日期: /

1. 不可以。因为关中断只能保证CPU不切换进程，而对于多核多CPU，不同进程仍可以在不同CPU上进入临界区。

2. 编译前会附加一个宏定义，使得libc文件在编译时会选择线程安全的实现。

3. 防止wait错过唤醒操作，保证解锁、等待唤醒、上锁是一个原子操作。

1. ABCDE ABDCE ADBCE DABCE ABDEC
ADBEC DABEC ADEBC DAEBC DEABC

2.3. // 初始

```
semaphore pass_mutex = 1;
int count0 = 0;
semaphore mutex0 = 1;
int count1 = 0;
semaphore mutex1 = 1;
// semaphore limit = k;
// 同步
```

日期: /

```
while (true) {
    p (mutex 0) ;
    if ( !count0)  p(pass-mutex) ;
    count 0 ++ ;
    v (mu tex 0);
    // p(limit);
    // 过桥
    // v(limit);
    p (mutex 0);
    count 0 -- ;
    if (!count 0 )  v(pass-mutex) ;
    v(mutex 0);
}
// 同步.
while (true) {
    p (mutex 1) ;
    if ( !count 1 )  p (pass-mutex);
    count 1 ++ ;
    v(mu tex 1);
    // p (limit);
    // 过桥
```

日期: /

```
// V(limit);
P(mutex1);
count1 --;
if (!count1)  V(pass_mutex);
    V(mutex1);
}
```

4. 假设已有管理 plate：

```
procedure 父母：
    while (true) {
        处理水果;
        plate.insert (对应水果);
    }
procedure 子女：
    while (true) {
        水果 = plate.remove();
        子女吃对应水果;
    }
```

5. (1) 可以，

初始为：Available = (1, 1, 2)

日期: /

	Current Available			Claim - Allocation			Current + Allocation			
Process	A	B	C	A	B	C	A	B	C	Possible
P0 ③	2	6	7	2	3	2	4	6	9	T
P1 ②	1	4	5	1	0	3	2	6	7	T
P2 ①	1	1	2	1	0	1	1	4	5	T
P3 ④	4	6	9	1	2	1	6	10	10	T
P4 ⑤	6	10	10	1	1	2	8	10	11	T

其中一个安全执行序列为 P2→P1→P0→P3→P4.

(2) 可以.

初始: Available = (0, 1, 0)

	Current Available			Claim - Allocation			Current + Allocation			
Process	A	B	C	A	B	C	A	B	C	Possible
P0 ③	4	3	5	2	3	2	6	3	7	T
P1 ②	3	1	3	1	0	3	4	3	5	T
P2 ④	6	3	7	1	0	1	6	6	10	T
P3 ⑤	6	6	10	1	2	1	8	10	11	T
P4 ①	0	1	0	0	1	0	3	1	3	T

其中一个安全执行序列为 P4→P1→P0→P2→P3.

日期: /