A weeks worth of claws-mail on OSX

After installing Snow Leopard I was left with a page of programs and plugins that needed attention because of the upgrade. Most of these were little nuisances and just needed a little reconfiguration. There were a couple of things where Apple decided to rename an app or hide it otherwise from view (Kerberos Ticket manager for example).

There were two things on the list which stopped me in my daily work. The first was was the need to recompile the complete macports collection. I won't go into the details of everything, but executing a complete reinstall of macports apps takes about a day, not counting the time needed to fix compile errors (wtf, all *.la files missing now?).

The second was the disfunctioning of a number of Mail.app plugins. I guess it's not really the plugin authors who have dropped the ball here, as Apple had an early release of SL and many developers were just not ready. However, that didn't change the fact my *nix tools and mail were both severely crippled now.

Not looking forward to spending two days to recompile and reconfigure everything and nothing new to show for I decided that a long time wish could be sneaked in. Leaving Mail.app behind and switching to Claws as a mail client.

Specifically, these are the main gripes I have with Mail.app:

- keyboard navigation is absent for all practical purposes;
- tabbing order in compose window or absence of shortcut keys to go to subject/body at once;
- weird threading decision; (altho i have to admit it works for the daily chores mail)
- no way to configure top/bottom posting;
- no nntp support;
- IMAP support does not include subscriptions;
- somehow it keeps locking up on me with tenacious regularity, i suspect sqlite here;
- the direction with yellowy sticky notes, the stationary things are all useless to me;
- proprietary, closed program (which I could live with. . .)
- but worse, it has a non-documented API for plugins, which is an insult to developers.

I've always liked claws on linux. Tremendously fast, great keyboard support, a feature-set which is more than I can handle in most areas and a small enough codebase which makes it realistic for me to patch it or write a plugin for, should I feel inclined to do so. The only problem I really had was running OSX, so the lightweightedness would be kinda lost, as I would need a while slew of dependencies. Still, most of them I already had because of our internal implementation of OpenERP

I knew I would be running into issues, so the first couple of days I ran with claws-mail inside VirtualBox running an Ubuntu install. This would give me sort of a reference to get everything working without having to worry about compilation digging etc. Installing a new mail application is easy but deciding if it is good enough to replace a program which you use every minute of the day is something else.

I sort of knew after half a day I was going to like it. The slowness of the virtual machine was a little bit in the way, as was its instability, but overall things were looking good. Ok, time to repeat the excercise for real on OSX.

Here's the configure step I used:

```
export CPPFLAGS=-I/opt/local/include
export LDFLAGS=-L/opt/local/lib

./configure \
  --disable-trayicon-plugin \
  --disable-manual \
  --enable-ipv6 \
  --disable-dillo-viewer-plugin \
  --enable-crash-dialog
```

The first two are perhaps redundant, but I wanted to make sure everything came from the ports collection and not from libraries supplied by Apple by default. Compiling afterwards gives you a running, but ugly claws-mail. So, first thing I did was install a bunch of gtk2 stuff to give me a bit of theme-ing options. I used Platypus to create a Claws.app; the OSX application bundle so I can have an icon in the Dock:

Apart from the icon, that does not give you much really. The app runs as a child below X11.app so it still misbehaves in many ways, but alas.

Next up was 'mailto' URI handling. I had thought this to be an easy thing. OSX refuses to accept a shell script as handler though, it must be an app bundle. So, what I needed was an app-bundle, as invisible as possible which can act as an URI handler for claws. The first thing I thought about was adapting MailToMutt to call out to claws instead of mutt. Looking at the source I saw it was pretty much suitable to mutt only, so I set out to write my own, How hard could this be?

