

툴체인

본 문서는 [Chap3. Tool Chain & Cross Compile 실습](#)을 원활하게 진행하기 위한 가이드입니다.

☑ 준비물

- 호스트 머신: VMWare
- 타겟 머신: Raspberry Pi

☑ 실습 1) 호스트 머신에 타겟 머신에 필요한 툴체인 다운로드

[1] vim 설치

```
>>> sudo apt-get install vim
```

[2] Arm-linux-gcc 설치

```
>>> sudo apt install gcc-aarch64-linux-gnu
```

[3] Make 설치

```
>>> sudo apt install make
```

- ☑ 실습 2) 호스트 머신에 설치한 툴체인들(vim, Arm-linux-gcc, Make)을 테스트하기 위해 아래 소스코드 생성 (/home/pi/emss/ch03 디렉터리 생성 후, 아래 파일 생성)

[1] main.c 생성

>>> *sudo vim main.c*

```
#include <stdio.h>
extern void sub1();
extern void sub2();

int main()
{
    printf("Hello Main!!\n");
    sub1();
    sub2();
    return 0;
}
```

[2] sub1.c 생성

>>> *sudo vim sub1.c*

```
#include <stdio.h>
void sub1(){
    printf("Hello Sub1!!\n");
}
```

[3] sub2.c 생성

>>> *sudo vim sub2.c*

```
#include <stdio.h>
void sub2(){
    printf("Hello Sub2!!\n");
}
```

[4] Makefile 생성

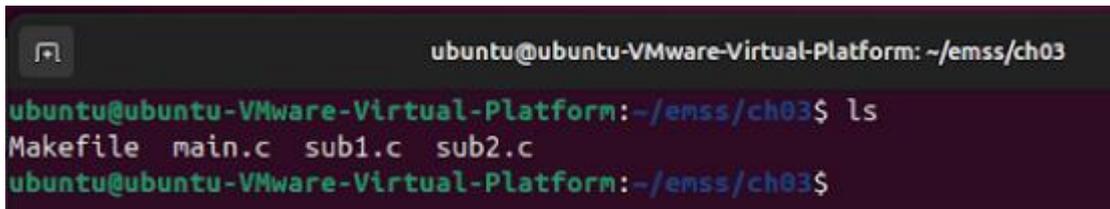
>>> *sudo vim Makefile*

```
CC = aarch64-linux-gnu-gcc
CFLAGS =
OBS = main.o sub1.o sub2.o
TARGET = hello_exe
```

```
$(TARGET) : $(OBJS)
    $(CC) -o $@ $(OBJS)
```

```
main.o : main.c
sub1.o : sub1.c
sub2.o : sub2.c
```

- [5] 최종적으로 생성된 4개의 파일(main.c, sub1.c, sub2.c, Makefile)이 ch03 디렉터리에 존재하는지 확인



```
ubuntu@ubuntu-VMware-Virtual-Platform: ~/emss/ch03
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ ls
Makefile main.c sub1.c sub2.c
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$
```

☑ 실습 3) 호스트 머신에서 Make를 사용하여 실행 파일 생성 후, 의존성 확인

[1] Make를 이용한 실행파일 빌드

>>> `sudo make`

```
ubuntu@ubuntu-VMware-Virtual-Platform: ~/emss/ch03
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ sudo make
aarch64-linux-gnu-gcc -c -o main.o main.c
aarch64-linux-gnu-gcc -c -o sub1.o sub1.c
aarch64-linux-gnu-gcc -c -o sub2.o sub2.c
aarch64-linux-gnu-gcc -o hello_exe main.o sub1.o sub2.o
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ ls
Makefile hello_exe main.c main.o sub1.c sub1.o sub2.c sub2.o
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$
```

➤ Makefile 내용에 따라 실행파일(hello_exe)와 목적파일들(main.o, sub1.o, sub2.o)이 생성됨을 확인

[2] 생성된 실행파일의 바이너리 형식 확인

>>> `file hello_exe`

```
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ file hello_exe
hello_exe: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically link
ed, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=2153abeea5e58a8682914023450e711
445b791a5, for GNU/Linux 3.7.0, not stripped
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$
```

➤ 해당 파일은 ARM aarch64 계열에서만 실행할 수 있는 바이너리 형태의 실행 파일로 호스트 머신에서는 실행할 수 없음

➤ 하지만, 타겟 머신은 ARM 코어이므로, 실행 가능함

[3] 호스트 환경에서의 실행 불가 검증

>>> `./hello_exe`

```
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ ./hello_exe
bash: ./hello_exe: cannot execute binary file: Exec format error
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$
```

