CbC による xv6 の FileSystem の書き換え

学籍番号：165723C 氏名：坂本昂弘 指導教員：河野真治

2020 年 2 月 17 日

1 section1

2 xv6

xv6[1] とは MIT のオペレーティングコースの教育目的で 2006 年に開発されたオペレーティングシステムである. xv6 はオリジナルである v6 が非常に古い C 言語で書かれている為, ANSI-C に書き換えられ x86 に再実装された. xv6 は read や write などの systemcall, プロセス, 仮想メモリ, カーネルとユーザーの分離, 割り込み, ファイルシステムなど Unix の基本的な構造を持っている.

sectionContinuation based C xv6 kernel 上で interface を実装する際, 当研究室で開発されたプログラミング言語 Continuation based C (CbC) を用いる. CbC は基本的な処理単位を CodeGear として定義し，CodeGea 間で遷移するようにプログラムを記述する C 言語と互換性のあるプログラミング言語である. CodeGear は返り値を持たない為, 関数内で処理が終了すると呼び出し元の関数に戻ることがなく別の CodeGear へ遷移する. 以下の Code1 に CodeGear 遷移時のコード例を示す.

```
__code cg0(Integer a, Integer b){
 int a_v = a->value;
 int b_v = b->value;
 Integer c = {a_v + b_v};
  goto cg1(c);
}
__code cg1(Integer c){
  goto cg2(c);
}
```

Code 1: CodeGear の継続の例

また CbC における CodeGear 間の継続にはスタックが使用できず, 呼び出し元の環境などを持たない為軽量継続と呼ぶ. 現在 CbC は C コンパイラである GCC 及び LLVM をバックエンドとした clang 上で実装されている.

3 context

context とは一連の実行が行われる際に使用される CodeGear と DataGear の集合である. 従来のスレッドやプロセスに対応する.Context は接続可能な CodeGear, DataGear のリスト. Data Gear を確保するメモリ空間, 実行される Task への Code Gear 等を持っている. CodeGear が別の CodeGear に遷移する際, 必ず context を参照し enum で定義された CodeGear の番号を指定し遷移する. ノーマルレベルで見た際の CodeGar,DataGer および context の関係を以下の図 1 に簡潔に示す.

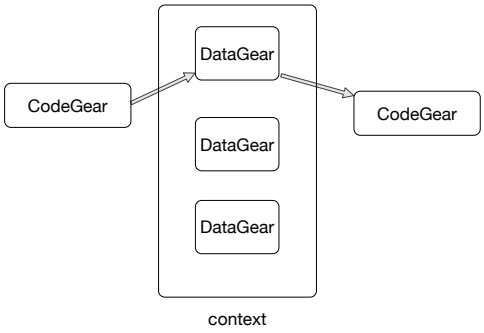

図 1: CodeGear,DataGear,contxt の関係図

4 CbC の Interface

先述した通り, CbC の Interface は Gears OS のモジュール化の仕組みである. Interface は呼び出しの引数になる Data Gear の集合であり，そこで呼び出される Code Gear のエントリである. 呼び出される Code Gear の引数となる Data Gear はここで 全て定義される. Interface を定義することで複数の実装を持つことができる. この Interface は,Java の Interface や Haskell の型クラスに対応し, 導入することで仕様と実装に分けて記述することが出来る.

5 xv6 の FileSystem

6 CbC による xv6 FileSystem の書き換え

7 まとめと今後の課題

参考文献

[1] Xv6, a simple Unix-like teaching operating system. https://pdos.csail.mit.edu/6.828/2019/xv6.html. (2019 年 10 月 19 日閲覧).

[2] Kaito TOKKMORI and Shinji KONO. Implementing continuation based language in llvm and clang. *LOLA 2015*, July 2015.

[3] Chris Lattner and Vikram Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

[4] GNU Compiler Collection (GCC) Internals. *http://gcc.gnu.org/onlinedocs/gccint/*.

[5] 伊波立樹, 河野真治. Gears os の並列処理. 琉球大学工学部情報工学科平成 30 年度学位論文 (修士), 2018.

[6] 宮城光希, 河野真治. 継続を基本とした言語による os のモジュール化. 琉球大学工学部情報工学科平成 31 年度学位論文 (修士), 2019.