目次

1.2. FIFO方式でのファイルキャッシュの挙動

以下の図2にFIFO方式でのファイルキャッシュの更新の様子を示す。

具体的には、キャッシュ可能なデータブロックが4の場合に `4, 2, 4, 3, 4, 5, 3, 1, 6` という順でデータブロックにアクセスした際、最後の6がアクセスされたときの更新の様子である。

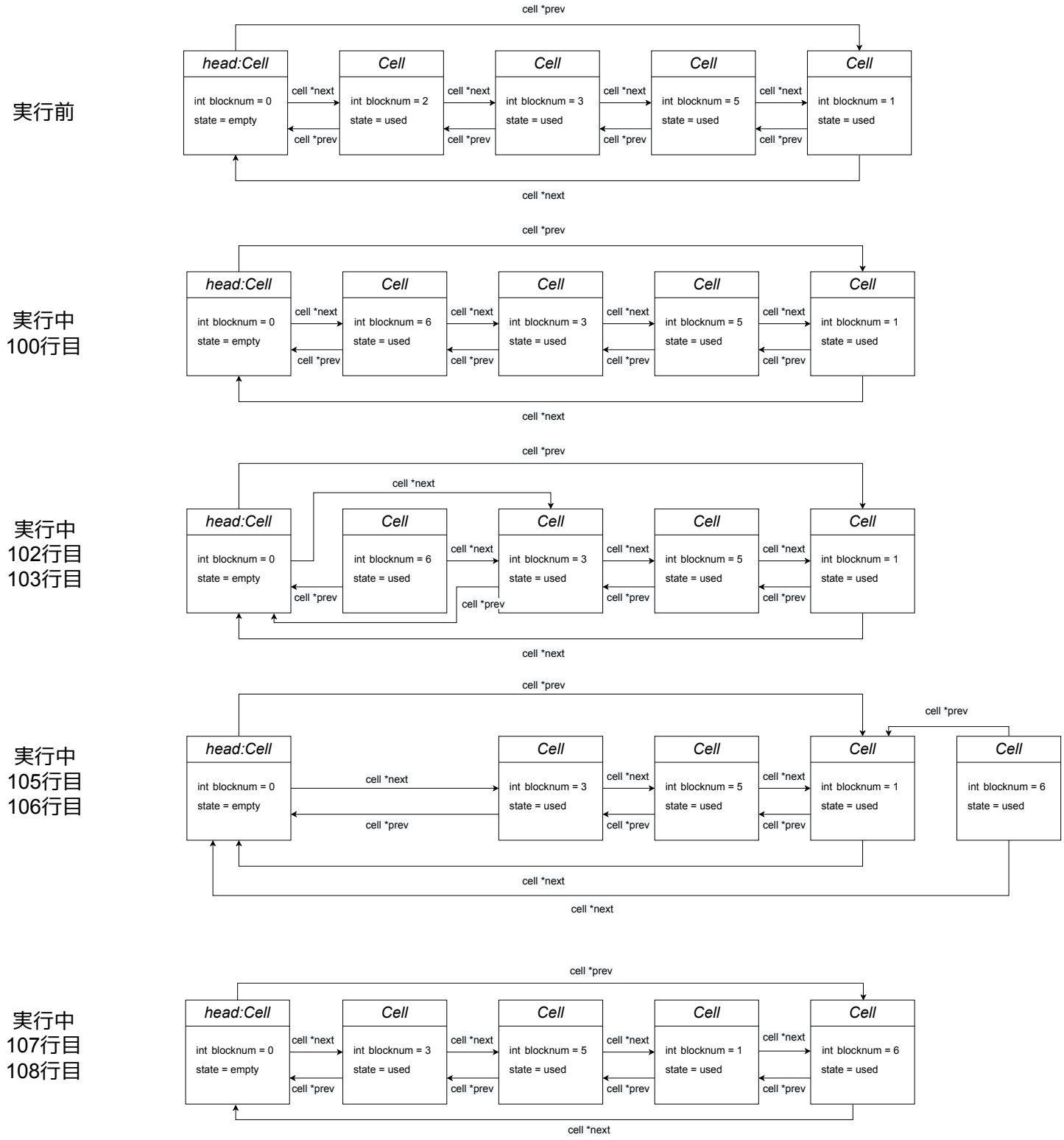

図2. FIFO方式でのファイルキャッシュの更新の様子

```
// runcmd() というインタフェースは変えたくないが
// コマンド文字列をパースするのは一度だけでいいため
// 実際にコマンドを実行する部分を runcmd_inner として分けた
void runcmd(char *buf)
{
        char *argv[ARGVSIZE];
        memset(argv, 0, ARGVSIZE);

        if (parsecmd(argv, buf, &buf[strlen(buf)]) > 0){
                int ecmd = 0;
                while(argv[ecmd] != NULL) ecmd++;
                runcmd_inner(argv, ecmd); // 再帰的にコマンド文字列を実行
        }
        exit(-1);
}
```

プログラム4. 拡張した simple_shell.c

4.2.3. 実験

実装した機能が正しく動作するかを確認する。
具体的には simple_shell と bash で同じコマンドを実行する。
simple_shell の出力が bash と同じであれば実装した機能は正しく動作していると言える。

4.2.3.1. リダイレクトの実験に使うプログラム

リダイレクトの実験には以下の標準出力に"stdout"、標準エラー出力に"stderr"と出力するプログラム `output.c` を用いる.

```
#include <stdio.h>

int main(){
    fprintf(stdout, "stdout\n");
    fprintf(stderr, "stderr\n");
}
```

プログラム5. output.c

4.2.3.2. 標準出力のリダイレクト(上書き)

プログラムの標準出力を `stdout.txt` に上書き保存する以下のコマンドを使ってテストする。

```
output.out > std.txt
```