现代操作系统第一次作业

陈涵 2019 级软件工程 1 班 201936380086

1. 图 2-2 中给出了三个进程状态。理论上，三个状态之间可以有六种转换，每个状态两个。但图中只给出了四种转换。其余两种转换是否可能发生?

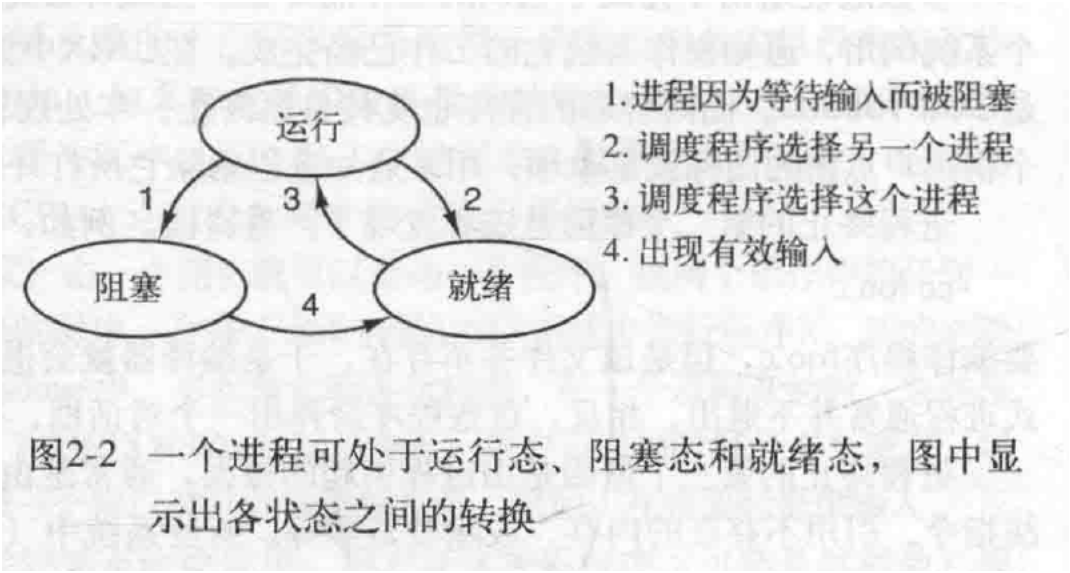

图2-2 一个进程可处于运行态、阻塞态和就绪态，图中显示出各状态之间的转换

答：

其余两种分别是从阻塞状态到运行状态，和从就绪状态到阻塞状态。对于第一种情况当某个进程被 I/O 阻塞，当 I/O 结束且 CPU 空闲时，可以从阻塞状态直接转换到运行状态。对于第二种情况，处在就绪状态未运行的进程，不可能被阻塞，因为只有运行中的进程会被阻塞。

4. 中断或系统调用把控制权转交给操作系统时，为什么通常会用到与被中断进程的栈分离的内核栈?

答：

一般来说用户程序的堆栈空间有大小限制，使用单独栈能防止系统因为超过栈空间而崩溃。同时，如果把数据保留在用户的进程空间中，会导致操作系统的数据泄漏，容易被恶意用户利用。

14. 既然计算机中只有一套寄存器，为什么图 2-12 中的寄存器集合是按每个线程列出而不是按每个进程列出?

每个进程中的内容	每个线程中的内容
地址空间	程序计数器
全局变量	寄存器
打开文件	堆栈
子进程	状态
即将发生的定时器	
信号与信号处理程序	
账户信息	

图2-12 第一列给出了在一个进程中所有线程共享的内容，第二列给出了每个线程自己的内容

答：

当一个进程结束时，其在寄存器中的数据将被转移到内存中保存，对于多线程的进程，每个线程都需要有储存自己寄存器中数据的内存空间。

18. 在用户态实现线程的最大优点是什么？最大缺点是什么?

