**UNIVERSIDAD AUTÓNOMA METROPOLITANA**

**UNIDAD IZTAPALAPA**

****

**Casa abierta al tiempo**

**Práctica 3**

**Arboles de procesos y Pseudo paralelismo**

**NOMBRE:**

Carrillo Pacheco Francisco Javier

2143008102

**MATERIA:**

**Sistemas Operativos**

**PROFESOR:**

Dr. Benjamín Moreno Montiel

# **18/FEB/2019**

# **Introducción**

Cuando un programa hace la llamada al sistema fork(), este se crea una copia exacta del proceso activo. Cada variable, constantes, metodos, estan hubicados en regiones de memoria diferentes y cada proceso tiene su identificador diferente.

La estructura de datos que puede representar perfectamente el comportamiento de las llamadas al sistema fork, son los arboles, aveces este puede ser balanceado, otras no, todo dependera de la aplicación.

Para esta primera parte de la practica, crearemos un arbol de procesos, cuya profundidad y numero de ramificación estara determinada por el usuario.

Este programa estara hecho con el lenguaje de programacion Python, a si que para poder implementarlo, tuvimos que instalar previamente Cygwin, el cual nos permitira correr la aplicacion como si estuvieramos en una terminal en linux, y tambien se debio de haber instalado su compilador con la versión 3 de python.

Importamos las librerias os, sys y time, estas nos serviran para poder llamar al sistema y poder manejar los procesos, posterior mente se le pedira al usuario ingresar la profundidad y numero de ramas y se tendra que implementar ciertas tecnicas para poder creear dicho arbol.

Para la segunda parte de la practica se tendra que implementar un esquema de pseudoparalelismo, el cual es una pequeña simulacion de como el sistema operativo pude generar diferentes procesos e implementaremos un esquema maestro-esclavo.

El programa iniciara con el proceso padre, la funcion principal de este proceso maestro es crear una lista de enteros aleatorios entre el 1 al 10000, posterior mente, creara 4 hijos, estos hijos seran los encargados de buscar la cantidad de numeros que hay de pares, impares, primos y numeros perfctos en el arreglo.

El proceso padre, esperar el resultado de cada uno de sus hijos y reportara el tiempo de ejecucion de la aplicacion

