GearsOSにおけるinodeを用いたFileSystemの設計

又吉　雄斗 河野研

GearsOSにおけるFileSystemの設計

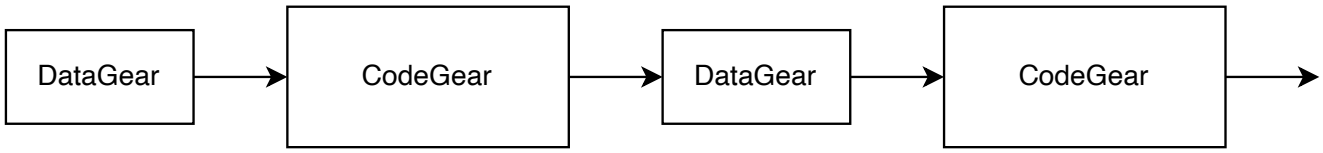
- アプリケーションの信頼性を保証するために，アプリケーションが動作するOSの信頼性を高める必要がある
- 当研究室では，信頼性の保証を目的としたGearsOSを開発している
- GearsOSで未実装の機能であるファイルシステムの実装を行う
- Unix likeな実装

inodeを用いたgearsDirectoryの実装

- 今回はディレクトリシステムを実装した
- GearsOSへUnixのFile systemの仕組みを取り入れるアプローチをとる
- Unixのinodeの仕組みを取り入れる
- GearsOSのディレクトリシステムであるgearsDirectoryについて説明する

Continuation based C

- Cの下位言語である
- function callの代わりにgotoによる継続を用いる
- プログラムはCodeGearと呼ばれる処理の単位で記述
- ノーマルレベルとメタレベルの処理を切り分けることが可能である

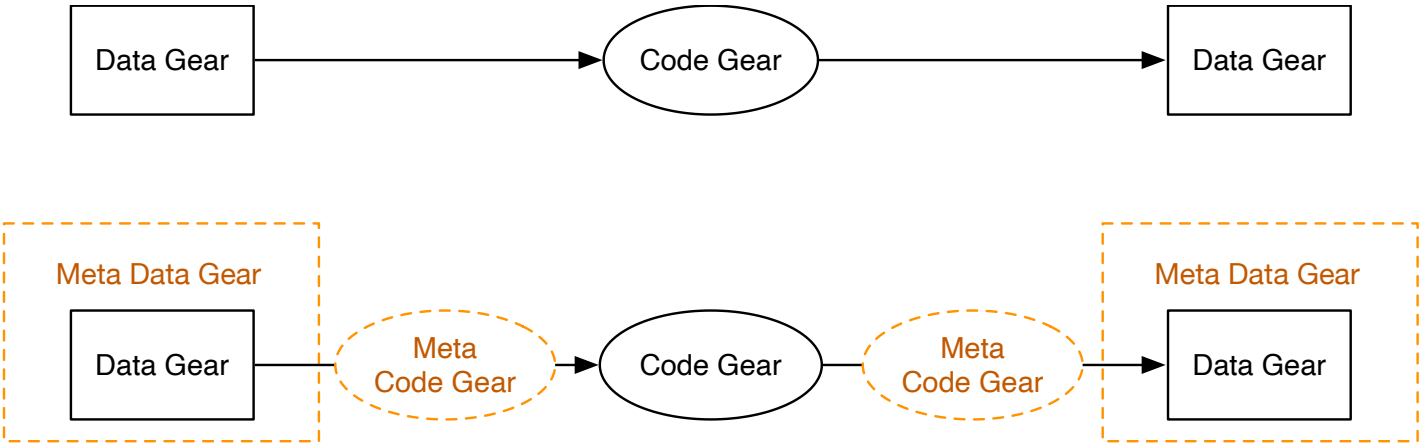


GearsOS

- 信頼性と拡張性の両立を目的として開発されている
- Gearという概念があり，実行の単位をCodeGear，データの単位をDataGearと呼ぶ
- 軽量継続を基本とし，stackを持たない代わりに全てをContext経由で実行する
- ノーマルレベルとメタレベルの処理を切り分けることができる
- 同様にGearの概念を持つCbC(Continuation based C)で記述されている．
- OSとして動作するために今後実装しなければならない機能がいくつか残っている．

