

Operating Systems

2006011139 박준규

2.3.

더블 버퍼링이란 무엇인가?

컴퓨터 그래픽스에서 효율적으로 깜빡임이나 잔상(artifact) 없이 그래픽스 출력을 하기 위한 테크닉. 싱글 버퍼링 환경에서는 한 개의 버퍼를 사용하는데 그래픽 출력을 한 다음 해당 버퍼를 다시 지우고 새로 그리는 과정을 반복하게 된다. 이 과정에서 깜빡임이거나 그리는 중간 과정이 화면에 나타나게 된다. 더블 버퍼링은 전면(front)과 후면(back)으로 이루어진 두 개의 버퍼를 이용하는데, 전면 버퍼는 화면에 실제로 출력되는 버퍼이며 후면 버퍼는 그리기 작업이 수행되는 버퍼이다. 그리기 작업이 완료된 후면 버퍼는 전면 버퍼와 교체(swap)되는데, 이렇게 되면 그리기가 완료된 모습만 화면에 나타나게 된다.

트리플 버퍼링은 어떻게 수행되는지를 설명하라.

더블 버퍼링의 variant이다. 보통 버퍼링은 모니터의 주사율(refresh rate)와 동기화되어 작동하는데 (이는 버퍼 교체 작업이 빨리 끝나면 모니터의 주사가 끝날때까지 기다린다는 의미) 이것은 자원의 낭비이므로, 두번째 버퍼 교체가 끝나면 세번째 버퍼로 바로 그리도록 한다. 이렇게 하여 더욱 부드러운 애니메이션 효과를 얻을 수 있다.

트리플 버퍼링을 언제 사용하는가?

빠르고 부드러운 애니메이션 효과가 필요할 경우에 사용한다.

4.23.

Dekker의 알고리즘에서 상호배제 문장 끝의 두문장 순서를 바꾸면 어떤 결과가 생기겠는가?

우선 Dekker's algorithm의 의사코드를 보자.

```
f0 := false
f1 := false
turn := 0 // or 1

p0:                                p1:
  f0 := true                        f1 := true
  while f1 {                        while f0 {
    if turn ≠ 0 {                   if turn ≠ 1 {
      f0 := false                   f1 := false
      while turn ≠ 0 {              while turn ≠ 1 {
        }                             }
      f0 := true                     f1 := true
    }                                  }
  }                                    }

// critical section                // critical section
...                                  ...
// remainder section                // remainder section
turn := 1                            turn := 0
f0 := false                          f1 := false
```

각각 프로세스를 p_0 , p_1 으로, 진입 플래그를 f 로 가정하자. 우선 p_0 이 임계구역에 진입하고, f_0 을 true로 설정한다. 이어서 p_1 이 임계구역에 진입하려고 하면 f_0 이 true이기 때문에 p_0 이 임계구역에서 나올 때까지 기다리게 된다. 하지만 마지막 두 문장의 순서를 바꾸면 첫 번째 루프의 조건을 불만족하기 때문에 p_1 도 임계구역에 들어가게 되어 교착 상태에 이르게 된다.

5.37.

감독자 기록자 문제에 대한 다음과 같은 변형들에 대하여 에이다와 모니터 해결안을 각각 제시하라.

가. 한 순간에 오직 하나의 감독자나 오직 하나의 기록자만 활동 중이도록 하라. 이것은 너무 제한적이어서 병행성의 효율을 떨어뜨리지만, 쉽게 구현할 수 있다. 이 방법은 무기한 연기를 방생시킬수 있는가? 한 번에 하나의 프로세스만 진입할 수 있으므로 무기한 연기는 발생하지 않을 것이다.

나. 한 순간에 오직 하나의 기록자나 다수의 감독자가 활동 중이도록 하라. 이것은 모든 감독자가 동시에 활동 중일 수 있도록 하며, 이것은 감독자들 상호간에 간섭하지 않기 때문에 가능하다. 이방법은 대기중인 기록자에게 무기한 연기를 발생시킬 수 있는가?

감독자는 서로 간섭하지 않고 기록자 작업이 끝난 다음에 접근이 가능하기 때문에 다수의 감독자들은 하나의 감독자와 같다고 볼 수 있다. 그러므로 하나의 감독자와 하나의 기록자가 있는 것과 마찬가지로 기때문에 무기한 연기는 발생하지 않을 것이다.

다. 하나의 기록자나 다수의 감독자를 허용하되, 기록자에게 더 높은 우선순위를 부여하라. 이것은 대기중인 기록자의 무기한 연기를 방지할 수 있으나 이제 기록자의 꾸준한 행렬이 감독자의 무기한 연기를 발생시킬 수 있다. 그러나, 어떤 상황에서 무기한 연기가 중대한 문제점이 되지 않겠는가?

기록 작업이 띄엄띄엄 발생하는 시스템의 경우 문제되지 않을 것이다.

라. 하나의 기록자나 다수의 감독자를 허용하며, 기록자에게 더 높은 우선순위를 부여하되, 각각의 기록자가 기록을 끝낸 후 모든 대기중인 감독자를 받아들이도록 하라. 이것은 대기중인 기록자나 감독자의 무기한 연기를 방지할 수 있다. 그러나, 어떤 상황에서 이 잘 조정된 해결안이 응답에 문제를 일으킬 수 있겠는가?

임의 선택 속성에 의해 대기중인 감독자가 먼저 실행될 경우