よりよいコメント記述のためのプログラミング環境

高野 保真
yasunao@ipl.cs.uec.ac.jp
電気通信大学 大学院 情報工学専攻

Yasunao Takano
Department of Computer Science, The University of Electro-Communications

概要

プログラムにおけるコメントは，プログラムの挙動を説明するために記述されるものであり，そのプログラムの理解のために大変重要なものである．しかし，これまでコメントはプログラム中にテキストで記述されていたため，その表現力，可読性の面において不十分であった．そこで，コメントの記述をプログラム本体と分離して保持し，コメントの表示・編集操作を提供するプログラミング環境を提案する．提案するプログラミング環境では，コメントとプログラム本体の分離により，図示によるコメントや複数のコメントの切り替えなどが可能である．本発表では，Emacs 上の実装について述べ，プログラム本体とコメント記述の分離による効果を示す．

1 はじめに

プログラムにおけるコメントは，そのプログラムの挙動を説明するために記述されるものである[*1]．よいコメントを記述しておけば，後でそのプログラムを読むときに大変有効であり，自分も含めそのプログラムを読む人のことを考えると，欠かせない作業である．特に，近年では，オープンソースプロジェクトの増加により，プログラムを大勢で共有・管理する機会が増え，コメント記述の重要性を再考しなくてはならない状況となっている．

ここで，現状のコメント記述について考えてみると，プログラミング言語で定義された記号を用いて，テキストでプログラム本体と入り交じって記述することとなっている．そのため，テキストで記述する代わりに図示などにより説明することはできない．また，プログラム本体とコメント記述の対応づけを明確にしようとするために，余計な改行などを挿入することが必要となってしまうことが多い．コメント記述が人間に対する記述であるということを考えると，テキストでの記述に限定されることによる表現力の乏しさは問題である．さらに，本来プログラムの可読性を高めるためのものであるコメントを挿入するために，コメントや改行などによりプログラム全体のつながりなどがみにくくなってしまうことは問題である．

この問題を解決するため， 本発表では，コメント記述をプログラム本体と分離して保持しつつ，ウィンドウシステム上で適切に対応づけを行い表示し，編集できるプログラミング環境を提案する．分離することにより，コメント記述がテキストに限定されないため，図示によるコメントが可能となる．また，あるプログラム本体に対して，複数のコメントを切り替えて表示することもできるようになる．

本発表では，提案機構をプログラミングでのコメントに適用した場合を中心に説明を行うが，実際には提案機構の適用範囲はプログラムに対するコメント記述のみでなく，任意の文書に対するコメント記述へと適用できる．

2 プログラム本体とコメント記述の分離

提案するプログラミング環境は，コメントをプログラム本体と分離して保持し，適切に表示を行い，編集を可能とする環境を提供する．分離されたコメントは，コメント記述ファイルとして，プログラム本体のどの部分へのコメントであるか保持し，プログラムの表示領域に合わせてコメントの表示を調整する．

[*1] 一時的に式・文を取り除く用途でのコメントはコメントアウトとして，本発表中では区別する

2.1 分離の利点

プログラム本体とコメント記述とを分離して保持することにより，主に以下のような利点がある．

- 図示によるコメントが可能となる．
  これまで，プログラム本体がテキストにより記述されているため，それに合わせてコメントもテキストとなっていたが，分離することによりテキストに限定されなくなる．図示によって説明することにより，たとえば，リストやスタックの操作などを明解に記述できるため，より読みやすいコメント “描く” ことができる．
- コメントの選択が可能となる．
  コメント記述を分離したことにより，あるプログラムに対して，複数のコメント記述を用意することができる．そのため，コメントの読み手に応じてコメント記述を切り替えることが可能となる．具体的な用途としては，あるプログラムに対して，日本語のコメントと英語のコメントとを別々に用意しておき適宜選択することができる．
- コメントの対応づけが明確になる．
  これまではプログラム本体のある行へのコメントといったようにコメントを記述していたが，ウィンドウシステム上で表示することにより，プログラム中の文字単位でどの領域へのコメントといった記述が可能となる．
- コメントのためにプログラム本体を変更する必要がない．
  これまで，コメントとの対応づけを明確にするためにプログラム本体中に改行などを入れることがあったが，提案機構が対応づけを行うので，プログラム本体を変更する必要がない．
- コメント記述の文字コードを考えなくて済む．
  プログラム本体の文字コードとコメント記述の文字コードを分けることができるため，日本語などのコメントを導入しやすくなる．

