UNIVERSIDAD SAN CARLOS DE GUATEMALA -USAC-
CENTRO UNIVERSITARIO DE OCCIDENTE
-CUNOC-
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA

DOCUMENTACIÓN PROYECTO 1 SISTEMAS OPERATIVOS 1

Celia Esmeralda Vargas Lopez 201730930

Ingeniería en Ciencias y Sistemas.

Ing. Francisco Rojas.

Quetzaltenango, abril de 2021

ÍNDICE

HERRAMIENTAS UTILIZADAS

Visual Studio Code:

- Visual Studio Code es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos personalizar y potenciar esta herramienta.
- Las extensiones de Visual Studio Code nos otorgan infinidad de opciones, como colorear tabulaciones, etiquetas o recomendaciones de autocompletado. También hay extensiones que nos ayudan con el lenguaje de programación que vayamos a usar, como por ejemplo para Python, C / C++, JavaScript, etc.

Lenguaje C:

- C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación.
- Este lenguaje ha sido estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Sin embargo, este lenguaje no está ligado a ningún sistema operativo ni a ninguna máquina

concreta. Se le suele llamar lenguaje de programación de sistemas debido a su utilidad para escribir compiladores y sistemas operativos, aunque de igual forma se puede desarrollar cualquier tipo de aplicación.

Glade:

- GTK son las siglas de GIMP Tool kit, ya que fue diseñado originalmente para implementar la aplicación GIMP (Programa de manipulación de imágenes GNU) .
- Es un conjunto de herramientas para crear aplicaciones GUI (interfaz gráfica de usuario) en múltiples plataformas. GTK se implementa como una biblioteca con un conjunto de widgets que se utilizan para crear aplicaciones GUI.

CÓDIGO:

Creación de procesos con la función Fork();

Cuando se llama la función fork, esta genera un duplicado del proceso actual. El duplicado comparte los valores actuales de todas las variables, ficheros y otras estructuras de datos. La llamada a fork retorna al proceso padre el identificador del proceso hijo y retorna un cero al proceso hijo.

1. **PID**

Las siglas PID, del inglés Process IDentifier, se usan para referirse al identificador de proceso.

2. **La función getppid**

El hijo puede obtener el identificador del proceso padre llamando a la función getppid .

3. **La variable $$**

La variable especial $$ contiene el PID del proceso actual. Es de sólo lectura.

En el caso de nuestro proyecto se implementó de la siguiente manera:

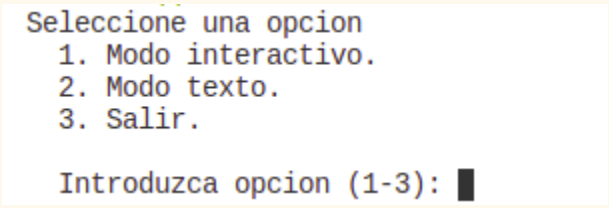
Se obtiene la planta a imprimir y se obtiene la cantidad de plantas y hojas en ese momento, ya que puede variar. Después de obtener las hojas y ramas se procede a crear los procesos.

```
void imprimirPlantas(char *numeroPlanta){
```

```
int numero = atoi(numeroPlanta);

   for (int i = 0; i < sizePlantas; i++) {

       if (arregloPlantas[i].idPlanta != 0) { //-1 es una planta no
creada

           if(arregloPlantas[i].idPlanta == numero){

               printf("\n\n PLANTA: %d RAMAS: %d HOJAS: %d",
arregloPlantas[i].idPlanta, arregloPlantas[i].cantidadRamas,
arregloPlantas[i].cantidadHojas);

               int ramas = arregloPlantas[i].cantidadRamas;

               int hojas=  arregloPlantas[i].cantidadHojas;

   pid_t pid;

               //en caso de solo ser la planta se imprime el arbol

           if(ramas == 0){

               pid = fork();

               if (pid) {//planta

                   printf("\n  #Planta %d", getpid());

                   int numeroP = getpid();

                   imprimirArbol(numeroP);

               }else{}

           }else{

           }

   int x, y;


   for (x = 1; x <= ramas; x++) {

       pid = fork();

       if (pid) {//planta
```

```
            printf("\n  #Planta %d", getpid());
            sleep(3);


        } else {
            printf("\n      #Rama %d de la planta %d",
getpid(),getppid());
            for (y = 1; y <= hojas; y++) {
                pid = fork();
                if (pid) {//rama
                } else {//hoja
                        printf("\n          #Hoja %d de la rama %d",
getpid(), getppid());
                    exit(0);
                }
            }
            int numeroPR = getppid();
            imprimirArbol( numeroPR);
            exit(0);
        }
    }
                }
        } else {
            //espacio vacion, no mostramos nada
        }
    }
```

