4Y-7　同時コンパイルを可能にするＡｄａプログラムライブラリ管理方式

梶原　清彦　　伊藤　光恭　　伊集院　正
ＮＴＴ　ソフトウェア研究所

１．はじめに

Ａｄａコンパイラはコンパイル単位間のインタフェースチェックをコンパイル時に行う。プログラムの誤りチェックが開発工程の初期で可能となり、生産性、品質の向上が図られている。本機能の実現のため、Ａｄａコンパイラは、プログラムライブラリ（Program Library，以後ＰＬと記す）に各コンパイル単位の構文・意味情報を格納し、参照する。＊

Ａｄａによりプログラム開発を行う際、ＰＬはプロジェクトに一個持ち、プログラマ間で共用するつことが有効である。共用することにより、プログラムの版管理の容易化、ディスクスペースの節約等が可能となるからである。一方、ＰＬの共用を可能とするためには、同時コンパイル時に行われるＰＬへのリード、ライトを管理する必要がある。

本研究では、ＰＬへのリード・ライトの排他制御をコンパイル単位ごと行い、同時コンパイルを実現するＡｄａプログラムライブラリ管理方式を提案する。

＊：通常はＤＩＡＮＡと呼ばれる中間言語形式となっている。本研究でもＤＩＡＮＡ形式を採用する。

２．ＰＬへの同時アクセスにおける問題点

２．１　ＰＬとコンパイル順序

Ａｄａでは、各コンパイル単位間のコンパイル順は以下の規則で決定される（図１参照）。

①本体は仕様より後にコンパイルされなければならない。
②サブ単位は親の単位より後にコンパイルされなければならない。
③ＷＩＴＨで参照する側は、ＷＩＴＨ先のコンパイル単位より後にコンパイルされなければならない。

ＰＬ内には、上述のコンパイル順序規則をチェックするための、各コンパイル単位間の参照関係を示す情報を持っている。コンパイル順序の規則①、②、③により、Ａｄａでは、コンパイル単位Ａを再コンパイルした場合、Ａを直接・間接に参照しているコンパイル単位も再コンパイルする必要がある。

２．２　問題点

ＰＬへの同時アクセスを可能とするためには、以下の３点が問題となる。

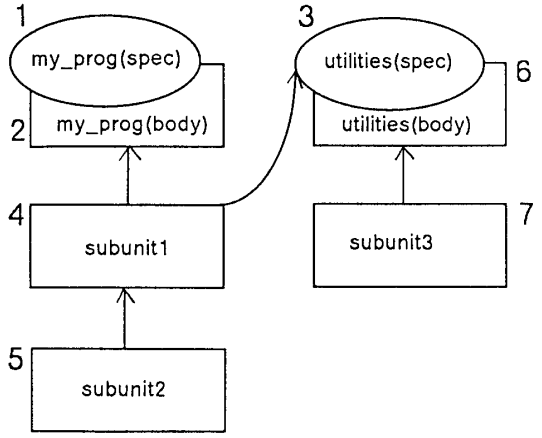

図1.Adaにおけるコンパイル順序例

①　ＰＬの整合性の維持

［ＰＬの整合性の定義］
ＰＬの整合性が保たれているとは，ＰＬ内の全てのコンパイル単位のコンパイル順序が正しく管理されている状態をいう。

コンパイル単位Ｘ，Ｙ，Ｚが次の様な関係にあった時を考える。
Ｘ－＞Ｚ：直接または間接に参照
Ｙ－＞Ｚ：直接または間接に参照
Ｘはコンパイル時にＺを参照するから、Ｘをコンパイル中に、Ｚを書き換えることはできない。一方、Ｙはコンパイル時にＺを読み出すのみであるから、ＸとＹは同時にコンパイルできる。従って、ＰＬへの同時アクセスを実現する際に、コンパイル単位個々のリード、ライトアクセスを独立して管理するのではなく、直接・間接に参照関係のあるコンパイル単位を意識した排他制御を行わなければ、ＰＬ内のコンパイルの整合性が破壊される（図２）。

②　デッドロックの回避
排他制御方式に伴ったデッドロックに加え、コンパイル時に、不可避に発生するシステムダウン等により、ＰＬアクセス不能となることを避けなければならない。

③　排他制御のオーバヘッド回避
通常のＰＬに格納されているコンパイル単位の数は、数１０個から数１００個になっている。このため、排他制御をコンパイル単位ごとに行う場合、コンパイル時間が増大することが予想され、このオーバヘッドをできるだけ小さくする必要がある。

On an Ada program library management method which enable concurent compilations
Kiyohiko Kajihara、Mitsutaka Itoh、Sei Ijuin
NTT Software Laboratories

