1. make (만든다 ?)

1.1 make 유틸리티

영어 사전에서 make란 뜻은 누구나 알듯이 '만들다'라는 뜻의 동사이다. 그럼 make유틸리티는 왜 이름이 make인지 알 필요가 있을 것 같다. man으로 찾아보면 make에 대해 다음과 같이 설명하고 있다.

make - GNU make utility to maintain groups of programs

> The purpose of the *make* utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

우리말로 하면 make는 프로그램 그룹을 유지하는데 필요한 유틸리티이다. make유틸리티의 목적은 프로그램 그룹 중에서 어느 부분이 새롭게 컴파일되어야 하는지를 자동적으로 판단해서 필요한 커맨드(gcc따위)를 이용해서 그들을 재컴파일 시킨다고 되어 있다.

make는 일련의 프로그램 개발에만 쓰이지 않고, 컴파일러처럼 일종의 명령어 방식으로 처리되는 모든 곳에서 쓰일 수가 있다. 가령 LaTeX와 같은 경우도 .tex 파일에서 .dvi 파일을 만들고 다시 .ps 파일로 만드는 과정을 make를 사용해서 간단하게 만들 수가 있다.

쉽게 말하면 다음과 같은 경우에 make를 쓰면 유리합니다.

1. *입력 파일이 바뀌면 자동적으로 결과 파일이 바뀌기를 원할 때, 기왕이면 좀 지능적으로 일이 수행되기를 바랄 때* 말입니다.
2. 위의 LaTeX 파일처럼 자동적으로 프로그램이 수행이 되기를 바랄 때... (배치(batch)의 개념이죠)

=> make는 위의 두 가지 개념을 모두 포함하고 있다고 봅니다. 보통 리눅스 프로그램에서는 make all을 입력하면 자세한 내막은 모르지만 자기가 알아서 모든 일을 다하죠... 그 다음으로 make install만 입력하면 되구요... 히...

GNU make는 보통 `GNUmakefile`, `Makefile`, `makefile` 중에서 하나가 있으면 그 파일을 읽게 된다. 하지만 일반적으로 *Makefile*을 추천하게 되는데, 그 이유는 우선 GNUmakefile은 기존의 make에서 인식을 못한다는 단점이 있고, makefile은 보통 소스 파일에 묻혀서 잘 안보이게 되기 때문이다.

`Makefile`은 make가 이해할 수 있도록 일종의 쉘 스크립트 언어같이 되어 있다(makefile database라 하기도 한다). 이 파일에는 결과 파일을 생성시키기 위한 파일들간의 관계, 명령어 등을 기술하고 있는데 이 강좌의 주된 목적이 바로 `Makefile`의 작성에 있다.

1.2 make의 필요성

우선은 make의 사용을 프로그램 개발과 유지 쪽으로 국한시키기로 한다. 보통 라인 수가 많아지면 여러 개의 파일로 나누어 (모듈로 나누어) 개발을 하게 된다. 이들은 알게 모르게 서로 관계를 가지고 있는데, 어느 하나를 필요에 의해 바꾸게 되었을 때 그 파일에 있는 함수를 이용하는 다른 파일도 새롭게 컴파일되어야 한다.

하지만 파일 수가 많은 경우 이를 일일이 컴파일을 하게 될 때, 그 불편함과 함께 컴파일하지 않아도 될 것도 컴파일을 하게 될 수도 있고, 컴파일해야 할 것도 미처 못하게 되는 경우가 있다(링크 에러의 원인이 되기도 하는데 에러의 원인을 제대로 찾기가 힘이 든다).

앞에서도 얘기했듯이 이런 상황에서 지능적으로 관계 있는 것만 새롭게 갱신을 할 필요가 있을 때 make파일은 빛을 발하게 된다.