ソフトウェア工学 71−15
（1990. 2. 9）

ソースコード検索閲覧システム

齊藤明紀　　辻野嘉宏　　都倉信樹

大阪大学基礎工学部情報工学科

既存のプログラムを理解する際には、手続きや変数や型といった対象物のの定義や参照に関する複数のファイルの渡ってのクロスリファレンス情報の検索、オンラインマニュアルの閲覧、読解結果の記録などを行いながらソースコードを読む必要がある。また複数のファイルを並行して読む必要性があることも大きな特徴である。

そこで、マルチウィンドウ環境上でソーステキストファイルやマニュアルページの提示と検索などソースプログラム読解に必要な機能を統合して提供するソースコード検索閲覧システムについて考察し、プロトタイプを試作した。

Source Program Browsing Tool

Akinori Saitoh　Yoshihiro Tsujino　Nobuki Tokura

Department of Information and Computer Sciences,
Faculty of Engineering Science,
Osaka University

1-1, machikaneyama-cho, toyonaka-shi, Osaka 560, Japan

Reading source program text is a common method to understand computer programs. On reading source code, it is needed to retrieve cross reference information seeking for definition or reference of objects such as procedures, types and variables , to invoke on-line manual and to record memorandum. It is also needed to display multiple file simultaneously.

We propose a source code browsing tool which provides an integrated environment. It display source texts and manual pages and retrieve cross reference information on multi-window system. We also make a prototype of the tool.

1. はじめに

既存ソフトウェアの改造，移植，保守に際してはそのソースプログラムを読んで内容を理解することが必要である．また，新規作成の場合でも使用するライブラリモジュールの使用法や制限の詳細を調べるために，マニュアル，ドキュメント(関数仕様など)の参照を行う．しかし，ドキュメントやマニュアルがなかったり，内容が不十分であるためにそのソースプログラムそのものを読んだり，同じライブラリを使用している他のソフトウェアのソースプログラムを読んで参考にしなければならない状況はよく起こる．

ソースプログラムは一般には複数のファイルに分割されていて相互に参照しあっている．そのためソースプログラムを読む場合には，文章を読む場合と違って先頭から順に全部を読むわけではない．不明な関数/変数が現れるたびにそれを調べに行くなどして，注目点が頻繁に移動する．また，注目点は移動したあと，しばしば戻ってきて，その続きを読む．たとえばある関数を使用している箇所とその関数そのもののように複数のファイルあるいは一つのファイルの複数の場所を並行して見る必要も起こる．プログラム全体に渡って詳細に理解するために読むことは少ないので必要な情報を得るのに最小限の部分を読むことが望ましい．ソースプログラムを読む手段は，プリントアウトしたハードコピーを読む方法と，計算機のディスプレイにソーステキストを表示させて読む方法に分類できる．それぞれの特徴を以下に述べる．

◎ハードコピー

・違う分冊ならば複数の箇所を同時に開くことができる．
・計算機ディスプレイの表示より文字品位が高く，良みやすい．
・メモを書き込むことができる．
・検索が手作業．
・あらかじめプリントアウトしておかなければならない．
・製本に手間がかかる
・ソースプログラムが変更されると無効になる．書き込んだメモの移しかえが面倒．
・保管場所が必要
・利用者間の共有が難しい

◎計算機上でエディタやテキスト表示ツールを使用する．

・文字列サーチなどが容易で高速
・クロスリファレンサなどの利用
・一度に1箇所(数十行)しか見えない
・メモが書き込めない

そのため，別のファイルや紙にメモを書き留めることになる．結果としてメモとメモの対象との対応がとりにくい，ある対象にメモが書かれているのかどうかが分からなくなるという欠点が生じる．

◎マークとしおりの比較

ハードコピーに対しては，しおりをはさんだりタグシールを張り付けることができる．計算機上でソースプログラムを読む場合，
エディタなどが持つマーク機能がこれに対応すると考えられる．しかし，マーク機能は，

・エディタ等のツールを終了すると消えてしまう
・マークがついているということを知ることができないあるいは手間がかかる
・タグシールのように，メモを書くことができない．
・一つのファイルの中だけで有効であることが多い．

というような欠点がある．

このように，ハードコピー，計算機ツール共に一長一短である．そこで，ソースプログラムの閲覧とその際必要な情報の検索を効率的に行うための読解支援ツールを作成することを考える．