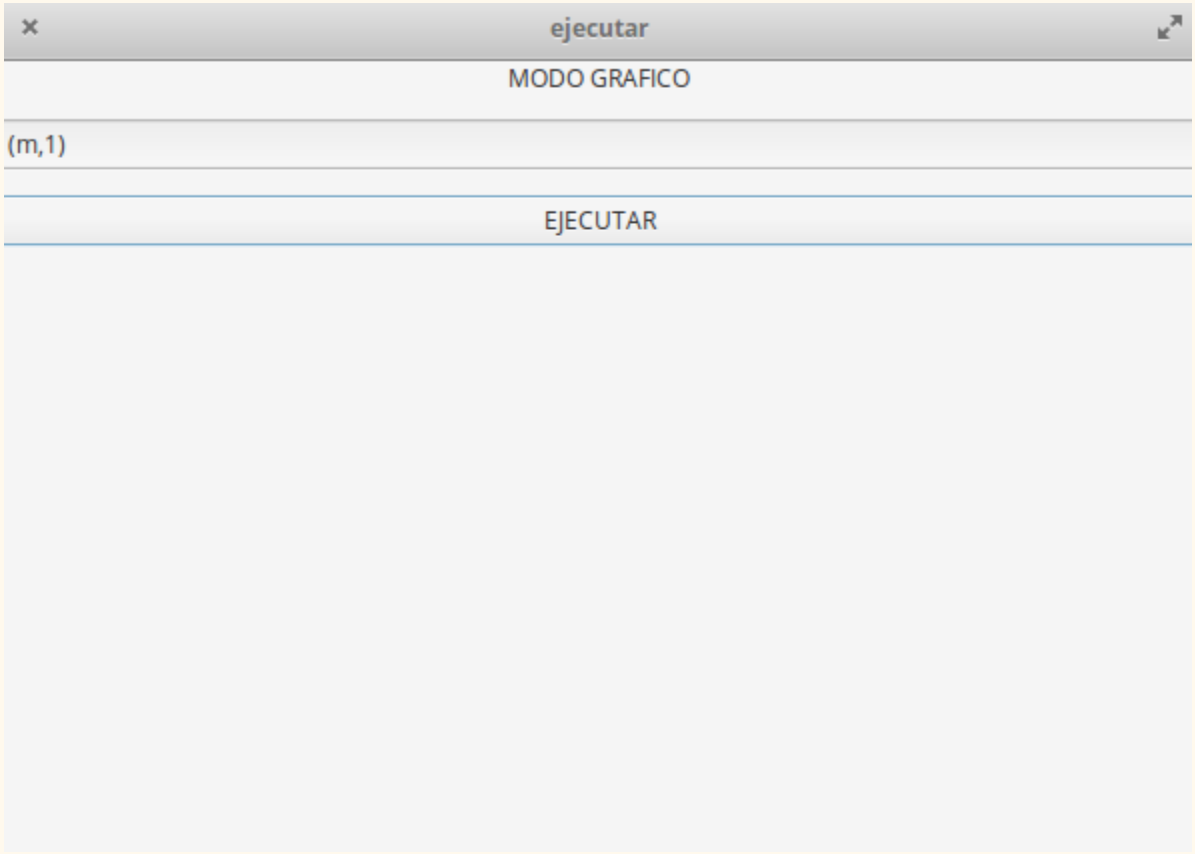
MENU

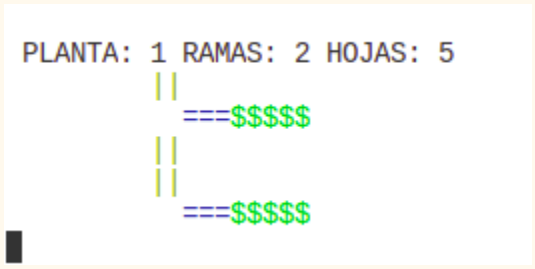


Opcion1:

Para la parte de la interfaz se utilizó Glade para crear una pantalla y recibiera instrucciones.

