lab2 part1

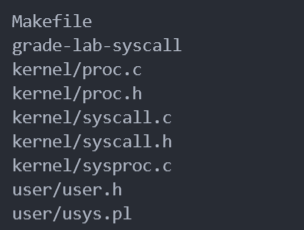
1、要求

系统调用信息打印。添加一个名为 trace 的系统调用。原型为 int trace(int mask)。参数mask：每一位对应一个系统调用，位的比特值指示是否需要追踪对应的系统调用。

2、思路

1.步骤一：在各个文件完成trace这个系统调用的注册。

变更的文件包含这些：



Makefile：增加$U/\_tarce\ 表项。

kernel/proc.c：修改fork的逻辑使其复制mask字段，代码： np->mask = p->mask;

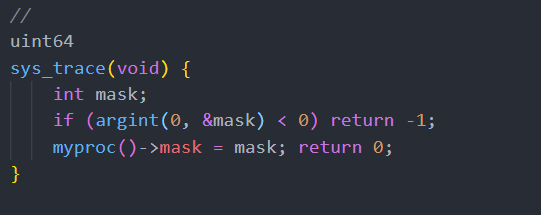
kernel/proc.h：修改struct proc 增加 int mask; 字段。

kernel/syscall.c：编写主要逻辑。

kernel/syscall.h：添加枚举 #define SYS\_trace 22

kernel/sysproc.c：该文件主要负责：Process-related system calls.

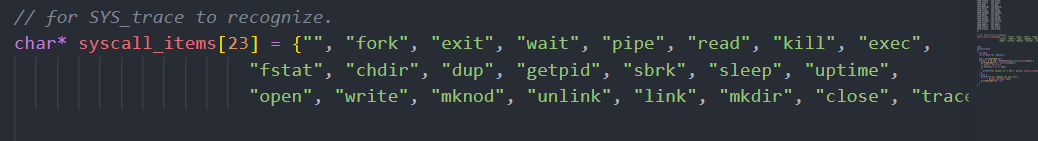
于是向其中添加的是sys\_trace这个调用的原型：



user/user.h：添加函数原型 int trace(int); 以通过编译。

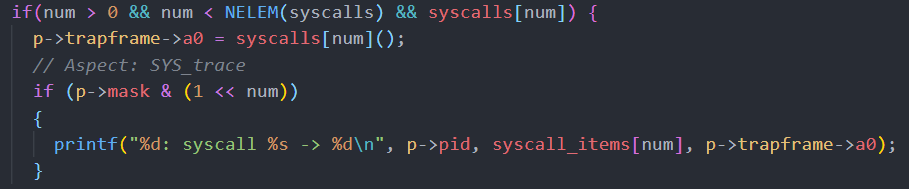
user/usys.pl：增加entry用于生成asm文件。

2.步骤二：在syscall.c源文件中编写trace调用的具体逻辑。

首先建立trace可识别的shell输入的命令的名称：

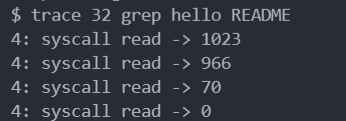
注意这里除了22个调用外还应当包含空“”。

接着在触发 p->trapframe->a0 = syscalls[num](); 即调用的时候，增加printf输出作为记录。本质上，**这个逻辑就是增加了一个Logger切面。**

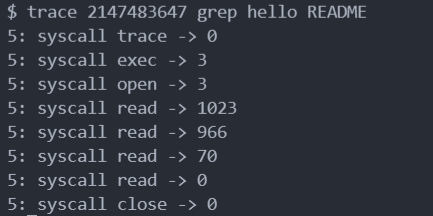


3、测试

测试1：



测试2：



测试3：空输入测试，测试目的是保证编写的trace调用不会对grep程序造成影响。

测试4：测试trace对子进程的功能

trace 2 usertests forkforkfork

这个输出比较长，未截图。

4、思考：

1.为什么trace的核心逻辑编写于syscall.c而不是其他调用一样写在sysproc.c？

解释：sysproc.c负责Process-related system calls，syscall.c负责Dispatch system calls to handling function.但是trace这个调用并不属于是Process-related，它只是一个日志切面，所以应当写在syscall.c这个dispatcher里面，而不是写在sysproc.c里面作为一个Handler。