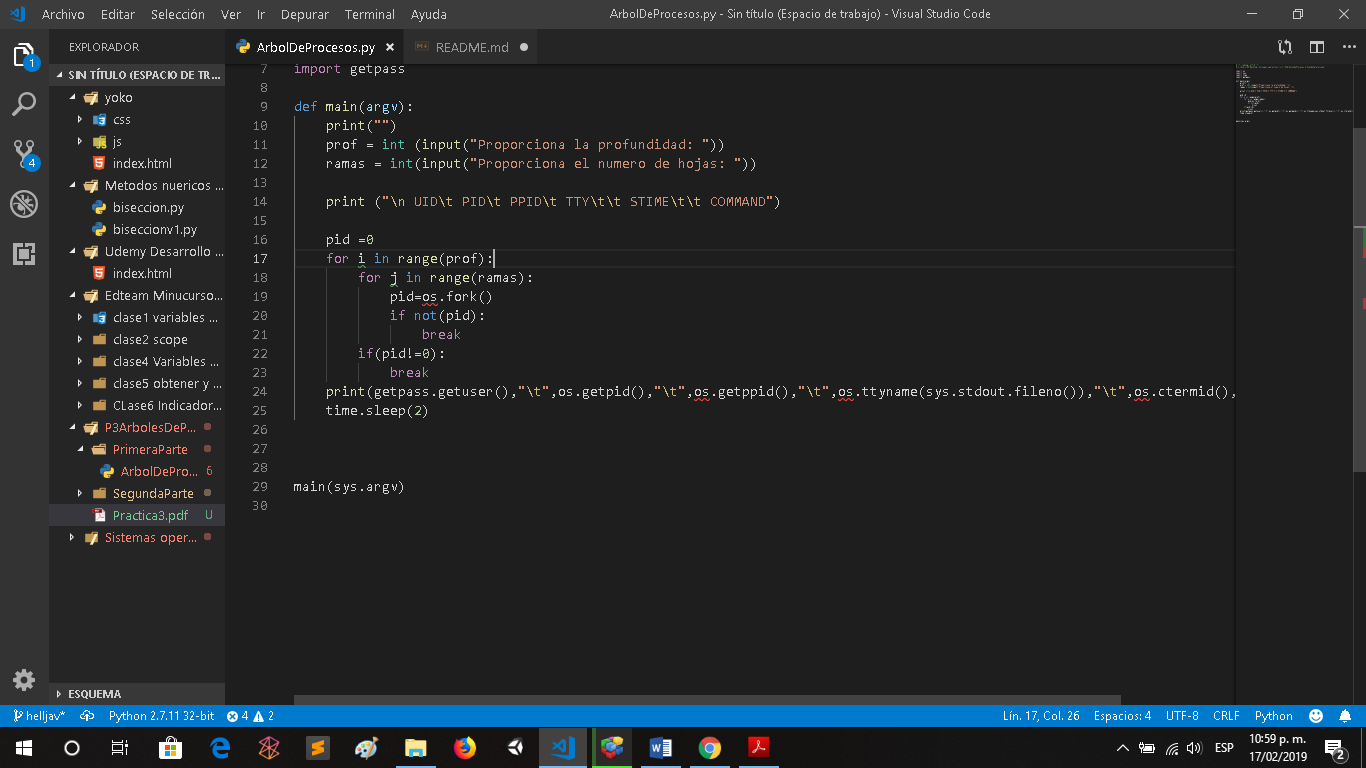
# **Desarrollo**

**Árbol de procesos**

Para la primera parte de la practica, importamos las librerias nesarias: os,sys,time y get pass, como se nos pidio implementar mostar un desplegado similar al comando ps-fe, este ultima librería nos permite mostrar el nombre de usuario de la maquina donde se estra mostrado el programa, la librería time se utilizara para dormir los procesos por unos segundos y las ultimas dos librería restantes la utilizaremos para obtener informacion al sistema y manejar los procesos.

Posteriormente, le pediremos al usuario que ingrese la profundidad y el numero de ramas del arbol para crear el arbol de procesos, estos datos los guarde en dos variables (prof y ramas).

Para creear el arbol, yo utilice dos for anidados, el primero indicara la profundidad del arbol y el degundo indicara el numero de ramas. Dentro del segundo for, hare las bifurcaciones con el metodo fork() y el valor que devuelva sera guardado en una variable antes creada llamada pid, para que despues sea evaluado, ya que si es un proceso hijo, se saldra del segundo for, ya que solo se quiere que los proceso padre, cree el total de las ramas, al finalizar el segundo for, se volvera a evaluar, si es un proceso padre o hijo, si es un proceso padre, se sale del primer for, ya que acabo de generar todas la ramas y si aun no se acaba de generar la profundidad el arbol solicitado, los hijos deben de continuar generando ramas, a si hasta que se alcance la profundidad del arbol.



