust them to the user's personal preferences. Multiple versions of the same command with different default values can also be created very easily. Form templates for new messages or for replies are easily changeable, and output is adjustable with format files. Almost every part of the system can be adjusted to personal preference.

The system is well scriptable and extensible. New MH tools are built out of or on top of existing ones quickly. Further more, MH encourages the user to tailor, extend and automate the system. As the MH tool chest was modeled after the Unix tool chest, the properties of the latter apply to the former as well.

Using MH

It is strongly recommended to have a look at the MH Book, which introduces well into using MH [Peek95, Part II]. Rose and Romine provide a deeper and more technical though slightly outdated introduction in only about two dozens pages [Rose85].

Following is an example mail handling session. It uses mmh but is mostly compatible with nmh and old MH. Details might vary but the look'n'feel is the same.

```
$ inc
Incorporating new mail into inbox...

   1+ 2012-05-16 11:16  meillo@dream.home  Hello
   2  2012-05-16 11:17  meillo@dream.home  book

$ show
Date:    Wed, 16 May 2012 11:16:00 +0200
To:      meillo
From:    <meillo@dream.home.schnalke.org>
Subject: Hello

part       text/plain                 13
mmh is great

$ next
Date:    Wed, 16 May 2012 11:17:24 +0200
To:      meillo
From:    <meillo@dream.home.schnalke.org>
Subject: book

part       text/plain                 79
Hello meillo,

have a look at the ''Daemon book''. You need to read that!

foo

$ rmm 1

$ scan
   2+ 2012-05-16 11:17  meillo@dream.home  book

$
```

1.2 NMH: CODE AND COMMUNITY

In order to understand the condition, goals and dynamics of a project, one needs to know the reasons. This section explains the background.

MH predates the Internet, it comes from times before networking was universal, it comes from times when emailing was small, short and simple. Then it grew, spread and adopted to the changes email went through. Its core-concepts, however, remained the same. During the Eighties students at UCI actively worked on MH. They added new features and optimized the code for the then popular systems. All this still was in times before POSIX and ANSI C. As large parts of the code stem from this time, today's nmh source code still contains many

ancient parts. BSD-specific code and constructs tailored for hardware of that time are frequent.

Nmh started about a decade after the POSIX and ANSI C standards had been established. A more modern coding style entered the code base, but still a part of the developers came from ‘‘the old days’’. The developer base became more diverse and thus resulted in code of different style. Programming practices from different decades merged in the project. As several peers added code, the system became more a conglomeration of single tools rather than a homogeneous of-one-cast mail system. Still, the existing basic concepts held it together. They were mostly untouched throughout the years.

Despite the tool chest approach at the surface – a collection of separate small programs – on the source code level it is much more interweaved. Several separate components were compiled into one program for efficiency reasons. This lead to intricate innards. Unfortunately, the clear separation on the outside appeared as being pretty interweaved inside.

The advent of MIME rose the complexity of email by a magnitude. This is visible in nmh. The MIME-related parts are the most complex ones. It's also visible that MIME support had been added on top of the old MH core. MH's tool chest style made this easily possible and encourages such approaches, but unfortunately, it lead to duplicated functions and half-hearted implementation of the concepts.

To provide backward-compatibility, it is a common understanding to not change the default settings. In consequence, the user needs to activate modern features explicitly to be able to use them. This puts a burden on new users, because out-of-the-box nmh remains in the same ancient style. If nmh is seen to be a back-end, then this compatibility surely is important. However, in the same go, new users have difficulties to use nmh for modern emailing. The small but matured community around nmh hardly needs much change as they have their convenient setups since decades.

1.3 MMH

I started to work on my experimental version in October 2011, at a time when there were no more than three commits to nmh since the beginning of the year. In December, when I announced my work in progress on the nmh-workers mailing list [ML:mmh_ann], nmh's community became active, too. This movement was heavily pushed by Paul Vixie's ‘‘edginess’’ comment [ML:edginess]. After long years of stagnation, nmh became actively developed again. Hence, while I was working on mmh, the community was working on nmh, in parallel.

The name *mmh* may stand for *modern mail handler*, because the project tries to modernize nmh. Personally however, I prefer to call mmh *meillo's mail handler*, emphasizing that the project follows my visions and preferences. (My login name is *meillo*.) This project model was inspired by *dwm*, which is Anselm Garbe's personal window manager – targeted to satisfy Garbe's personal needs whenever conflicts appear. Dwm had retained its lean elegance and its focused character, whereas its community-driven predecessor *wmii* had grown fat over time. The development of mmh should remain focused.

