Practica 2: Árbol no Balanceado

Equipo: Not False

Grupo: 2CM1

Fecha: Marzo 3, 2016

Contents

[Planteamiento del problema 2](#_Toc444801767)

[Objetivo 2](#_Toc444801768)

[Procedimiento 2](#_Toc444801769)

[Resultados 3](#_Toc444801770)

[Conclusiones 5](#_Toc444801771)

[Cortes Francisco Juan Salvador 5](#_Toc444801772)

[Lara Hernández Ricardo 5](#_Toc444801773)

[Romero Téllez Rolando de Jesús 5](#_Toc444801774)

# Planteamiento del problema

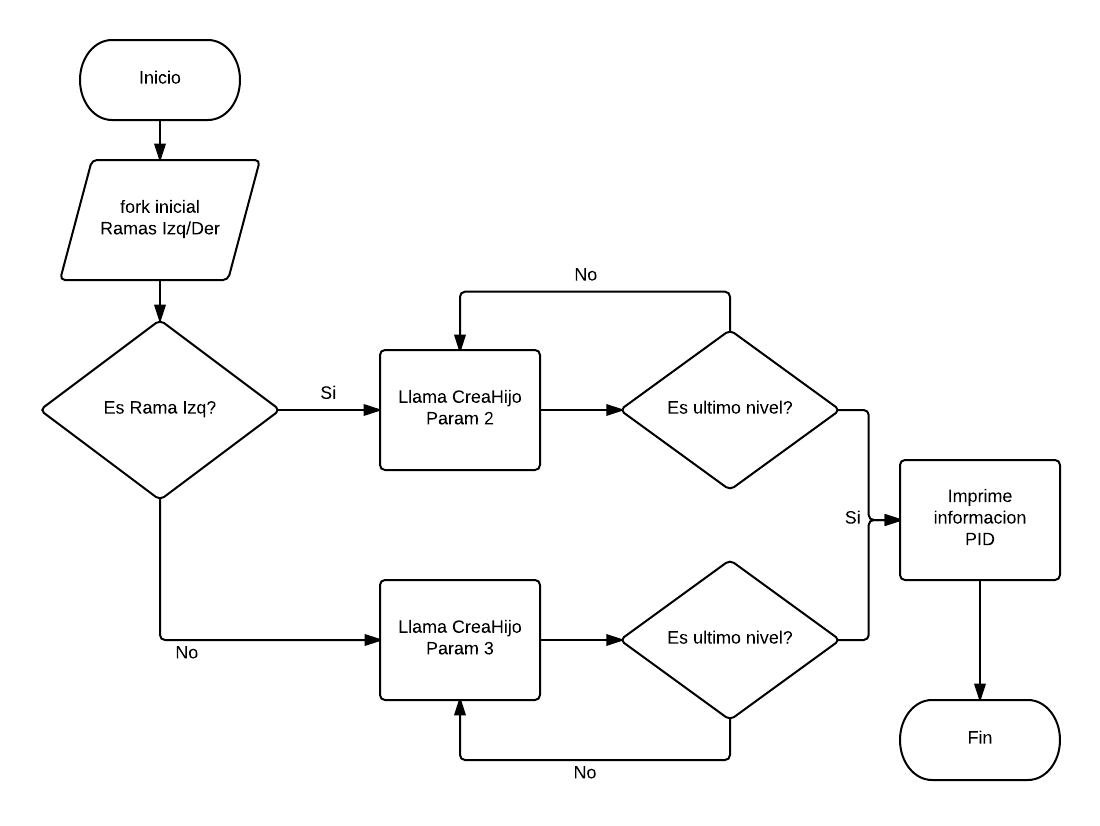
Se requiere formar un árbol no balanceado de forma que los hijos de la rama izquierda generen 2 hijos y los de la rama derecha generen 3, hasta un nivel N especificado por el usuario