また，プログラミング環境としてコメント機能を提供することで，プログラミング言語ごとのコメント記述子を考慮することなくコメントを記述できるという利点もある．

2.2 提案機構の動作

プログラムを編集している際にコメントの記述が必要となった際には，コメントが対応するプログラム上の領域を指定することでコメントに対応する編集用のウィンドウが表示される．記述されたコメントは表示されているプログラムに対応して表示が更新され，コメントが張られているプログラムの領域が分かるようになっている．動作例は，図 1 のようになる．

左側がプログラム本体を編集中のウィンドウで，右側に並んでいるウィンドウが各コメントに対応している．この例では，表示中のプログラム本体中の四ヶ所にコメントが張られている．また，左側のウィンドウでは，コメントの張られた領域は背景色により，コメントが張られていることが分かる．さらに，右側に並んでいるウィンドウは対応するプログラムの横に表示されるようになっていることからもコメントの張られているプログラムとの対応づけが可能である．

3 実装

Emacs の機能拡張として，Emacs Lisp により実装し，マイナーモードとして提供する．ここで，コメントを記述するウィンドウも Emacs のフレームとして提供することで，コメントの記述も Emacs キーバインドなどを用いて統一的に行うことができる．このマイナーモードでは，プログラムを編集しているウィンドウのスクロールと編集操作をフックし，プログラム本体のうちどこの領域が現在表示されているかを取得する．取得した領域に合わせて，対応するコメントが表示されるように調整を行う．

コメント記述ファイルについては，それぞれのコメントに対応するエントリからなり，コメントエントリは張られているプログラムの開始ポイント（その文書中の何文字目であるか）により整列して保存する．現状のコメントファイルのエントリは図 2 のようになっている．ここで，一つ前のコメントからの文字数を保持することで，プログラム本体の各編集操作において，編集が起こった行の周辺のコメントエントリのみを調整すればよいようにしてある．そのため，コメントの開始ポイントが実際のものと一時的にずれることがあるが，プログラムへの編集操作ごとに編集が起こった行以下に対応するコメントエントリをすべて更新することなく実現できる．