以下，2節で，ソースプログラム読解作業及びその手段についての検討を行い，3節で支援に必要な機能の検討と閲覧ツールの設計について，4節でプロトタイプの作成について述べる．

2 ソースプログラム読解作業の分析

C言語の場合を例に，ソースプログラムの読解作業について考察する．ソースプログラムの読解の過程で行う行為は次の5種類が考えられる．

1）名前からそれに関する情報を検索する．
2）参照関係を調べる
3）情報を記録する
4）ソーステキストファイルの構造を調べる
5）ソースプログラムに関する文章を読む

これをさらに細かく分類すると，以下のようになる．

1）名前からそれに関する情報を検索する．
・ある名前が関数か変数かマクロかなど名前の種類を調べる．
・関数，変数，マクロの定義，宣言を調べる．
・関数，変数，マクロを定義している部分のソーステキストを呼び出す
2）参照関係を調べる
・ファイルの中で定義されている関数名を調べる
・ファイル，関数の中で定義されている変数名，マクロ名を調べる
・関数，変数，マクロを参照している関数名を調べる
・ファイル，関数の中で呼び出されている関数名を調べる
・変数，マクロを参照しているファイル，関数を調べる
・ファイルの中でインクルードされているファイル名を調べる
・ファイルをインクルードしているファイル名を調べる
3）情報を記録する
・わかったことをメモにして，記録する．
4）ソーステキストファイルの構造を調べる
・ソースプログラムの制御構造を理解する．
・データ型の構成を調べる
5）ソースプログラムに関する文章を読む
・マニュアルやドキュメントを読む．

また，ソースプログラム読解時に以下のような特徴がある．

・調べるのを中断したり，複数の場所を見るとき，あとで読みやすいように印を付ける
・複数のファイルを並行に読む
・長期間(数日～)にわたって読解を間欠的に継続する

3 ソースプログラム閲覧検索ツールの検討設計

3.1 支援機能についての検討

既存のソースプログラム閲覧方法の主な欠点は，利用者が個別のツールを駆使して自分で必要な情報を捜したり，メモを管理したりしなければならないことである．そこで，2節で述べた読解時に行っている行為を支援することについて検討する．

ソースプログラム読解を支援するため必要な機能を整理すると以下のようになる．

1）注目している名前に関する情報（種類や定義や参照関係）を利用者が必要なときに，検索し，表示する機能（情報検索機能）

2）ある対象(関数，ファイル，ソーステキストのある領域など)について利用者が残しておきたい文章(メモ)を記録し，その文章を保存する．そして対象物とそれについて書かれたメモの間で相互に参照を可能にする．（メモ管理機能）

3）注目しているソーステキストを分かりやすく表示する機能（テキスト表示機能）

4）ソースプログラム内で任意の位置に印をつけ，あとで印の付いている場所を再表示する機能．（マーク機能）

5）メモをソースプログラムの適切な位置に挿入した形でのハードコピーをとる機能

1）から4）の機能は，対象物がある．まず利用者が対象物を指定するための操作について考察し，続いて機能の呼び出しについて考察する．

3.2 機能の対象

上に述べた機能の対象となるものは，次の6つである．

・ファイル
・関数
・変数
・マクロ
・メモ
・領域

ファイル，関数，変数，マクロはテキストである．また，メモは利用者が文章にして残している記録，領域は連続した複数の文字である．

ファイル，関数，変数，マクロは，それぞれファイル名，関数名，変数名，マクロ名を指定することにより選ぶことができる．画面に表示されている名前に関しては，その名前をマウス等のポインタで指示することで対象物を指定するのが便利だと考えられる．表示されていない名前に関しては，キーボードから入力するしかない．

また，メモは，表示されているメモそのものを指定することにより選ぶことができる．

3.3 メモ管理機能

ソースプログラム読解時において，得られた情報を，ハードコピーの余白や別紙に記入しておくことはよくある．これを電子的に行うメモ機能を持たせる．記入をメモとして自動的に管理し，後で必要とする時に容易に取り出せる機能を実現する．また，読解結果の共有が可能なように，メモ情報も利用者間で共有可能であることが望ましい．