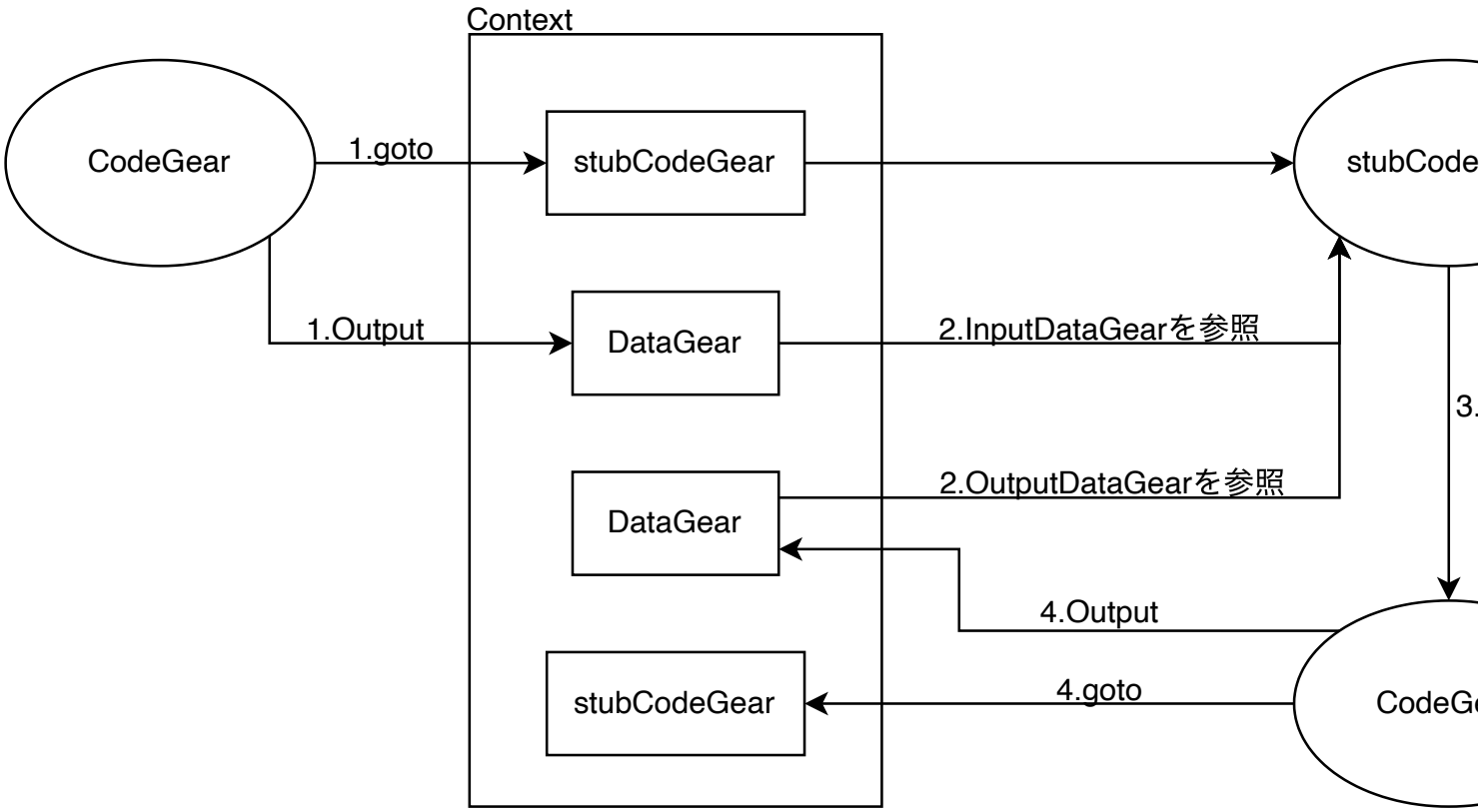
GearsOS

CodeGearとmetaCodeGearの関係


Data Gear
Code Gear
Data Gear
Meta Data Gear
Data Gear
Meta
Code Gear
Code Gear
Meta
Code Gear
Meta Data Gear
Data Gear

