山东大学 计算机科学与技术 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201818130189 | 姓名：王硕 | | 班级：18级2班 |
| 实验题目：进程同步实验 | | | |
| 实验学时：4学时 | | 实验日期：2020 11 20 | |
| 实验目的：  加深对并发协作进程同步与互斥概念的理解，观察和体验并发进程同步与互斥操作的效果，分析与研究经典进程同步与互斥问题的实际解决⽅案。了解系统中进程同步工具的⽤法，练习并发协作进程的同步与互斥操作的编程与调试技术。 | | | |
| 硬件环境：  双核CPU、4GB内存 | | | |
| 软件环境：  Parallels Desktop、Ubuntu 18.04、macOS Catalina | | | |
| 实验内容与设计：   1. 实验内容   抽烟者问题。假设⼀个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽 烟。抽烟者卷起并抽掉⼀颗烟需要有三种材料：烟草、纸和胶⽔。⼀个抽烟者有烟草，⼀个有纸，另⼀个有胶⽔。系统中还有两个供应者进程，它们⽆限地供应所有三种材料，但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉⼀颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这⼀过程重复进⾏。   1. 实现方法   （1）信号量：总共设置了五个信号量，分别是三个消费者的同步信号量、生产者的同步和互斥信号量；    （2）缓冲区：存放生产者生产的产品，消费者取得产品的途径；    （3）生产者：  遵循先同步，再互斥的原则，通过产生随机数的方式，让共享内存的值在 ‘A’ ‘B’ ‘C’之间变化，每个字符都对应一种生产的情况；然后先唤醒阻塞的生产者，再唤醒阻塞的对应的消费者。    （4）消费者（以消费者1为例）：  当通过生产者将对应的消费者唤醒之后，同样也是遵循先同步，再互斥的原则，但是由于每次生产者会根据随机数而固定唤醒，所以消费者之间的互斥性其实默认就满足了。唤醒之后输出消费者已经取得产品的信息，然后将被阻塞的生产者唤醒，继续生产。 | | | |
| 结论分析与体会：  1、并发进程同步机制是怎样实现和解决同步问题的  进程间的两种关系：同步和互斥。所谓同步就是把异步环境下的一组并发进程，因直接制约而互相发送消息二进行互相合作、互相等待，使得各进程按一定的速度执行的过程；互斥是指不允许两个以上的共享该资源的并发进程同时进入临界区，是进程之间发生的一种间接性作用；  2、信号灯机制是怎样完成进程的互斥和同步的？ 其中信号量的初值和其值的变化的物理意义是什么？  信号量是一种软件资源，是最早出现的用来解决进程同步与互斥问题的机制，包括一个称为信号量的变量及对它进行的两个原语操作，每个信号量至少须记录两个信息：信号量的值和等待该信号量的进程队列。  信号量机制的原理：在操作系统中，信号量sem是一整数。在sem大于等于零时代表可供并发进程使用的资源实体数，但sem小于零时则表示正在等待使用临界区的进程数。显然，用于互斥的信号量sem初值应该大于零。信号量数值仅能由P,V原语操作改变。执行一次P操作意味着请求分配一个单位资源，因此S的值减1；当S<0时，表示已经没有可用资源，请求者必须等待别的进程释放该类资源，它才能运行下去。而执行一个V操作意味着释放一个单位资源，因此S的值加1；若S=0，表示有某些进程正在等待该资源，因此要唤醒一个等待状态的进程，使之运行下去。  3、使⽤多于 4 个的⽣产者和消费者，验证程序的可行性  若使用多个生产者，因为在定义的时候是定义过生产的互斥信号量的，所以生产者在生产的时候是会根据启动的先后顺序，往共享内存中存入数据，所以就算是很多个生产者，必须得满足互斥性，所以程序仍然是能够执行的。执行图如下： | | | |

