Clase 12. Pipes y Librerias estándar.

Librerias Estándar: stdlib.h

stdlib.h, variables

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm

Variable	Description
size_t	En general es un entero sin signo y puede almacenar el tamaño del objeto mas grande posible que puede ser reservado.
wchar_t	Es un caracter **wide**, usualmente del tamaño de un entero, sirve para codificar mas símbolos que los del codigo ASCII.
div_t	Estructura para almacenar el resultado y residuo de una división realizada por la función *div*.
ldiv_t	Similar al anterior pero con tipos long.

```
1 typedef struct {
    int quot;
3   int rem;
} div_t;
5 typedef struct {
    long int quot;
7   long int rem;
} ldiv_t;
```

stdlib.h, constantes (macros)

- NULL El valor de un apuntador constante que no apunta a un lugar valido de la RAM, usualmente (void *)0.
- EXIT_FAILURE es un macro que indica que la función terminó con algún error.
- EXIT_SUCCESS es un macro que indica que la función terminó exitosamente.
- RAND_MAX Máximo valor que regresa la función *rand*.
- MB_CUR_MAX Es el tamaño maximo que puede tener un caracter multibyte. Los caracteres que no son ASCII se pueden representar como wchar o como caracteres multibyte.

stdlib.h, funciones

- **double atof(const char *str)**. string a double (Ya visto).
- **int atoi(const char *str)**. String a entero (Ya visto).
- **long int atol(const char *str)**. string a long (Ya visto).
- **double strtod(const char *str, char **endptr)**. Convierte la parte inicial de una cadena de caracteres en un número double. Si la cadena contiene mas texto, endptr se actualiza a la posición desde donde comienza el texto (ver ejemplo).
- **long int strtol(const char *str, char **endptr, int base)**. Similar al anterior para un número entero en base **base**.
- **unsigned long int strtoul(const char *str, char **endptr, int base)**. Versión sin signo del anterior.
- **void *calloc(size_t nitems, size_t size)**. Requiere memoria dinámica para **nitems** de tamaño **size**. La diferencia con malloc es que calloc si pone la memoria a cero.
- **void free(void *ptr)**. Libera memoria (Ya visto).
- **void *malloc(size_t size)**. Requiere memoria (ya visto).
- **void *realloc(void *ptr, size_t size)**. Intenta requerir memoria a partir de un bloque requerido anteriormente con malloc o calloc.(Ver ejemplo).

stdlib.h, ejemplos strtod y strtol


```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
  char cadena[]={"-3.19653e-4 -110101 Fin"};
  char *next, *fin;
  double num=strtod(cadena,&fin);
  long int num2;
  printf("Tamanio de la cadena=%d\n",sizeof(cadena));
  printf("Numero extraido=%lf\n", num);
  printf("Fin de la cadena 1=%s \n",fin);
  next=++fin;
  num2=strtol(next,&fin,2);
  printf("Numero entero=%ld\n", num2);
  printf("Fin de la cadena 2=%s \n",fin);
return 0;
}
```

```
Tamanio de la cadena=24
Numero extraido=-0.000320
Fin de la cadena 1= -110101 Fin
Numero entero=-53
Fin de la cadena 2= Fin
```

stdlib.h, ejemplos, calloc y realloc


```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
  int n=10;
  char *str=calloc(n,sizeof(char));
  char *tmp;
  for (int i=0; i<n; i++)
    printf("%c",str[i]);
    printf("\n");
  n=20; tmp=str;
  str=realloc(str,n*sizeof(char));
  printf("tmp =%p str = %p \n",tmp, str);
  free(str);
return 0;
}
```

realloc: Si preserva el contenido de la memoria!

```
tmp =0x51e0040 str = 0x51e0090
==19087==
==19087== HEAP SUMMARY:
==19087==     in use at exit: 0 bytes in 0 blocks
==19087==   total heap usage: 2 allocs, 2 frees, 30 bytes allocate
==19087==
==19087== All heap blocks were freed -- no leaks are possible
```

stdlib.h, funciones

- **void abort(void)**. Causa una terminación anormal del programa.
- **int atexit(void (*func)(void))**. Hace que se llame la función **func** cuando el programa termina.
- **void exit(int status)**. Termina normalmente el programa.
- **char *getenv(const char *name)**. Obtiene el valor de una variable de entorno.
- **int system(const char *string)**. Ejecuta un comando (o programa) del sistema.
- **void *bsearch(const void *key, const void *base, size_t nitems, size_t size, int (*compar)(const void *, const void *))**. Realiza una búsqueda binaria. (Ver ejemplo)
- **void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*))**. Ordenamiento con quick sort. (Ver ejemplo)

