Clase 14. Propuestas de proyectos y librerias estándar.

Proyectos

Requisitos

- En cuanto a programación utilizar:
  - Funciones.
  - Tipos de definidos por el usuario.
  - Estructuras.
  - Memoria dinámica.
  - Lectura y escritura de archivos.

Proyecto 1, descripción

Considere la siguiente figura:

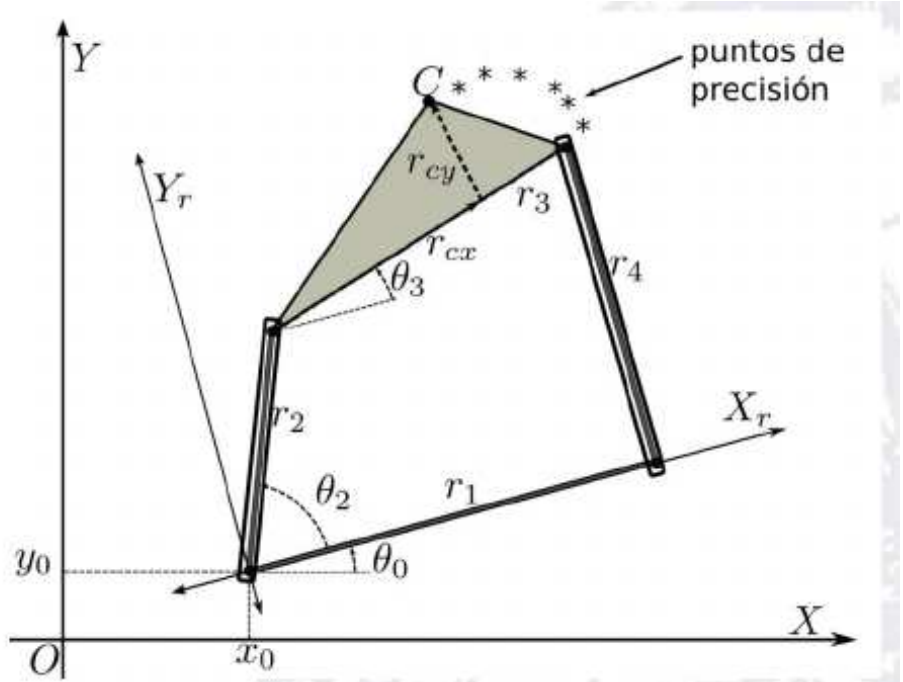


Figura 1. Esquema cinemático de un mecanismo plano de cuatro barras.

El objetivo es que el organo terminal C pase o se aproxime lo más posible a cada uno de los puntos de precisión. El par motriz es el relativo a θ_2, dado este valor, la longitud de los eslabones $r_2, r_3, r_4, r_{cx}, r_{cy}$, las coordenadas x_0, y_0 y rotación θ_0 del origen del sistema de coordenadas relativas, es posible determinar las coordenadas del punto C.

Proyecto 1

Se tienen n puntos de precisión. Y n puntos correspondientes donde se medira la cercanía con los puntos de precisión en C. A estos últimos les llamaremos $C = [Cx_i, Cy_i]$ para $i = 1, 2, .., n$. Siendo n el número de puntos de precisión.

NOTA: La verificación de las ecuaciones es responsabilidad del estudiante. Las coordenadas C cercanas a los puntos de precisión pueden ser determinadas como:

Con respecto al sistema de coordenadas relativo a x_0, y_0 y θ_0.

$$\vec{C^0} = \vec{r_2} + \vec{r_{cx}} + \vec{r_{cy}} \quad (1)$$

$$C_x^0 = r_2 cos(\theta_2) + r_{cx} cos(\theta_3) + r_{cy}(-sin(\theta_3)) \quad (2)$$

$$C_y^0 = r_2 sin(\theta_2) + r_{cx} sin(\theta_3) + r_{cy}(cos(\theta_3)) \quad (3)$$

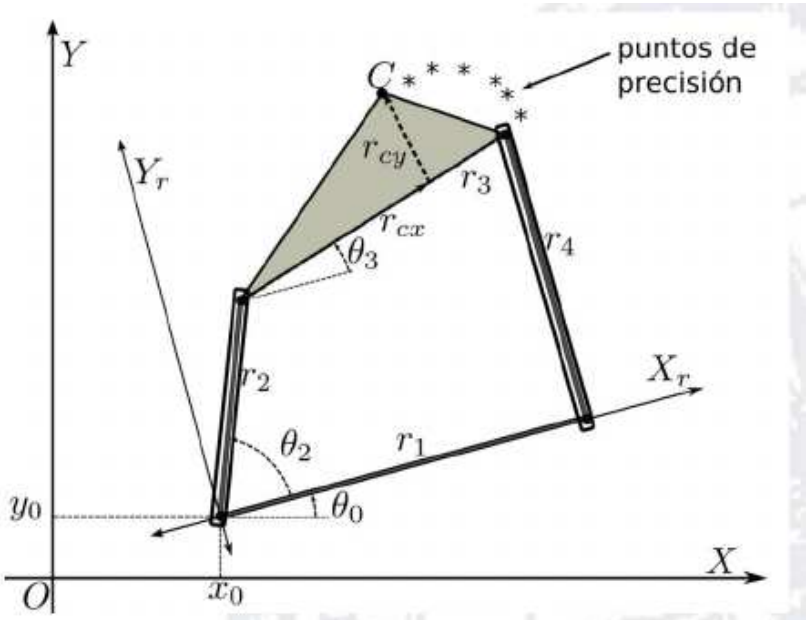


Figura 1. Esquema cinemático de un mecanismo plano de cuatro barras.