答：

最大的优点是效率的提升，不需要为了切换进程而陷入内核。最大的缺点是如果一个线程被阻塞，整个进程都会被阻塞。

29. 将生产者一消费者问题扩展成一个多生产者-多消费者的问题，生产(消费)者都写(读)一个共享的缓冲区，每个生产者和消费者都在自己的线程中执行。图 2-28 中使用信号量的解法在这个系统中还可行吗?

```
#define N 100                                   /* 缓冲区中的槽数目 */
typedef int semaphore;                          /* 信号量是一种特殊的整型数据 */
semaphore mutex = 1;                            /* 控制对临界区的访问 */
semaphore empty = N;                            /* 计数缓冲区的空槽数目 */
semaphore full = 0;                             /* 计数缓冲区的满槽数目 */

void producer(void)
{
    int item;

    while (TRUE) {                              /* TRUE是常量1 */
        item = produce_item();                  /* 产生放在缓冲区中的一些数据 */
        down(&empty);                           /* 将空槽数目减1 */
        down(&mutex);                           /* 进入临界区 */
        insert_item(item);                      /* 将新数据项放到缓冲区中 */
        up(&mutex);                             /* 离开临界区 */
        up(&full);                              /* 将满槽的数目加1 */
    }
}

void consumer(void)
{
    int item;

    while (TRUE) {                              /* 无限循环 */
        down(&full);                            /* 将满槽数目减1 */
        down(&mutex);                           /* 进入临界区 */
        item = remove_item();                   /* 从缓冲区中取出数据项 */
        up(&mutex);                             /* 离开临界区 */
        up(&empty);                             /* 将空槽数目加1 */
        consume_item(item);                     /* 处理数据项 */
    }
}
```

图2-28 使用信号量的生产者-消费者问题

答：

可行。由于有控制对临界区访问的信号量，所以在特定的时刻，只有一个生产者或消费者可以使用缓冲区，多对多也不会产生冲突。

34. 如果线程在内核态实现，可以使用内核信号量对同一个进程中的两个线程进行同步吗？如果线程在用户态实现呢？假设其他进程中没有线程需要访问该信号量。请解释你的答案。

答：

对于内核线程，线程可以通过信号量阻塞，然后内核可以运行该进程中的其他线程，所以可以用内核信号量同步。但是对于用户线程，当某个线程在信号量上阻塞时，内核会认为整个进程都被阻塞而不再执行。

39. 考虑以下 C 代码：

```
void main() {
    fork();
    fork();
    exit();
}
```

程序执行时创建了多少子进程？

答：

创建了三个子进程。主进程执行后第一次 fork() 创建子进程 1，子进程 1fork() 创建子进程 2，主进程第二次 fork()创建子进程 3，如下图所示。

45. 有 5 个批处理作业 A~E，它们几乎同时到达一个计算中心。估计它们的运行时间分别为 10、6、2、4 和 8 分钟。其优先级(由外部设定)分别为 3、5、2、1 和 4，其中 5 为最高优先级。对于下列每种调度算法，计算其平均进程周转时间，可忽略进程切换的开销。
(a)轮转法
(b)优先级调度
(c)先来先服务(按照 10、6、2、4、8 次序运行)
(d)最短作业优先
对于(a)，假设系统具有多道程序处理能力，每个作业均公平共享 CPU 时间，对于(b)~(d), 假设任一时刻只有一个作业运行，直到结束。所有的作业都是 CPU 密集型作业。

答：

(a)对于时间片轮转，在第一个 10 分钟内，每个作业获得 1/5 的 CPU 时间。在第 10 分钟结束时，作业 C 执行完毕。在接下来的 8 分钟内，每个作业获得 1/4 的 CPU 时间，接着 D 执行完毕。在接下来 6 分钟内，每个作业获得 1/3 的时间，然后 B 执行完毕。在接下来的 4 分钟内，A，E 各获得 1/2 的时间，E 执行完毕，剩下 2 分钟 A 执行完毕。有周转时间：

