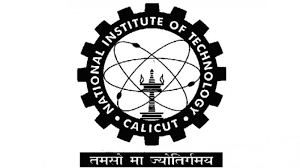
**Project Report**

**Kernel syscall that displays process details by using its pid**



*Submitted to*

*Dr. Jayraj P B*

*Dept. of Computer Science and Engineering*

*NIT Calicut*

*By*

*Arjun Nandlal Bhardwaj*

*Roll No: M210683CA*

## Introduction

Process descriptor is a simple kernel system call that takes input as process id from the user and display its details on screen

It shows the details such as Process name, Process ID,

Process priority, Process parent name and Process exit state.

In this we are creating a new system call that directly calls to our custom function.

## Requirements

* Linux Kernel
* Ubuntu

## Major Tasks

1. **Creating a new system call**

**Experiment procedure**

1. **Instalation environment**

**1 | #sudo apt-get install libncurses5-dev**

**2 | #sudo apt install build-essential**

**3 | #sudo apt-get install libncurses5-dev**

**4 | #sudo apt-get install flex**

**5 | #sudo apt-get install bison**

1. **Add system calls to kernel**

**(Drag the downloaded kernel into ubuntu)**

**1| #sudo mv linux-5.15.77.tar.gz/usr/src**

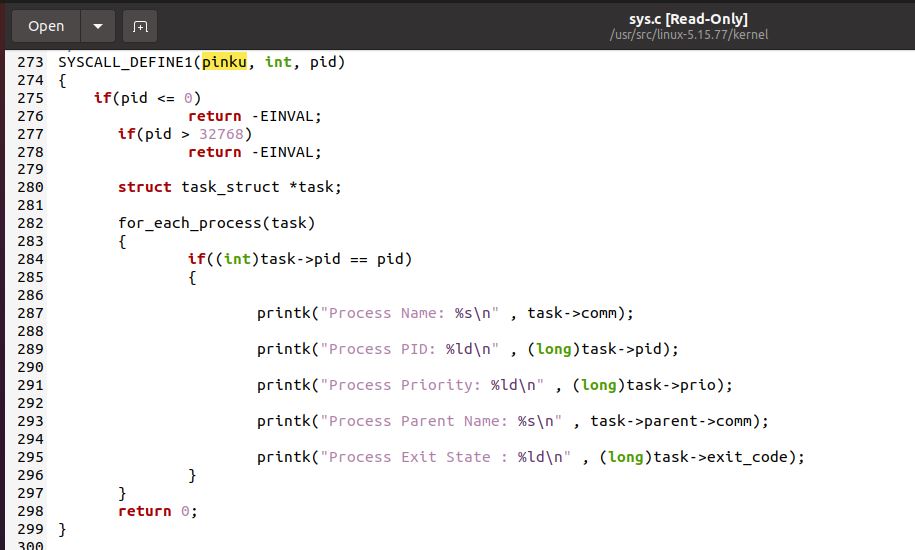
**2| #subo tar –xvf linux-5.11.7.tar.gz**