まず，その対象とする事柄によりメモは次の2種類に分類できる（図3.1）．

・名前の実体に関するメモ

プログラム中のファイル名，関数名，変数名，マクロ名の表す実体に関する情報を記録するために用いる．例えば関数に関するメモは，その関数名によって呼び出す．

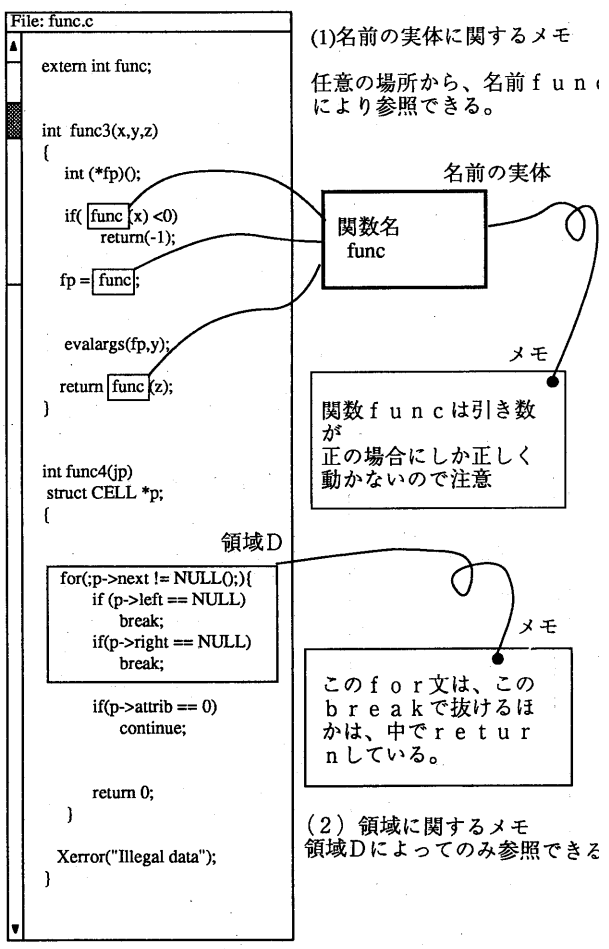


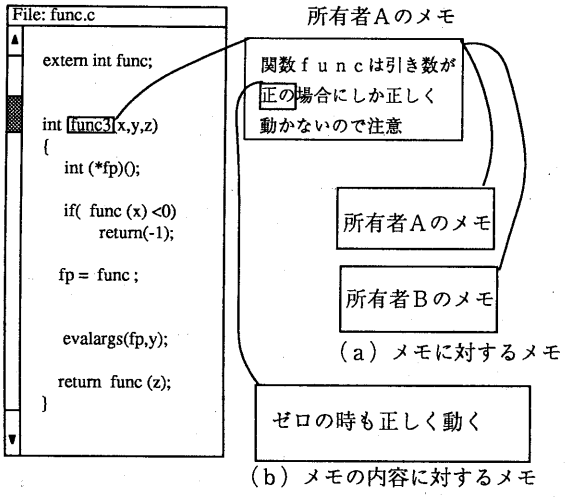
図3.1 メモの分類

・領域に関するメモ

プログラムテキスト中の各領域に関する情報を記録するのに用い，そのメモのついた領域内の任意の場所から参照できる．名前の実体に関するメモと異なり，そのメモのついた領域からでないと参照できない．

一つの対象（文字列または領域）に関して，複数のメモが存在したり，一つのメモに関して，複数のメモが付いている状況がおこりうる（図3.2）．また，利用者の要求にしたがってメモは表示するだけでなく，メモが存在することを利用者に知らせる機能が重要となる．