GearsOS

Contextを参照する流れ



UnixのFile system

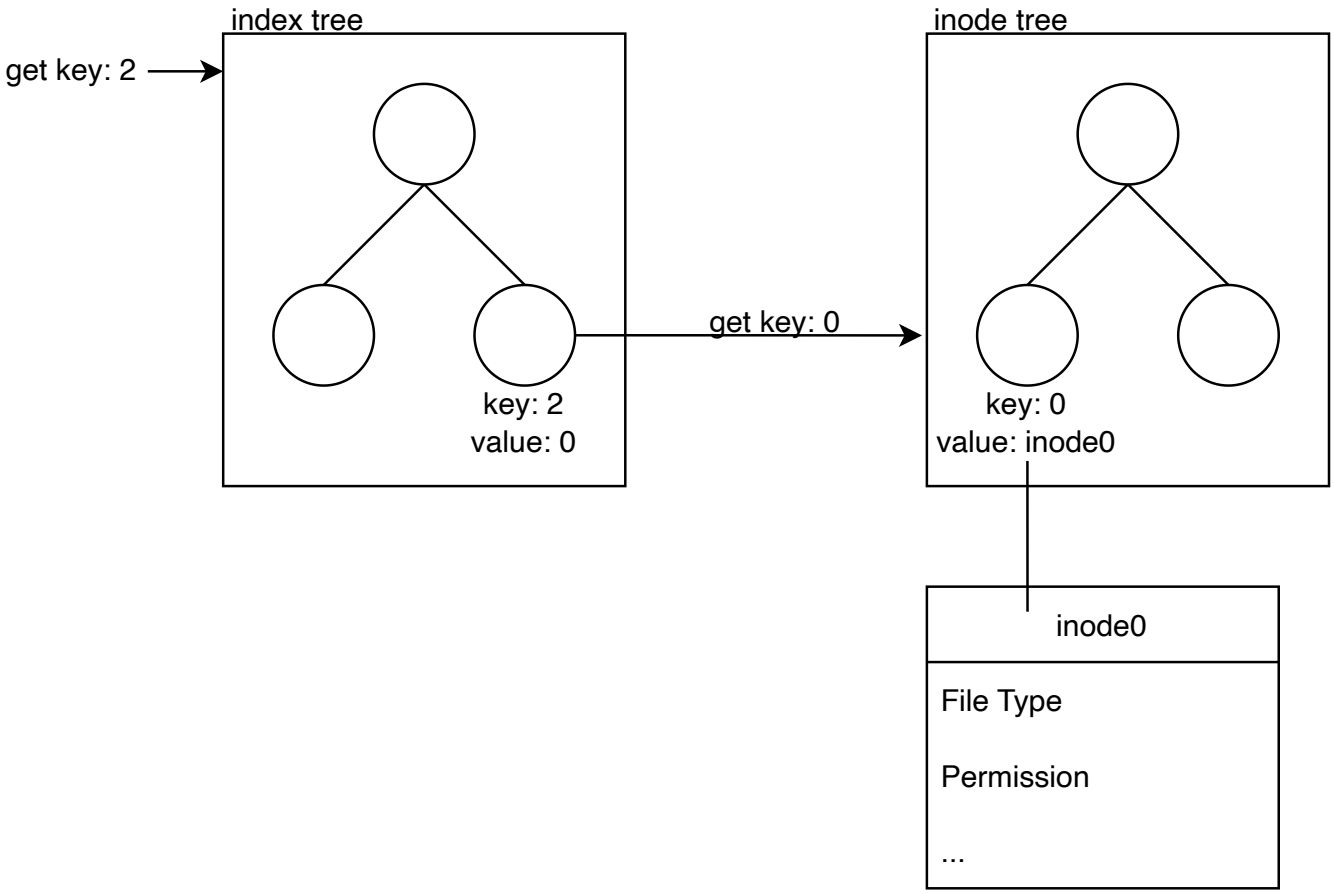
xv6

- MITで教育用の目的で開発されたOS
- Unixの基本的な構造を持つ
- 当研究室ではxv6のCbCでの書き換え，分析を行なっている
- File systemではinodeの仕組みが用いられている

inode

- ファイルの属性情報が書かれたデータである
- 識別番号としてinode numberを持つ
- inodeはファイルシステム始動時にinode領域をディスク上に確保する

