日曜研究室

技術的な観点から日常を綴ります

[xv6 #65] Chapter 5 – File system – Code: Path names

テキストの73〜74ページ

本文

ディレクトリのように、パス名を実現するために必要となる追加のコードは少ししかない。
なぜなら、dirlookup関数を再帰的に呼び、namei関数などの関連がある関数を使うだけだからである。
namei関数は、引数のpathを階層的なパス名として評価し、それに一致するinodeを返す。
nameiparent関数は、namei関数の変種である。
nameiparent関数は、最後の要素の直前で止まり、親ディレクトリのinodeを返し、最後の要素を引数nameにコピーする。
両方の関数とも、実際の仕事をさせるために、一般化された関数であるnamex関数を呼ぶ。

fs.cのnamex, namei, nameiparent関数

```
01 // Look up and return the inode for a path name.
02 // If parent != 0, return the inode for the parent and copy the
   final
03 // path element into name, which must have room for DIRSIZ bytes.
04 static struct inode*
05 namex(char *path, int nameiparent, char *name)
06 {
07   struct inode *ip, *next;
08
09   if(*path == '/')
10     ip = iget(ROOTDEV, ROOTINO);
11   else
12     ip = idup(proc->cwd);
13
14   while((path = skipelem(path, name)) != 0){
15     ilock(ip);
16     if(ip->type != T_DIR){
17       iunlockput(ip);
18       return 0;
19     }
20     if(nameiparent && *path == '\0'){
```

```
21        // Stop one level early.
22        iunlock(ip);
23        return ip;
24      }
25      if((next = dirlookup(ip, name, 0)) == 0){
26        iunlockput(ip);
27        return 0;
28      }
29      iunlockput(ip);
30      ip = next;
31    }
32    if(nameiparent){
33      iput(ip);
34      return 0;
35    }
36    return ip;
37  }
38
39  struct inode*
40  namei(char *path)
41  {
42    char name[DIRSIZ];
43    return namex(path, 0, name);
44  }
45
46  struct inode*
47  nameiparent(char *path, char *name)
48  {
49    return namex(path, 1, name);
50  }
```

namex関数は、まずパスをどこから評価するかを決定する。
パスが”/”で始まっている場合、評価はルートディレクトリから開始される。
パスが”/”で始まっていない場合は、現在のディレクトリから評価が開始される。
そしたら、パスの各要素について処理を行うためにskipelem関数を使う。
ループ中の各反復では、現在のinodeであるipからnameを探さなければならない。
ループ中は、まずipをロックし、それがディレクトリであるかどうかをチェックする。
ディレクトリでなければ、namexは失敗となり0を返す。
(ipのロックが必要なのは、ip->typeが他のプロセスによって変更される可能性があるからではない。それは不可能である。ipのロックが必要なのは、ilockが実行されるまでは、ip->typeが、ディスクから読み込まれた直後のままであるかどうかは保証されないからである。)
nameiparent関数から呼び出され、かつパスの最後の要素の場合、nameiparent関数の定義に基づきループは早めに止まる。
最後のパスの要素は、すでにnameにコピーされているので、namex関数はアンロックされたipを返すだけでよい。
最後に、ループ中ではdirlookup関数を使ってパスの要素を調べ、ip = nextを実行することによって次の反復に備える。
ループでパスの要素を走査し尽くしたときは、ipを返す。

感想

すぐ下のディレクトリの層を使って階層的なパスをどう表現するかについてです。

ディレクトリの層では、各ディレクトリは名前とinodeの組をデータ構造上は1階層分しか持ちません。
そのinodeがディレクトリの場合、階層構造を成します。
今回は、その階層構造と、パス（各ディレクトリ階層を”/”で区切った文字列表現）を対応付けるコードの説明となります。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/4/19 木曜日 [http://peta.okechan.net/blog/archives/1663] |