stdlib.h, ejemplos, abort

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm


```c
#include <stdio.h>
#include <stdlib.h>
int main (){
   FILE *fp;
   printf("Going to open nofile.txt\n");
   fp = fopen( "nofile.txt","r" );
   if(fp == NULL){
      printf("Going to abort the program\n");
      abort();
   }
   printf("Going to close nofile.txt\n");
   fclose(fp);
   return(0);
}
```

```
Going to open nofile.txt
Going to abort the program
```

stdlib.h, ejemplos, atexit

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm


```c
#include <stdio.h>
#include <stdlib.h>
void functionA (){
   printf("This is functionA\n");
}

int main (){
   /* register the termination function */
   atexit(functionA );
   printf("Starting  main program...\n");
   printf("Exiting main program...\n");
   return(0);
}
```

```
Starting main program...
Exiting main program...
This is functionA
```

stdlib.h, ejemplos, getenv

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm


```c
#include <stdio.h>
#include <stdlib.h>
int main ()
{
   printf("PATH : %s\n", getenv("PATH"));
   printf("HOME : %s\n", getenv("HOME"));
   return(0);
}
```

```
PATH : /usr/lib64/mpi/gcc/openmpi/bin:/home/ivvan/bin
HOME : /home/ivvan
ROOT : (null)
```

stdlib.h, ejemplos, bsearch

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm


```c
#include <stdio.h>
#include <stdlib.h>
int cmpfunc(const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}
int values[] = { 5, 20, 29, 32, 63 };
int main (){
   int *item;
   int key = 32;
   /* using bsearch() to find value 32 in the array */
   item = (int*) bsearch (&key, values, 5, sizeof (int)
    , cmpfunc);
   if( item != NULL ) {
      printf("Found item = %d\n", *item);
   }
   else    {
      printf("Item = %d could not be found\n", *item);
   }
   return(0);
}
```

stdlib.h, ejemplos, qsort

http://www.tutorialspoint.com/c_standard_library/stdlib_h.htm


```c
#include <stdio.h>
#include <stdlib.h>
int values[] = { 88, 56, 100, 2, 25 };
int cmpfunc (const void * a, const void * b){
   return ( *(int*)a - *(int*)b );
}

int main()
{
   int n;
   printf("Before sorting the list is: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }
   qsort(values, 5, sizeof(int), cmpfunc);
   printf("\nAfter sorting the list is: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }
  return(0);
}
```

Tarea 7

prog7.1 Escriba un programa que:

1. Defina un tipo estructura que contiene dos miembros double.
2. Requiere memoria dinámica para un arreglo de n (parámetro) de usuario elementos del tipo definido anteriormente.
3. Inserta números aleatorios reales (double) entre 0 y 100 en cada uno de los miembros del arrglo de la estructura.
4. Tiene una funcion *compara(a,b)* (como se requiere para qsort y para bsearch) que devuelve un entero positivo si el mayor elemento de a es mayor que el mayor elemento de b, negativo en caso contrario y 0 si son iguales.
5. Utilice la función qsort para ordenar su arreglo de estructuras.
6. Pida al usuario (o lea de archivo) un número entero entre 0 y 100. Utilizando *bsearch* verifique si la parte entera del elemento mayor de alguna de las estructuras del arreglo es igual que el número dado por el usuario imprima que se ha encontrado (o que no).

stdlib.h, funciones

- **int abs(int x)**. Regresa el valor absoluto de x.
- **div_t div(int numer, int denom)**. Divide dos numeros enteros y devuelve el resultado y residuo.
- **long int labs(long int x)**. Valor absoluto de un long.
- **ldiv_t ldiv(long int numer, long int denom)**. División de un long.
- **int rand(void)**. Genera números aleatorios.
- **void srand(unsigned int seed)**. Fija la semilla de generador de aleatorios.
- **int mblen(const char *str, size_t n)**. Devuelve el largo de un char multibyte.
- **size_t mbstowcs(wchar_t *dest, const char *src, size_t n);**. Convierte un arreglo de char multibyte en un arreglo widechar.
- **int mbtowc(whcar_t *pwc, const char *str, size_t n)**.Convierte un multibyte a n wide caracter.
- **size_t wcstombs(char *str, const wchar_t *pwcs, size_t n)**. Convierte un arreglo de wide caracters en un multibyte.
- **int wctomb(char *str, wchar_t wchar)**. Convierte un wide a multibyte.