As it turns out, such an app in its simplest form would require the following:

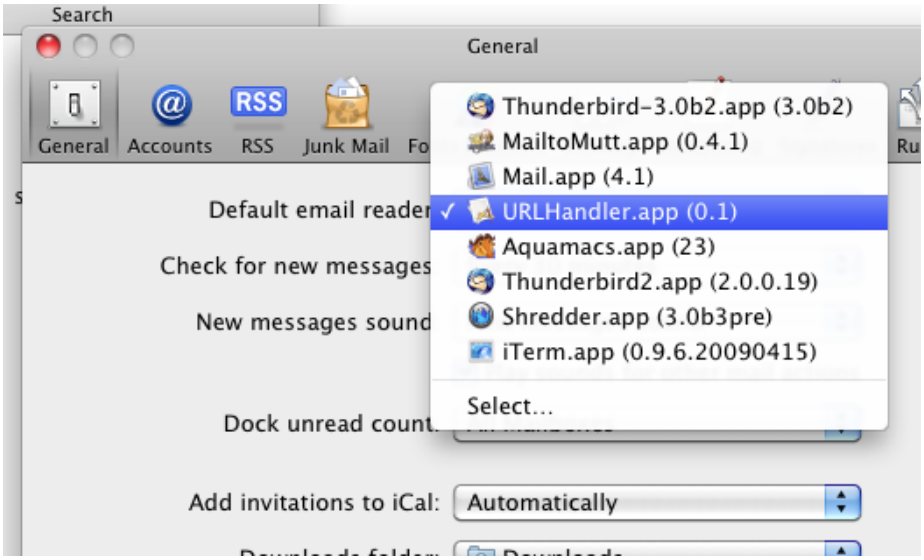
1. An info.plist file, as all app bundles have, with some special entries;
2. A declaration of a handler on how to react to the involved 'Apple Events'
3. The code of that handler.

The plist has to look something like this:

There are 2 things in there which make this interesting. First, the highlighted line

marks this application as an 'Agent' which just means to not show it anywhere on screen.

Second, a list of URL-types on which this URIhandler app should react. As you can see I threw a bunch in there which I thought would come in handy someday. What this does is make the app known to OSX as being capable of handling these types of URLs. This is used, for example by the Mail.app application for listing clients in its preference for default Email program:



The second file needed in the XCode project is URLHandler.scriptSuite:

```
{
    Name = URLHandler;
    AppleEventCode = "UrHD";

    Commands = {
        "GetURL" = {
            CommandClass = URLHandlerCommand;
            AppleEventCode = GURL;
            AppleEventClassCode = GURL;
        };
    };
}
```

which registers `URLHandlerCommand` as the handler for Events of type GURL (get an URL).

And finally, the third file, `URLHanderlerCommand.m` which implements that handler:

```
@implementation URLHandlerCommand

- (id)performDefaultImplementation {

    NSString *urlString = [self directParameter];
    NSURL    *url = [NSURL URLWithString: urlString];

    // Log what we got
    NSLog(@"url = %@", urlString);

    // Launch _handler script in path
    NSTask *task = [NSTask new];
    NSString *script = [NSString stringWithFormat:@"~/bin/%@_handler",[url scheme]];
    NSLog(@"launchtarget = %@", script);
    [task setLaunchPath:script];

    // Give the URI-string as parameter to that script
    [task setArguments:[NSArray arrayWithObject:urlString]];

    // Catch stdout / stderr
    [task setStandardOutput:[NSPipe pipe]];
    [task setStandardError:[task standardOutput]];

    // Run it
    [task launch];

    // Log the output, if any
    NSData* output = [[[task standardOutput] fileHandleForReading] readDataToEndOfFile];
    NSString* out_string = [[[NSString alloc] initWithData:output encoding:NSUTF8StringEncod
    NSLog(@"%@",out_string);

    // If we are not there we cant do no harm, so quit.
    [[NSApplication sharedApplication] terminate:nil];

    return nil;
}
@end
```

In words: It takes the URL that the user clicked on "mailto:somethin g@something.org", turns it into an URL object and calls out to the file <code>mailto$_{\text{handler}}$</code> in the Users bin directory.