# Objetivo

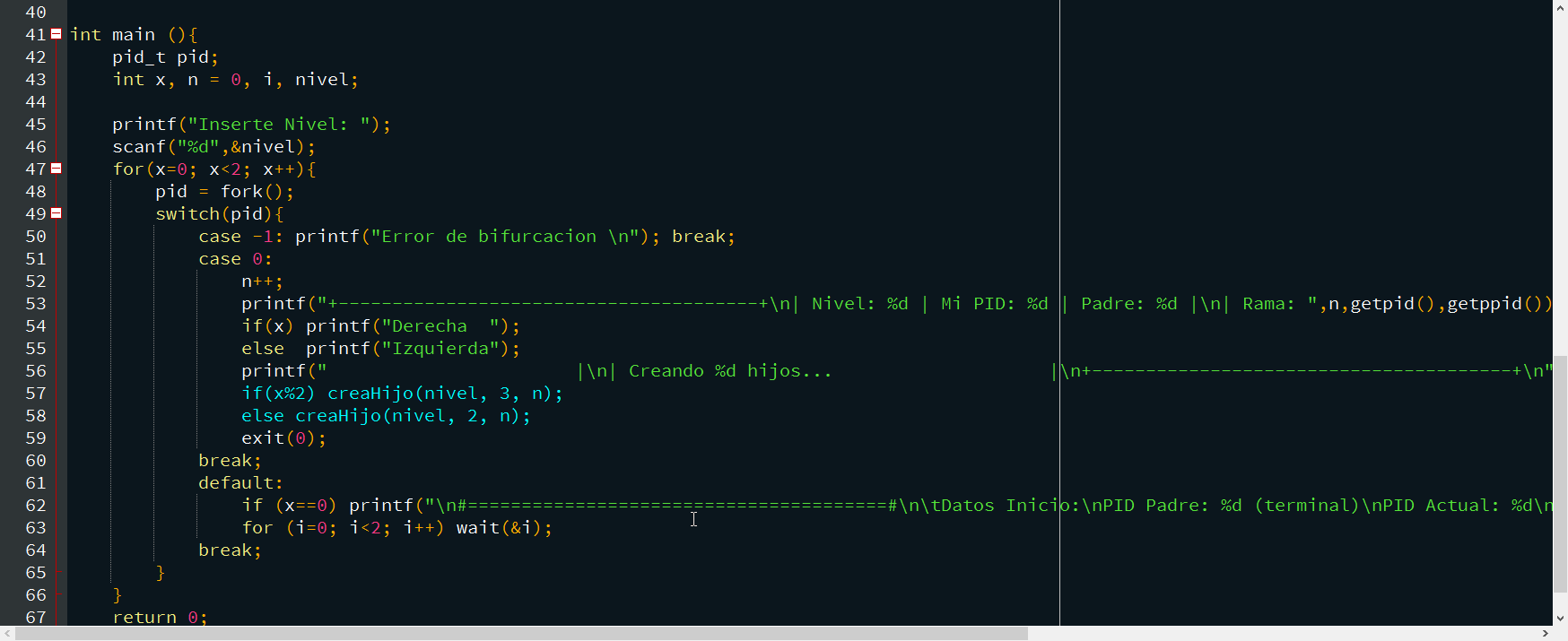
Es diseñar un programa que sea capaz de utilizar procesos para la generación de un árbol con la siguiente forma.