GearsFileSystemにおけるdirectoryの構成



Unix Like な interface

mkdir

```
__code mkdir(struct GearsDirectoryImpl* gearsDirectory, struct Integer* name, __code next
    struct FTree* newDirectory = createFileSystemTree(context, gearsDirectory->currentDir
    Node* inode = new Node();
    inode->key = gearsDirectory->INodeNumber;
    inode->value = newDirectory;
    struct FTree* cDirectory = new FTree();
    cDirectory = gearsDirectory->iNodeTree;
    goto cDirectory->put(inode, mkdir2);
}

__code mkdir2(struct GearsDirectoryImpl* gearsDirectory, struct Integer* name, __code nex
    Node* dir = new Node();
    dir->key = name->value;
    Integer* iNum = new Integer();
    iNum->value = gearsDirectory->INodeNumber;
    dir->value = iNum;
    gearsDirectory->INodeNumber = gearsDirectory->INodeNumber + 1;
    struct FTree* cDirectory = new FTree();
    cDirectory = gearsDirectory->currentDirectory;
    goto cDirectory->put(dir, next(...));
}
```

Unix Like な interface

ls

```
__code ls(struct GearsDirectoryImpl* gearsDirectory, struct Integer* name, __code next(..
    Node* dir = new Node();
    dir->key = name->value;
    struct FTree* cDirectory = new FTree();
    cDirectory = gearsDirectory->currentDirectory;
    goto cDirectory->get(dir, ls2);
}

__code ls2(struct GearsDirectoryImpl* gearsDirectory, struct Node* node, __code next(...)
    printf("%d\n", node->key);
    goto next(...);
}
```