2.为什么测试能返回稳定的结果？

解释：首先要明白“随机”和“稳定”的区别。随机可以理解为在程序运行前运行结果无法预判，稳定可以理解为多次执行同一个程序能得到相同的结果。

显然，结果的稳定性是结果可测试的基础。在这里输入字符串引起不稳定的因素主要是pid等。

可以证明，pid是稳定的。虽然内核调度算法的执行过程是随机的，但是由于xv6的调度算法比较简单，所以是稳定的。也就是跑两次测试程序，trace结果中分配的pid始终是固定的。

lab2 part2

1、要求

在该任务中，你需要加入一条新的系统调用，叫做sysinfo。该系统调用将收集xv6运行的一些信息和当前进程的一些信息。

freemem：xv6当前剩余的内存 字节 数

nproc： xv6状态为UNUSED 的进程个数

freefd：当前进程可用文件描述符的数量，即 尚未使用 的文件描述符数量

2、思路

这里就需要编写一个具体的Handler了，并且需要调用sysproc.c中的一些getter函数。另外值得注意的是这里引入了内存相关的调用。

之前仍然是先在各个文件里面注册sysinfo调用。

sysinfo的主要逻辑就是填充结构体

struct sysinfo {

  uint64 freemem; *// amount of free memory (bytes)*

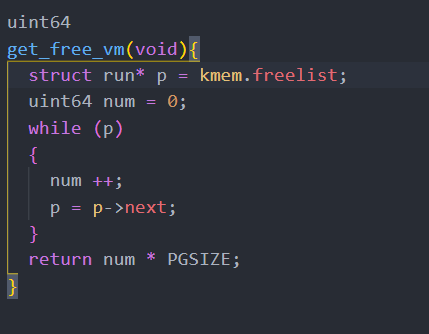
  uint64 nproc; *// number of process*

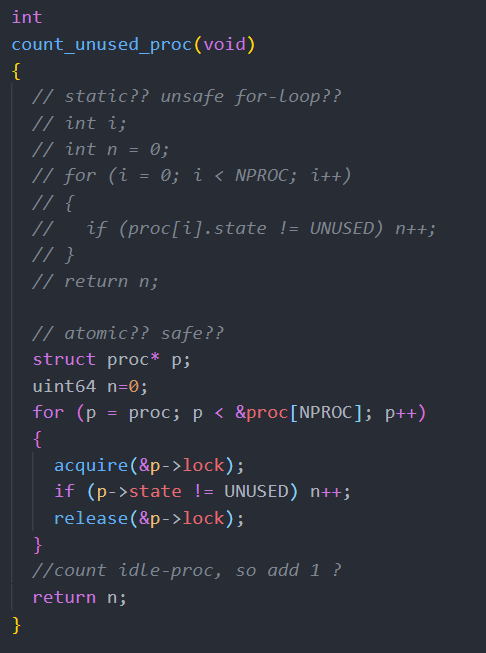
  uint64 freefd; *// number of free file descriptor*

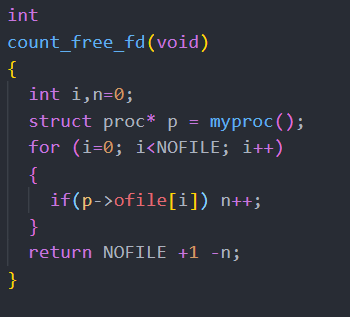
};

中的三个字段。

分别编写getter函数如下：

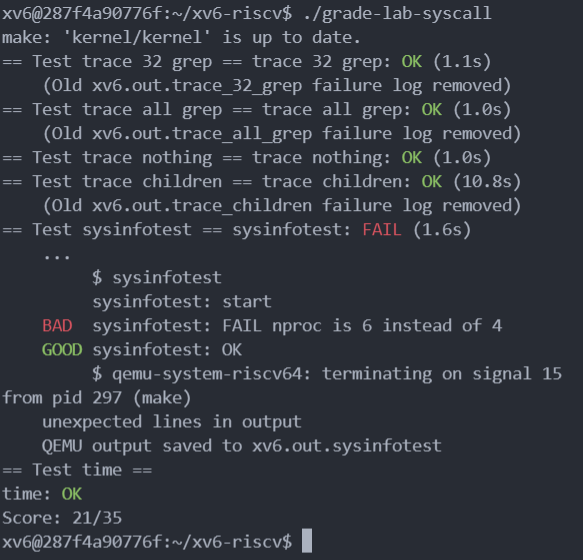






实际上，由于进程是在动态创建的，所以在count\_unused\_proc中推荐下面一种写法，是原子的计数。

最终测试结果



可以看到，syscall trace全部通过。

但是syscall sysinfo还存在Bug，但是我实在是找不出来原因了。