第4章 ワンランク上の活用をしたい人向け

最適化/解析ソフト・モジュール「Pass」のメカニズム

東 工太郎

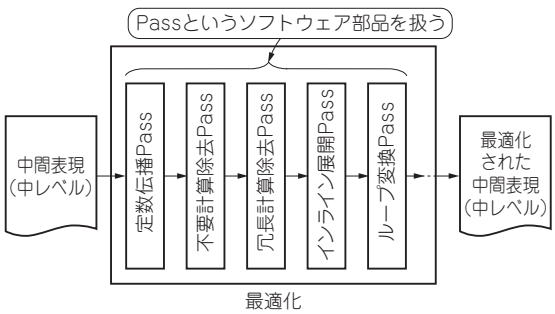

図1 最新コンパイラのGCCやLLVMは中身を好きに入れ替えられるようにソフトウェア・モジュールPass単位で最適化/解析処理を行うようになっている

● 用途に応じたコード最適化/解析を手軽に実現できる構造になっている

GCCやLLVMを使う理由として，ユーザが用途に応じて独自のコード最適化/解析を少ない労力で実現できる点が挙げられます．

独自の最適化/解析を実現する労力を減らすために，GCCやLLVMはコンパイラの内部構造の設計を工夫しています．具体的には，多くの最適化/解析の実現で共通に使用できるデータ構造やアルゴリズムをライブラリ化(部品化)して用意しています．

また，既存のコード最適化/解析とともに，独自の最適化/解析を柔軟に組み合わせられるよう，個々の最適化/解析をPassという単位で管理し，用途に応じて最適化フェーズにPassを着脱できる図1のようなしくみを備えています．これらのしくみにより，GCCやLLVMでは，用途に応じて独自の最適化/解析を手軽に実現できます．

本稿では，用途に応じた最適化/解析を少ない労力で実現可能にするしくみを解説します．

いろいろできると便利！ コード最適化/解析あれこれ

例えば，メモリ容量が大きいサーバ機上のアプリケーション・コードと，メモリ容量が小さい組み込みシステム上のコードとでは，必要とされる最適化が異なる場合があります．

現状のコンパイル技術では，全ての応用領域に対してコンパイラが全自動で最適なコードを生成することはできません．各応用領域に応じてユーザが適切な最適化/解析を選択したり実現したりする必要があります．

● メモリがたくさん使える場合…サイズと引き換えにガンガン高速化

パソコンやサーバ機の場合，使用できるメモリ容量が大きいため，ループ展開やインライン展開などの最適化によって，コード実行時間の短縮が見込めます．

ループ展開はループ本体中の文を複製することで，ループの条件判定にかかるオーバヘッドを削減する最適化です．

インライン展開は関数呼び出しを関数本体に置き換えることで関数呼び出しにかかるオーバヘッドを削減する最適化です．

いずれもコード・サイズの増加と引き換えにオーバヘッド(実行命令数)の削減を図る手法です．メモリ容量が大きな環境では，ループ展開やインライン展開の後もコード実行のためのメモリ容量が十分に残されるため，これらの最適化によってコード全体の実行効率の向上が見込めます．

● 使えるメモリがプアな場合…高速化したつもりなのにかえって遅くなる場合も！

メモリ容量の小さい組み込みシステムなどの環境では，ループ展開やインライン展開により全体の実行効率が悪化する危険性があります．ループ展開やインライン展開によるコード・サイズの増大がメモリ領域を圧迫し，効率的コード実行に必要なメモリ領域を十分に確保できない場合があるからです．このように，メモリ容量の小さな計算環境では，コード・サイズが増大する最適化を抑制するとともに，積極的にコード・サイズを削減する最適化が求められます．