Lab 1 report

20170628 조우석

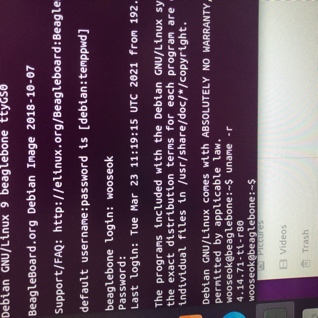
1. Purpose

이 과제의 목적은 beaglebone에 debian을 설치하고, ubuntu 환경에서 beaglebone 을 위한 cross compile 환경을 조성하고, nfs를 통해 ssh로 매번 파일을 전송하는 번거로움을 해결한 공유 폴더 시스템을 활용해 보는 것이었다. 그리고 컴파일 오류가 있는 taylor\_ce.c 라는 파일을 문법부터 안의 타이포 오류까지 한단계씩 고쳐보는 과정을 gdb라는 디버거를 활용해 마지막에는 올바른 taylor sin 값이 나오도록 하는 과제였다.

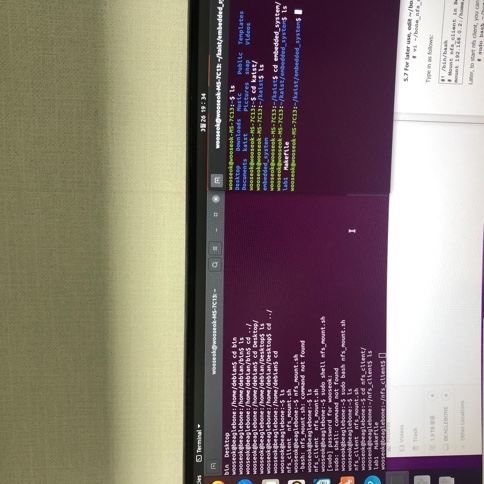
1. Experiment sequence

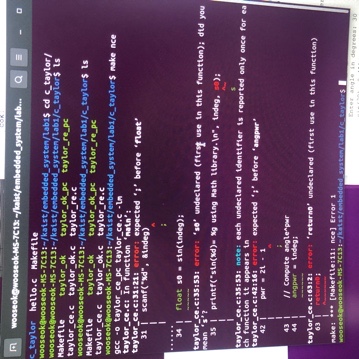
우선 나는 외장하드의 파티션을 나누어 우분투를 설치하였고, bios를 이용해 외장하드를 꽂으면 우분투로 부팅이 되도록 만들어 주어 데스크탑을 우분투 환경으로 부팅하였다.

다음으로 Debian 파일을 받아 pdf와 klms의 영상을 참고하여 beaglebone microSD에 debian을 설치해 주었다. 그리고 sudo minicom -a 명령어를 통해 설정을 아래 그림처럼 바꾸어 주었다.

그리고 minicom 으로 로그인해 wooseok이라는 super user를 추가하는 과정을 거치면 로그인이 가능해 지게 된다.

다음으로 이제 원래의 ubuntu 데스크탑에서 gcc와 gcc-arm-linux-gnueabihf(크로스컴파일러)를 설치해준다.

이제 매번 scp command를 이용해 파일을 전송하기 매우 번거로우므로 nfs를 통해 공유 폴더를 만든다. Pdf의 지시대로 쭉 따라하면 다음 아래 그림처럼 공유 폴더가 만들어지고 매번 명령어를 쳐서 mount 시키기 힘드므로 shell script 또한 nfs\_mount.sh로 만들어주었다. 실제로 이제 pc에서 자료를 만들면 beaglebone안의 폴더에도 똑같은 파일이 생기는 것을 확인 할 수 있다.

