電子メールクライアント Sylpheed の内部構造

山本 博之
株式会社グッデイ
yamamoto@good-day.co.jp

平成 13 年 7 月 31 日

1 はじめに

Sylpheed[1] は GTK+ ベースの電子メールクライアントであり，軽快，高速，高機能，そして優れたユーザインタフェースを目標にして開発されている．

Sylpheed の開発における哲学とも呼べるものは「美しさ」である．ここでいう美しさとは，単なる外観にとどまらず，設計や実装の美しさ，そしてコードの美しさといったあらゆることを含んでいる．

本論文ではこれまであまりドキュメントとして公開されることのなかった Sylpheed の内部構造について詳細に解説する．また， Sylpheed の開発を通して明らかになったオープンソースによる開発の利点についても触れる．

2 設計と実装

2.1 基本的な設計方針

Sylpheed は C 言語で記述されているが，様々なオブジェクト指向の概念，例えば抽象化，メッセージパッシング，継承，ポリモーフィズム (多態) といったものを積極的に採り入れている．その実装には GTK+ にヒントを得ている．また，各機能のモジュール化が積極的に行われ，ユーザインタフェースと実際にデータ処理などを行うロジック部分の分離が進められている．これは，コードの見通しを良くし，保守性，拡張性の向上といったことを実現するためである．

2.2 ユーザインタフェース

従来の Unix 系のアプリケーションでは，ユーザインタフェースについて真剣に考慮されることが少なく，逆に軽視されてきた面もあるように思う．Sylpheed にはそのような状況を打破するという目的もある．

Sylpheed のユーザインタフェースはできるだけ直観的でかつ効率的なものを目指している．画面構成は Windows 系メーラでは標準的で，現時点では UI として最も優れていると思われる 3 ペイン方式を採用し，またキー操作については， Unix 系ではデファクトスタンダードともいえる Mew と Wanderlust のものを可能な限り採用している．

Sylpheed では GUI ツールキットとして GTK+ を使用しているが，標準のウィジェットだけでは機能的に不十分なことも多い．そのため，Sylpheed ではいくつか独自のウィジェットを実装して機能を拡張している．現在メインツリーにあるものは GtkSCTree と GtkSHRuler である．現在別ブランチで GtkText を改良した GtkSText が開発中である．

また，将来的には GTK+ 以外，たとえばコンソールモードなどのインタフェースも提供する予定である．さらに， Mac OS X の Cocoa 環境に移植することも視野に入れている．

UI の各部は，図 1 に示すようにそれぞれが別個のオブジェクトとして設計されている．各オブジェクトは互いに情報を受け渡し合い，全体として協調して動作を行っている．

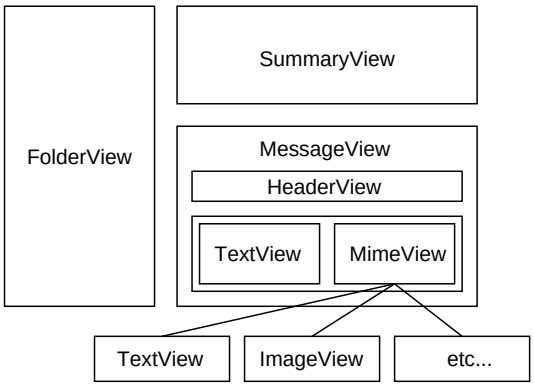

図 1: UI オブジェクトの構造

FolderView

FolderView は Sylpheed が内部で持っているメールボックスのフォルダの階層構造を目に見える形にするためのオブジェクトである.

SummaryView

SummaryView は現在開かれているフォルダ内のメッセージを一覧として表示するためのオブジェクトである.

MessageView

MessageView は SummaryView で選択されているメッセージの内容を表示するためのオブジェクトである.

MessageView はさらにいくつかのオブジェクトから構成されている. HeaderView はヘッダの内容を簡略表示するためのオブジェクトである. X-Face の画像もここに表示される. TextView は通常のプレーンテキスト形式のメッセージを表示する. MimeView は MIME による様々な形式のデータを表示し, マルチパート MIME にも対応する. MimeView はデータ形式に応じて TextView, ImageView といったオブジェクトを切り替えて表示する.