メモは，表示されるだけでソースプログラムは書き換えないことにする．これは，既存のプログラミング環境との親和性を増すためである．



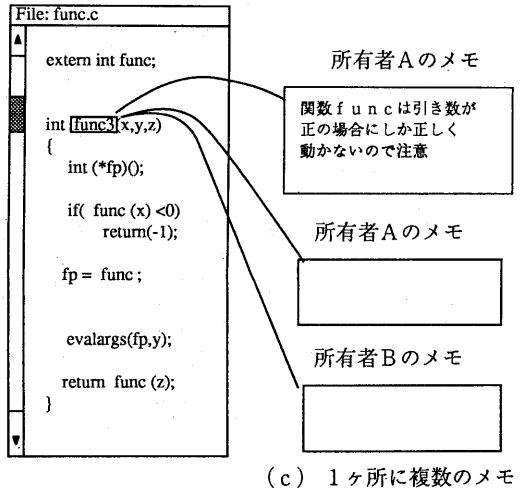


図３.２ メモの形態と所有者

メモの表示方法，メモの存在を知らせる方法について検討する．

１）メモの表示方法

2種類が考えられる．

・ソースを表示している画面以外にメモ用の画面をつくり，そこにメモを表示する

・あたかもハードコピーの余白に書きこむように，ソーステキスト中に挿入する

前者では，一つの対象に対して複数のメモを付けることができ，他人のメモに対して自分のメモを書くのにも適していると思われる．

後者は余分な領域を必要としないので，画面の利用効率が良いが，逆にメモの分量が多いとソーステキストの表示量が減ってしまう．

２）メモの存在の告知方法

他人が書いたメモも読むことを可能にすると，知識の蓄積ができるという点でソースプログラム読解にとって非常に強力な機能になる．メモのついている対象物をうまく利用者に知らせる方法を考える必要がある．ここでは，性質の違いのため，名前の実体に付くメモと領域に付くメモを分けて検討する．

名前の実体に付くメモの場合，メモのついている実体を表す名前を点滅させたり，反転させたりして，利用者にメモがあることを視覚的に知らせればよい（図３．３）．これは，メモの所有者により，表示方法を変えるのも有効だと思われる．

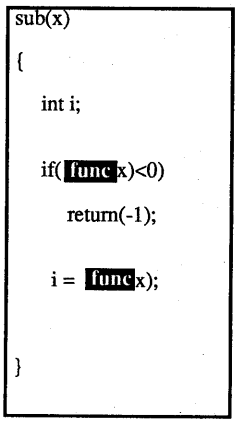


図３.３ 名前の実体につくメモの存在場所の表示方法

領域に関するメモの場合，いくつかの方法が考えられる．

１）メモのついている領域のすべての行の第一カラムに印を付ける．

２）名前の実体に関するメモと同様に，メモの付いている領域を点滅または，反転させたりして，視覚的にメモのあることを利用者に知らせる．

３）メモの付いているそれぞれの領域の右端にメモの付いていることを表す線を表示する．（図3.４）．

３．４ テキスト表示機能

前後のスクロールや文字列検索などの機能はまず必要である．そのほかに，ソースプログラムの制御構造など，概略を知りたいことはよくある．そこで，指定された領域内の不必要な部分を省略することにより，その領域の〝骨組み〟を表示する機能があれば便利である．骨組み表示の手法について考察する．

・指定された領域内のトップレベルの文だけを表示する．例えば，ファイル全体を指定した場合，指定されたファイル内で定義されている関数の定義部分の第一行だけを表示する（図3.５ａ参照）．

