

lab4 实验报告

姓名：刘佳璇

学号：231220105

邮箱：13573529579@163.com

实验进度

我完成了lab 4.1，lab 4.2，lab 4.3，lab 4.4，其中lab 4.4实现了生产者-消费者问题。

实验结果

QEMU - Press Ctrl-Alt to exit mouse grab

```
Input:" Test %c Test %6s %d %x"  
Test a Test oslab 2025 0xadc  
Ret: 4; a, oslab, 2025, adc.
```

```
QEMU
Parent Process: Semaphore Initializing.
Parent Process: Sleeping.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Parent Process: Semaphore Posting.
Parent Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Parent Process: Semaphore Posting.
Parent Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Destroying.
Parent Process: Semaphore Posting.
Parent Process: Sleeping.
Parent Process: Semaphore Posting.
Parent Process: Semaphore Destroying.
-
```

```
QEMU - Press Ctrl-Alt to exit mouse grab
=====TEST SHAREDVARIABLE=====
Parent Process: create Shared Variable: 0 with value: 114514
Child Process: readShared Variable: 0 get value: 114514
Parent Process: writeShared Variable: 0 with value: -5678
Child Process: readShared Variable: 0 get value: -5678
Child Process: writeShared Variable: 0 with value: 2333
Parent Process: readShared Variable: 0 get value: 2333
Parent Process: destroyShared Variable: 0
```

```
QEMU - Press Ctrl-Alt to exit mouse grab
Producer 2: produce
Producer 3: produce
Consumer: consume
Producer 4: produce
Producer 5: produce
Producer 2: produce
Producer 3: produce
Consumer: consume
Producer 4: produce
Producer 5: produce
Consumer: consume
Consumer: consume
Consumer: consume
Consumer: consume
Consumer: consume
Consumer: consume
```

实验修改的代码位置

lab 4.1

1. kernel/kernel/irqHandle.c:

- void sysReadStdIn(struct StackFrame *sf); // 添加函数实现
- void keyboardHandle(struct StackFrame *sf); // 添加键盘按键的串口回显和控制进程同步的代码

lab 4.2

1. kernel/kernel/irqHandle.c: 添加函数实现

- void sysSemInit(struct StackFrame *sf);
- void sysSemWait(struct StackFrame *sf);
- void sysSemPost(struct StackFrame *sf);
- void sysSemDestroy(struct StackFrame *sf);

lab 4.3

1. kernel/kernel/irqHandle.c: 添加函数实现

- void sysSVarCreate(struct StackFrame *sf);
- void sysSVarDestroy(struct StackFrame *sf);
- void sysSVarRead(struct StackFrame *sf);
- void sysSVarWrite(struct StackFrame *sf);

2. lib/syscall.c: 添加函数实现

- `int createSharedVariable(sharedvar_t *svar, int value);`
- `int destroySharedVariable(sharedvar_t *svar);`
- `int readSharedVariable(sharedvar_t *svar);`
- `int writeSharedVariable(sharedvar_t *svar, int value);`

3. kernel/kernel/kvm.c: 添加函数实现

- `initSharedVariable(void);`

lab 4.4

1. lib/lib.h

- 添加宏定义 `#define SYS_GETPID 6`，并修改原有的宏定义 `#define SYS_SEM 7` 和 `#define SYS_SHAREDVAR 8`
- `pid_t getpid();` // 添加函数定义

2. lib/syscall.c

- `pid_t getpid();` // 添加函数实现

3. kernel/kernel/irqHandle.c

- 添加宏定义 `#define SYS_GETPID 6`，并修改原有的宏定义 `#define SYS_SEM 7` 和 `#define SYS_SHAREDVAR 8`
- `void syscallHandle(struct StackFrame *sf);` // 添加 case SYS_GETPID
- `void sysGetPid(struct StackFrame *sf);` // 添加函数定义和实现

4. app/main.c

- 添加宏定义
 - `int item[N] = {}`
- 添加全局变量
 - `sem_t full;`
 - `sem_t empty;`
 - `sem_t mutex;`
 - `int in = 0;`
 - `int out = 0;`
 - `int id;`
- `void produce();` // 添加定义和实现
- `void consume();` // 添加定义和实现
- 添加测试程序，每个生产者生产2次，消费者消费8次