LAB6-Extra

背景说明

本次LAB6-Extra主要是让大家根据**课程组官方给定的规范**实现一套PV操作的系统调用接口。

用语说明

- pv_id非法：该pv_id无法映射到任何一个已经分配了的信号量。

接口说明

本次实验需要你实现以下几个系统调用函数。

syscall_init_PV_var

说明

用于初始化信号量，并返回该信号量的id。id是信号量的唯一标识，且不小于0。如果此时信号量都已经被分配，即，此时没有空闲的信号量，则返回负值。

规格

```
int syscall_init_PV_var(int init_value);
```

参数说明

init_value：该信号量的初始值。

返回值说明

分配的信号量的id。id是信号量的唯一标识。id规则除必须不小于0以及可以作为唯一标识以外没有规则设计约束，不会对此进行考察。如果此时系统的信号量资源已经用尽或处理过程中出错，则返回一个负值(可以自定义)。

syscall_P

说明

行为参考操作系统课件。为便于查阅，标程实现的PV操作规范将在后文给出。

此外，有几点补充说明。

- 如果给出的pv_id非法，则不执行任何修改信号量以及进程调度的操作。
- 等待该信号量的进程的的管理需要采用FIFO的调度。

规格

```
void syscall_P(int pv_id);
```

参数说明

pv_id：进行P操作的信号量的id。

syscall_V

说明

行为参考操作系统课件。为便于查阅，标程实现的PV操作规范将在后文给出。

此外，有几点补充说明。

- 如果给出的pv_id非法，则不执行任何修改信号量以及进程调度的操作。
- 等待该信号量的进程的的管理需要采用FIFO的调度。

规格

```
void syscall_V(int pv_id);
```

参数说明

pv_id：进行P操作的信号量的id。

syscall_check_PV_value

说明

查询某个信号量当前的值。需给出该信号量的pv_id，如果pv_id非法则可以返回任意值，否则返回该pv_id对应信号量的值。

规格

```
int syscall_check_PV_value(int pv_id);
```

参数说明

pv_id：待查询信号量的pv_id。

返回值说明

查询结果

syscall_release_PV_var

说明

释放信号量，将该信号量的状态设置为未分配。如果有进程在等待该信号量，则将这些进程结束(env_free)，即调用该接口的进程去kill掉等待该信号量的全部进程。**为了保证不影响测试，请不要注释掉以下调试信息**。如果pv_id非法则不执行任何操作。

```c
void
env_free(struct Env *e)
{
    Pte *pt;
    u_int pdeno, pteno, pa;

    /* Hint: Note the environment's demise.*/
    printf("[%08x] free env %08x\n", curenv ? curenv->env_id : 0, e->env_id);

    /* Hint: Flush all mapped pages in the user portion of the address space */
    for (pdeno = 0; pdeno < PDX(UTOP); pdeno++) {
        /* Hint: only look at mapped page tables. */
        if (!(e->env_pgdir[pdeno] & PTE_V)) {
            continue;
        }
        /* Hint: find the pa and va of the page table. */
        pa = PTE_ADDR(e->env_pgdir[pdeno]);
        pt = (Pte *)KADDR(pa);
        /* Hint: Unmap all PTEs in this page table. */
```

规格

```c
void syscall_release_PV_var(int pv_id);
```

参数说明

pv_id: 待查询信号量的pv_id。

测试方式

官方测试

机制

共两组测试点。每组测试点的测试方法为，编写一个用户级程序(**内部会使用fork函数**)，调用学生提供的PV操作接口，实现有关输出。

测试内容

详细测试点不会公布。但可以保证所测试的PV操作有关行为皆在前文进行表述中。此外，出错处理的规格仅为说明的完整性，测试不涉及。

如果其他lab有Bug，自然可能导致出错。

补充说明

测试点通过几十万次(至少10万)的大循环来保证不同进程的执行到PV操作相关代码的顺序，基本可以排除因为进程执行速度的原因导致无法通过测试。

测试约束

- 所测试的PV操作行为皆在规格说明中给出
- 错误处理不涉及，仅为了规格的完整。
- **至少支持5个PV变量**
- 每个PV变量至少支持**10个等待进程**
- pv操作的值为32位有符号整数
- 运行时间10s(避免你们担心测试样例的大循环导致TLE)

自行测试建议与要求

- 参考lab4进行测试
- **必须编写生产消费者模型。**不怕简单，不怕naive，就怕你不写(可能会在答辩时抽查)

实现说明

建议

- 参考lab4的系统调用实现方式
- 定义新的数据结构，可以参考进程控制块等(实现数组)。
- 建议直接在已有的文件里修改

要求

- 必须在`user/lib.h`里添加有关声明！
- 不要修改官方给的`env_free`函数的调试信息！
- 为了便于测试，我们检查了进程的编号。要求第一个进程的id为0x800,第二个为0x1001，第三个为0x1802。(实际上，没有随便碰官方给的进程编号代码，就没问题。以防万一，建议自行测试一下)

补充说明

球球您们直接在照着lab4的系统调用去加，结构体定义的直接参考`Env`，全局变量声明参考进程控制块数组，定义就放在`lib/env.c`。这样应该能减少一定的神必问题，秋梨膏！

提示

基本上已经在补充说明里说了

想说的话

这次实验本意是让大家体验一下PV操作，并动手尝试实现简单的PV操作。我们的规格有不少漏洞，但是体验PV操作还是足够的。助教的标程就不放出来了，随手搓的，太丑了，有点丢人。希望大家能够享受一下实现一些简单小功能的快乐。以上。

联系我们

题意

王子牧：14317593@qq.com 微信：一只无奈的猫(除非能忍受逢年过节撒狗粮，否则不建议添加)

评测环境等神必问题

胡柯杨、张佳辉

PV规范说明

P操作的主要动作是：

①S减1；

②若S减1后仍大于或等于0，则进程继续执行；

③若S减1后小于0，则该进程被阻塞后放入等待该信号量的等待队列中，然后转进程调度。

V操作的主要动作是：

①S加1；

②若相加后结果大于0，则进程继续执行；

③若相加后结果小于或等于0，则从该信号的等待队列中释放一个等待进程，然后再返回原进程继续执行或转进程调度。