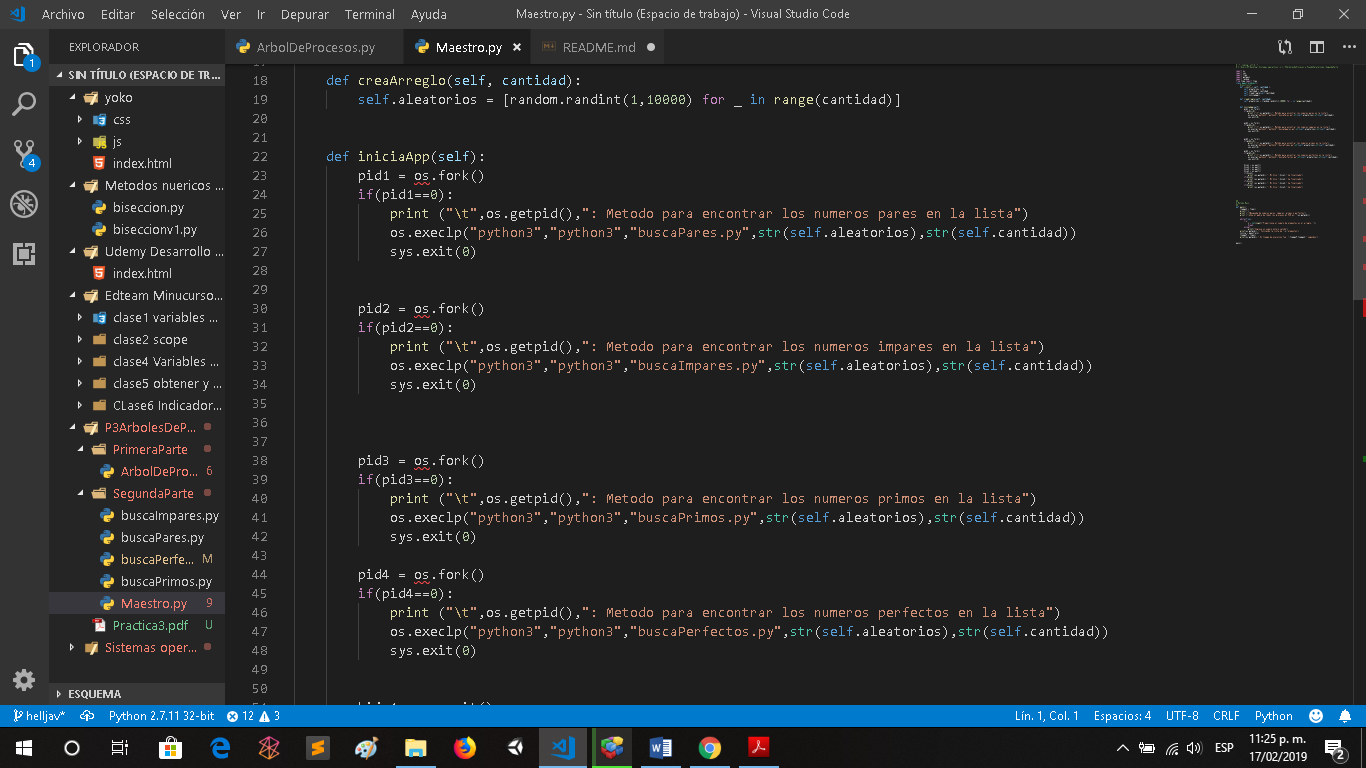
Posteriormente, cada proceso, mandara a imprimir a consola, su propia informacion, como el nombre de usuario, su pid, el pid de su padre,el tty, el nombre del archivo, etc.

Para lograr la simulacion del comando ps-fe.