➤ `./hello_exe` 명령어를 통해 호스트 머신에서 실행파일을 실행하면 파일 포맷 에러가 발생함

☑ 실습 4) 호스트 머신에 생성된 실행파일을 타겟 머신으로 전송

[1] 타겟 머신의 원격 제어 및 업로드를 위해 호스트 머신에 SSH 설치

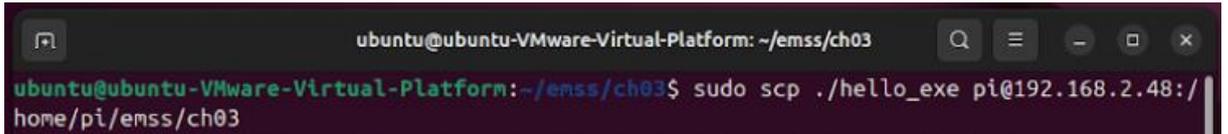
```
>>> sudo apt install ssh
```

[2] 타겟 머신을 켜서 와이파이(iptime)에 연결 후, *ifconfig*를 통해 타겟 머신의 ip 주소 확인

[3] 타겟 머신 /home/pi 위치에 emss/ch03 디렉터리 생성

[4] 호스트 머신에서 타겟 머신으로 실행파일 전송

```
>>> sudo scp ./hello_exe pi@xxx.xxx.xxx.xxx:/home/pi/emss/ch03
```



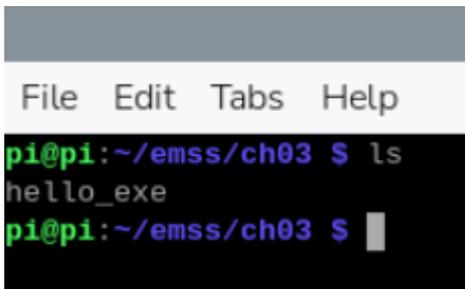
```
ubuntu@ubuntu-VMware-Virtual-Platform: ~/emss/ch03
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ sudo scp ./hello_exe pi@192.168.2.48:/home/pi/emss/ch03
```

➤ 형식은 scp [보낼파일] [타겟이름] @ [타겟ip주소]:[타겟에서 파일 저장 경로]

➤ xxx로 표현되어 있는 것은 [2]에서 확인한 타겟 머신의 ip 주소입니다.

➤ 패스워드를 입력하라고 하면, 타겟 머신의 패스워드를 입력하면 됩니다.

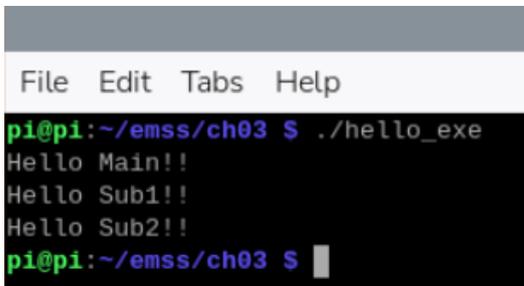
[5] 전송이 성공적으로 완료되면, 타겟 머신 /home/pi/emss/ch03 경로에서 *ls* 명령어 실행하여 hello_exe 파일이 정상적으로 존재하는지 확인



```
File Edit Tabs Help
pi@pi:~/emss/ch03 $ ls
hello_exe
pi@pi:~/emss/ch03 $
```

[6] 타겟 머신 환경에서 호스트 머신에서 받아온 hello_exe 실행파일 실행

```
>>> ./hello_exe
```



```
File Edit Tabs Help
pi@pi:~/emss/ch03 $ ./hello_exe
Hello Main!!
Hello Sub1!!
Hello Sub2!!
pi@pi:~/emss/ch03 $
```

➤ 위와 같이 정상적으로 실행되지 않는다면, 파일의 실행 권한과 소유자 변경 진행 후 다시 실행

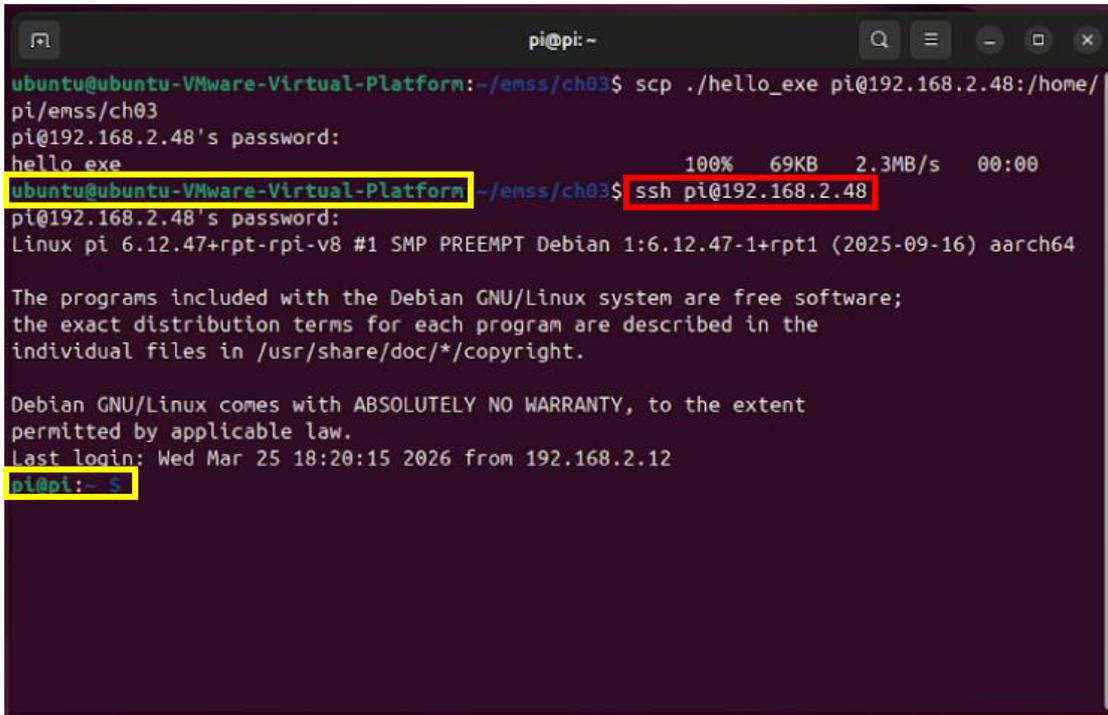
```
>>> sudo chown root:root hello_exe
```

```
>>> sudo chmod 777 hello_exe
```

