Chapter 21. Asynchronous Notification IO 모델 연습 확인문제

1. 동기 입출력과 비동기 입출력이 무엇인지, send & recv 함수를 기준으로 설명해보자. 그리고 동기 입출력의 단점은 무엇이고 이것이 비동기 입출력을 통해서 어떻게 해결이 되는지도 함께 설명하자.

Send( ) 함수는 전송이 완료된 시점에 함수의 반환이 이루어지고, recv( ) 함수는 수신이 완료된 시점에 함수의 반환이 이루어진다. 이처럼 이벤트의 발생 시점과 함수의 반환 시점이 동일한 입출력 방식을 동기 입출력이라고 한다. 동기 입출력은 전송 또는 수신이 시작된 시점부터 완료할 때까지 함수의 반환이 이루어지지 않는다. 이는 CPU가 I/O burst에 할애하는 시간을 증가시켜 CPU 효율성이 저하됩니다.

이에 반해, 비동기 입출력은 전송 또는 수신이 시작된 시점에 함수의 반환이 이루어지기 때문에, 완료 이벤트가 발생하는 시점과 함수의 반환 시점이 달라 비동기 입출력이라고 합니다.

대신 함수의 호출과 반환이 동시에 이루어지기 때문에, CPU의 I/O burst을 줄일 수 있고 전송 또는 수신이 진행되는 동안 CPU가 다른 작업을 수행할 수 있도록 하는 것을 가능하게 해줍니다.

1. 모든 경우에 있어서 비동기 입출력이 최선의 선택이 될 수는 없다. 그렇다면 비동기 입출력의 단점은 무엇인가? 그리고 어떠한 경우에 동기 입출력이 좋은 선택이 될 수 있겠는가? 이에 대한 답을 내리기 위해서 비동기 입출력 관련 소스코드를 참고하기 바라며, 쓰레드와 관련해서도 의견을 제시해보자.

비동기적 입출력은 함수 호출과 동시에 반환이 이루어진다. 이러한 특성 때문에 입출력 버퍼로의 데이터 전송과 수신을 보장해주지 않는다. 데이터가 적은 경우라면, 동기 입출력을 사용하더라도 프로그램 수행에 끼치는 영향이 적겠지만, 데이터 입출력이 많은 프로그램이라면 오랜 시간동안 프로그램 수행이 중단될 위험이 있습니다. 따라서 다중 쓰레드를 이용하여 입출력을 담당하는 쓰레드와 프로그램 수행을 하는 쓰레드를 별도로 두어, 프로그램이 중단없이 원활히 수행될 수 있도록 하는 것이 좋다고 생각합니다.

1. Select 방식과 관련된 다음 설명이 맞으면 O, 틀리면 X를 표시해보자.

* Select 방식은 호출된 함수이 반환을 통해서 IO 관련 이벤트의 발생을 알리니, Notification IO 모델이라 할 수 있다. ( O )
* Select 방식은 IO 관련 이벤트의 발생 시점과 호출된 함수의 반환 시점이 일치하기 때문에 비동기 모델이 아니다. ( O )
* WSAEventSelect 함수는 select 방식의 비동기 모델이라 할 수 있다. IO 관련 이벤트이 발생을 비동기의 형태로 알리기 떄문이다. (

1. Select 함수를 이용하는 방식과 WSAEventSelect 함수를 이용하는 방식의 차이점을 소스코드의 관점에서 설명해보자.
2. Chapter 17에서 소개한 epoll은 엣지 트리거 모드와 레벨 트리거 모드로 동작한다. 그렇다면 이 중에서 비동기 입출력이 잘 어울리는 모드는 무엇인가? 그리고 그 이유는 또 무엇인가? 이와 관련해서 포괄적인 답을 해보자.
3. 리눅스의 epoll 역시 비동기 입출력 모델이라 할 수 있다. 그렇다면 이를 비동기 입출력 모델이라 할 수 있는 이유에 대해서 설명해 보자.
4. WSAWaitForMultipleEvents 함수가 관찰할 수 있는 최대 핸들의 수는 어떻게 확인이 가능한가? 이의 확인을 위한 코드를 작성해서, 이 값을 실제로 확인해보자.
5. 비동기 Notification IO 모델에서 Event 오브젝트가 manual-reset 모드이어야 하는 이유를 설명해보자.
6. 이번 Chapter에서 설명한 비동기 Notification IO 모델을 바탕으로 채팅 서버를 구현해보자. 이 채팅 서버는 Chapter 20에서 소개한 채팅 클라이언트인 예제 chat\_clnt\_win.c와 함께 동작이 가능해야 한다.