A 30 分钟、B 24 分钟、C 10 分钟、D 18 分钟、E 28 分钟
平均时间=(30+24+10+18+28)/5=22 分钟。

(b)根据优先级，完成顺序为 B(5)、E(4)、A(3)、C(2)、D(1)。

作业编号	B	E	A	C	D
优先级	5	4	3	2	1
运行时间	6	8	10	2	4
周转时间	6	14	24	26	30

有平均时间=(6+14+24+26+30)/5=20 分钟。

(c)先来服务作业。

作业编号	A	B	C	D	E
运行时间	10	6	2	4	8
周转时间	10	16	18	22	30

有平均时间=(10+16+18+22+30)/5=19.2 分钟。

(d)最短作业优先。

作业编号	C	D	B	E	A
运行时间	2	4	6	8	10
周转时间	2	6	12	20	30

有平均时间=(2+6+12+20+30)/5=14 分钟。

60. 假设某大学谁备把美国最高法院的信条“平等但隔离其本身就是不平等”(Separate but equal is inherently unequal) 既运用在种族上也运用在性别上，从而结束校园内长期使用的浴室按性别隔离的做法。但是，为了迁就传统习惯，学校颁布法令:当有一个女生在浴室里时，其他女生可以进入，但是男生不行，反之亦然。在每个浴室的门上有一个滑动标记，表示当前处于以下三种可能状态之一:

- 空
- 有女生
- 有男生

用你喜欢的程序设计语言编写下面的过程:
woman_wants_to_enter, man_wants_to_enter, woman_leaves, man_leaves。可以随意使用计数器和同步技术。

答：

Filename: bathroom.c

```
/*filename bathroom.c
 *for Modern Operation System homework1 task60
 *Created by concyclics
 */

#include <stdio.h>

enum bathroomStates {EMPTY,GIRLSin,BOYSin};
enum returnStates {SUCCESS,FAILED};

struct Bathroom
{
        enum bathroomStates States;
        unsigned int caps;
};
struct Bathroom bathroom={.States=EMPTY,.caps=0};

enum returnStates woman_wants_to_enter()
{
        if(bathroom.States==GIRLSin||bathroom.States==EMPTY)
        {
                bathroom.caps++;
                if(bathroom.States==EMPTY)bathroom.States=GIRLSin;
                puts("Woman enter success.");
                return SUCCESS;
        }
        else
        {
```

```
                puts("Woman enter failed!");
                return FAILED;
        }
}

enum returnStates man_wants_to_enter()
{
        if(bathroom.States==BOYSin||bathroom.States==EMPTY)
        {
                bathroom.caps++;
                if(bathroom.States==EMPTY)bathroom.States=BOYSin;
                puts("Man enter success.");
                return SUCCESS;
        }
        else
        {
                puts("Man enter failed!");
                return FAILED;
        }
}

enum returnStates woman_leaves()
{
        if(bathroom.States==GIRLSin&&bathroom.caps>0)
        {
                bathroom.caps--;
                if(bathroom.caps==0)
                {
                        bathroom.States=EMPTY;
                }
                puts("A woman leaves.");
                return SUCCESS;
        }
        else return FAILED;
}

enum returnStates man_leaves()
{
        if(bathroom.States==BOYSin&&bathroom.caps>0)
        {
                bathroom.caps--;
                if(bathroom.caps==0)
                {
                        bathroom.States=EMPTY;
```

```
                }
                puts("A man leaves.");
                return SUCCESS;
        }
        else return FAILED;
}

void showStates()
{
        if(bathroom.caps>0)
        {
                printf("%u ",bathroom.caps);
                if(bathroom.States==GIRLSin)puts("woman in the bathroom.");
                else if(bathroom.States==BOYSin)puts("man in the bathroom.");
                else puts("Error");
        }
        else puts("The bathroom is empty.");
}

int main()
{
        showStates();
        woman_wants_to_enter();
        showStates();
        man_wants_to_enter();
        showStates();
        woman_leaves();
        showStates();
        man_wants_to_enter();
        showStates();
        man_leaves();
        showStates();
}
/*运行结果


*The bathroom is empty.
*Woman enter success.
*1 woman in the bathroom.
*Man enter failed!
*1 woman in the bathroom.
*A woman leaves.
*The bathroom is empty.
*Man enter success.
```

```
*1 man in the bathroom.
*A man leaves.
*The bathroom is empty.
*/
```