Pipes

Posix System

- **Portable Operating System Interface.**
  Son un conjunto de estándars definidos por la IEEE, en particular, definen una API y un conjunto de shells.

Posix System

- **Portable Operating System Interface.**
  Son un conjunto de estándars definidos por la IEEE, en particular, definen una API y un conjunto de shells.
- De acuerdo con la compatibilidad con el estándar un sistema operativo puede ser parcialmente POSIX o completamente POSIX. Existen versiones de Linux y Unix completamente POSIX, y la mayoría son parcialmente POSIX. Windows no es POSIX aunque han existido versiones de Windows que habilitan interfaces de este tipo.

Posix System

- **Portable Operating System Interface.**
  Son un conjunto de estándars definidos por la IEEE, en particular, definen una API y un conjunto de shells.
- De acuerdo con la compatibilidad con el estándar un sistema operativo puede ser parcialmente POSIX o completamente POSIX. Existen versiones de Linux y Unix completamente POSIX, y la mayoría son parcialmente POSIX. Windows no es POSIX aunque han existido versiones de Windows que habilitan interfaces de este tipo.
- El que un sistema sea POSIX, proveé compatibilidad con otros sistemas POSIX, lo que hace el código mas portable entre estos sistemas aunque a bajo nivel no funcione igual.

Posix System

- **Portable Operating System Interface.**
  Son un conjunto de estándars definidos por la IEEE, en particular, definen una API y un conjunto de shells.
- De acuerdo con la compatibilidad con el estándar un sistema operativo puede ser parcialmente POSIX o completamente POSIX. Existen versiones de Linux y Unix completamente POSIX, y la mayoría son parcialmente POSIX. Windows no es POSIX aunque han existido versiones de Windows que habilitan interfaces de este tipo.
- El que un sistema sea POSIX, proveé compatibilidad con otros sistemas POSIX, lo que hace el código mas portable entre estos sistemas aunque a bajo nivel no funcione igual.
- **C POSIX library** Un estándar de librerías de C para sistemas POSIX. Este estándar fue definida al mismo tiempo que el ANSI C. Y es comúnmente implementada junto con la librería ANSI.

Posix System

- Dentro del estándar POSIX están los **pipes**.

Posix System

- Dentro del estándar POSIX están los **pipes**.
- Un **pipe** básicamente es un archivo compartido entre dos programas, la salida de uno puede ser la entrada de otro y viceversa. Una función de *alto nivel* para acceder a los pipes es **popen**.

Posix System

- Dentro del estándar POSIX están los **pipes**.
- Un **pipe** básicamente es un archivo compartido entre dos programas, la salida de uno puede ser la entrada de otro y viceversa. Una función de *alto nivel* para acceder a los pipes es **popen**.
-

```
1 #include <stdio.h>
  FILE *popen(const char *command, const char *type);
3 int pclose(FILE *stream);
```

stdio.h, ejemplo 1, popen

Note que: este programa no incluye a stdlib.h, y que solo lee y escribe de la entrada y salida estándar.

```c
#include <stdio.h>
int main()
{
  char input[256];
  scanf("%s",input);
  if(strcmp(input,"fin")!=0){
    while(strcmp(input,"fin")!=0){
      printf("Programa 2 lee: %s\n",input);
      scanf("%s",input);
    }
  }else{
    printf("Programa 2 sale: %s\n",input);
  }
  printf("Sale programa 2\n");
return 0;
}
```

stdio.h, ejemplo 2, popen

Note que: este programa incluye a stdlib.h, y lee y escribe al pipe.

