Taller #8 Métodos Computacionales

Daniel Lozano Gómez
d.lozano343@uniandes.edu.co

September 2018

1. Sección #2: Makefile

En el siguiente taller se practicaran algunas de las habilidades aprendidas para la escritura de archivos makefile. Para ello será provisto con los siguientes códigos:

- **datos.cpp**
- **funcion.cpp**
- **plotea.py**

El primero de estos archivos es un código en C++ que genera un números N de puntos equiespaciados entre 0 y 2π (N es ingresado por el usuario) e imprime el resultado en un archivo llamado **output.txt**.El segundo lee el archivo output.txt y genera otro archivo llamado **sin.txt** que contiene los datos x y $sin(x)$. Finalmente, el tercero es un código en python que, dado el archivo sin.txt, genera una imagen llamada **sin.png**, este código NO muestra la imagen.

1. (5 Puntos) **Separe los puntos del taller por comentarios**.
2. Genera un archivo makefile que realice las siguientes acciones:
   - (10 Puntos) Borrar todos los archivos .txt y .png al ejecutar el comando : make clean.
   - (10 Puntos) Compilar el archivo datos.cpp: $g++ -o datos\_x\ datos.cpp$.
   - (10 Puntos) Compilar el archivo funcion.cpp: g++ -o sin_x funcion.cpp.
   - (15 Puntos) Ejecuta *datos_x* para generar el archivo output.txt.
   - (15 Puntos) Ejecuta *sin_x* para generar el archivo sin.txt.
   - (20 Puntos) Generar la figura sin.png: Esta figura depende del código de python como del archivo sin.txt.
   - (15 Puntos) Abrir la figura sin.png en caso de que todo este actualizado (para abrir desde terminal se usa el comando display).

La calificación de este make file depende de 1) Si este comando funciona al ser ejecutado desde 0. 2) Si este comando funciona bien ante actualizaciones de los archivos.

#include < iostream >	Básico para imprimir
#include < cmath >	Funciones matematicas típicas
#include < cstdlib >	Para números random
#include < ctime >	Para tomar el tiempo actual
#include < fstream >	Para trabajar con archivos