操作系统

实 验 报 告

实验名称：信号量的实现

学　　员：侯华玮　　学　　号：202102001015

培养类型：无军籍学员　　年　　级：21 级

专业：计算机科学与技术（天河拔尖班）　所属学院：计算机学院

指导教员：文艳军　　职　　称：教授

实 验 室：　　实验日期：2023.05.12

国防科技大学训练部制

《实验报告》填写说明

1. 学员完成人才培养方案和课程标准要所要求的每个实验后，均须提交实验报告。

2．实验报告封面必须打印，报告内容可以手写或打印。

3．实验报告内容编排及打印应符合以下要求：

（1）采用 A4（21cm×29.7cm）白色复印纸，单面黑字打印。上下左右各侧的页边距均为 3cm；缺省文档网格：字号为小 4 号，中文为宋体，英文和阿拉伯数字为 Times New Roman，每页 30 行，每行 36 字；页脚距边界为 2.5cm，页码置于页脚、居中，采用小 5 号阿拉伯数字从 1 开始连续编排，封面不编页码。

（2）报告正文最多可设四级标题，字体均为黑体，第一级标题字号为 4 号，其余各级标题为小 4 号；标题序号第一级用“一、”、“二、”……，第二级用“（一）”、“（二）”……，第三级用“1.”、“2.”……，第四级用“（1）”、“（2）”……，分别按序连续编排。

（3）正文插图、表格中的文字字号均为 5 号。

一、实验目的和内容

实验目的：

- 加深对进程同步与互斥概念的认识；
- 掌握信号量的使用，并应用它解决生产者——消费者问题；
- 掌握信号量的实现原理。

实验内容：

- 在 Ubuntu 下编写程序，用信号量解决生产者——消费者问题；
- 在 0.11 中实现信号量，用生产者—消费者程序检验之。

二、操作方法与实验步骤

信号量系统调用

```
sem_t* sys_sem_open(const char* name, unsigned int value)

int sys_sem_wait(sem_t *sem)

int sys_sem_post(sem_t *sem)

int sys_sem_unlink(const char* name)
```

- `sem_open()` 的功能是创建一个信号量，或打开一个已经存在的信号量。
  - `sem_t` 是信号量类型，根据实现的需要自定义。
  - `name` 是信号量的名字。不同的进程可以通过提供同样的 name 而共享同一个信号量。如果该信号量不存在，就创建新的名为 name 的信号量；如果存在，就打开已经存在的名为 name 的信号量。
  - `value` 是信号量的初值，仅当新建信号量时，此参数才有效，其余情况下它被忽略。当成功时，返回值是该信号量的唯一标识（比如，在内核的地址、ID 等），由另两个系统调用使用。如失败，返回值是 NULL。

● `sem_wait()` 就是信号量的 P 原子操作。如果继续运行的条件不满足，则令调用进程等待在信号量 sem 上。返回 0 表示成功，返回 -1 表示失败。

● `sem_post()` 就是信号量的 V 原子操作。如果有等待 sem 的进程，它会唤醒其中的一个。返回 0 表示成功，返回 -1 表示失败。

● `sem_unlink()` 的功能是删除名为 name 的信号量。返回 0 表示成功，返回 -1 表示失败。

生产者消费者问题

利用上面实现的信号量系统调用，编写一个应用程序“pc.c”来模拟经典的生产者 消费者之间的同步。

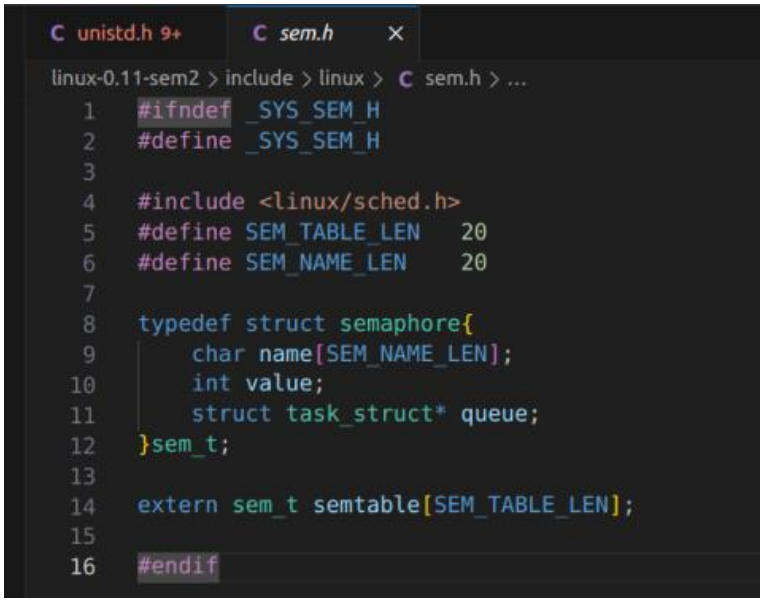
在这个程序中，要建立 1 个生产者进程，5 个消费者进程；用文件建立一个共享缓冲区；生产者进程依次向缓冲区写入整数 0,1,2,...,499；每个消费者进程从缓冲区中读取 100 个数，每读取一个数字就打印到标准输出上；缓冲区文件最多只能保存 10 个数。