**Pseudoparalelismo**

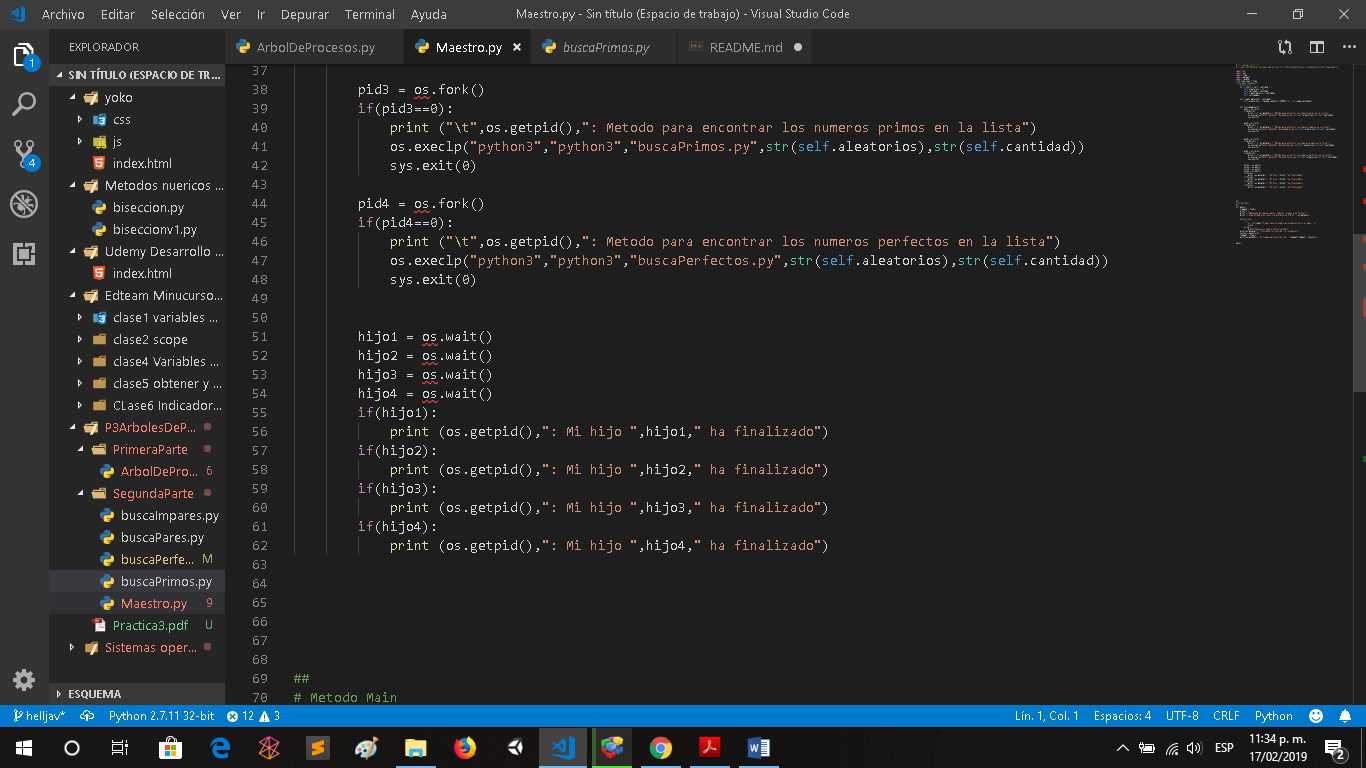
Para esta segunda parte de la practica, iniciamos con un proceso inicial, yo cree una clase llamada maestro que recibe en su contructor el tamaño del arreglo, para que al momento de llamarlo, cree el arreglo de numeros enteros aleatorios, este arreglo sera de forma global, y tambien se inicie automaticamente un metodo llamado iniciaApp.

Lo que hace este metodo es que el proceso padre, haga el uso del fork(), posterior mente a ese hijo creado, se le asignara una tarea con su propio codigo, este codigo va ser un nuevo arhivo que sera ejecutado por el hijo,esto se puede hacer gracias a la funcion os.execlp, al finalizar la tarea, finalizara con el comando sys.exit(0)



Para el hijo que se encargaria de encontrar los numeros pares, recorri todo el arreglo que es mandado por el padre, en la funcion os.execlp llamado aleatorios, acada elemento recorrido le saque modulo 2, si este era igual a cero, un acontador aumentaba en uno, hasta finalizar el recorrido del arreglo. Para el hijo que debia de calcular los numero impares, solo modifique la condicion del codigo anterior, ya que si el modulo 2 de cada elemento evaluado, era diferente de cero, el contador aumentaba a uno y a si cada hijo ejecutaba su propio codigo, pasandole solo como argumentos el arreglo y el tamaño del arreglo (self.cantidad)

Hasta el ultimo el proceso padre solo esperaba la finalizacion de cada uno de sus hijos.



# **Conclusión**

Se me dificultaron ciertas partes de la práctica, sobre todo como pasar cierta información del proceso padre al hijo, para que este pudiera hacer uso de el en su propio código, ya que tiene tiempo que tome el curso de programación currente, así que tuve que volver a repasar un poco, de ahí en fuera, me resulto muy entretenida, y sacar los números perfectos fue un poco atareado