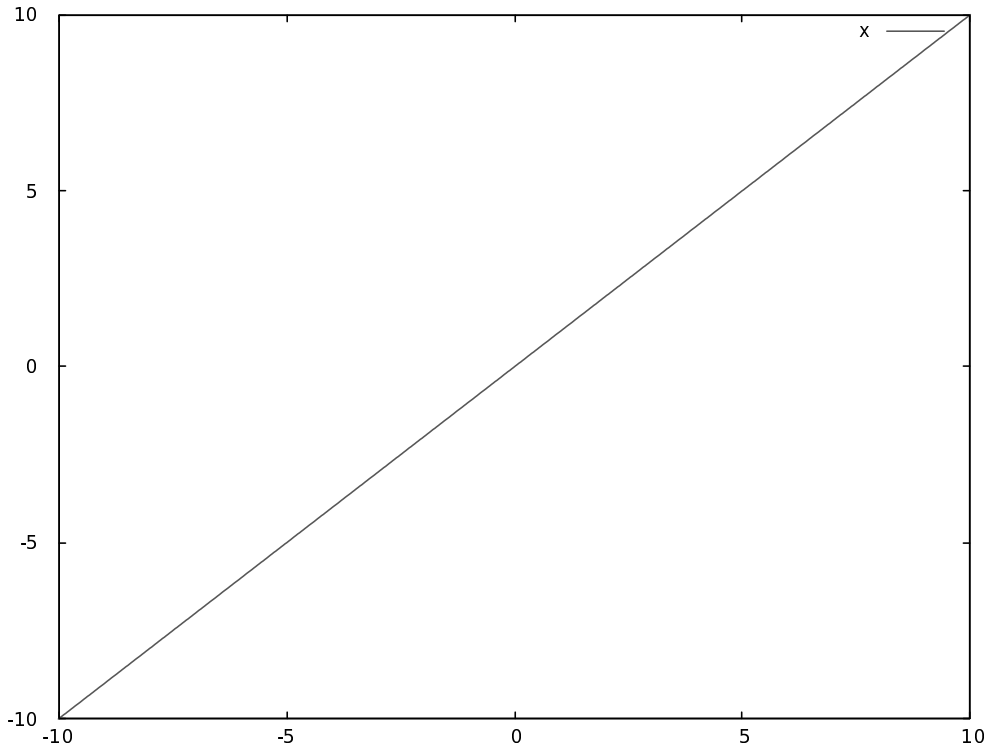
ej06 es el programa anterior.


```c
#include <stdio.h>
int main(){
 FILE *pipeOut=popen("ej06","w");
 if (pipeOut){
   char input[256];
   scanf("%s",input);
   if(strcmp(input,"fin")!=0){
     while(strcmp(input,"fin")!=0){
       fprintf(pipeOut,"%s\n",input); fflush(pipeOut);
       scanf("%s",input);
     }
     fprintf(pipeOut,"%s",input); fflush(pipeOut);
   }else{
     fprintf(pipeOut,"%s",input); fflush(pipeOut);
   }
 }
 pclose(pipeOut);
 printf("Sale programa 1\n");
return 0;
}
```

GnuPlot usando pipes

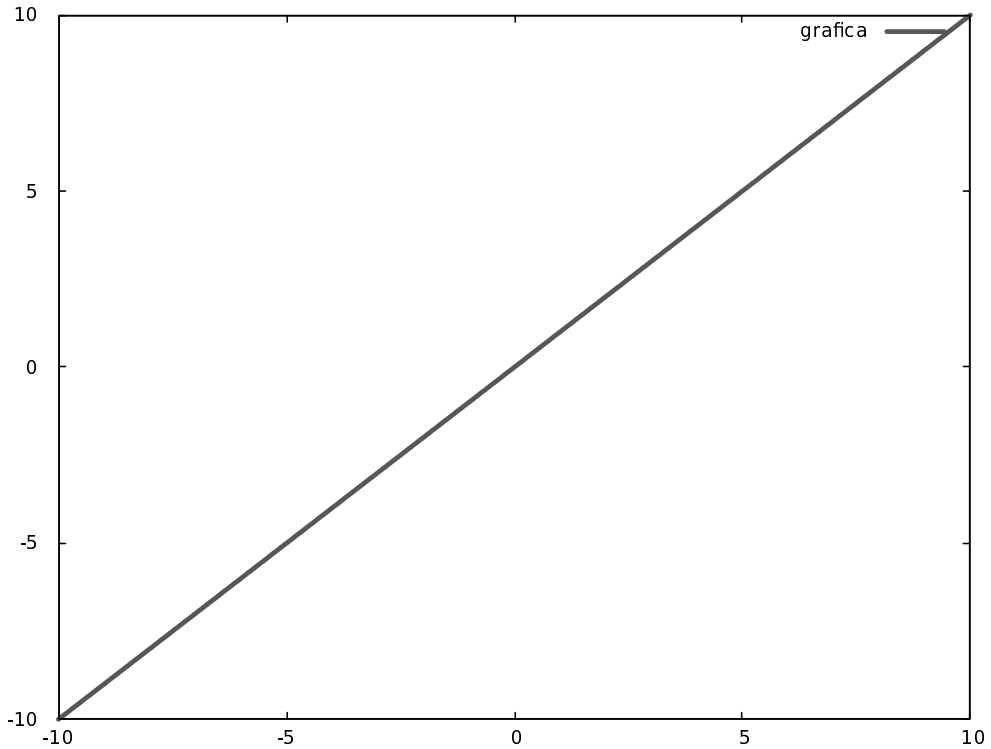
GnuPlot Comandos básicos


```
#rango de las x
set xrange [-10:10]
#rango de las y
set yrange [-10:10]
#grafica y=x
plot x
```