・指定された領域内で，トップレベルの文と共に，制御の流れを表す予約語（if,while,for,など）の存在する行を

表示する（図3.5b参照）．

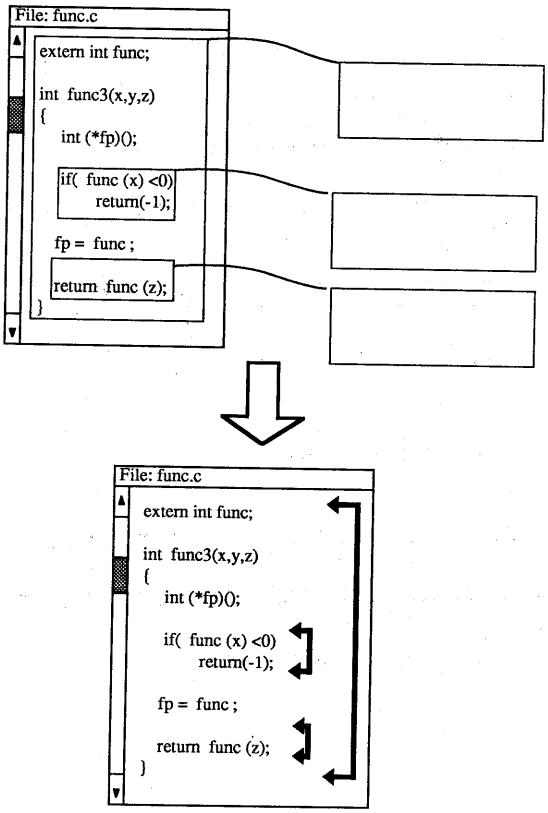


図3.4 領域に対するメモの存在場所の表示方法

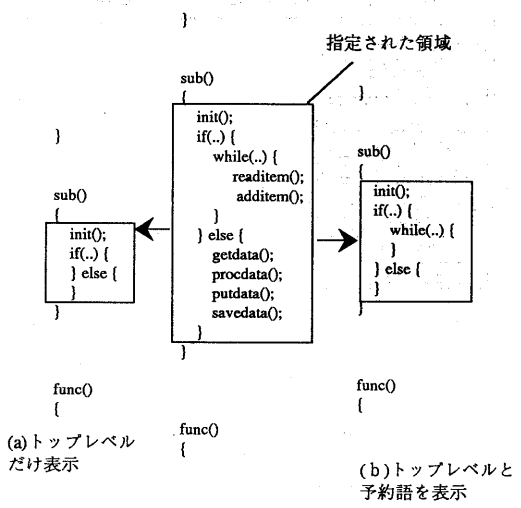


図3.5 骨組み表示

これらの方法は，状況により使い分けるべきである．明らかに，後者の方法の方が前者よりも表示領域が多いので，指定したブロックが非常に大きなものの場合は，前者の方法，比較的小さな場合は，後者の方法を用いるといったように使い分けるのが適当である
と思われる．

また，利用者は，あるブロックの骨組みを見たあとで，その骨組み表示の結果省略されている領域の骨組みを知りたい場合がある．
また，表示されている骨組みを含む大きな領域の骨組みを知りたい場合もある．そこで，表示されている骨組みの中の任意のブロックの骨組みを会話的に表示できることが望ましい．

3．5 ハードコピー

ソースプログラム読解のため，そのプログラムのハードコピーを読むことは，一般的によく行われている．これは，主にハードコピーを持ち運ぶことができるためと思われる．ソースプログラムを持ち運ぶことは計算機では支援できないので，ここでは，ハード
コピーを読むことを支援する機能について検討する．

ソーステキストの他にクロスリファレンスリストをプリントアウトすることは普通に行われているが，記録されたメモをソースプログラムの中に挿入した形でハードコピーをとる機能があれば便利だと思われる．

メモをハードコピー上に挿入してプリントアウトする．このとき，メモはソースプログラムと簡単に区別できなければならない．
そのためにはメモのフォントを変えるなどの方法が考えられるが，利用者が見なれているコメント文として挿入するという方法も考えられる．

先に述べたように，メモには名前の実体に付くメモと領域に付くメモがある．名前の実体に付くメモの場合，メモの付いた実体を表す名前が現れるたびにそのメモを挿入するとソースプログラムが読みにくくなる．そこで，名前の実体に付くメモはその実体が定義されている場所だけに挿入することにする．

3．6 全体設計

本閲覧ツールの構成は，図3．7 のようになる．全体制御部がデータ収集部 （指定されたファイルから読解の際必要なデータを抽出する部分）を用いて必要なデータを抽出する．

本ツールで表示のために使うウィンドウは次の3種類にする．

1）閲覧ツール全体に関する制御や状態の変更を行うためのウィンドウ（制御ウィンドウ）

2）利用者から要求があった場合にソースプログラムまたは必要な情報を表示するためのウィンドウ（情報表示ウィンドウ）

3）利用者が自由に書き込むことのできるウィンドウ（スクラップブックウィンドウと呼ぶ）制御ウィンドウはボタンだけのウィンドウである（図3．7）．

制御ウィンドウは一つだけ，スクラップブックおよび情報表示ウィンドウは必要に応じて複数個開く．

操作はメニューとボタンで行うファイルの呼び出しや，新規ファイルの登録は制御ウィンドウで行う．ボタンをクリックするとファイル一覧のメニューが現れる．

画面に表示されている対象(名前あるいは領域)に関する情報を検索するには，マウスでその対象物を選択する．このとき，全体制御部は選んだものの種類とそれに関して利用者が要求可能な情報を知らせる．この情報はメニューとして，表示する．利用者がこのメニューの中からひとつの項目を選ぶことにより必要な機能を選択することにする．

全体制御部は利用者が選択した要求に従い，必要な情報を調べ，調べたデータを表示する．表示されていない対象物に対する情報を検索するには，まずスクラップブックウィンドウにその対象物の名前を入力し，それを選択する．

