**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**

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

1. 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 Orphan\_Process [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 Orphan\_Process [root@10.10.1.27:/User\_Programs](mailto:root@10.10.1.27:/User_Programs)

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

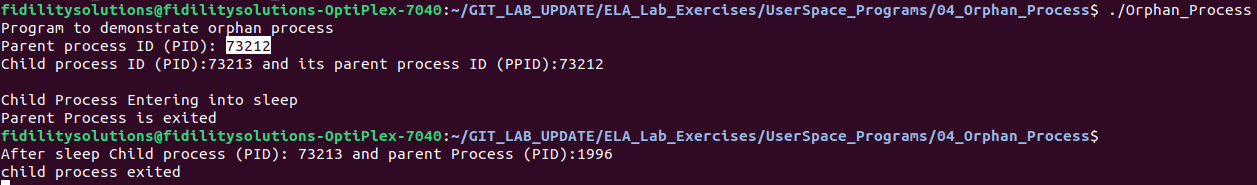
**Running on Platform (x86, Raspberry Pi, BBB)**

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

1. Once you're in the correct directory, execute the generated executable file using the **./filename** command.

$ ./Orphan\_Process

1. The overall output will be as given below:



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

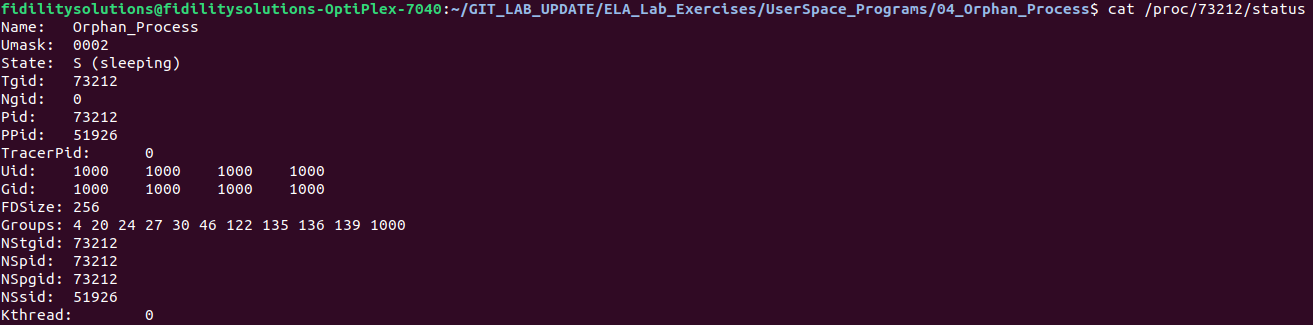
# **Understanding Processes Using /proc Interface and ps Command**

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

## **Using /proc Interface**

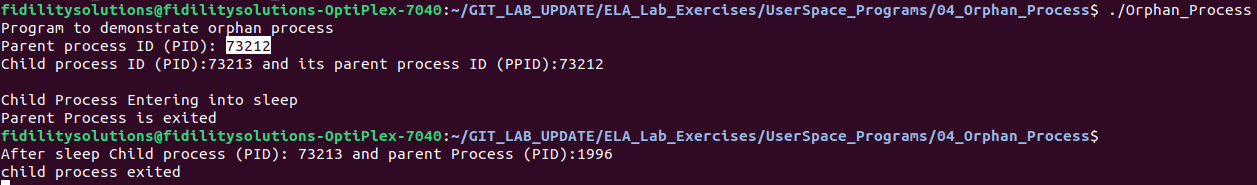
1. Identify the PID (Process ID) of a running process -> 73212
2. /proc/$pid/status

$ cat /proc/73212/status



## **Using ps Command**

1. The **ps -ef/ps aux** command provides a concise overview of all processes running on the system, displaying detailed information in a full-format listing.
2. The following image showing the child process entered sleep for long time so the parent process exited before child process, in that case the kernel allocated new parent to child process and the process ids shown below.



1. The image depicts entry of child process present in process table.

$ ps aux

