操作系统(D)

项目 8

李子龙 518070910095

2021 年 4 月 11 日

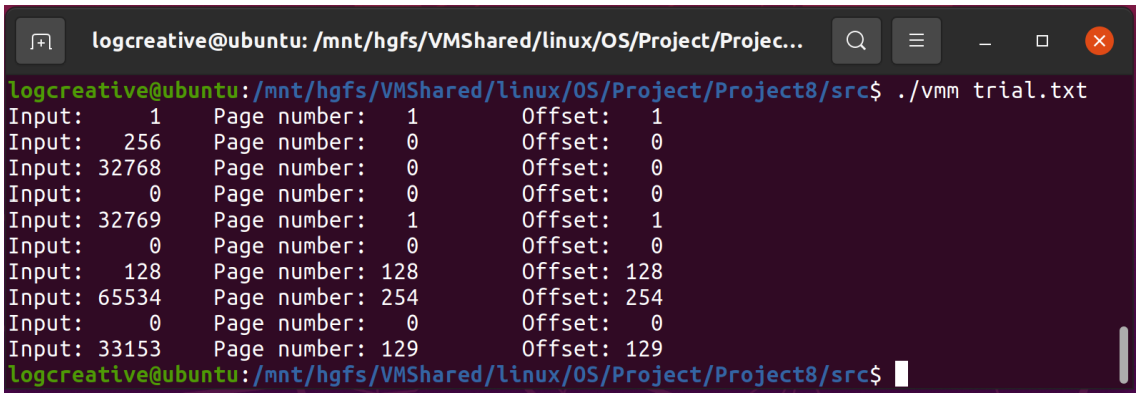
设计虚拟内存管理器

1. 起步

首先将测试用的地址写入 trial.txt，以测试 addext 内存地址分析模块的正确性。

Listing 1: src/trial.txt

```
1
256
32768
32769
128
65534
33153
```



定义地址结构和地址提取器如下：

Listing 2: src/addext.h

```c
#include <stdlib.h>

typedef struct address
{
    int number;
    int offset;
} add;
```

```
add addext(int _rline);
int getAdd(add _addin);
```

Listing 3: src/addext.c

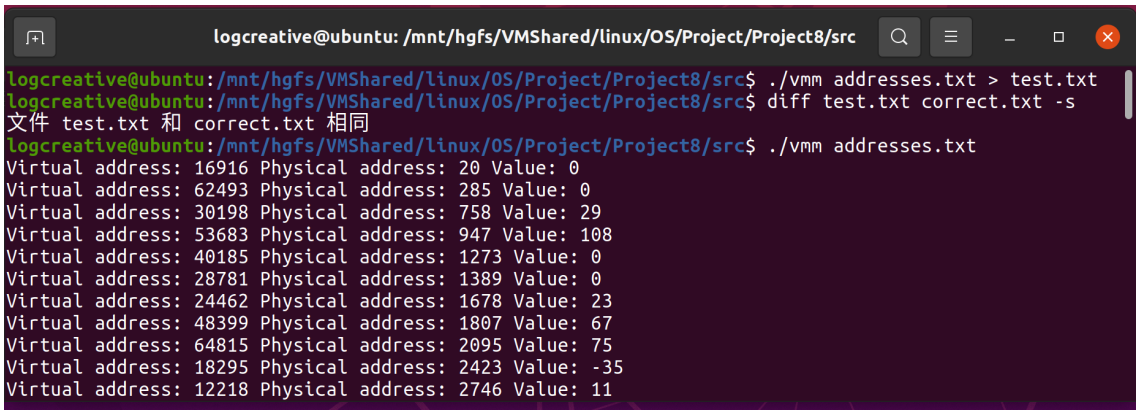
```
#include "addext.h"

add addext(int _rline) {
    add add_;
    _rline = _rline & 0x0000FFFF;
    add_.number = (_rline & 0x0000FF00) >> 8;
    add_.offset = _rline & 0x000000FF;
    return add_;
}

int getAdd(add _addin) {
    return (_addin.number << 8) + _addin.offset;
}
```

2. 处理页面错误

接着，先不考虑 TLB，只使用页表。将输出结果与正确参考比较，结论是正确：



首先对输入流分析，在 main 函数里的情形如下：

```
    while(fgets(addline, MAXLINE, addfile)!=NULL){
        int rline = atoi(addline);
        add viradd = addext(rline);
        add phyadd = getPhyAdd(viradd);
        fprintf(stdout, "Virtual address: %d Physical address: %d Value: %d\n",
            getAdd(viradd), getAdd(phyadd), getValue(phyadd));
    }
```

获取值是直接从内存中获得对应位置的值：

```
int getValue(add _phyadd) {
    return mem[_phyadd.number][_phyadd.offset];
}
```

其中 mem 是用 char 存储的：

Listing 4: src/memory.h

```
#ifndef MEMORY
#define MEMORY 1

#include <stdio.h>

#define MEMSIZE 256
#define FRAMESIZE 256

char mem[MEMSIZE][FRAMESIZE];

int read_frame(int page_number);

#endif
```

当前只使用页表是不需要考虑 TLB 的获取物理地址的函数如下：

```
add getPhyAdd(add _inadd) {
    add phyadd;

    if (!page_table[_inadd.number][1])
        handle_pagefault(_inadd.number);

    phyadd.number = page_table[_inadd.number][0];
    phyadd.offset = _inadd.offset;
    return phyadd;
}
```

一旦有页面错误就会触发对应的函数，将内容存放到内存中去：

```
void handle_pagefault(int page_number) {
    int frame_number = read_frame(page_number);
    page_table[page_number][0] = frame_number;
    page_table[page_number][1] = 1;
}
```

由于现在的内存充足，帧码直接用静态变量 `frame_number` 递增存储。

```
int read_frame(int page_number) {
    static int frame_number = 0;

    FILE* backstore;
    if ((backstore = fopen("BACKING_STORE.bin", "rb")) == NULL) {
        fprintf(stderr, "Empty file storage!\n");
        return -1;
    }

    int frame_number_ = frame_number++;
    long pos = page_number * FRAMESIZE;
    fseek(backstore, pos, SEEK_SET);
    fread(mem[frame_number_], sizeof(char), FRAMESIZE, backstore);
    fclose(backstore);

    return frame_number_;
}
```

这里使用了二进制文件读取的方式复制到内存中去。