2.3 メールボックスの構造

Sylpheed ではメールボックスについてもオブジェクト的手法を用いて管理している. それは図 2 のように階層構造を持ったクラスで実現されている. 先頭に Folder クラスがあり, ローカルフォルダとリモートフォルダに対してそれぞれ LocalFolder と RemoteFolder クラスが派生する. これらからさらに各種フォーマットやプロトコルに応じて MHFolder, MboxFolder, MaildirFolder, IMAPFolder, NewsFolder のように派生する[1].

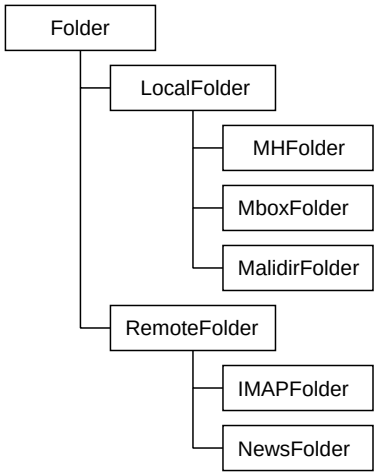

図 2: Folder クラスの構造

Folder クラスは図 3 のようなメソッドを持ち, 派生したクラスに応じて実装の内容は異なる. これによりどの形式のフォルダに対しても同じ API を提供することができる (オブジェクト指向の世界では多態, ポリモーフィズムと呼ばれる).

Folder クラスはまたメールボックスのツリー構造を持ち, ツリーの各ノードの情報は FolderItem オブジェクトに保持される (図 4). Sylpheed 内部では Folder オブジェクトを図 4 のように連結リストを用いて保持している.

[1] MboxFolder, MaildirFolder は未実装.

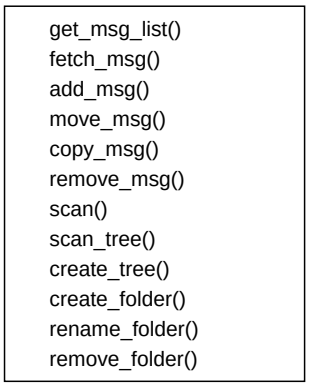

図 3: Folder クラスのメソッド

2.4 サーバとの通信

Sylpheed はサーバとの通信時においてもオブジェクトを作成して管理している．開発の経緯とプロトコルの特性により，POP3 と NNTP, IMAP4 では実装方法が異なっている．

POP3 プロトコルによる受信時には， GTK+ のソケット監視機能と，有限状態オートマトン[2]を応用して，処理の簡略化を行っている．POP3 を状態遷移図で表すと図 5 のようになるが， Sylpheed ではこれをそのままコードで表現している．実際には，各状態をそれぞれ関数で表し，関数の戻り値で次の状態を決定している．サーバから応答があると，現在の状態に応じた関数が呼ばれ，適切な処理を行った後，状態を遷移して次の応答を待つ．処理が終了すれば自動的に接続を切断する．

NNTP あるいは IMAP4 においては，接続が必要になった時点 (フォルダやメッセージに対するアクセスがあったときなど) で自動的に接続を行う．接続後，その状態を Session オブジェクトとしてタイムアウトになるまで常に保持する．ユーザからの入力があった時点で Folder オブジェクトのメソッドが呼び出され， Session オブジェクトを経由してサーバとの通信を行う (図 6)．

[2] 自動機械などと呼ばれる．内部状態を持ち，入力データを受け取ったときに規則に従って次の状態に遷移する仮想的な機械．有限とは，状態数が有限であるという意味．

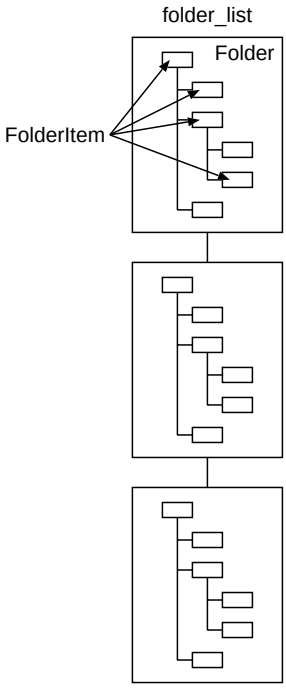

図 4: フォルダの構造

2.5 データの保存形式

Sylpheed ではアドレス帳とフォルダ情報の保存に XML (eXtensible Markup Language) フォーマットを用いている．XML パーサには機能を最小限に抑えた独自のコンパクトな実装を用いている．フォーマットに XML を採用した理由としては，木構造の表現や将来のデータ形式の拡張が容易であるということがあげられる．