**Step 1: Add a custom function**

**1| #sudo cd/usr/src/linux-5.15.77**

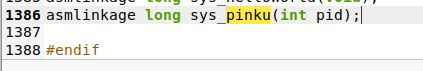
**2|#sudo gedit kernel/sys.c**

**Here we write the functionality of our syscall**



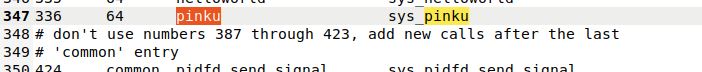
**Step 2: Declare the header file**

**1| #sudo gedit include/linux/ syscalls.h**



**Step 3: Add call number**

**1| # gedit arch/x86/entry/syscalls/syscalls\_64.tbl**



1. **Compile the kerel**

**1 | #sudo make clean**

**2 | #sudo make mrproper**

**3 | #sudo make menuconfig**

**Modify the configuration file**

**1|# sudo gedit .config**

CONFIG\_SYSTEM\_TRUSTED\_KEYS //comment out this

**1|# sudo make –j4 //used to assign cores to syscall**

**( the compilation process takes 3-4 hrs on avg. )**

1. **Install the kernel**

**1 | #sudo make madules\_install**

**2 | #sudo make install**

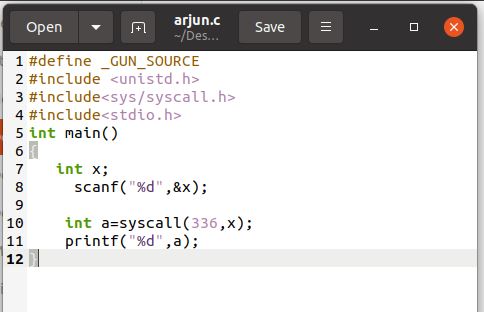
**3 | #sudo reboot**

**1| #sudo uname -r**

**(Appearing the corresponding kerel version means success)**

1. **Edit the test file**

**1| #sudo gedit arjun.c**



1. **Running the system call**

**1 | #sudo gcc arjun.c –o arjun**

**2 | #sudo ./arjun**

**3 | #sudo dmesg**

**{here give pid as user input}**

**The output will be displayed**

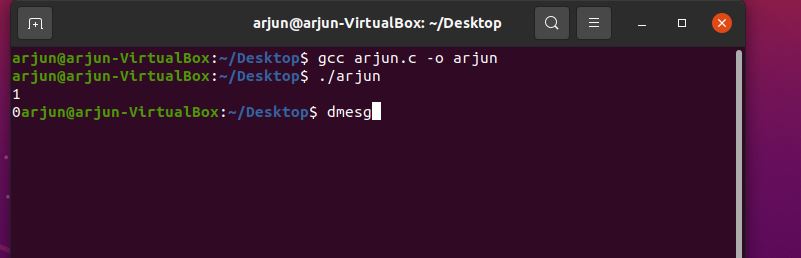
1. **Testing the syscall**
2. **Give pid as random number**

**(helps to check whether the output is displayed or not)**

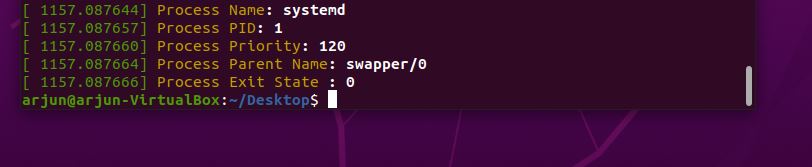
**Example:**

**Input:**

**Here arjun.c contains syscall function and 1 is pid input**



**Output:**



1. **Give pid as Known number**

**(helps in verification of output)**

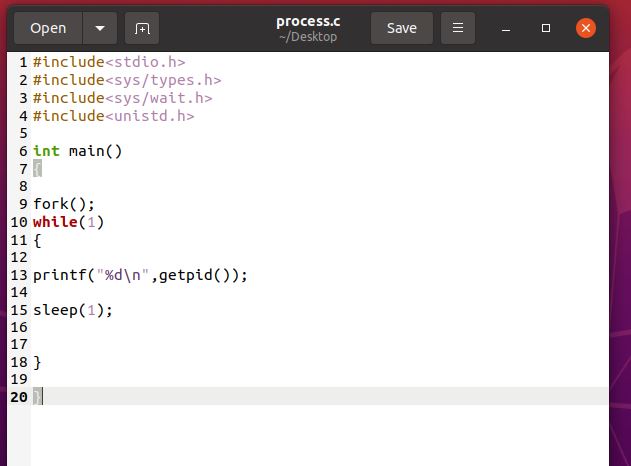
**Example :**

**We created a dummy program named as “process.c”**

**This program prints Pid of parent process as well as**

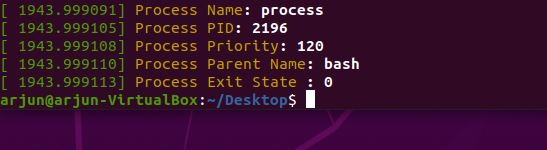
**Pid of choild process**

**Code:**

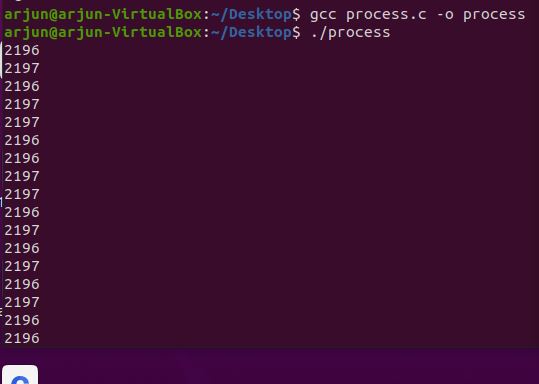


**Lets run it on terminal\_1**

**Output:**

****

**OUTPUT ( In Terminal\_2 )**

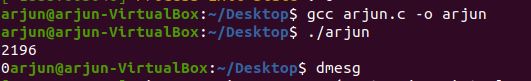


**Here note that**

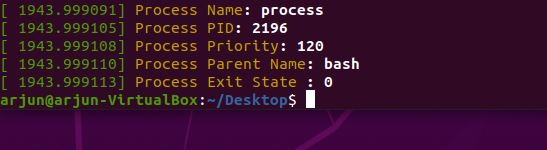
**2196 is pid of parent process and**

**2197 is pid for child process**

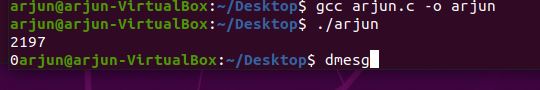
**Now run this on Terminal\_2**

** Input Parent Pid :**

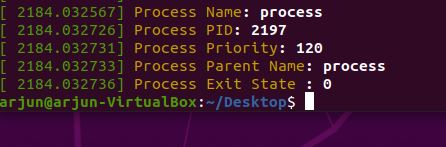
**Output :**



**Input Child Pid :**

****

**Output:**

****