OS2022 Project1 - Thread Management

> Department : 機械所系控組
>
> ID : R10522845
>
> Name : 郭忠翔

Why the result is not congruent with expected?

- 一開始給的代碼中，可以發現所有程序的 physicalPage 都共用，因此在test1和test2匯入後，彼此重疊到正在使用的page，也就是程式的虛擬記憶體映射到同一塊實體記憶體，所以得到不如預期的結果，因此要修改addrspace的內容

How to solve the issue?

- 要記錄main memory page的狀況，確認是否正在使用，因此使用一個布林變數`PhyPageState`標記，並且宣告instance，因為程式的大小必須要載入程式才會知道，所以`AddrSpace`把註解掉，將其內容移到`AddrSpace::Load`中

```
// usrprog/addspace.h
Class AddrSpace{
    public:
        // AddrSpace();
        ...
    private:
        ...
        static bool PhyPageState[NumPhysPages];
};

// usrprog/addrspace.cc
#include "copyright.h"
    ...
bool AddrSpace::usedPhyPage[NumPhysPages] = {0};
    ...
```

```
// move to Load
/*
AddrSpace::AddrSpace()
{
    pageTable = new TranslationEntry[NumPhysPages];
    for (unsigned int i = 0; i < NumPhysPages; i++) {
        pageTable[i].virtualPage = i;   // for now, virt page
# = phys page #
        pageTable[i].physicalPage = i;
//      pageTable[i].physicalPage = 0;
        pageTable[i].valid = TRUE;
//      pageTable[i].valid = FALSE;
        pageTable[i].use = FALSE;
        pageTable[i].dirty = FALSE;
        pageTable[i].readOnly = FALSE;
    }
    // zero out the entire address space
    // bzero(kernel->machine->mainMemory, MemorySize);
}
*/
```

- 當將程序載入記憶體時，要去填入 `pageTable` 映射到的 `physicalPage`，先建立一個跟實體記憶體一樣大的 pageTable，並線性搜尋到第一個未被使用的page填入，找到後使用`bzero`清空分配到的page

```
// usrprog/addrspace.cc
AddrSpace::Load()
{
    ...
    pageTable = new TranslationEntry[numPages];
//establish pageTable
    for(unsigned int i = 0, j = 0; i < numPages; i++) {
        pageTable[i].virtualPage = i;
        while(j < NumPhysPages && AddrSpace::PhyPageState[j])
j++; // if the page is used, find next
        bzero(&kernel->machine->mainMemory[j * PageSize],
PageSize);// free preallocated page
        AddrSpace::PhyPageState[j] = true;
        pageTable[i].physicalPage = j;
        pageTable[i].valid = true;
        pageTable[i].use = false;
        pageTable[i].dirty = false;
        pageTable[i].readOnly = false;
    }
    ...
```

```
}
```

- 找出映射到的位置，先用`virtualAddr`除`PageSize`，可以知道是哪個`page`，接著丟進`pageTable`找到對應的physical page，乘上`PageSize`後得到實體記憶體，再加上 page offset，就知道是在這頁的哪裡，而page offset是地址對`PageSize`取餘數

```
// usrprog/addrspace.cc
AddrSpace::Load()
{
    ...
    if(noffH.code.size>0){
        ...
        executable->ReadAt(
            &(kernel->machine->mainMemory[pageTable[noffH.code.virtualAddr/PageSize].physicalPage *             PageSize + (noffH.code.virtualAddr%PageSize)]),
            noffH.code.size, noffH.code.inFileAddr);
        ...
    }
    if(noffH.initData.size>0){
        ...
        executable->ReadAt(
            &(kernel->machine->mainMemory[pageTable[noffH.code.virtualAddr/PageSize].physicalPage *             PageSize + (noffH.code.virtualAddr%PageSize)]),
            noffH.code.size, noffH.code.inFileAddr);
        ...
    }
    delete executable;
    return TRUE;
}
```

- 程式執行完後，把程式占用的physical page釋放掉，讓之後的程式可以使用

```
// usrprog/addrspace.cc
AddrSpace::~AddrSpace()
{
    for(int i = 0; i < numPages; i++)
        AddrSpace::PhyPageState[pageTable[i].physicalPage] = false;
    delete pageTable;
}
```

Experiment result

```
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:6
return value:0
Print integer:25
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 300, idle 8, system 70, user 222
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Discussion

- 下方的程式是未修改的部分，最後一行被註解了，實際上應該要清掉記憶體才對，而且放的位置也好像不太適合，放在最後面是記憶體分給多個程式後就把所有記憶體清零，應該要分配一頁後清空一頁的記憶體，所以我認為是放在for-loop中，在這整段改到`AddrSpace::Load`後並加入

```
userprog/addrspace.cc
AddrSpace::AddrSpace()
{
    pageTable = new TranslationEntry[NumPhysPages];
    for (unsigned int i = 0; i < NumPhysPages; i++) {
        pageTable[i].virtualPage = i;   // for now, virt page # =
phys page #
        pageTable[i].physicalPage = i;
//      pageTable[i].physicalPage = 0;
        pageTable[i].valid = TRUE;
//      pageTable[i].valid = FALSE;
        pageTable[i].use = FALSE;
        pageTable[i].dirty = FALSE;
        pageTable[i].readOnly = FALSE;
    }
    // zero out the entire address space
    // bzero(kernel->machine->mainMemory, MemorySize);
}
```