Unix Like な interface

cd

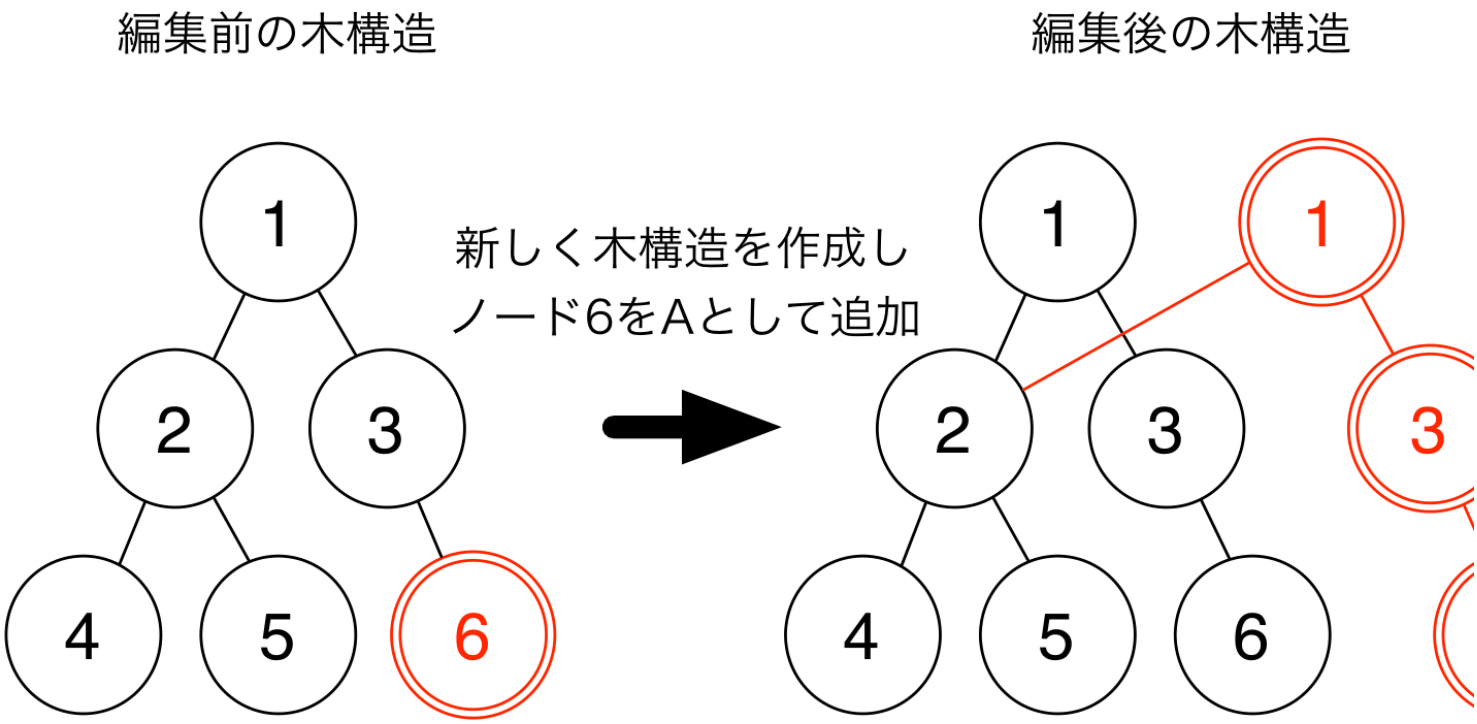
```
__code cd2Child(struct GearsDirectoryImpl* gearsDirectory, struct Integer* name, __code n
    struct FTree* cDirectory = new FTree();
    cDirectory = gearsDirectory->currentDirectory;
    struct Node* node = new Node();
    node->key = name->value;
    goto cDirectory->get(node, cd2Child2);
}

__code cd2Child2(struct GearsDirectoryImpl* gearsDirectory, struct Node* node, __code nex
    struct FTree* iNodeTree = new FTree();
    iNodeTree = gearsDirectory->iNodeTree;
    goto iNodeTree->get(node->value, cd2Child3);
}

__code cd2Child3(struct GearsDirectoryImpl* gearsDirectory, struct Node* node, __code nex
    gearsDirectory->currentDirectory = node->value;
    goto next(...);
}
```

GearsDirectoryの非破壊的編集によるバックアップ

- GearsOSにおける永続データは非破壊的な編集を行う木構造を用いて保存する
- ディレクトリシステム自体にバックアップの機能を搭載することが可能と考える



GearsFileSystemの今後

GearsShell

- 現状のGearsOSはユーザーの入力を受け付けることが出来ず，プログラミングインターフェースの様に機能している．
- gearsFileSystemなどGearsOSの各機能と接続し，今回作成したcdやlsの様なコマンドを受け付けるGearsShellを作成したい．

gearsDirectory filename

- 現状はgearsDirectoryのfilenameはIntegerの構造で管理されている
- filenameは一般的に文字列型であるためIntegerから文字列型に変更する必要がある

GearsFileSystemの今後

gearsDirectory path

- gearsDirectoryにはpathの機能が実装されていない
- full path指定のlsなどが実装できない状態である
- FileSystemTreeを拡張し，ノードをたどりpathを生成する様な機能を実装する必要がある

ファイルのバックアップ

- レコードのDataをファイルの差分履歴として保持し，日時情報を付け加えることでVersion Control Systemのような機能を持たせることが可能であると考えられる

まとめ

- gearsDirectoryの実装について説明した
- RedBlackTreeのシンプルなinterfaceにより比較的容易に実装を行うことができた
- 形式手法とファイルシステムの機能の両面で信頼性の向上が図れると考える
- RedBlackTreeを用いてinodeの仕組みを構築し，ls，cd，mkdirを作成するなどして， Unix Likeに構築することが出来た．