附录：

product.c

1. #include "ipc.h"
3. **int** main(**int** argc, **char** \*argv[]) {
4. **int** rate;
5. //可在在命令⾏第⼀参数指定⼀个进程睡眠秒数，以调解进程执⾏速度
6. **if** (argv[1] != NULL) rate = atoi(argv[1]);
7. **else** rate = 3; //不指定则默认为 3 秒
9. //共享内存使⽤的变量
10. buff\_key = 101;//缓冲区任给的键值
11. buff\_num = 1;//缓冲区任给的⻓度
12. pput\_key = 102;//⽣产者放产品指针的键值
13. pput\_num = 1; //指针数
14. shm\_flg = IPC\_CREAT | 0644;//共享内存读写权限
16. //获取缓冲区使⽤的共享内存，buff\_ptr 指向缓冲区⾸地址
17. buff\_ptr = (**char** \*)set\_shm(buff\_key, buff\_num, shm\_flg);
19. //获取⽣产者放产品位置指针 pput\_ptr
20. pput\_ptr = (**int** \*)set\_shm(pput\_key, pput\_num, shm\_flg);
22. //信号量使⽤的变量
23. prod\_key = 201;//⽣产者同步信号灯键值
24. pmtx\_key = 202;//⽣产者互斥信号灯键值
25. cons1\_key = 301;//消费者1同步信号灯键值
26. cons2\_key = 302;//消费者2同步信号灯键值
27. cons3\_key = 303;//消费者3同步信号灯键值
28. sem\_flg = IPC\_CREAT | 0644;
30. //⽣产者同步信号灯初值设为缓冲区最⼤可⽤量
31. sem\_val = buff\_num;
32. //获取⽣产者同步信号灯，引⽤标识存 prod\_sem
33. prod\_sem = set\_sem(prod\_key, sem\_val, sem\_flg);
35. //⽣产者互斥信号灯初值为 1
36. sem\_val = 1;
37. //获取⽣产者互斥信号灯，引⽤标识存 pmtx\_sem
38. pmtx\_sem = set\_sem(pmtx\_key, sem\_val, sem\_flg);
40. //消费者初始⽆产品可取，同步信号灯初值设为 0
41. sem\_val = 0;
42. //获取消费者同步信号灯，引⽤标识存 cons\_sem
43. cons1\_sem = set\_sem(cons1\_key, sem\_val, sem\_flg);
44. cons2\_sem = set\_sem(cons2\_key, sem\_val, sem\_flg);
45. cons3\_sem = set\_sem(cons3\_key, sem\_val, sem\_flg);
47. **while**(1)
48. {
49. //如果缓冲区满则生产者阻塞
50. down(prod\_sem);
51. //如果另一生产者正在放产品，本生产者阻塞
52. down(pmtx\_sem);
54. buff\_ptr[\*pput\_ptr] = 'A' + rand()%3;
56. **if**(buff\_ptr[\*pput\_ptr] == 'A'){
57. printf("The producer(%d) gives tobacco and paper\n",getpid());
58. //唤醒阻塞的⽣产者
59. up(pmtx\_sem);
61. up(cons1\_sem);
62. }
63. **if**(buff\_ptr[\*pput\_ptr] == 'B'){
64. printf("The producer(%d) gives tobacco and glue\n",getpid());
65. //唤醒阻塞的⽣产者
66. up(pmtx\_sem);
68. up(cons2\_sem);
69. }
70. **if**(buff\_ptr[\*pput\_ptr] == 'C'){
71. printf("The producer(%d) gives glue and paper\n",getpid());
72. //唤醒阻塞的⽣产者
73. up(pmtx\_sem);
75. up(cons3\_sem);
76. }
77. sleep(rate);
79. }
80. **return** EXIT\_SUCCESS;
81. }

consumer1.c

1. #include "ipc.h"
3. **int** main(**int** argc, **char** \*argv[]) {
4. **int** rate;
5. //可在在命令⾏第⼀参数指定⼀个进程睡眠秒数，以调解进程执⾏速度
6. **if** (argv[1] != NULL) rate = atoi(argv[1]);
7. **else** rate = 3; //不指定为 3 秒
9. //共享内存 使⽤的变量
10. buff\_key = 101; //缓冲区任给的键值
11. buff\_num = 1; //缓冲区任给的⻓度
12. cget\_key = 103; //消费者取产品指针的键值
13. cget\_num = 1; //指针数
14. shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限
16. //获取缓冲区使⽤的共享内存，buff\_ptr 指向缓冲区⾸地址
17. buff\_ptr = (**char** \*)set\_shm(buff\_key, buff\_num, shm\_flg);
19. //获取消费者取产品指针，cget\_ptr 指向索引地址
20. cget\_ptr = (**int** \*)set\_shm(cget\_key, cget\_num, shm\_flg);
22. //信号量使⽤的变量
23. prod\_key = 201; //⽣产者同步信号灯键值
24. pmtx\_key = 202; //⽣产者互斥信号灯键值
25. cons1\_key = 301; //消费者同步信号灯键值
26. sem\_flg = IPC\_CREAT | 0644; //信号灯操作权限
28. //⽣产者同步信号灯初值设为缓冲区最⼤可⽤量
29. sem\_val = buff\_num;
30. //获取⽣产者同步信号灯，引⽤标识存 prod\_sem
31. prod\_sem = set\_sem(prod\_key, sem\_val, sem\_flg);
33. //消费者初始⽆产品可取，同步信号灯初值设为 0
34. sem\_val = 0;
35. //获取消费者1同步信号灯，引⽤标识存 cons1\_sem
36. cons1\_sem = set\_sem(cons1\_key, sem\_val, sem\_flg);
38. //循环执⾏模拟消费者不断取产品
39. **while** (1) {
40. //如果⽆产品，消费者阻塞
41. down(cons1\_sem);
43. //⽤读⼀字符的形式模拟消费者取产品，报告本进程号和获取的字符及读取的位置
44. sleep(rate);
45. printf("%d consumer(have glue):now get tobacco and paper\n", getpid());
47. //唤醒阻塞的⽣产者
48. up(prod\_sem);
49. }
50. **return** EXIT\_SUCCESS;
51. }

