**COMPILATION ON DIFFERENT TARGETS**

Document outlining the compilation steps on different targets (x86 Ubuntu, Raspberry Pi, and BeagleBone), follow these general steps:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Build steps for X86**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Run the **make** command to compile your code:

$ make

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Build steps for BBB**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **Step 1: Environment Variables Setup**

1. **Set the ARCH and CROSS\_COMPILE** **environment Variables**

$ export ARCH=arm

$ export CROSS\_COMPILE=arm-linux-gnueabihf-

1. **Set the PATH to the Cross-Toolchain:**

$ export PATH=${HOME}/ela\_lab\_exercises/bbb\_build/toolchain/gcc-linaro-7.5.0-2019.12-x86\_64\_arm-linux-gnueabihf/bin/:$PATH

## **Step 2: Compilation**

$ make

## **Step 3:** **Transfer Binary file to target**

$ scp <binary\_file> <username>@<ip\_address>:<destination\_directory>

**Example**: $scp Thread\_IDs [root@10.10.3.233:/User\_Programs](mailto:root@10.10.3.233:/User_Programs)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Build steps for Raspberry Pi 4B**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **Step 1: Environment Variables Setup**

1. **Set the ARCH and CROSS\_COMPILE environment Variables**

$ export ARCH=arm64

$ export CROSS\_COMPILE=aarch64-linux-gnu-

1. **Set the PATH to the Cross-Toolchain**

$ export PATH=${HOME}/ela\_lab\_exercises\_rpi/rpi\_build/toolchain/gcc-linaro-7.5.0-2019.12-x86\_64\_aarch64-linux-gnu/bin/:$PATH

## **Step 2: Compilation**

$ make

## **Step 3: Transfer Binary file to target**

$ scp <binary\_file> <username>@<ip\_address>:<destination\_directory>

**Example**: $scp Thread\_IDs [root@10.10.1.27:/User\_Programs](mailto:root@10.10.1.27:/User_Programs)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Running on Platform (x86, Raspberry Pi, BBB)**

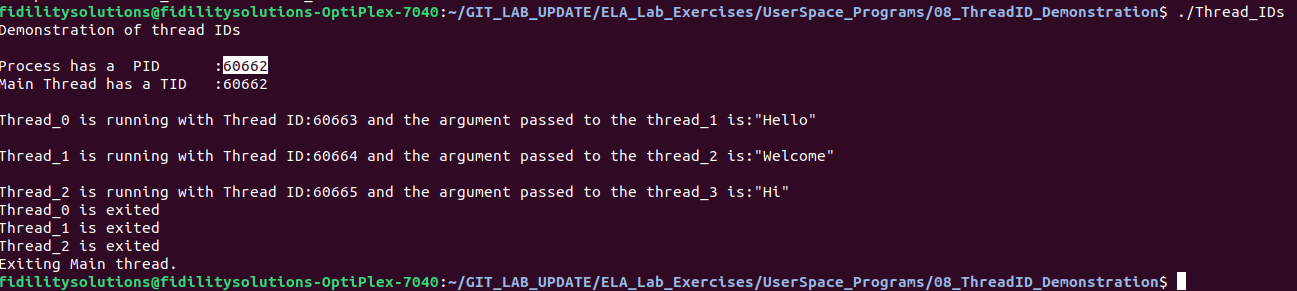
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Once you're in the correct directory, execute the generated executable file using the **./filename** command. Replace **filename** with the name of your executable file.

$ ./filename

**Ex: $ ./Thread\_IDs**

* The overall output will be as given below:



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

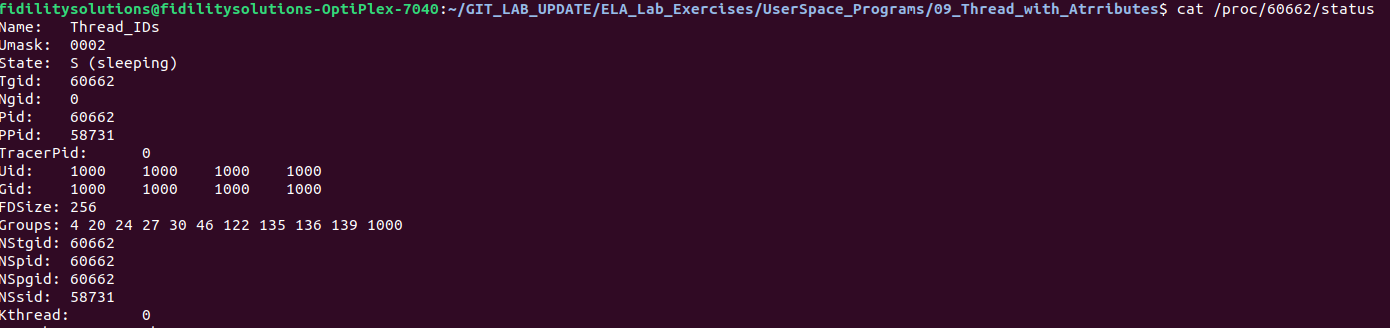
# **Understanding Processes Using /proc Interface and ps Command**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **Using /proc Interface**

* Identify the process ID -> 60662
* cat /proc/$pid/status.

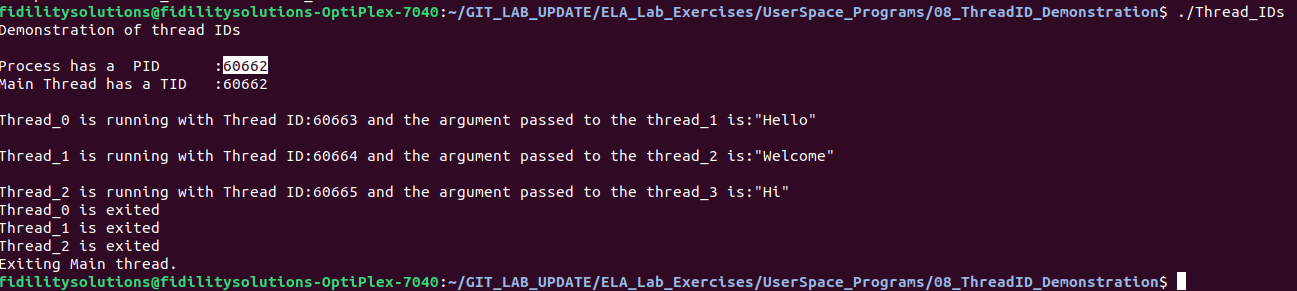
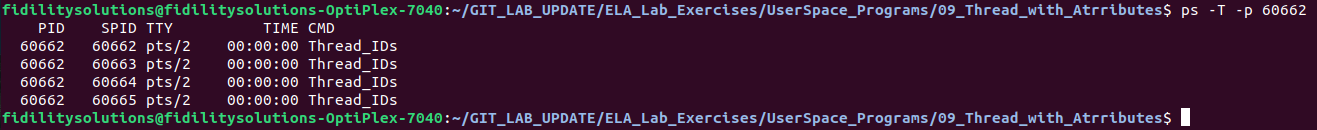
$ cat /proc/60662/status



## **Using ps Command**

* To displays thread information for a specific process ID **-> ps -T -p <TID>**

$ ps -T -p <TID>

* The image depicts four threads, with one identified as the main thread and three as threads created within the main thread. Each thread is associated with a unique thread ID, and the image displays the arguments passed from their respective function calls.
* when you run **ps -T -p <TID>**, you will get information about the threads associated with the process identified by **<TID>**