Se captura instrucción ingresada por parte del usuario en la caja de texto cuando le da click en el botón ejecutar, seguido de eso el sistema evaluará la información , en el caso de ser mostrar, mostrará en consola la planta dibujada.



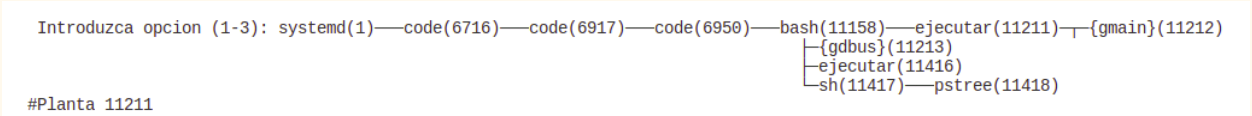


OPCION 2:

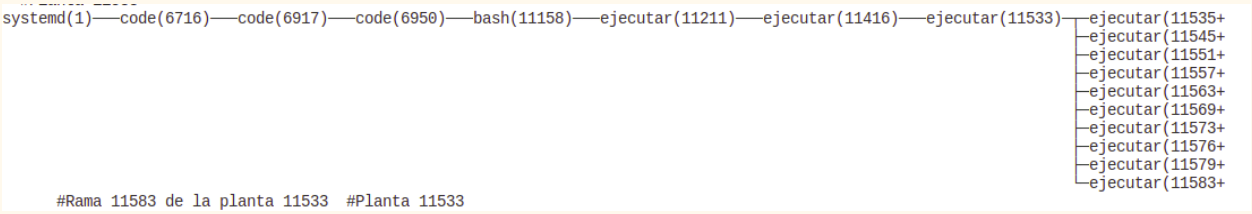
Lee un archivo de texto llamado instrucciones.txt que está ubicado en la misma carpeta del ejecutable.

En caso de que el archivo tenga la opcion de imprimir se mostrara la planta con sus respectivos procesos en un pstree.

Planta 1 0 ramas 0 hojas.

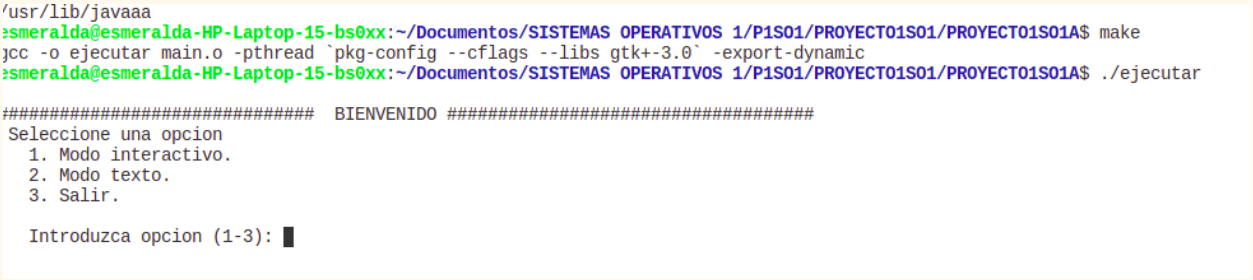


Planta 8 con 5 ramas 0 hojas.



Ejecución de aplicación

El usuario debe de ejecutar los siguientes comandos para que se ejecute la aplicación desde una terminal ubicándose en la carpeta donde se ha guardado el codigo,



Requisitos del sistema:

- SO Linux.
- Instalar Visual Studio Code u otro editor de texto.

https://code.visualstudio.com/download

- Instalar Glade

https://prognotes.net/2015/12/installing-gtk-3-and-glade-development-tools-in-linux/

REPOSITORIO DE CODIGO

https://github.com/CELIAVARGAS/PROYECTO1SO1.git