データ収集部は，読解の際必要となるデータを調べ，ファイルに格納する．

すべての閲覧の対象のファイル及びその中にインクルードされているファイルからデータを抽出する．調べる項目はファイル，ユーザ定義関数（閲覧の対象としているソースプログラム内で定義されている関数），大域変数，マクロ，型について定義の位置やそれぞれ
の相互の参照関係等についてのデータである．

テキストのスクロールや文字列検索などは表示ウィンドウだけの力で独自に行われる．対象物の選択とそれに対して適用する機能の選択(メニューによる）が行われると，その<対象物，機能>の組が全体制御部に渡される．全体制御部はデータ収集部が収集した
データを検索する．そして新たに表示ウィンドウを作り，そこに検索結果(テキスト，マニュアルなど)を表示する．

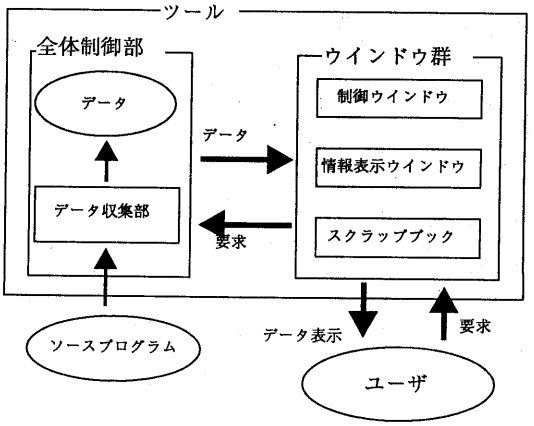


図３．６ツールの構成

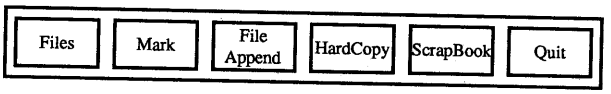


図３.７ 制御ウインドウ

４．プロトタイプの作成

プロトタイプを以下の環境で作成した．

・使用計算機 ＳＵＮ３
・プログラミング言語 Ｃ
・ウィンドウシステム Ｘウィンドウ
　　バージョン１１リリース３

閲覧ツールは，2節で述べた全体制御部とデータ収集部を担当するひとつのプロセス（以後全体制御プロセスと呼ぶ）が，担当のウィンドウを管理する複数のプロセス（以後ウィンドウ管理プロセスと呼ぶ）を制御する形にした．ウィンドウ管理プロセスは，３種類あり，それぞれが４節で述べた３種類のウィンドウと対応している．また，ウィンドウ管理プロセスどうしはそれぞれ並列に独立して働く．全体制御プロセスは，ウィンドウ管理プロセスと連携して働く．

プロトタイプの動作の概略は以下の通りである．

起動時に収集済みの検索情報をロードする．検索対象ファイルの追加を指示された場合には，そのファイルに関するデータも収集し，ファイルにセーブする．そして制御ウィンドウをディスプレイの右上に表示する（図4.1）．その後，制御ウィンドウのボタンが押されるか，表示ウィンドウからの検索要求が来るのを待つ．

