实验三-实现新的系统调用

本文为草稿，<>中为待补充内容

一、实验目的

通过为xv6内核添加新的系统调用，理解系统调用的工作原理，进一步熟悉xv6内核的内部结构，巩固有关系统调用的教学内容

二、预备知识

1. C 语言编程
2. 系统调用相关知识

三、实验环境

预置实验环境

四、实验内容

1.实现新的系统调用sysinfo()

题目信息

难度：中等/困难

分数：100 分

所需时间：60min

题目介绍

在这个实验题目中，你将为xv6内核添加一个系统调用 sysinfo()，用来收集运行系统的信息，通过编写这个程序，你将进一步加深对系统调用相关教学知识的理解。

题目要求

在这个实验题目中，你需要为xv6内核添加一个系统调用 sysinfo()，用来收集运行系统的信息。这个系统调用接受一个参数：一个指向 sysinfo结构体的指针（见kernel/sysinfo.h）。内核应该填写这个结构体的freemem字段，这个字段应该被赋值为可用内存的字节数，nproc字段应该被赋值为状态不处于「UNUSED」的进程数。我们提供了一个测试程序 sysinfotest，如果你编写的程序正确，运行它就会输出 "sysinfotest: OK"。为确保评测程序能够正确地评测你的代码，应确保系统调用的名字为sysinfo。

运行效果

```
(在 xv6 shell 中)
$ hw
Hello World!
```

提示

1. 可以参考xv6下已有系统调用（比如open(),pipe()等）的实现来辅助你完成代码的编写，这些代码位于 kernel/syscall.c 和 sysfile.c 等代码文件中。
2. 我们为本实验设计了很多的评测点，评测点通过的难度由易到难逐步递进，即使你没有实现全部功能也无需担心，你仍然可以将代码上传至评测系统。评测系统会根据你的完成程度来进行阶段性的打分，比如你实现的sysinfo系统调用能正常地被调用，但是其什么操作都没有做（没有打印系统信息），这样仍能获得部分测试点的分数。
3. 为了向操作系统内核注册这个新系统调用，你需要在user/user.h中加上这个新系统调用的定义（函数原型），例如：

```
struct sysinfo;//提前声明sysinfo结构体
int sysinfo(struct sysinfo *);
```

   并在user*usys.pl*中加上这个系统调用的入口*(entry)*，以便于*xv6*在编译时自动地生成这条系统调用的入口（一些汇编语句）。于此同时，系统调用号也应被注册至 *kernel*syscall.h中。
4. 获取从用户空间来的系统调用参数的工具函数位于kernel*syscall.c*中，例如*argaddr*可以从系统调用参数中获取一个来自用户空间的指针值。你可以在*kernel*sysproc.c中看到它们的使用示例。关于xv6中系统调用是内核空间和用户空间的地址转换的相关知识，建议通过阅读xv6 book 文档来进行了解。
5. sysinfo系统调用返回事，需要将sysinfo结构体拷贝回用户空间；参见sys_fstat() (kernel*sysfile.c)*和*filestat() (kernel*file.c)所做的操作，了解如何使用copyout()进行从内核空间到用户空间的拷贝。
6. 建议做实验前先完整阅读xv6 book 文档，重点阅读里面关于系统调用的章节。
7. <这个实验的提示还不够全面，后续仍需补充>

<后续可以把这个实验拆成两个题目，一个简单些（系统调用输出helloworld），一个困难些（sysinfo）>