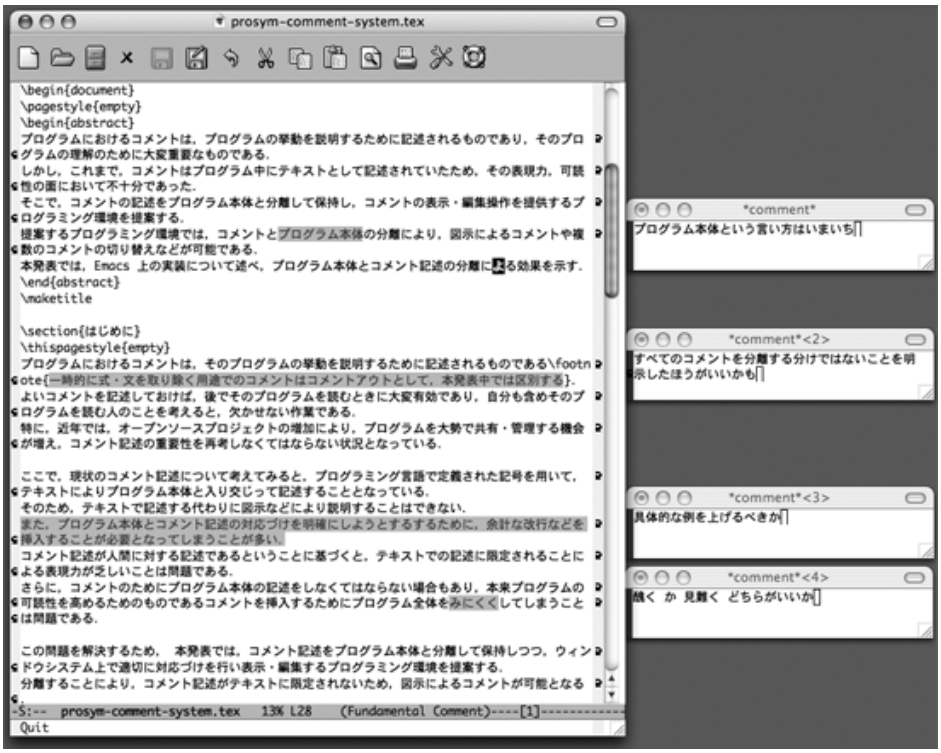

図 1　動作例

```
-------------
このエントリの末尾への行数
1 つ前のコメント (の先頭) からの文字数
コメント開始のポイント
コメント張られているプログラムの長さ
コメントの張られているプログラムの行数
コメントの行数
コメントの張られているプログラム
...
コメント
...
このエントリの先頭行への行数
-------------
...
```

図 2　コメント記述ファイルのフォーマット

内部構造としては，コメント記述ファイルを Emacs 上のバッファとして開き，そのバッファ中の現在表示されているプログラムに対応するコメントのエントリへの Emacs 上のマーカとして保持している．Emacs のマーカは編集に合わせてポイントを自動で調整するため，本機構中でのコメント挿入などによる調整は必要ない．

4　関連研究

既存の Microsoft Office Word や Apple iWorks Pages などの文書処理アプリケーションでは，文書へのコメントを付加することができる．しかし，現状ではコメントファイル自体を分離して保持していないため，複数のコメントファイルを切り替えて使うことはできないようである．

提案機構と Literate Programming [1] の WEB はプログラムにおけるコメントの記述に注目した点が同じであるが，WEB はプログラムをドキュメント中に記述するように推進しているアプローチをとっているため本発表の主張とは大きく異なる．WEB はプログラムのドキュメントの作成に主眼を置いているため，コメントとプログラムが混在することを問題にしていなく，組版されたドキュメントとして読む場合には有効な手法である．しかし，プログラム編集中の可読性を考えると，テキストとしてプログラム本体とコメントを渾然一体としてしまっているため，2.1 節で述べたようなコメントとプログラムを分

離したことによる利点を得ることができない．

5　まとめと今後の課題

本発表では，プログラム本体とコメント記述を分離することにより，コメントの表現力とプログラムの可読性を高めることができることを示した．Emacs 上に実現したプログラミング環境では，ウィンドウシステム上で分離されているコメントを適切に対応づけ，表示・編集を行うことができる．

今後の課題としては，本機構を用いないでプログラムのみが編集された場合のコメントとの対応づけ，Emacs 以外への対応があげられる．Emacs 以外の任意のアプリケーションに関してコメントを付けることができるようになれば，さらなる発展を見込むことができる．たとえば，一般の文書処理アプリケーションに対応することができれば，ある文書に対する部外秘のコメントを作成し，文書本体とは別に配布するといった用途も可能である．また，ウェブブラウザに同様の機能を追加することにより，コメントの選択を可能とし，ウェブ上の任意のコンテンツに対してコメントを記述し，配布することができるようになる．そうすれば，コメントを取捨選択できるようになり，ユーザが好きなコメントのみを読むような環境を構築することができると考えている．

謝辞

本プロジェクトは未踏ソフトウェア創造事業未踏ユースによりご支援いただいております．ご支援いただいております IPA，竹内郁雄教授，株式会社 創夢のみなさまに感謝いたします．

参考文献

[1] Donald E. Knuth: Literate Programming. Computer Journal. 27(2), 97-111 (1984)