# Procedimiento

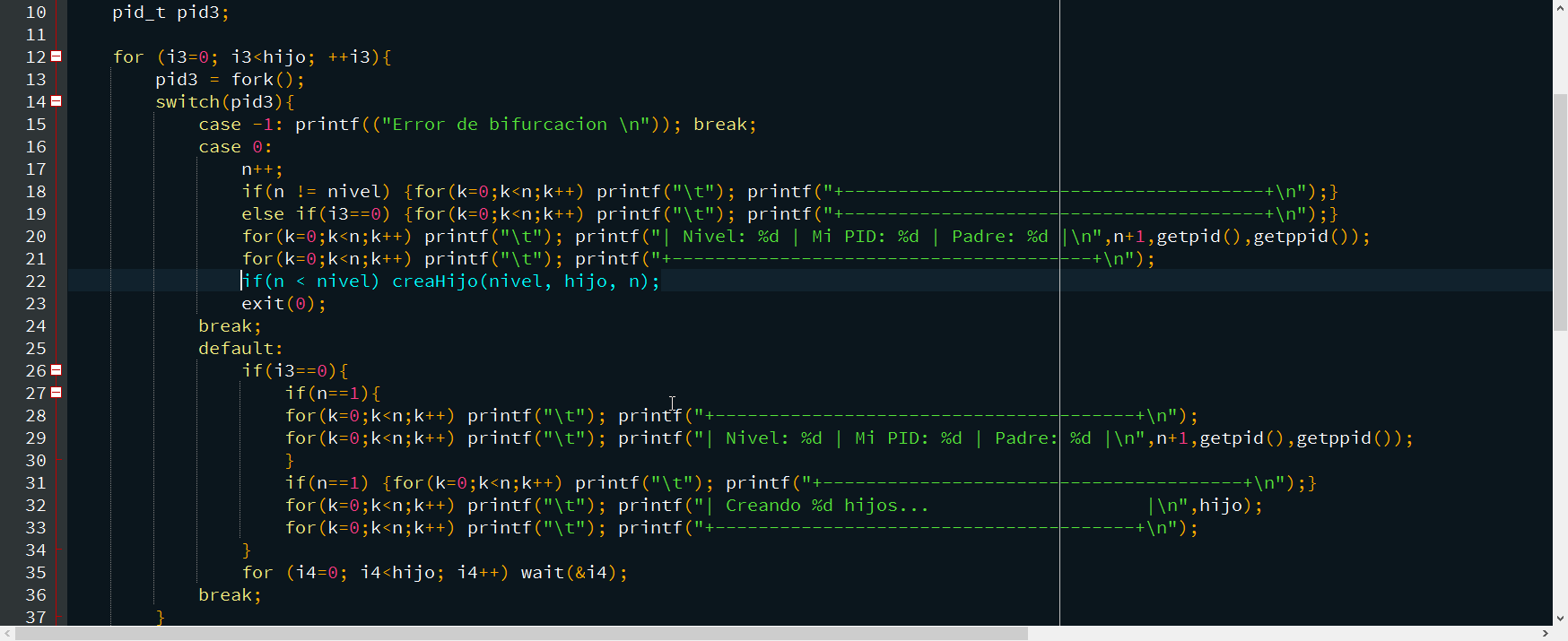
Para esta práctica diseñamos una función recursiva que dependiendo de sus parámetros generaba 2 o 3 hijos, y se llamaba recursivamente hasta llegar al límite especificado por el usuario.



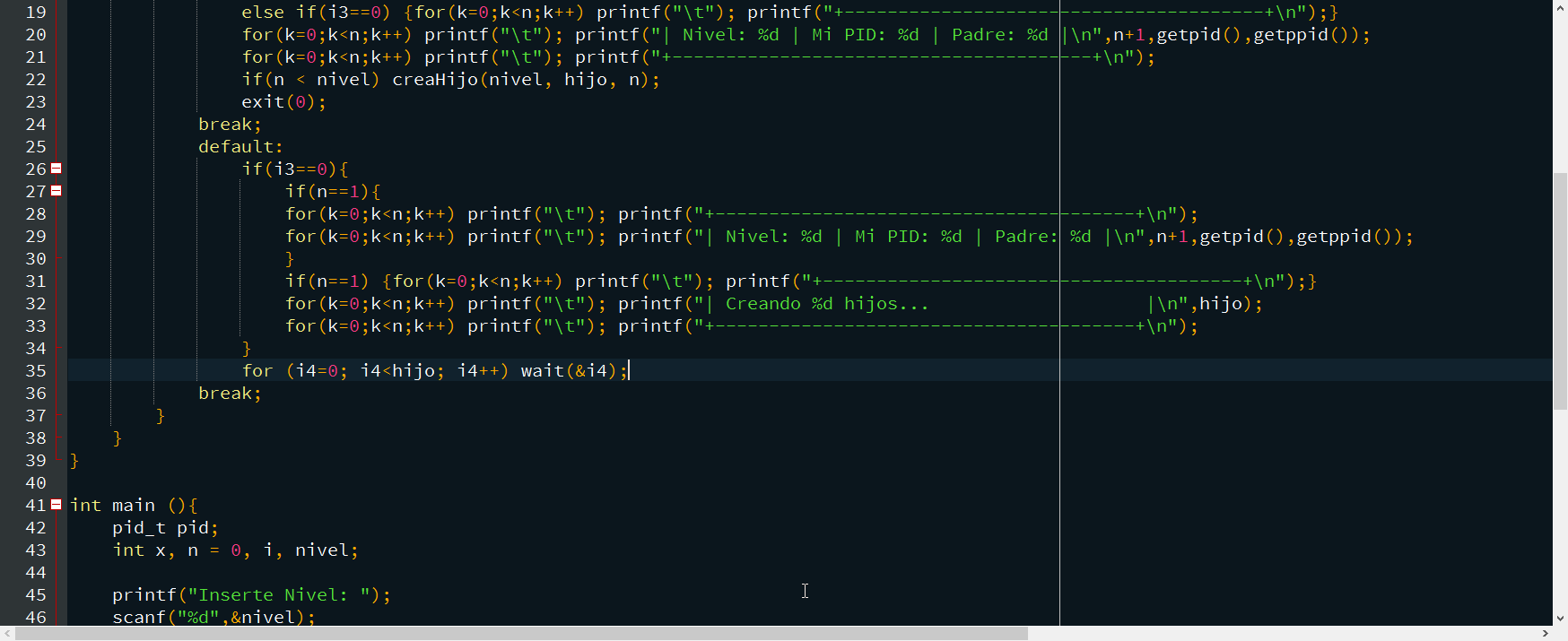
Basándonos en las practicas anteriores decidimos usar un switch con varios casos dependiendo de si era hijo o padre, primeramente creamos los 2 primeros procesos con los que inicia el árbol (las ramas izq/der) y a partir de ahí llamamos a la función creaHijo() con el primer parámetro siendo el ultimo nivel, el segundo el número de hijos a generar, y el ultimo el nivel actual. Para la decisión de rama utilizamos un modulo 2.