实验步骤

1 添加信号量系统调用

1. 修改 unist.h 与 sys.h 文件，添加有关信号量系统调用的信息。
2. 添加信号量头文件 sem.h 与信号量系统调用实现文件 sem.c。

头文件中声明了信号量的结构体，包括 name, value, queue 三个成员。



信号量系统调用实现：

sys_sem_open

sys_sem_unlink

```
int sys_sem_unlink(const char *name){
    if(sem_cnt > SEM_TABLE_LEN - 1){
        return -1;
    }

    char kernelname[50];
    int i = 0;
    int name_len = 0;

    while (get_fs_byte(name + name_len) != '\0')
        name_len ++;
    if( name_len > SEM_NAME_LEN)
        return -1;
    for(i = 0; i < name_len; i++){
        kernelname[i] = get_fs_byte(name + i);
    }

    int isExist = 0;
    for(i = 0; i< sem_cnt; i++){
        if(!strcmp(name, semtable[i].name)){
            isExist = 1;
            break;
        }
    }
    if(isExist){
        int j = 0;
        for(j = i; j < sem_cnt; j++){
            semtable[j]=semtable[j+1];
        }
        sem_cnt --;
        return 0;
    }
    else{
        return -1;
    }
}
```

2 编写生产者-消费者程序

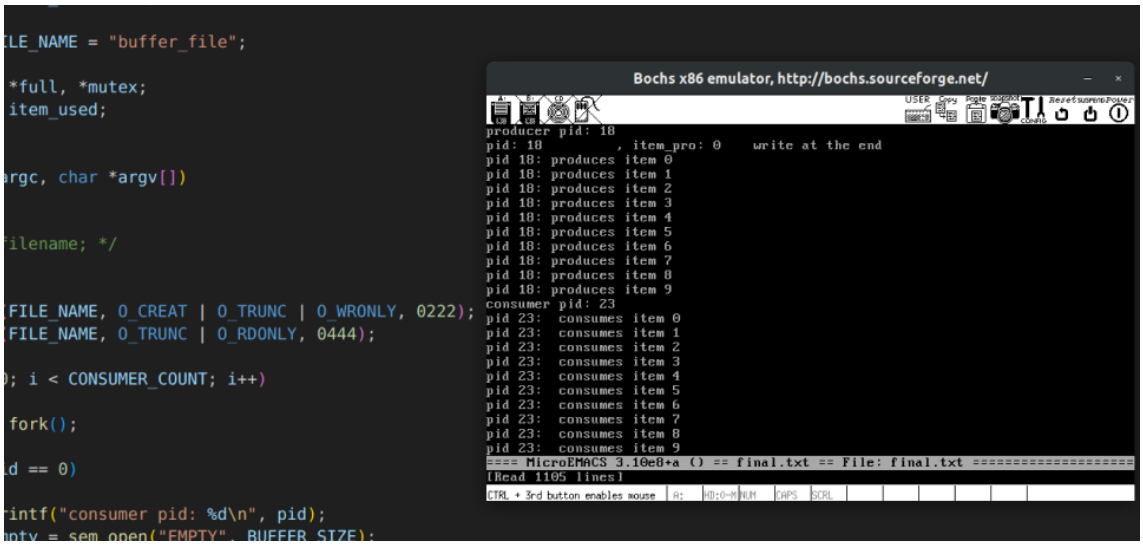
生产者-父进程

```
/* 生产者进程 */
pid = getpid();
printf("producer pid: %d\n", pid);
for (item_pro = 0; item_pro < ITEM_COUNT; item_pro++)
{
    sem_wait(empty);
    sem_wait(mutex);
    /* 读写文件 */
    if (!(item_pro % BUFFER_SIZE)){
        printf("pid: %d \t, item_pro: %d \t write at the end\n", pid, item_pro);
        lseek(fw, 0, 0);
    }
    write(fw, (char *)&item_pro, sizeof(int));

    printf("pid %d:\tproduces item %d\n", pid, item_pro);
    fflush(stdout);
    sem_post(mutex);
    sem_post(full);
}
```

消费者-子进程

```c
for (i = 0; i < CONSUMER_COUNT; i++)
{
    pid = fork();

    if (pid == 0)
    {
        pid = getpid();
        printf("consumer pid: %d\n", pid);
        fflush(stdout);

        while (1)
        {
            sem_wait(full);
            sem_wait(mutex);
            /*  读写文件 */

            /* read(fr, (char *)&item_used, sizeof(int)); */
            if (!read(fr, (char *)&item_used, sizeof(item_used)))
            {
                printf("pid: %d \t read at the end\n", pid);
                lseek(fr, 0, 0);
                read(fr, (char *)&item_used, sizeof(item_used));
            }
            printf("pid %d:\t consumes item %d\n", pid, item_used);
            fflush(stdout);

            sem_post(empty); /* 唤醒生产者进程 */
            sem_post(mutex);

            if (item_used == ITEM_COUNT)
                goto OK;
        }
    }
}
```

三、实验结果与分析



经测试，该生产者消费者程序成功实现了预定功能，说明信号量相关系统调用实现正确。

四、问题与建议