トップエスイー修了制作

静的解析ツールによるオープンソースの問題検出

（株）富士通コンピュータテクノロジーズ　松本 博郎　matsumoto.hiroo@jp.fujitsu.com

開発における問題点

オープンソースを用いた流用開発では，ソースコードの品質を確認する為に，人手によるコードインスペクションが必要となる．確認すべき品質として，ソースコードの構造に起因する既知問題の有無等がある．しかし、コードインスペクションに工数を割きすぎるとオープンソースの利点から離れてしまう．

手法・ツールの適用による解決

ソースコードの構造に起因した既知問題のひとつであるLinuxカーネルの497日問題について，静的解析による手法を用いることで問題個所を自動的に検出し，調査工数の削減を目指す．検出結果は人手によるコードインスペクションの場合と同等のものを目指す．

497日問題

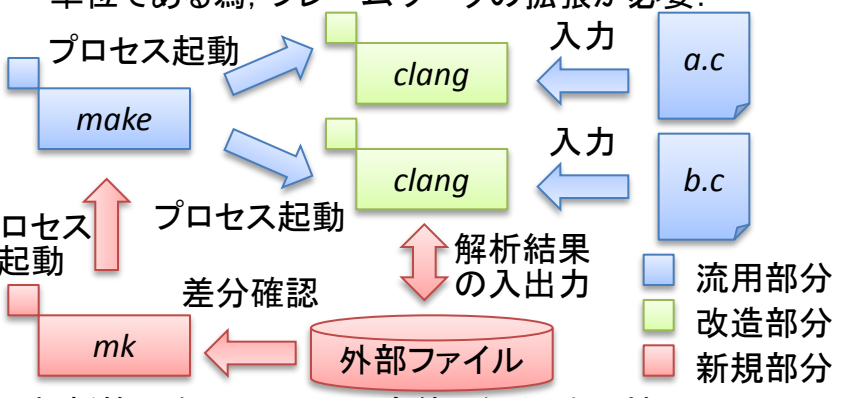
- 時間管理の為にOSが内部で保持する32bitのカウンタが497日周期でオーバフローした際に、カウンタを用いた判定文が予期せぬ結果となる問題．
- Linuxカーネルでは外部変数jiffiesを用いて時間管理を実現している．
- 497日問題になる場合とならない場合がある（差分を大小比較している場合は問題とならない）．

	なる場合 j1 > j2 + 1	ならない場合 j1 - j2 > 1
j1 = 3, j2 = 0	TRUE	TRUE
j1 = 1, j2 = 0xfffffffe	**FALSE** **予期せぬ結果**	TRUE

予めj1, j2がjiffies由来の値かどうかを知っている必要がある
（jiffiesは代入や関数経由でソースツリー全体に伝播する）

clangの拡張

- 任意の静的解析コードを追加可能なコンパイラ
- clangのデフォルトでは，最大の解析範囲は関数単位である為，フレームワークの拡張が必要．



解析範囲をソースツリー全体となるように拡張し，
ソースツリー全体を跨いだ変数伝播の追跡を実現
（外部ファイルには外部変数と外部関数の変数伝播を記録）

解析の流れ

1. jiffiesの伝播を追跡するフェーズ

<mk> 外部ファイルが収束するまでmakeを実行する

<make> ソースツリー全体を解析する

<clang> 外部ファイルから変数伝播を読み込む

<clang> 収束するまでファイル内の変数伝播を追跡する

<clang> 外部ファイルへ変数伝播を書き込む

2. 問題となる大小比較箇所を検出するフェーズ

<mk> 外部ファイルが収束したのでmakeを1回実行する

<make> ソースツリー全体を解析する

<clang> 外部ファイルから変数伝播を読み込む

<clang> 収束するまでファイル内の変数伝播を追跡する

<clang> 問題となる大小比較箇所を検出する

ソースツリー全体で変数伝播を収束させ，大小比較箇所を検出

評価と課題

評価

- 問題となる大小比較個所を自動で検出でき，調査工数を大幅に削減できた．
- 人手で検出していた問題個所は全て検出でき，人手の場合と同等の検出精度を得た．
- ただし、差分を大小比較しているにも関わらず，問題ありと判定されてしまう誤検出が発生している．これは差分算出に用いる片方の変数について，変数伝播の追跡が不十分であることが原因である．

課題

- 関数ポインタ経由等の変数伝播追跡の実装
- 変数伝播の履歴機能の実装
- クロスコンパイルの適用
- 変数伝播の追跡が必要となる他問題への適用