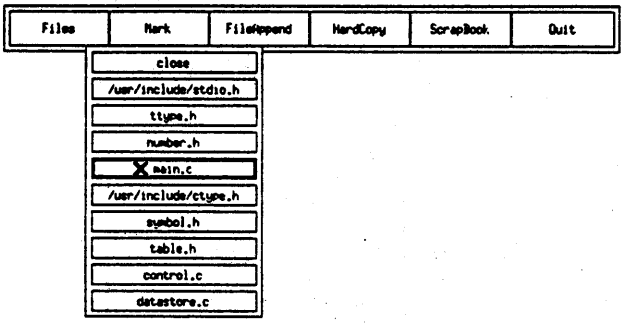


図４.１

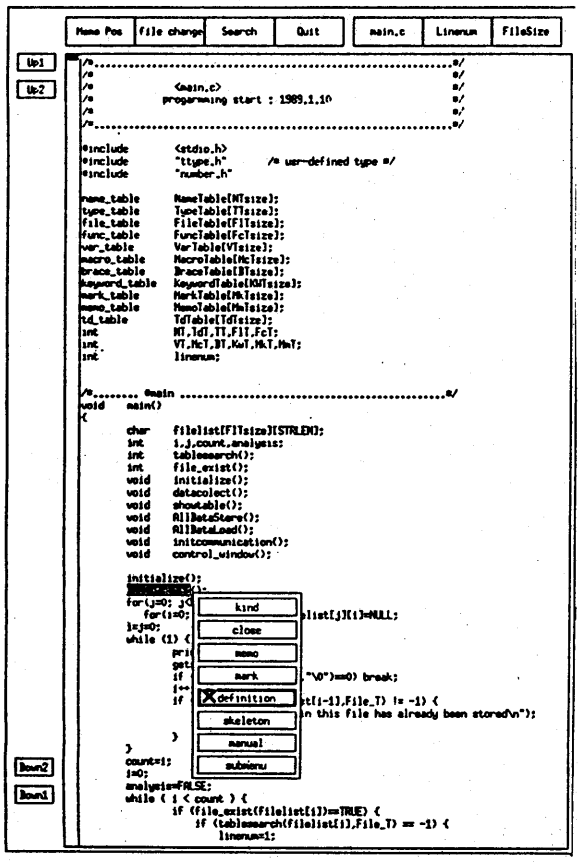


図4.2

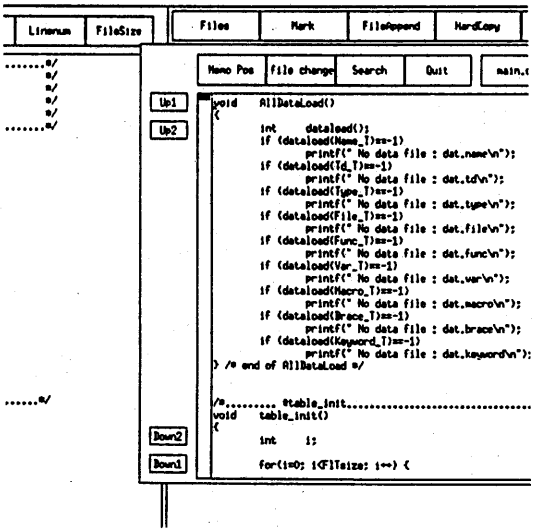


図4.3

Ｆｉｌｅｓというボタンを選べば，メニューが表示される．このメニューの各項目はデータを抽出した読解対象のファイルを表す．
この中から表示したいファイルを選択できる．選択されたファイルを情報表示ウィンドウに表示する．このウィンドウはディスプレイの左半分に配置する （図 4.2）．

ソースプログラムから情報を知りたいときは，知りたいものの上にカーソルをうごかし，マウスボタンをクリックすればよい．クリックすると知ることのできる情報を表すメニューが現れる．この中から必要な情報を表すラベルを選べば，指定した情報が情報表示ウィンドウに表示される（図4.3）．さらにこのウィンドウからも必要な情報は引き出せる（図4.4，4.5）．閲覧中にこのように情報を表示する場合は多いので，情報表示のためのウィンドウはソースプログラム表示ウィンドウに比べて約半分の大きさにした．また，ウィンドウどうしが重ならないように工夫した．

4．2 評価

試作したプロトタイプでは，次の機能を実現した．

1)指定されたファイルから必要なデータを抽出する．

2)あるウィンドウに表示されているソースプログラムの中から注目する文字列の定義部分を別のウィンドウに表示する．

3)あるウィンドウに表示されているソースプログラムの中から注目する文字列に関するマニュアルがあれば，別のウィンドウにそのマニュアルを表示する．

表示ウィンドウは任意のテキストデータを表示できる．よって関数一覧やメモもソースやマニュアルページと同じ手法で表示可能なので，プロトタイプでは実現しなかった．

プロトタイプを作成し実際に使用した結果，次の長短所が明らかになった．

［長所］

・マウスだけを用いてほとんどの操作ができるのでキーボード入力に比べて操作が簡単である．

・ディスプレイに表示されている全ての文字列に対して必要な情報を取り出すことができるので，キーボードからの入力を減らすことができた．また，表示データを入力ウィンドウにコピーする場合よりも手間が少ない．