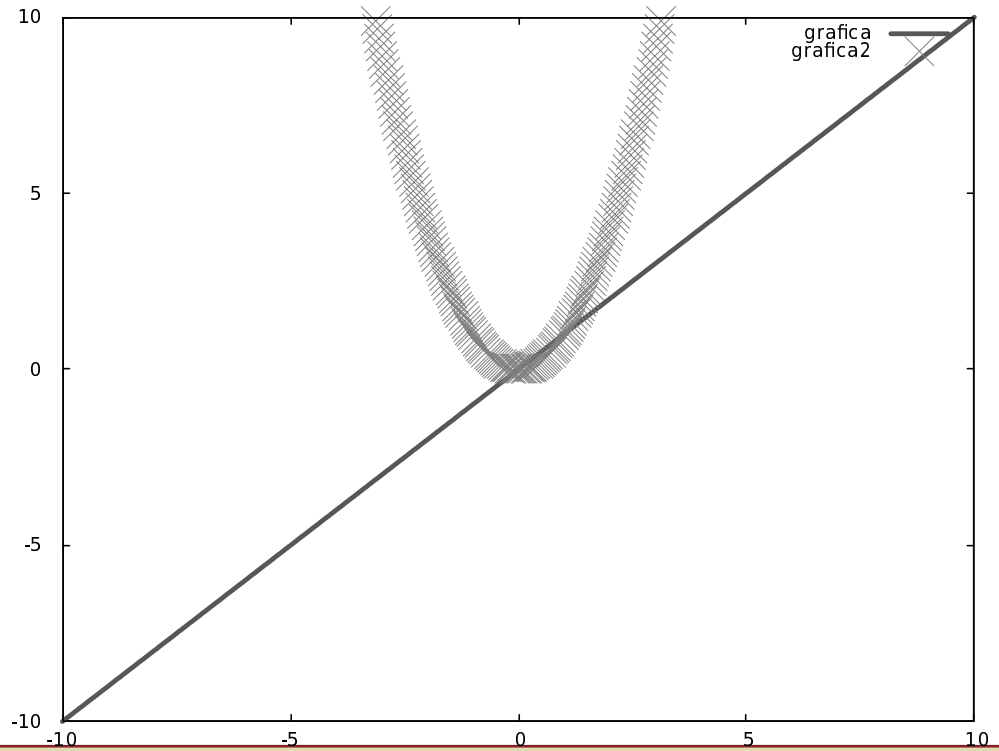
GnuPlot Comandos básicos


```
#w=with l=lines, ti=title lw=linewidth
plot x w l ti "grafica" lw 3
```



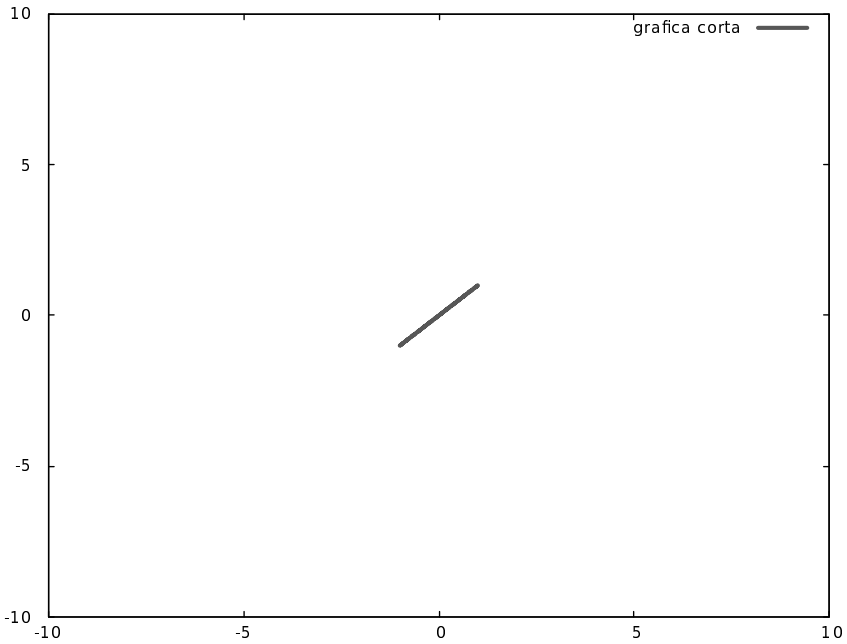
GnuPlot Comandos básicos


```
#Define una funcion
f1(x)=x
#Define una segunda funcion
f2(x)=x**2
#p=points, ps=pointsize
plot f1(x) w l ti "grafica" lw 3, f2(x) w p ti "
    grafica2" ps 3
```



