60200307 우연정

목차

1. Inter-Process Communication

프로세스는 CPU가 프로그램을 실행할 때 생기며, 메모리에 올라와 있을 때만 CPU가 그 내용을 읽고 실행할 수 있다. 메모리에 올라가 있다는 것은 cpu가 읽을 수 있는 상태라는 것이다.

프로그램을 프로세스라는 단위로 분리해 놓았다. 왜 프로세스를 분리해 놓았을까? 분리는 독립적인 context(문맥)를 갖는다는 뜻이다. 문맥은 내가 어떤 정보를 공유했을 때 그 정보가 공유될 수 있는 영역이다. 영역이 다르면 전달하는 무언가(미디어)가 있어야 한다. 이를 통신이라고 한다. 데이터 통신은 떨어진 두 개의 context가 무언가를 주고받는 것을 의미한다. context는 내부주소가 통하지 않는다. 하나의 프로그램에서 한 객체가 다른 객체에게 메시지를 보낸다는 것은 그 메시지의 내부 주소(메모리)를 서로 알고 있는 것이다. 함수호출을 한다는 것은 메모리 주소로 cpu의 실행순서가 가는 것이다. 이는 주소를 서로 알고 있기 때문에 가능하다.

context가 다르다는 것은 여러 문제를 포함하고 있다. 주소 문제, 문법적 문제, 언어 문제 등 여러 가지 문제가 발생한다. 이를 해결하기 위해서는 두 개의 네트워크 안에 표준화된 주소체계가 필요하다.

인터넷은 표준화를 포함하고 있다. 인터넷은 “아르파네트(ARPANET)에서 시작된 세계 최대 규모의 컴퓨터 통신망이다.”[두산백과] “참고로 복수의 통신망을 집합시킨 광역 통신망을 뜻하는 일반명사를 ‘인터네트워크(internetwork)’라고 하는데, 미 국방성의 아르파네트(ARPANET)는은 이러한 인터네트워크를 본격적으로 구축한 최초의 사례였다. 아르파네트(ARPANET)는 당초에 연구용으로만 쓰였으나 시간이 흐르고 참여 기관이 늘어나면서 다양한 목적으로 이를 쓰고자 하는 요구가 많아졌다. 또한 컴퓨터의 종류가 다양해지면서 프로토콜(protocol, 컴퓨터끼리의 공통된 통신 규약)을 재정비할 필요성이 부각되었다.” [네이버 지식백과] 인터넷 [Internet] - 전세계를 하나로 연결하는 정보의 바다 (용어로 보는 IT, 김영우, IT 동아)

클라이언트와 서버로 객체들을 컴퓨터로 나눠놓았는데, 프로세스가 나누어지면 하드웨어가 나누어진 것의 여부와 큰 차이가 없다. 하나의 하드웨어에 다른 프로세스가 한꺼번에 있어도 두 개의 하드웨어에 다른 프로세스로 있던 것과 다른 점이 없다. 즉 프로세스가 다르다면 컨텍스트가 다르기 때문에 다른 컴퓨터에 있는 것과 같다고 할 수 있다.

서버는 서비스를 제공할 준비를 하고 있어야 하고 클라이언트는 서버의 주소를 알고 있어야 한다. 프로세스가 다를 때, 클라이언트는 서버의 IP 주소를 알면 하드웨어(컴퓨터)까지는 연결이 된다. 서버를 알려면 서버의 주소를 알아야 한다.

1-1. Independent Context

Inter-Process Communication은 각 프로세스가 독립적인 Context를 가진다는 전제 하에 이루어진다. 즉, 서로 다른 프로세스가 독립적으로 정보를 처리하고 저장한다는 것을 의미한다.

1-2. Remote Addressing

서로 다른 프로세스 또는 컴퓨터 간의 통신을 위해서는 Remote Addressing(원격 주소 지정)이 가능해야 한다.

1-2-1. IP Address

클라이언트가 서버에 접속하기 위해서는 그 서버의 IP 주소를 알아야 한다.

1-2-2. Port Number

서버를 알려면 IP 주소 뿐만 아니라 포트 번호까지 알아야 한다. 서버를 만들면 서버가 올라가면서 자신의 Port Number를 지정한다. 이 포트 번호를 모르면 서버에 접속하지 못한다.

1-3. Protocol

통신을 하기 위해서는 통신 규약이 필요하다. 프로토콜은 데이터를 주고받는 문법이다. 데이터의 전송 형식이나 순서를 정의한다. 문법은 형식적으로 만드는 것이 가장 좋다. 클라이언트와 서버 사이에서 read/write를 할 때 여러 가지 프로토콜이 있다. 프로토콜의 종류로는 이메일, ftp, http, utf-8, pop3 등이 있다.

2. Socket Programming

소켓 프로그래밍은 네트워크 상에서 두 프로세스 간의 통신을 가능하게 하는 방법이다. 처음에 클라이언트가 서버에게 아이피 주소와 포트 번호로 연결을 하면 connect를 한다. 그리고 데이터를 전송한다는 의미인 write를 한다. 그리고 서버가 답장을 하면 읽는다(read). 그리고 끝나면 close를 한다. 통신을 종료하고 소켓을 ekesme는 의미이다. 그리고나서 다시 포트를 재사용할 수 있게 된다. connect를 할 때 서버가 반응을 안 하면 될 때까지 기다린다(wait). 이는 순차적으로 하는 방법이고, 또 다른 방법은 연결이 되는 순간 담당자를 하나 만드는 방벙 있다.

클라이언트와 서버의 동작은 이렇다.

서버	클라이언트
소켓 생성	소켓 생성
binding - IP와 포트 번호를 bind	connect - IP와 포트번호로 서버에 연결 시도

listening - 연결 요청을 기다림	
accept - 클라이언트의 연결 요청 수락	서버와 데이터 통신 시작
통신 종료, 소켓 닫기	통신 종료, 소켓 닫기