［短所］

・利用者が操作を行ってから必要な情報が得られるまで時間がかかる．これはプロトタイプの実装の問題である．

・データを抽出していないファイルに関する情報は得られない．

既存ツールを使った場合と比較すると，

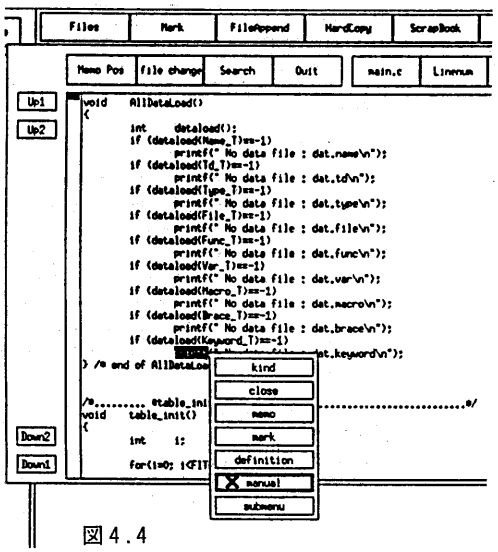


図4.4

・コマンドインタプリタの動作しているウィンドウへマウスを動かし
・情報を検索するコマンド列をキーボードから入力し，
・テキスト表示ツールを起動するコマンド列を入力する
・先ほどの検索の結果得られた位置まで注目点を移動させる
という操作が，対象の選択とその場でのポップアップメニューによる機能選択だけで実現できている．

5．むすび

本報告では，ソースプログラム読解について分析し，読解を支援するための機能について考察した．

支援すべき点は主に以下の３つであることを示した．
・必要な情報を検索する
・メモを管理する
・ブロックの大まかな構造を教える

さらに，支援するツールのプロトタイプを作成した．
今後の課題として以下の問題がある．
1）本ツールでは，ソースプログラムは変更しないことを前提としてきたが，作成/修正途上のプログラムも扱えることが望ましい．
その場合，ソースプログラムの修正前後の関数，変数などのデータおよび，メモの情報の一貫性を維持する手法について研究する必要がある．
2） 検索に必要なデータは利用者間で共有できることが望ましいことは一節で述べたが，データの共有性は今回作成したプロトタイプでは実現していない．このことはさらに検討する必要がある．

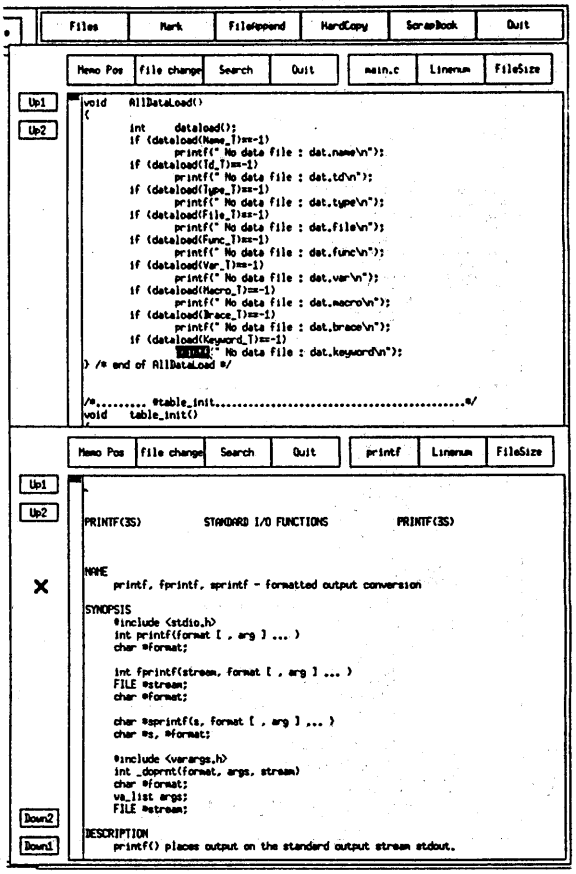


図4.5

参考文献

[1]Ralph R. Swick and Terry Weisman : "X Toolkit Athena Widgets - C Language Interface", DEC
[2]Joel McCormack,Paul Asente,and Ralph R. Swick : "X Toolkit Intrinsics -C Language Interface ", DEC
[3]David S.H.Rosenthal : "X Window System, Version 11 Inter-Client Communication Convertions Manual Public Review Draft", Sun Microsystems
[4] : "ctags", UNIX Programmers manual, BSD 4.3 virtual VAX 11 version