☑ 실습 5) 호스트 머신에서 타겟 머신의 ip 주소를 이용해서 원격 접속

[1] ssh를 활용해 호스트 머신에서 타겟 머신으로 원격 접속

>>> ssh pi@xxx.xxx.xxx.xxx



```
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ scp ./hello_exe pi@192.168.2.48:/home/pi/emss/ch03
pi@192.168.2.48's password:
hello_exe                               100% 69KB  2.3MB/s  00:00
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ ssh pi@192.168.2.48
pi@192.168.2.48's password:
Linux pi 6.12.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1 (2025-09-16) aarch64

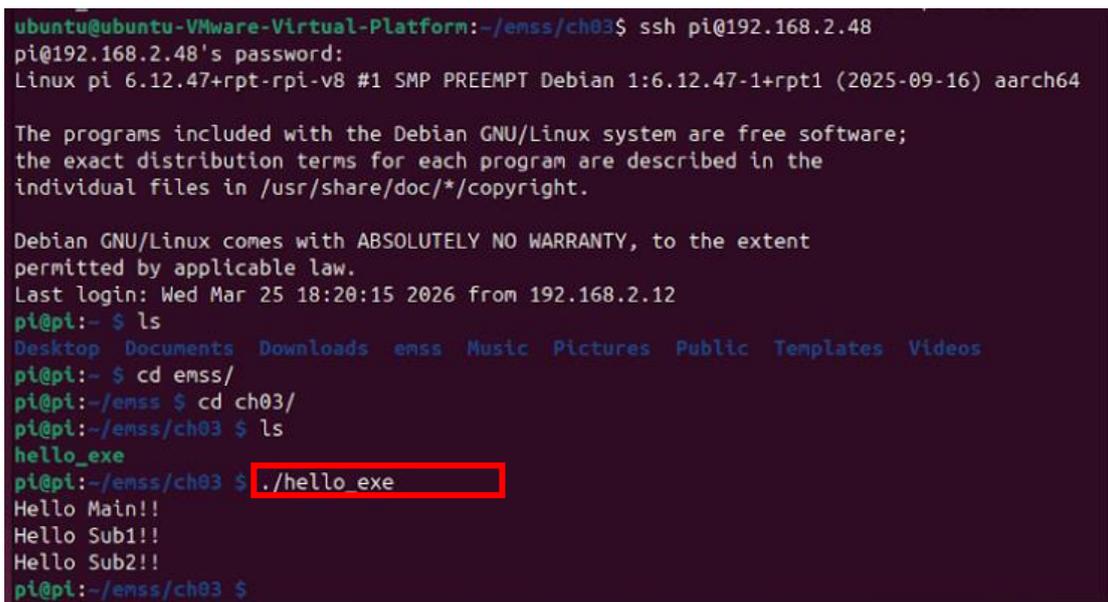
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 25 18:20:15 2026 from 192.168.2.12
pi@pi:~$
```

- 실습 4에서 확인한 타겟 머신 ip 주소 활용할 것
- ssh를 활용하여 타겟 머신으로 원격 접속하게 되면, 터미널 창의 호스트 명이 타겟 머신의 호스트 명으로 변경되는 것을 확인할 수 있음 (ubuntu → pi)

[2] 원격 접속한 상태에서 hello_exe 파일 실행 (hello_exe 파일이 있는 경로로 이동할 것)

>>> ./hello_exe



```
ubuntu@ubuntu-VMware-Virtual-Platform:~/emss/ch03$ ssh pi@192.168.2.48
pi@192.168.2.48's password:
Linux pi 6.12.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1 (2025-09-16) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 25 18:20:15 2026 from 192.168.2.12
pi@pi:~$ ls
Desktop Documents Downloads emss Music Pictures Public Templates Videos
pi@pi:~$ cd emss/
pi@pi:~/emss$ cd ch03/
pi@pi:~/emss/ch03$ ls
hello_exe
pi@pi:~/emss/ch03$ ./hello_exe
Hello Main!!
Hello Sub1!!
Hello Sub2!!
pi@pi:~/emss/ch03$
```