Motivation

MH is the most important of very few command line tool chest email systems. Tool chests are powerful because they can be perfectly automated and extended. They allow arbitrary kinds of front-ends to be implemented on top of them quickly and without internal knowledge. Additionally, tool chests are much better to maintain than monolithic programs. As there are few tool chests for emailing and as MH-like ones are the most popular among them they should be developed further. This keeps their conceptional elegance and unique scripting qualities available to users. Mmh will create a modern and convenient entry point to MH-like systems for new and interested users.

The mmh project is motivated by deficits of nmh and my wish for general changes, combined with the nmh community's reluctancy to change.

nmh hadn't adjusted to modern emailing needs well enough. The default setup was completely unusable for modern emailing. Too much setup work was required. Several modern features were already available but the community didn't wanted to have them as default. mmh is a way to change this.

In my eyes, MH's concepts could be exploited even better and the style of the tools could be improved. Both would simplify and generalize the system, providing cleaner interfaces and more software leverage at the same time. mmh is a way to demonstrate this.

In providing several parts of an email system, nmh can hardly compete with the large specialized projects that focus on only one of the components. The situation can be improved by concentrating the development power on the most unique part of MH and letting the user pick his preferred set of other mail components. Today's pre-packaged software components encourage this model. mmh is a way to go for this approach.

It's worthwhile to fork nmh for the development of mmh, because the two projects focus on different goals and differ in fundamental questions. The nmh community's reluctance to change conflicts with my strong will to change. In developing a separate experimental version new approaches can easily be tried out without the need to discuss changes beforehand. In fact, revolutionary changes are hardly possible otherwise.

The mmh project provides the basis to implemented and demonstrated the listed ideas without the need to change nmh or its community. Of course, the results of the mmh project shall improve nmh, in the end.

Target Field

Any effort needs to be targeted towards a specific goal in order to be successful. Following is a description of the imagined typical mmh user. mmh should satisfy his needs. Actually, as mmh is my personal version of MH, this is a description of myself.

The target user of mmh likes Unix and its philosophy. He likes to use programs that are conceptionally appealing. He's familiar with the command line and enjoys its power. He is at least capable of shell scripting and wants to improve his productivity by scripting the mail system. He naturally uses modern email features, like attachments, non-ASCII text, and digital cryptography. He is able to setup email system components besides mmh, and actually likes the choice to pick the ones he prefers. He has a reasonably modern system that complies to standards, like POSIX and ANSI C.

The typical user invokes mmh commands directly in an interactive shell session, but as well, he uses them to automate mail handling tasks. Likely, he runs his mail setup on a server machine, to which he connects via ssh. He might also have local mmh installations on his workstations, but does rather not rely on graphical front-ends. He definitely wants to be flexible and thus be able to change his setup to suite his needs.

The typical mmh user is a programmer himself. He likes to, occasionally, take the opportunity of Free Software to put hands on and get involved in the software he uses. Hence, he likes small and clean code bases and he cares for code quality. In general, he believes that:

- Elegance – i.e. simplicity, clarity and generality – is most important.
- Concepts are more important than the concrete implementation.
- Code optimizations for anything but readability should be avoided if possible.
- Having a lot of choice is bad.
- Removed code is debugged code.

Goals

The general goals for the mmh project are the following:

Stream-lining
: Mmh should be stripped down to its core, which is the MUA part of emailing. The feature set should be distilled to the ones really needed, effectively removing corner-cases. Parts that don't add to the main task of being a conceptionally appealing MUA should be removed. This includes, the MTA and MRA facilities. Choice should be reduced to the main options.

Modernizing
: Mmh's feature set needs to become more modern. Better support for attachment and digital cryptography needs to be added. MIME support needs to be integrated deeper and more naturally. The modern email features need to be readily available, out-of-the-box. And on the other hand, bulletin board support and similar obsolete facilities need to be dropped out. Likewise, ancient technologies, like hardcopy terminals, should not be supported any further.

Code style
: Mmh's source code needs to be updated to modern standards. Standardized library functions should replace non-standard versions whenever possible. Code should be separated into distinct modules when possible. Time and space optimizations should to be replaced by clear and readable code. A uniform programming style should prevail.

Homogeneity
: The available concepts need to be expanded as far as possible. A small set of concepts should prevail thoroughly throughout the system. The whole system should appear to be of-one-style. It should feel like being cast as one.