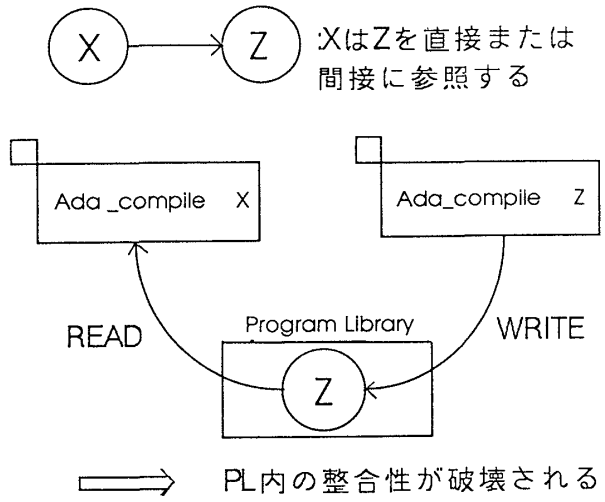

図2.PL内の整合性

3．問題点の解決策

3．1　ＰＬの整合性の保証

ＰＬの整合性を保つためには，以下の２点を解決すればよい。

①コンパイル時に直接・間接参照されているコンパイル単位の再コンパイルを禁止する。
②更新中のコンパイル単位については、参照、更新禁止にする。

本管理方式では参照禁止フラグ・更新禁止カウンタを用いて以上の２点を解決した。各コンパイル単位は、付属情報としてＰＬ内に参照禁止フラグ、更新禁止カウンタを持ち、参照禁止フラグがonの時には，そのコンパイル単位を参照できないことを表わし，更新禁止カウンタの値が0より大きいときには，そのコンパイル単位を更新しては行けないことを表わす。図３にコンパイル単位を参照する場合のアルゴリズムをアルゴリズムを示す。

参照禁止フラグ・更新禁止カウンタの自身の排他制御については、排他制御用のファイルを用意し，参照禁止フラグや新禁止カウンタを更新する際には，そのファイルを他プロセスからアクセス不能なモードでオープンし，ＰＬ内の参照禁止フラグ・更新禁止カウンタを更新して，その後ファイルをクローズすることにより、参照禁止フラグ、更新禁止カウンタの排他制御を実現した。

3．2　デッドロックの回避

４．１の排他制御方式では，参照しているコンパイル単位の解放待にならないので，コンパイラが正常に終了する限りにおいては，デッドロックに陥ることはない。しかし、システムダウン、コンパイラのバグにより参照禁止フラグや更新禁止カウンタが正しく設定されなかった場合には，ＰＬがロックされてしまい、デッドロックに陥る可能性がある。そのため，参照・更新フラグの整合性を図るためのツールを提供することとした。

```
if 参照したいコンパイル単位の
                参照禁止フラグ = on then
   return;
end if;
更新禁止フラグ := 更新禁止フラグ + 1;
参照したいコンパイル単位を参照する
更新禁止フラグ := 更新禁止フラグ - 1;
```

図３　コンパイル単位参照アルゴリズム

3．3　排他制御のオーバヘッド回避

実際のＰＬ内のコンパイル単位の中には、参照しか行わないものが存在する（例えばシステム既定義のＴＥＸＴ＿ＩＯ、ＤＩＲＥＣＴ＿ＩＯ、ＳＥＱＵＥＮＴＩＡＬ＿ＩＯ等）。そこで、参照しか行わないことがコンパイル以前にわかっているコンパイル単位については、あらかじめリードオンリとして扱い、その他のものについてのみ、リード、ライトアクセス制御を行い、排他制御のオーバヘッドを軽減した。（図４参照）。

4．おわりに

今後、本方式の効果を評価して行くと共に、さらに高性能なＡｄａコンパイラの実現法を検討していく。

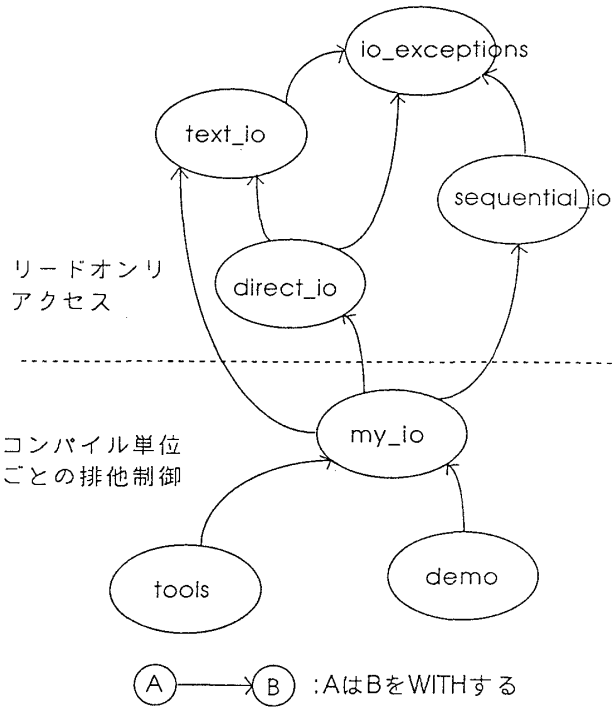

図4.排他制御の効率化

［参考文献］
[1] U.S.Department of Defense : ANSI/MIL_STD_1815A_1983, Ada Programming Language, American National Standards Institute Inc. (1983)
[2] DIANA REFERENCE MANUAL Revision 3, Tartan Laboratories Incorporated (1983)