# OSLAB-4实验报告

`OSLAB讲义`


> 南京大学匡亚明学院 刘志刚
> 学号：141242022
> 邮箱：njuallen@foxmail.com


---

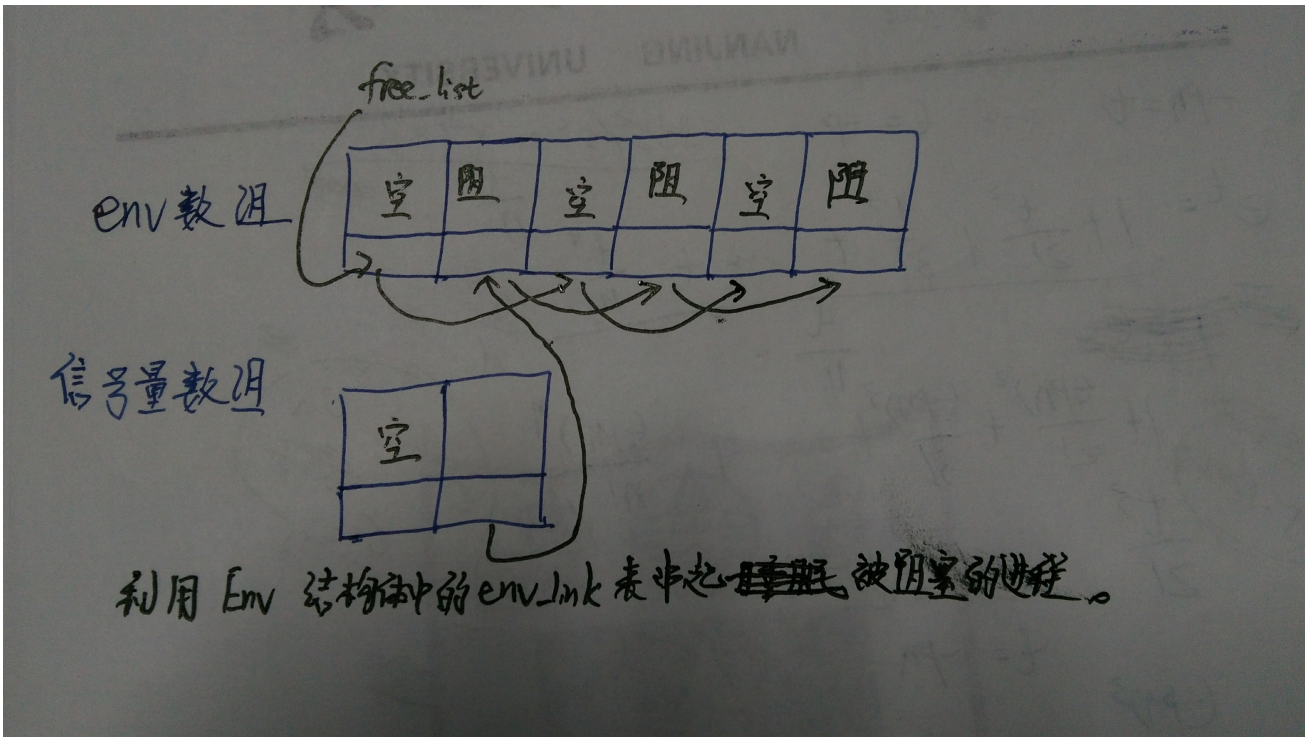
## 实验进度

总体进度：完成了lab4。

- [x] 具名信号量
- [x] 多线程
- [x] 共享内存
- [x] 生产者消费者问题（分别基于多进程与多线程）

具体说明：

- 我的具名信号量的实现就是在kernel中加入了一些用于处理信号量的函数（详见kern/sem.c）。信号量的标识是用一个整型数，可以看作是一个信号量描述符。通过sem_open获得一个信号量描述符，其他的操作都接受一个信号量描述符来指定信号量。我在内核中实现的是一般信号量。
- 信号量具体在内核中的组织是这样的：有一个信号量数组，并在其上构建了一个信号量的free_list。free_list串起当前可用的信号量。同时每一个信号量都应该有一个对应的阻塞队列，为了使实现更简洁，我采取了一些小trick。注意到：已分配的Env结构体中的env_link域实际上是未被使用，因此我们可以通过它来串起一个阻塞队列。示意图如下：




- jthread线程库
  参照了讲义上的提示，我的线程本质上就是共享同一套地址空间的进程。为了实现进程地址空间的共享，我新增加了一个vfork系统调用，它接受一个参数stack_top，指定新的进程的栈的位置。vfork创建一个子进程，它的cr3的值与父进程相同。该系统调用同时会增加父进程页目录表的引用计数。为了防止一个线程的终止导致整个地址空间都被释放，我修改了env_free函数，使得只有当父进程页目录表的引用计数为0时，才真正释放地址空间。但我这样子就有一个问题是，即使主线程退出了，整个进程还不会终止，必须等所有线程都退出了，整个进程才终止。
  参照jos微内核的实现风格，我在内核中只实现了一个最简单的vfork。我在用户态还增加了一系列的封装函数，提供了jthread_create，jthread_exit，jthread_yield等一系列接口。由于我们的线程退出后，线程占用的资源（栈所占用的内存空间）无法被释放，为了能更有效地利用资源。我在用户态lib中，对当前进程中的线程创建情况进行了记录。每当有线程退出时，这个线程所占有的资源将会被放到空闲链表的头部。这样线程占有的栈空间虽然不会被释放，但可以被重复利用。同时，通过用户态的库的封装，我的jthread在创建时可以传递参数，同时return之后会自动销毁。
  我在jthread库中还实现了一个信号量，是使用jos原有的ipc机制做的。但在实现了进程间的信号量之后，这个东西似乎就没有什么必要了。

- 共享内存
  jos之前的实验已经让我们实现了共享内存，通过page_map，ipc_send即可实现进程间内

存的共享。

- 生产者消费者问题
  我利用进程模型和线程模型解决了生产者消费者问题。分别是在user/ppc.c和user/tpc.c中，其中进程模型只是实现了同步，并没有实现数据的共享与传递。而线程模型通过一个全局变量buffer，可以传递数据。默认的buffer大小为10，生产者消费者各为5个。
  运行`make run-ppc-nox`可以查看进程模型的输出。
  运行`make run-tpc-nox`可以查看线程模型的输出。
  同时我还提供了一个程序，用于检查输出的结果的正确性（只检查生产消费序列对不对，不检查生产消费的产品对不对）。pcchecker这个程序会检查produce、consume序列，如果出现了某个时刻buffer中的产品数比buffer大或者当buffer已经为空时还有consumer在消费的情况，就会输出wrong。否则，它会在输入结束时输出correct。pcchecker接受一个命令行参数N，即buffer的大小。
  我们可以通过管道让pcchecker检查ppc和tpc程序的输出。例如运行`make run-ppc-nox | ./pcchecker 10`我们可以检查在buffer大小为10时，程序输出是否正确。