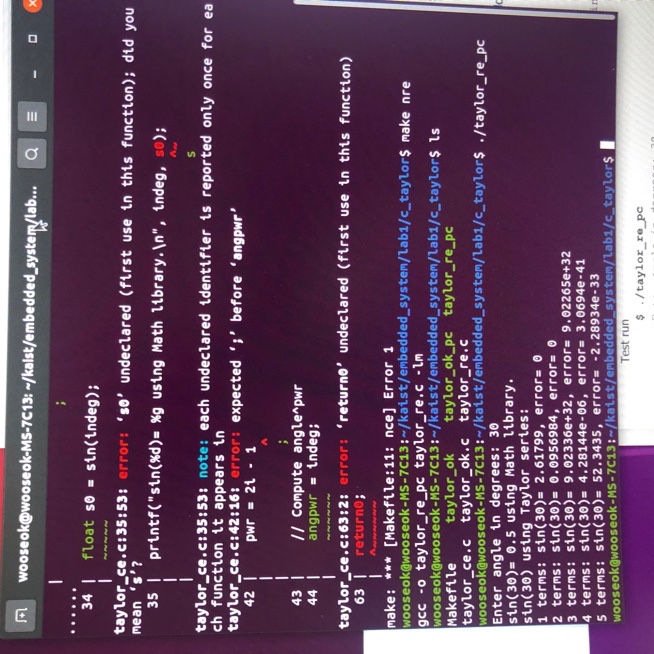
이제 taylor 함수를 위한 working directory를 만들고 오류를 수정하는 과정을 거친다.

처음 make file을 만들고 make nce 명령어를 하면 gcc 컴파일러로 taylor\_ce.c를 컴파일 하게 되는데 많은 오류들이 나와서 컴파일이 되지 않는다.

이제 이 taylor\_ce.c를 taylor\_re.c로 복사를 한다음 수정을 해본다. 나는 처음에는 문법적인 오류만 고쳤다. 그리고 나서 taylor\_re.c를 gcc 컴파일러로 컴파일 해보면 에러가 없이 컴파일 되는 것을 확인 할 수 있었다. 하지만 이 떄 sin 30과 같은 값이 올바르지 않게 나오는 것을 확인 할 수 있었다.

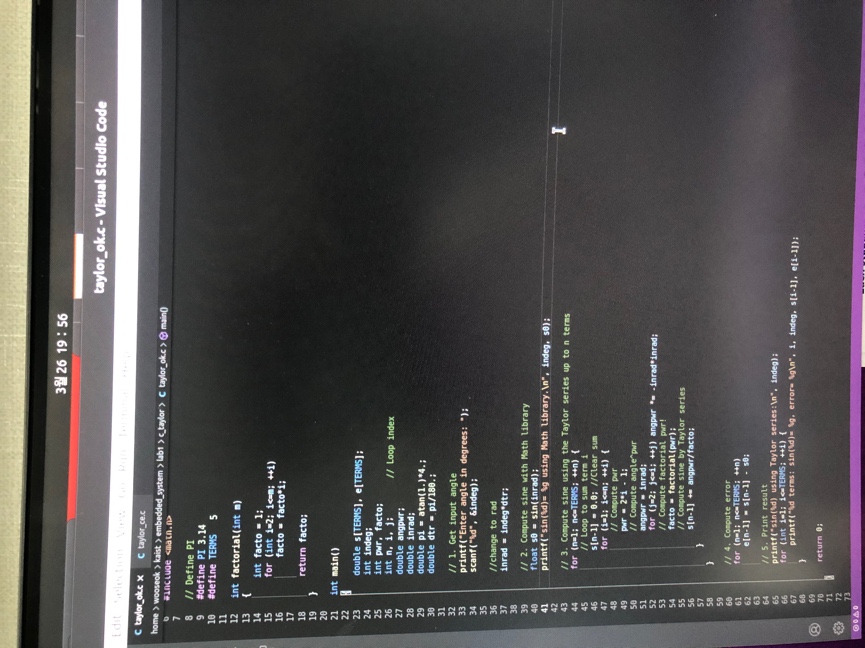
이는 rad값이 아닌 30이라는 숫자를 그대로 사용하고 있었기 때문이었고 이 오류와 pi=3.14로 근사해서 발생하는 오류를 고쳐준 결과 다음과 같은 결과가 나오게 되었다.

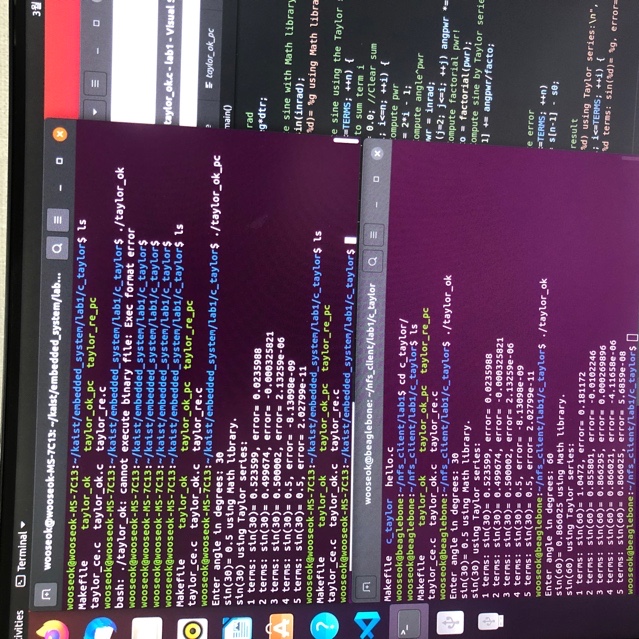
하지만 여전히 taylor 급수와 관련한 항들은 오류가 많이 발생하는 것을 확인할 수 있었다. 우리는 0.5 와 근사한 값을 얻어야 하지만 나의 경우 1 Term의 결과값을 2.617로 계산하는 것을 확인 할 수 있었다.