A partir de ahí se genera la función recursiva que, de igual manera está dividida en casos, para el caso hijo imprime su información y en caso de que el nivel actual sea menor al nivel requerido vuelve a llamar a la función.

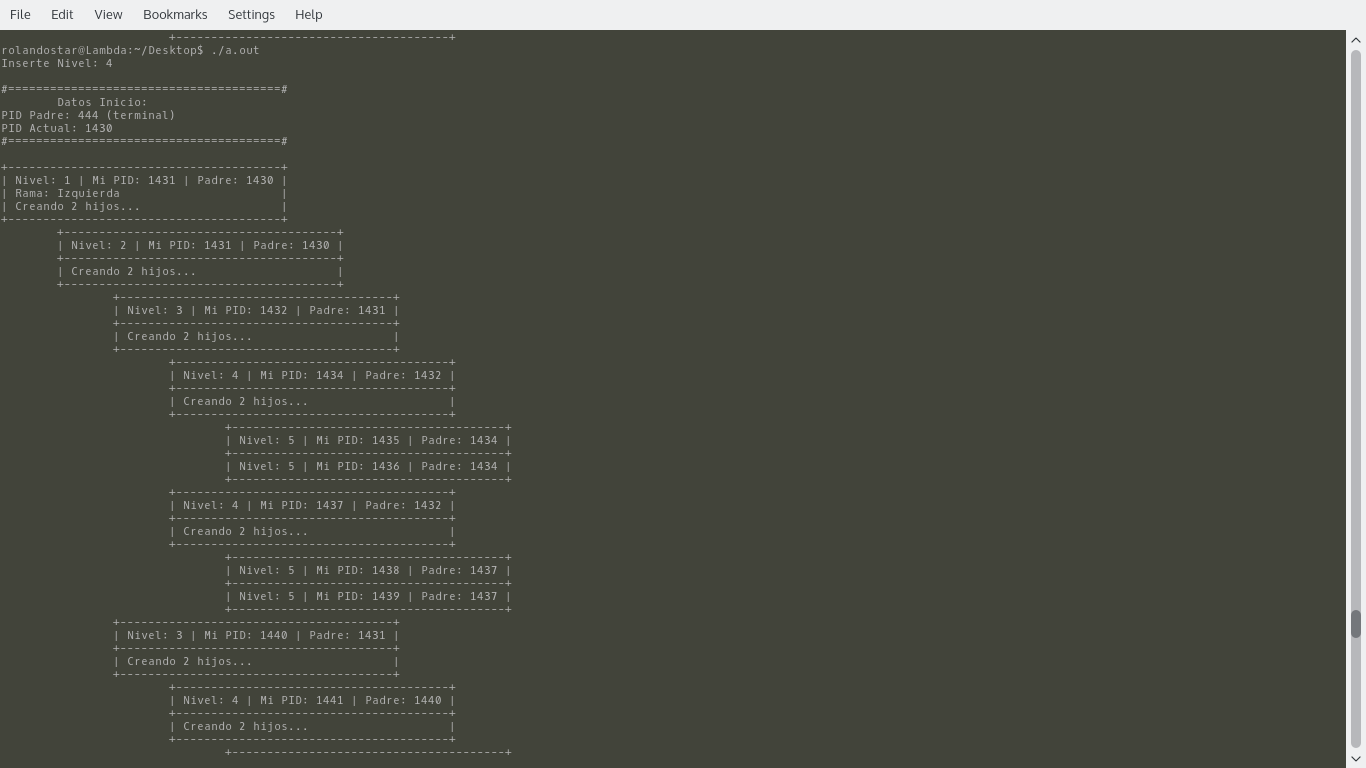


Mientras que en el caso del padre, además de imprimir su propia información, espera a que cada uno de sus hijos termine.