具体的には，アドレス帳のデータは図 7 のような形式で保存されている．

2.6 文字コード変換モジュール

Sylpheed では文字コードの処理のために libkcc と libjconv の二種類のライブラリを用いている．libkcc は日本語文字コードの変換フィルタである kcc をライブラリ化したものであり，それに autoconf 化など少し手を加えたものを Sylpheed に統合している． libjconv は iconv のラッパーとして Kondara チームにより開発されたもので

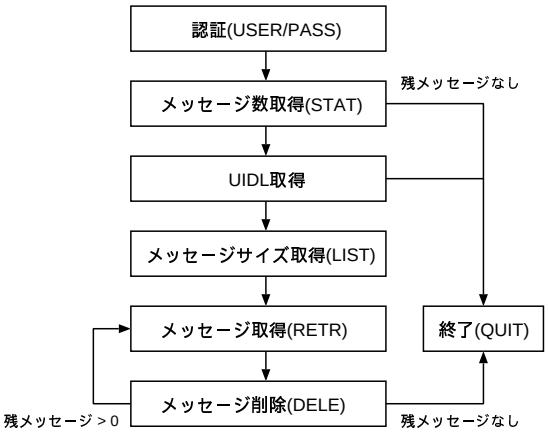

図 5: POP3 の状態遷移図

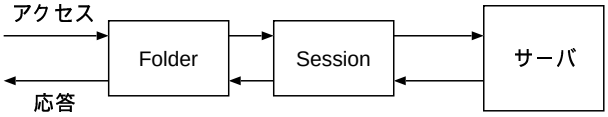

図 6: Session の概念図

ある．libkcc で扱えない文字コード (UTF-8，日本語以外の文字など) に関してはこちらを用いている．

Sylpheed では文字コード変換用のモジュールとして codeconv.c を用意し，これらの処理を抽象化している．そのため，将来文字コード周りの実装を変更したとしても，コードのあちこちを変更して回ることがないようになっている．

原則として，入力コードはメッセージの `Content-Type` ヘッダにより判定し，表示あるいは出力コードは現在指定されている locale，つまり `LANG` や `LC_CTYPE` などの環境変数を元に決定する．`Content-Type` が指定されていない場合は，入力コードは本来なら `US-ASCII` として扱うべきであるが，利便性を考慮して現在の locale にとって適切であると思われる文字コードを仮定する (`ja_JP.eucJP` の場合は `ISO-2022-JP`)．入出力コードについてはユーザが指定することも可能であるが，GTK+ の実装からくる制限のため，処理できるコードは現在の locale が扱えるものに限られる．

```
<?xml version="1.0" encoding="EUC-JP"?>
<addressbook>

<common_address>
    <folder name="ML">
        <folder name="Linux">
            <item>
                <name>linux-users</name>
                <address>linux-users@linux.or.jp</address>
                <remarks>Linux Users Mailing List</remarks>
            </item>
            <item>
                <name>debian-users</name>
                <address>debian-users@debian.or.jp</address>
                <remarks>Debian Users Mailing List</remarks>
            </item>
        </folder>
        <item>
            <name>Sylpheed-JP ML</name>
            <address>sylpheed-jp@good-day.net</address>
            <remarks>Sylpheed ML (Japanese)</remarks>
        </item>
    </folder>
</common_address>

<personal_address>
</personal_address>

</addressbook>
```

図 7: アドレス帳のデータ形式

3　オープンソースによる開発

Sylpheed は GPL2 に基づいて配布されており，その開発体制もオープンなものとなっている．開発中のコードは CVS を通じていつでも入手することができるようになっている．

ここでは，Sylpheed の開発の経緯と，現在の開発体制について説明する．

3.1　開発の開始と発展

Sylpheed の開発を開始したのは 1999 年の 9 月頃である．その頃は Linux 上で Mew と Wanderlust を使用していたが，その処理速度の遅さに納得がいかなかったのが開発を開始した動機の一つである．

Sylpheed を初めて正式に公開 (バージョン 0.1.0) したのが 2000 年 1 月 1 日である．驚いたのが，公開後ほんの数日でいくつかのバグ報告や Solaris へ対応するためのパッチなどをいただいたことである．オープンソースの利点については Eric S. Raymond 氏の論文などを通して知っていたが，実際にそれを体験できたことは大きな収穫であった．この流れは現在でも続いており，バグ報告や要望，パッチなどが頻繁に届いている．