For a 'feed' URL the app would call the file `feed_handler` and likewise for the other protocols. The `mailto_handler` file is just a shell script containing:

```
/usr/local/bin/claws-mail --compose $$1
```

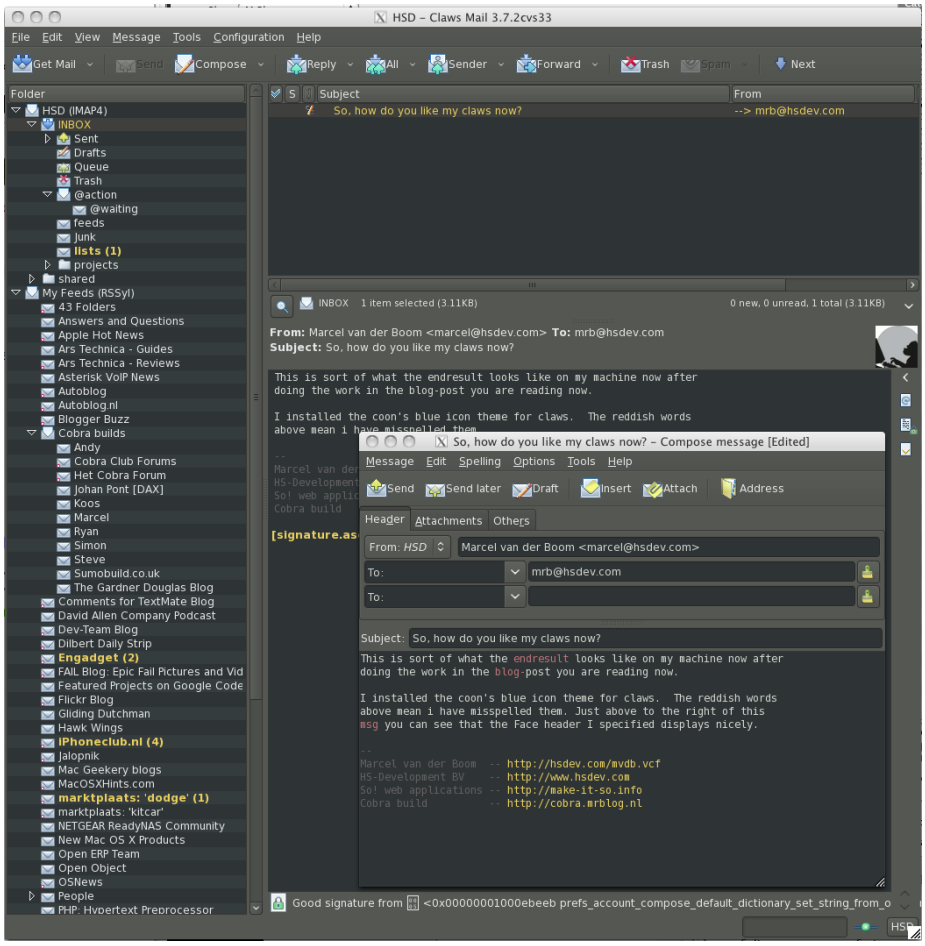
and my `~/bin/feed_handler` contains:

```
/usr/local/bin/claws-mail --subscribe $$1
```

If you would like a copy of the source of this program, go here: [/files/2009/09/URLHandler.zip](/files/2009/09/UI

The rest of the configuration went pretty smooth. I had a little hickup getting GPG signing to work with both PGP and S/MIME signatures. But a quick recompile of gpgme to include s/mime and a [specific configuration](http://www.beitz.org/node/85) which was not obvious to me for gpg-agent solved that.

Here's a picture of the end result (click on it for large version):



So, is this install perfect? Is it even better than Mail.app? I'd say, not yet. But the big difference is that I have control now and there's only a gap of time and energy between me solving any gripe I might have.

Which of the gripes have I solved with this? Let's recap:

I'd say that is a pretty good score! If it feels like that in the real world? Talk to

me in 3 months :D I'm also pretty sure there will be a whole set of new gripes waiting for me around the corner.