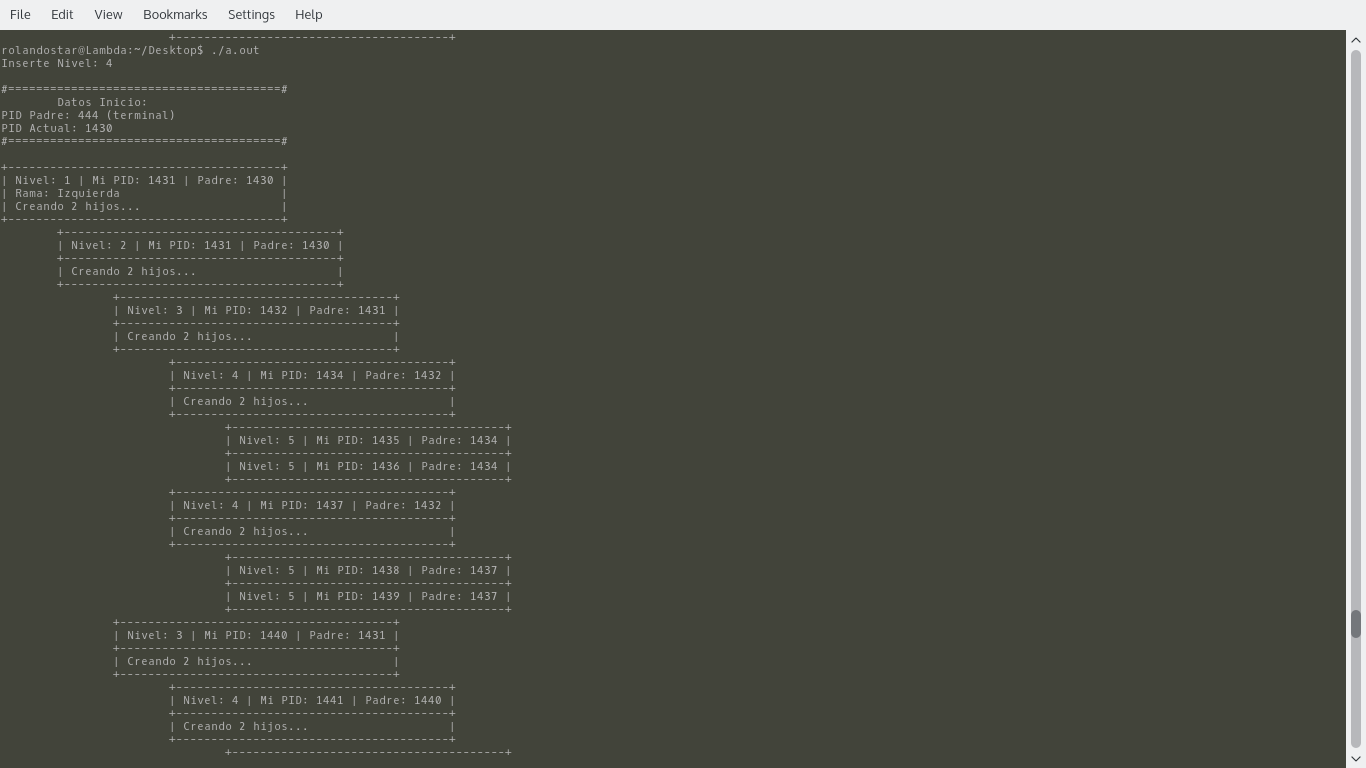


# Resultados

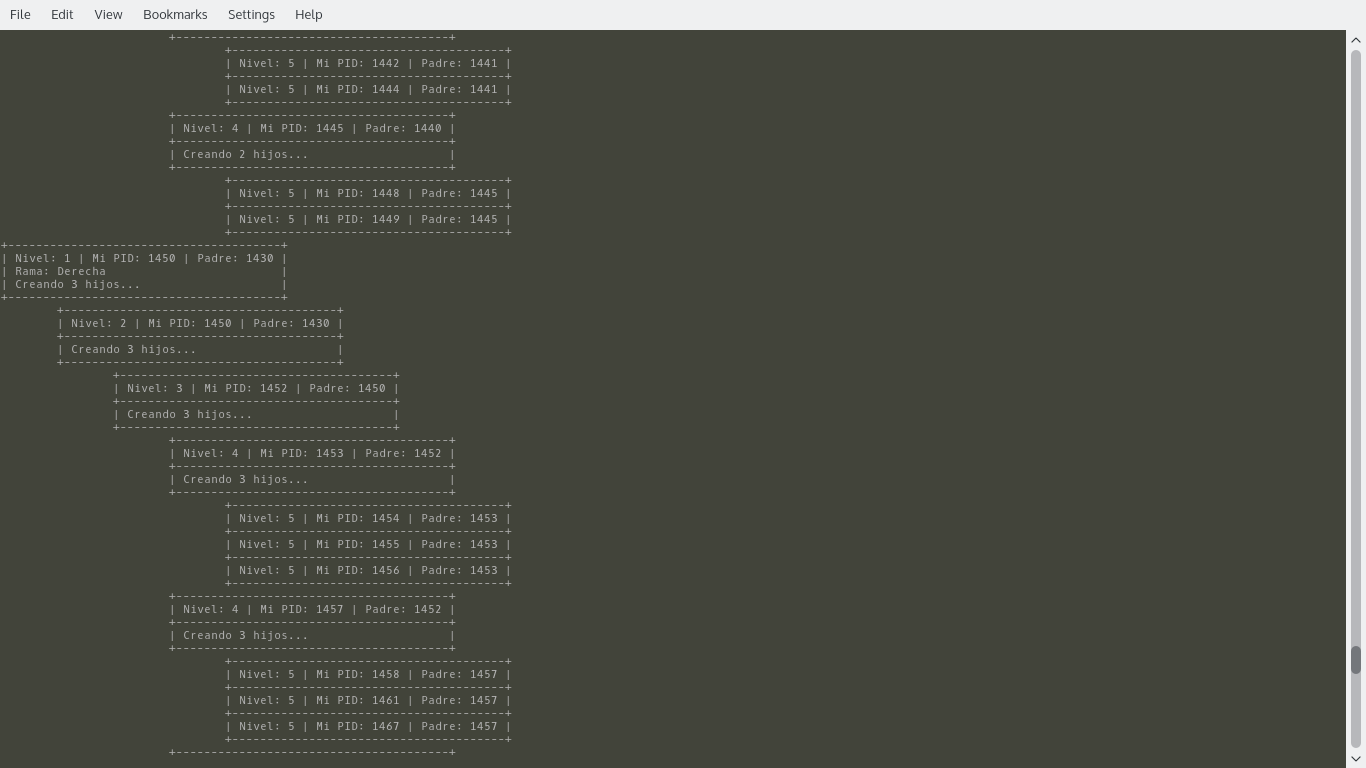
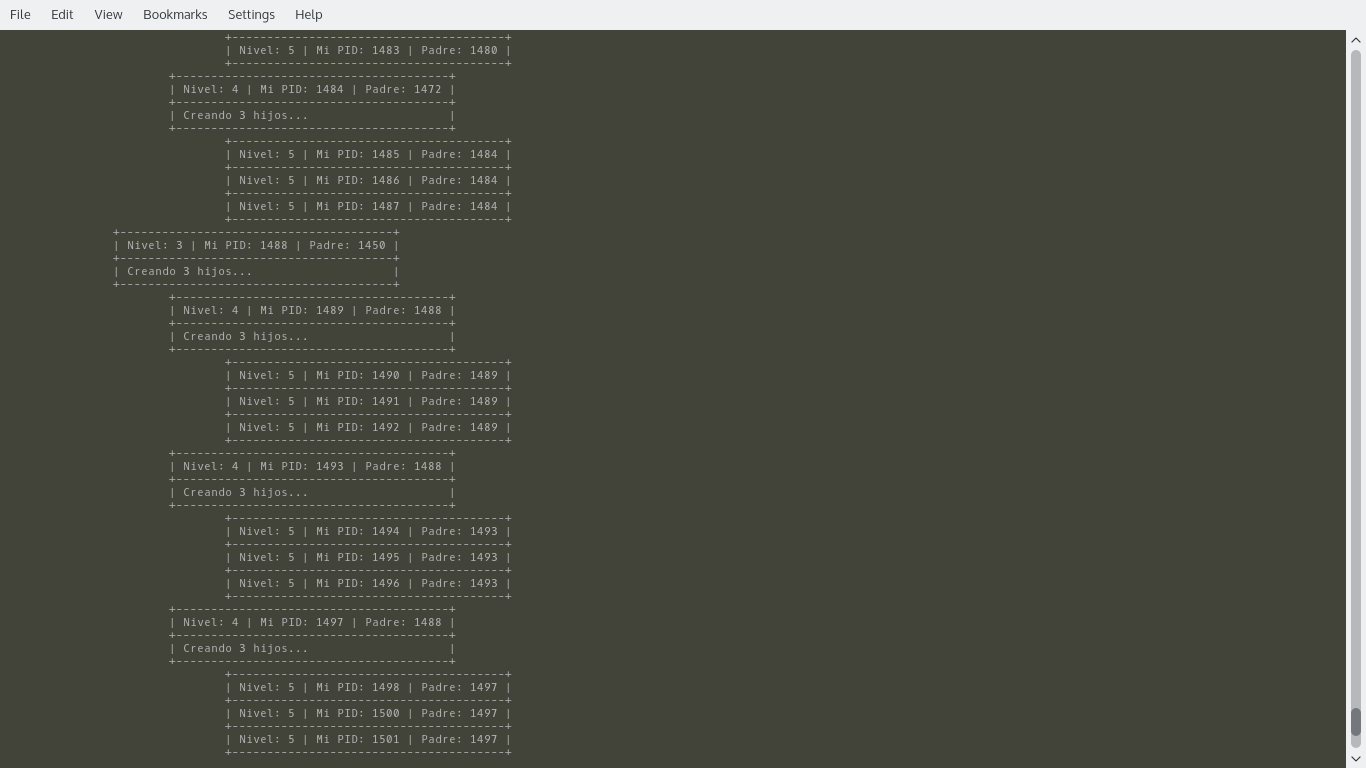
La primera línea de la salida de nuestro programa es una petición a introducir el nivel limite hasta donde llegara el árbol. En este ejemplo se uso 4.



