**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 Zombile\_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 Zombile\_Process [root@10.10.1.27:/User\_Programs](mailto:root@10.10.1.27:/User_Programs)

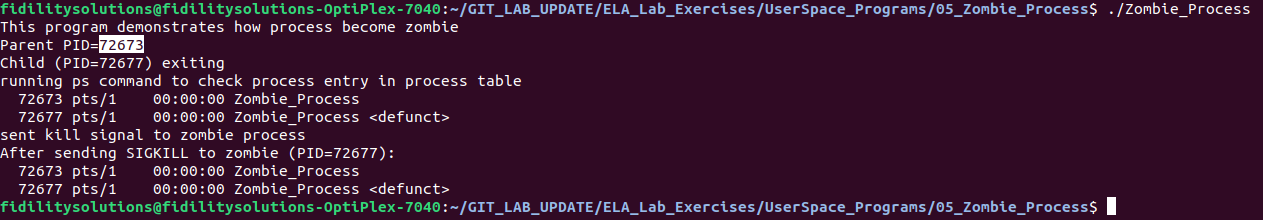
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

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

$ ./Zombile\_Process

* The overall output will be as given below:

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

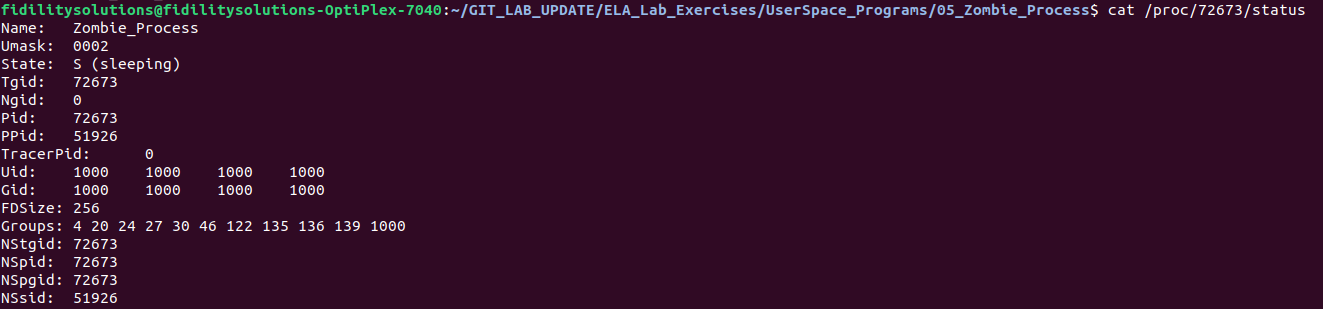
# **Understanding Processes Using /proc Interface and ps Command**

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

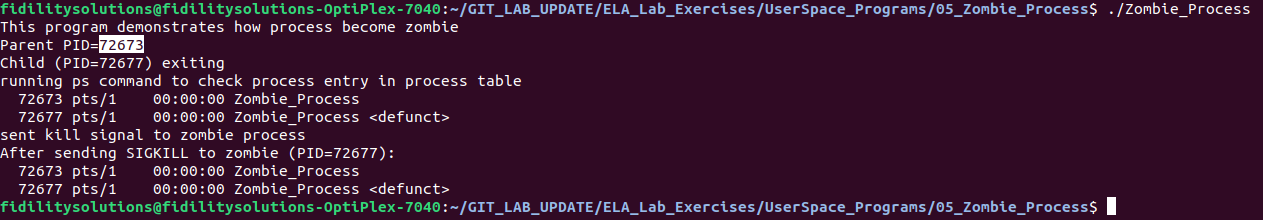
## **Using /proc Interface**

* Identify the PID (Process ID) of a running process-> 72673
* cat /proc/$pid/status

$ cat /proc/72673/status



## **Using ps Command**

1. **ps -ef** and **ps aux** commands display active processes on Unix-like systems.
2. The following image showing the child process completed its execution, but parent process entered sleep for long time so the parent process has not yet called wait () to obtain child exit status.
3. Parent process sent the signal to kill the zombile process, but it remains alive.
4. Since the zombies cannot be killed by a signal, the only way to remove them from the system is to kill their parent (or wait for it to exit), at which time the zombies are adopted and waited on by init, and consequently removed from the system
5. The image depicts entry of process present in process table, which means the child process completed its execution but still have entry because the parent of child not yet called wait () system call to wait for termination of child process. In image Z indicates that is Zombie Process.



$ ps aux