自分でも予想外だったことが，海外からの反響がかなりあったことである．Freshmeat[2] で 2000 年 8 月頃に初めてアナウンスしたところ予想以上の反響があり，それによって国際化の実装も急速に進展した (それまでは日本語に決め打ちされている部分がかなりあった)．それまでは日本語のメーリングリストのみであったが，海外からの参加者がかなり増えてきたこともあり英語のメーリングリストを作ることとなった．現在では英語のメーリングリストのほうが一日の投稿数が多いほどである．

3.2 現在の開発体制

現在では紆余曲折を経て，二つの CVS ブランチが稼働している．私のみが管理しているメインブランチと， Sourceforge 上で稼働しているブランチ [3] がある．前者は正式なリリースのためのものであり，後者は新しい実装を積極的に試すものである．両者は定期的に変更点をマージしている．

本来私は複数のブランチを作ることは好まなかったのであるが，開発者の増加に伴って自分が管理しきれる以上のパッチが頻繁に届くようになってしまい，パッチを取り込んでいるだけで時間がなくなってしまうという状況が起こっていた．そのために，ある開発者の提案により別ブランチを作ることとなった．この方法は現在の妥協点としては悪くないものだと思っている．しかし，本来なら取り込まれない，あるいは十分議論がなされるべきであるようなコードもそのまま入れられてしまうような問題もあり，今後このような問題をどのように解決していくかが課題である．

4 今後の課題

Sylpheed は電子メールクライアントとしての必須の機能は一通り実現されており，通常の使用に際してはほぼ実用的に使用できるレベルに達していると思う．しかし，Sylpheed はまだまだ発展途上の段階であり，様々な未実装の機能や解決すべき問題が残されている．具体的には以下のようなものがあげられる．

- MH 形式以外のメールボックスへの対応 (Maildir など)

  現在ネイティブに対応しているものは MH 形式のみであり， MBOX 形式はインポート・エクスポートにのみ対応している．将来は Maildir 形式にも対応する予定である．

- ネットワークのマルチスレッド化

  現在はネットワークとそれ以外の部分が単一スレッドで動作しており，ユーザインタフェースがネットワークの速度の影響をまともに受けてしまう．また，ネットワークに不具合があるとその時点でプログラム全体が固まってしまい，プログラムを強制終了させなければならない事態に陥ることもある．

- サマリ表示の高速化 (新たな GTK+ウィジェットの実装)

  現在サマリ表示には GtkCTree ベースのウィジェットを用いているが，これには要素数が増えてくると (だいたい 10000 個位) 極端に処理速度が低下するという問題を抱えている．これは GtkCTree 内部で連結リストを逐一たどっているためであり，$O(n^2)$ のオーダーで処理量が増加していることになる．これを解決するには独自のウィジェットを新たに実装する必要がある．

- 高度なフィルタリング (振り分け) 機能

  現在のフィルタリング機能は非常にシンプルなものであり，ヘッダ名とヘッダのボディにキーワードがマッチした場合に振り分け先を指定することができるだけである．将来的にはそれ以外にも，様々な動作を指定可能にしたり，正規表現に対応したりする予定である．

- キーバインドのフルカスタマイズ

  Sylpheed は Mew や Wanderlust ライクなキーバインドが使用できるが，これは今のところハードコーディングされている (メニ

ューのショートカットについては自由に設定可能)．将来はこれをユーザが自由に指定できるようにする予定である．

- プラグイン機能

  現在では Sylpheed に何か機能を拡張しようと思ったときにはソースのあちこちを変更する必要に迫られる．プラグイン機能により，Sylpheed の基本部分と拡張部分を明確に切り分けることができるようになるので，拡張性が向上し，よりシンプルな実装が可能になると期待している．

また，ユーザインタフェース全般においてもまだ十分に洗練されているとはいえず，依然として改良の余地が残されている．

5　おわりに

Sylpheed は自分にとっては単なる自作のソフトウェアというだけではなく，自己の表現手段としての作品でもある．感覚的には絵画や文学作品の創作に近いものがあるように思える．それをインターネットという手段で世界中の人々に公開することが手軽にできるようになったのは本当に素晴らしいことだと改めて思う．

最後に， Sylpheed の開発に協力してくださった大勢の方々に感謝します．

参考文献

[1] http://sylpheed.good-day.net/

[2] http://freshmeat.net/projects/sylpheed/

[3] http://sourceforge.net/projects/sylpheed-claws/