consumer2.c

1. #include "ipc.h"
3. **int** main(**int** argc, **char** \*argv[]) {
4. **int** rate;
5. //可在在命令⾏第⼀参数指定⼀个进程睡眠秒数，以调解进程执⾏速度
6. **if** (argv[1] != NULL) rate = atoi(argv[1]);
7. **else** rate = 3; //不指定为 3 秒
9. //共享内存 使⽤的变量
10. buff\_key = 101; //缓冲区任给的键值
11. buff\_num = 1; //缓冲区任给的⻓度
12. cget\_key = 103; //消费者取产品指针的键值
13. cget\_num = 1; //指针数
14. shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限
16. //获取缓冲区使⽤的共享内存，buff\_ptr 指向缓冲区⾸地址
17. buff\_ptr = (**char** \*)set\_shm(buff\_key, buff\_num, shm\_flg);
19. //获取消费者取产品指针，cget\_ptr 指向索引地址
20. cget\_ptr = (**int** \*)set\_shm(cget\_key, cget\_num, shm\_flg);
22. //信号量使⽤的变量
23. prod\_key = 201; //⽣产者同步信号灯键值
24. pmtx\_key = 202; //⽣产者互斥信号灯键值
25. cons2\_key = 302; //消费者同步信号灯键值
26. sem\_flg = IPC\_CREAT | 0644; //信号灯操作权限
28. //⽣产者同步信号灯初值设为缓冲区最⼤可⽤量
29. sem\_val = buff\_num;
30. //获取⽣产者同步信号灯，引⽤标识存 prod\_sem
31. prod\_sem = set\_sem(prod\_key, sem\_val, sem\_flg);
33. //消费者初始⽆产品可取，同步信号灯初值设为 0
34. sem\_val = 0;
35. //获取消费者同步信号灯，引⽤标识存 cons2\_sem
36. cons2\_sem = set\_sem(cons2\_key, sem\_val, sem\_flg);
38. //循环执⾏模拟消费者不断取产品`
39. **while** (1) {
40. //如果⽆产品，消费者阻塞
41. down(cons2\_sem);
43. //⽤读⼀字符的形式模拟消费者取产品，报告本进程号和获取的字符及读取的位置
44. sleep(rate);
45. printf("%d consumer(have paper):now get tobacco and glue\n", getpid());
47. //唤醒阻塞的⽣产者
48. up(prod\_sem);
49. }
50. **return** EXIT\_SUCCESS;
51. }

consumer3.c

1. #include "ipc.h"
3. **int** main(**int** argc, **char** \*argv[]) {
4. **int** rate;
5. //可在在命令⾏第⼀参数指定⼀个进程睡眠秒数，以调解进程执⾏速度
6. **if** (argv[1] != NULL) rate = atoi(argv[1]);
7. **else** rate = 3; //不指定为 3 秒
9. //共享内存 使⽤的变量
10. buff\_key = 101; //缓冲区任给的键值
11. buff\_num = 1; //缓冲区任给的⻓度
12. cget\_key = 103; //消费者取产品指针的键值
13. cget\_num = 1; //指针数
14. shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限
16. //获取缓冲区使⽤的共享内存，buff\_ptr 指向缓冲区⾸地址
17. buff\_ptr = (**char** \*)set\_shm(buff\_key, buff\_num, shm\_flg);
19. //获取消费者取产品指针，cget\_ptr 指向索引地址
20. cget\_ptr = (**int** \*)set\_shm(cget\_key, cget\_num, shm\_flg);
22. //信号量使⽤的变量
23. prod\_key = 201; //⽣产者同步信号灯键值
24. pmtx\_key = 202; //⽣产者互斥信号灯键值
25. cons3\_key = 303; //消费者同步信号灯键值
26. sem\_flg = IPC\_CREAT | 0644; //信号灯操作权限
28. //⽣产者同步信号灯初值设为缓冲区最⼤可⽤量
29. sem\_val = buff\_num;
30. //获取⽣产者同步信号灯，引⽤标识存 prod\_sem
31. prod\_sem = set\_sem(prod\_key, sem\_val, sem\_flg);
33. //消费者初始⽆产品可取，同步信号灯初值设为 0
34. sem\_val = 0;
35. //获取消费者同步信号灯，引⽤标识存 cons3\_sem
36. cons3\_sem = set\_sem(cons3\_key, sem\_val, sem\_flg);
38. //循环执⾏模拟消费者不断取产品
39. **while** (1) {
40. //如果⽆产品，消费者阻塞
41. down(cons3\_sem);
43. //⽤读⼀字符的形式模拟消费者取产品，报告本进程号和获取的字符及读取的位置
44. sleep(rate);
45. printf("%d consumer(have tabacco):now get glue and paper\n", getpid());
47. //唤醒阻塞的⽣产者
48. up(prod\_sem);
49. }
50. **return** EXIT\_SUCCESS;
51. }