운영체제 레포트

2016115805 박미선

5.4

(a)

tally 변수가 가질 수 있는 최대 값은 2개의 total()함수가 번갈아 가면서 실행되고 전역변수 이므로 메모리를 공유할 때 나타난다. 그러므로 50+50, 즉 100일 때가 최대 값이 된다.

최소 값은 1개의 total()함수만이 계속 실행해서 그대로 출력하는 것이다. 그러므로 50이 최소 값이 된다.

(b)

병렬 처리되는 프로세스의 개수에 따라서 50\*(실행되는 수)가 최종적으로 마지막에 값이 출력이 된다. 그 이유는 병렬 처리 될 때 전역변수인 tally를 공유하므로 각 프로세스가 50번씩 반복함에 따라 tally가 값을 누적하는 것이다.

5.6

P(0)와 P(1)이 병행 처리되는데 동시에 수행되었을 때 blocked[0]=true, blocked[1]=true가 되고 P(1)는 turn=0이고 id=1이므로 while문에 들어가서 true=1로 만들고 P(0)는 id=0 true=0 이므로 while문에 들어가지 않고 임계영역을 수행한다. 이렇게 cpu가 2개인 경우에는 상호배제가 가능해지지만 cpu 1개인 경우에는 프로세스를 1개씩만 수행 시키므로 오류가 발생한다.

먼저 P(0)가 blocked[0]=true를 수행하고 P(1)이 수행되면 blocked[1]=true, turn=0이고 id=1이므로 while문에 들어가서 true=1로 만든다. 그리고 P(0)이 수행되면 turn=1이고 id=0이므로 while문에 들어가서 true=0로 만든다. 둘 다 true이므로 두 개의 프로세스는 무한 루프를 돌게 되고 교착상태에 빠지게 된다.

5.7

(a)

Number 배열에서 가장 큰 값에 +1을 한 값을 할당한다. 그리고 for문을 통해 number에서 0이 아닌 가장 작은 값이 for문을 빠져나오고 서로 같은 값은 j<I 일 때 for문을 빠져 나온다. 이렇게 되면 가장 작은 값이 나오는데 이렇게 우선순위를 할당하게 된다. 그리고 임계 영역에 들어가서 수행한 후 해당 number을 초기화하고 다시 반복하게 된다.

(b)

단일프로세서에서 2개의 프로세스가 서로 스케쥴링하면서 병행수행하고 있다고 가정했을 때 for문에서 while(choosing[j])를 통해 할당 받지 않은, 즉 true인 choosing을 루프를 돌면서 할당을 받고 false가 될 때까지 기다린다. 이러한 과정을 통해 분야에 따라 겹치는 부분을 확실하게 나누므로 데드락이 발생하지 않는다.

(c)

상호 배제가 되기 위해서는 먼저 6가지 조건을 완성해야한다.

1. 단 하나의 프로세스만이 임계 영역에 진입할 수 있다.
2. 임계 영역이 아닌 곳에서 수행이 멈춘 프로세스는 다른 프로세스의 수행을 간섭해서는 안된다.
3. 교착상태 및 기아가 발생하지 않아야 한다.
4. 임계 영역이 비어있을 때 진입하려고 하는 프로세스는 즉시 임계 영역에 들어갈 수 있다.
5. 임계 영역에 들어간 프로세스는 일정한 시간 내에 임계영역에서 나와야만 한다.
6. 프로세서의 개수나 상대적인 프로세스 수행 속도에 대한 가정이 없어야 한다.

제일 작은 값을 가진 프로세스를 제외하고는 임계 영역에서 나올 때까지 while루프를 계속 돌게 되므로 1번, 2번 조건이 만족한다. 3번 조건은 (b)를 통해 만족한다는 것을 알 수 있다. 그리고 임계 영역이 지나면 해당 배열이 0으로 되므로 그 다음으로 작은 값이 for문 밖으로 나오게 된다. 그러므로 1개씩 들어가게 되므로 4번 조건을 만족하고 기아가 발생하지 않는다. 임계 영역에 들어간 것은 루프를 돌고 있지 않으므로 언젠가 임계 영역을 나오게 되고 5번 조건을 만족한다. 프로세서의 개수와 수행 속도에 상관없이 프로그램이 작동하므로 6번 조건 또한 만족한다.

이렇게 모든 조건을 만족하므로 상호 배제를 보장한다.

5.8

단일프로세서에서 2개의 프로세스가 서로 스케쥴링 하면서 병행수행하고 있다고 가정했을 때 메모리를 공유하므로 i=1인 프로세스가 실행될 때 number[1]=1이 들어가고 for문으로 들어가면 while문이 거짓이 되므로 임계 영역에 들어간다. 그 다음 i=0이 실행되면 number[0]=2가 들어간다. for문에 들어가서 j=0일 때 while문의 조건식이 (1, 0) < (1, 0)으로 거짓이므로 i=0인 프로세스 또한 임계영역에 들어가 상호 배제 조건을 만족할 수 없게 된다.

5.14

(a)

5개의 프로세서, pro1, pro2, pro3, pro4, pro5가 병행수행하고 있다고 가정한다. 이 때 pro1과 pro2는 버퍼에 진입이 가능하다. 이때 pro3이 진입을 할 때에는 active==3이 참이 되므로 must\_wait부분이 true가 됩고 버퍼에 진입한다. 이렇게 active를 변환하고 검사하는 과정은 세마포어를 통해 한번에 1개의 프로세스만이 들어갈 수 있게 된다. 그 다음 pro4, pro5가 시행 될 경우 must\_wait가 true이므로 block 됩니다. 이 처리 또한 세마포어로 인해 1개의 프로세스만이waiting을 변경시킬 수 있게 된다. 그리고 세마 포어로 인해 1개의 프로세스만 코드를 실행시킨다. 만약 active가 모두 빠져 나왔다면 즉, 더 이상 임계 영역에 있는 프로세스가 없다면 기다리고 있는 프로세스을 3개씩 임계 영역에 진입시킨다.

이러한 과정은 한번의 수행과정마다 1개의 프로세스만이 값을 변경시킬 수 있으므로 상호배제의 조건을 만족한다. 따라서 생산자 소비자에서 문제가 되던 둘 중 하나만 한 시점에서 버퍼에 접근 할 수 있도록 하고 일정 버퍼가 지나면 더 이상 접근할 수 없게 하는 조건 또한 성립한다. 따라서 생산자와 소비자문제를 해결할 수 있게 된다.

(b)

만약 버퍼가 다 차고 block을 시켰을 때 새로운 프로세스가 계속 들어온다고 가정한다. semSignal(block)을 실행시킬 때 새로운 프로세스부터 프로세스를 block에서 해제하므로 기존의 프로세스들은 cut in line으로 인해 기아상태에 빠질 수 있게 된다.

(c)

세마포어를 이용해 한 개의 프로세스만이 조건식을 구동하도록 하고, 3개가 임계 영역에 들어가게되면 다른 프로세스들은 block에 빠지게 된다. 그리고 모든 프로세스들이 임계영역에서 나오게 되면 버퍼의 크기만큼 block된 프로세스들을 깨워서 임계영역으로 보낸다. 이렇게 하면 프로세스들이 서로의 값에 영향을 주지 않게 되고 버퍼의 크기만큼 들어가게 된다. 또한 메모리를 한 프프로세스 사용할 때 기다리므로 병행성에서도 적절하게 사용될 수 있다. 하지만 새로운 프로세스가 있을 때 cut in line이 발생하여 기아상태가 발생할 수도 있다.