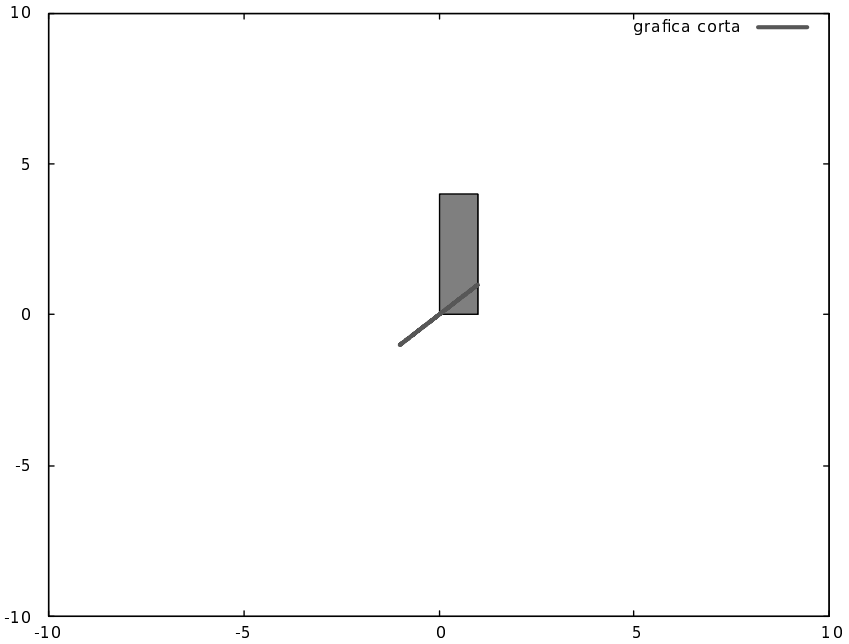
GnuPlot Comandos básicos


```
#La funcion solo esta definida de -1 a 1
f3(x)=x>-1 & x<1 ? x : 1/0
#numero de puntos para graficar
set sample 10000
plot f3(x) w l ti "grafica corta" lw 3
```



GnuPlot Comandos básicos


```
#Define un rectangulo de 0,0 a 1,2, no grafica
set object 1 rect from 0,0 to 1,4 fc lt 2
#grafica la linea y el rectangulo
plot f3(x) w l ti "grafica corta" lw 3
#elimina el rectangulo
unset object 1
#Vuelve a graficar sin rectangulo
plot f3(x) w l ti "grafica corta" lw 3
```



GnuPlot en un pipe


```
#include <stdio.h>
#include <unistd.h> #libreria posix: usleep
int main(){
 FILE *pipeOut=popen("gnuplot --persist","w");
 double theta=0.0,r=5.0;
 if (pipeOut){
    fprintf(pipeOut,"set terminal wxt size 600,600\n");
    fprintf(pipeOut,"set sample 10000\n");
    fprintf(pipeOut,"set xrange [-10:10]\n");
    fprintf(pipeOut,"set yrange [-10:10]\n");
    fprintf(pipeOut,"r=%lf\n",r);
    for (theta=0.0; theta<15.0; theta=theta+0.01){
     fprintf(pipeOut,"theta=%lf\n",theta);
     fprintf(pipeOut,"f(x)= (x>0 & x<(r*cos(theta)
     +0.0001)) || (x<0 & x>(r*cos(theta)+0.0001)) ? x/(
     cos(theta)+0.0001)*(sin(theta)+0.0001): 1/0\n");
     fprintf(pipeOut,"plot f(x) w l lw 3 ti \"x\" \n");
     fflush(pipeOut);
     usleep(100);
    }
 }
 pclose(pipeOut);
return 0;
}
```

Tarea 7

BONUS Escriba un programa que simule el comportamiento de un mecanismo biela-manivela-corredera utilizando pipes y gnuplot. Las dimensiones de los eslabones son parámetro de usuario.