아직 여전히 오류가 있다는 것이고 이것을 gdb 디버거로 break point를 정해서 디버깅을 해 수정해 보았다. 아래는 gdb를 실행시킨 사진을 첨부하였다.



이제 모든 수정이 끝이 났다. 수정한 부분들중에 핵심적인 부분은 우선 pdf처럼 오류를 좀더 섬세하게 구하기 위해 float s[TERMS], e[TERMS]; 를 double로 바꾸어 주었다. 그리고 pi의 값을 atan(1.)\*4. 로 해주어 정확한 pi 값을 사용하였고, degree로 나온 각을 rad으로 바꾸어서 계산할수 있도록 dtr이라는 double형의 변수를 추가하였고 이와 더불어 indeg를 rad으로 바꾼 inrad 변수를 만들어 rad을 사용해야 하는 경우들에 inrad 변수들을 넣어주었다. 다음으로 sum 을 clear할수 있도록 for loop 가 돌때 값을 0으로 바꾸어 줄 수 있도록 해주었고, pwr를 계산할때 c언어에서 곱하기 기호를 생략할 수 없으므로 2\*i-1로 바꾸어 주었다. 다음으로 매번 더할때 -부호가 번갈아 가면서 등장해야 하기 때문에 angpwr에 곱해지는 값에 – 부호를 붙여 주었다. 그리고 n값을 사용해야 하지만 i변수를 잘못 사용하고 있는 경우들을 고쳐주었다. 그래서 최종으로 만들어진 코드는 다음과 같다.

1. Experimental results

이제 make nok와 make cok 를 하여 gcc와 cross compiler로 컴파일을 해준다. 그리고 gcc로 컴파일 된 파일을 ubuntu에서, crosscompiler 로 컴파일 된 파일을 beaglebone에서 실행시켜 정상작동이 되는지 확인을 해본다.

이러면 왼쪽의 그림과 같이 정상적으로 둘다 작동이 되는 것을 확인 할 수 있었다.

1. Discussion

이번 과제는 기존에 있던 코드의 오류를 수정하는 것이라 엄청 어렵지는 않았다. 하지만 처음 셋팅을 하는데 시간이 오래 걸렸고, 코드를 수정할 때 이것이 내가 처음부터 짠 코드가 아니여서 n과 i와 같이 미세한 오류를 찾기는 어려웠지만 gdb를 이용하여 찾아낼 수 있었다. 그리고 cross compile을 해보는 경험을 하게 되었다. 또한 nfs사용해 보는 경험을 할 수 있었다. 또한 linux terminal에서 sudo user를 추가하는등 linux 관련 지식도 더 많이 배운 것 같다.

또한 gcc로 Build 한파일은 beagle bone환경에서 실행되지 않았고, crosscompiler를 통해 build 한 파일은 우분투에서 실행되지 않는 것을 확인할 수 있었다. ubuntu라는 하나의 시스템에서 만들어진 파일이 다른 두 운영체제에서 돌아갈 수 있도록 각각 compile을 해주었다. Beaglebone은 ubuntu와 다른 환경인 Debian 운영체제를 가지므로 ubuntu환경에서 작동하도록 build 된 파일은 실행될 수 없다. 마찬가지로 반대로 beaglebone에서 작동하로독 build된 파일은 우분투에서 돌아갈 수 없기에 이런 현상이 발생하는 것이다.

Beaglebone과 같이 임베디드 시스템들은 리소스가 훨신 적고, 내부에서 컴파일 툴이 돌아갈 여유가 크지 않기에 이렇게 crosscompile을 하는 것으로 생각된다.

1. References
2. Pdf file from class material