Inmediatamente después muestra la información del nodo padre general.



Seguido del árbol cuya indentacion indica el nivel de profundidad



# Conclusiones

## Cortes Francisco Juan Salvador

En esta práctica repasamos más afondo la creación de hijos y como los padres los deben de esperar, aunque los procesos deberían ser completamente aleatorios en su ejecución, podíamos ver una estructura clara gracias a los "wait" y eso le daba una mejor vista a nuestra salida, lo que nos costó trabajo fue el hecho de que no teníamos una idea clara de cómo pasar de manera recursiva al programa pero cuando lo descubrimos funciono de manera correcta.

## Lara Hernández Ricardo

Con esta práctica pude observar cómo se comportan de manera más clara los procesos y como es el funcionamiento del mismo al crear hijos, sufrimos un poco a la hora de realizar el árbol tal cual ya que al simplificar la función y meter las instrucciones en una función recursiva, el funcionamiento era errático, una vez corregido simplemente le dimos un buen formato de salida

## Romero Téllez Rolando de Jesús

Durante esta práctica tuvimos problemas al momento de implementar la recursividad ya que en seguidas ocasiones ocurrían inconsistencias entre ejecuciones lo cual hacia complicado el proceso de debugging, al punto que tuvimos que reescribir el programa un par de veces hasta encontrar un método consistente de obtener los resultados deseados.