Proyecto 1

La ecuación de lazo cerrado para el mecanismo es:

$$\vec{r_2} + \vec{r_3} - \vec{r_4} - \vec{r_1} = 0 \quad (4$$

$$r_2 cos(\theta_2) + r_3 cos(\theta_3) - r_4 cos(\theta_4) - r_1 = 0 \quad (5$$

$$r_2 sin(\theta_2) + r_3 sin(\theta_3) - r_4 sin(\theta_4) = 0 \quad (6$$

$$k_1 cos(\theta_3) + k_2 cos(\theta_2) + k_3 = cos(\theta_2 - \theta_3) \quad (7$$

Para $k_1 = r_1/r_2$, $k_2 = r_1/r_2$, $k_3 = (r_4^2 - r_1^2 - r_2^2 - r_3^2)/(2r_2r_3)$.

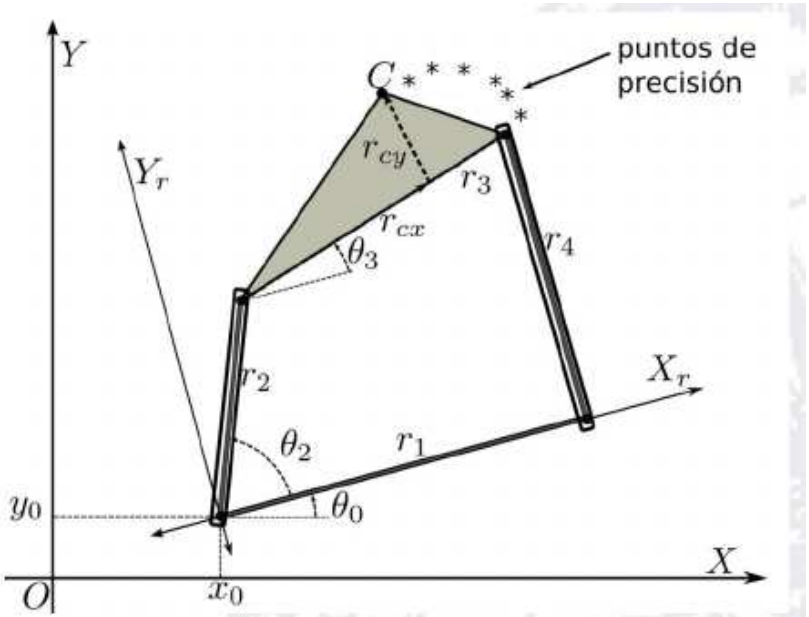


Figura 1. Esquema cinemático de un mecanismo plano de cuatro barras.

de lo anterior:

$$\theta_3 = 2tan\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right), \quad (8)$$

Para $a = cos(\theta_2) - k_1 + k_2 cos(\theta_2), b = -2sin(\theta_2)$ y $c = k_1 + (k_2 - 1)cos(\theta_2) + k_3$.

Raices complejas es que el mecanismo no se conecta, raices reales usando el signo positivo: configuración cruzada, signo negativo: configuración abierta.

Proyecto 1

Todo lo anterior fue tomando como referencia los ejes relativos, en los ejes absolutos podemos calcular el valor de C como:

$$C_x = x_0 + C_x^0 cos(\theta_0) - C_y sin(\theta_0)$$
$$C_y = y_0 + C_x^0 sin(\theta_0) + C_y sin(\theta_0) \quad (10)$$

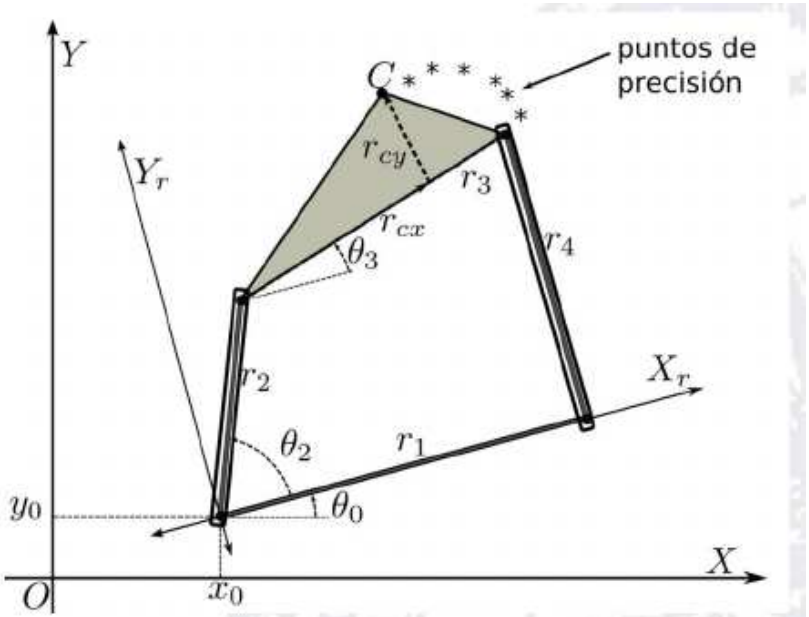
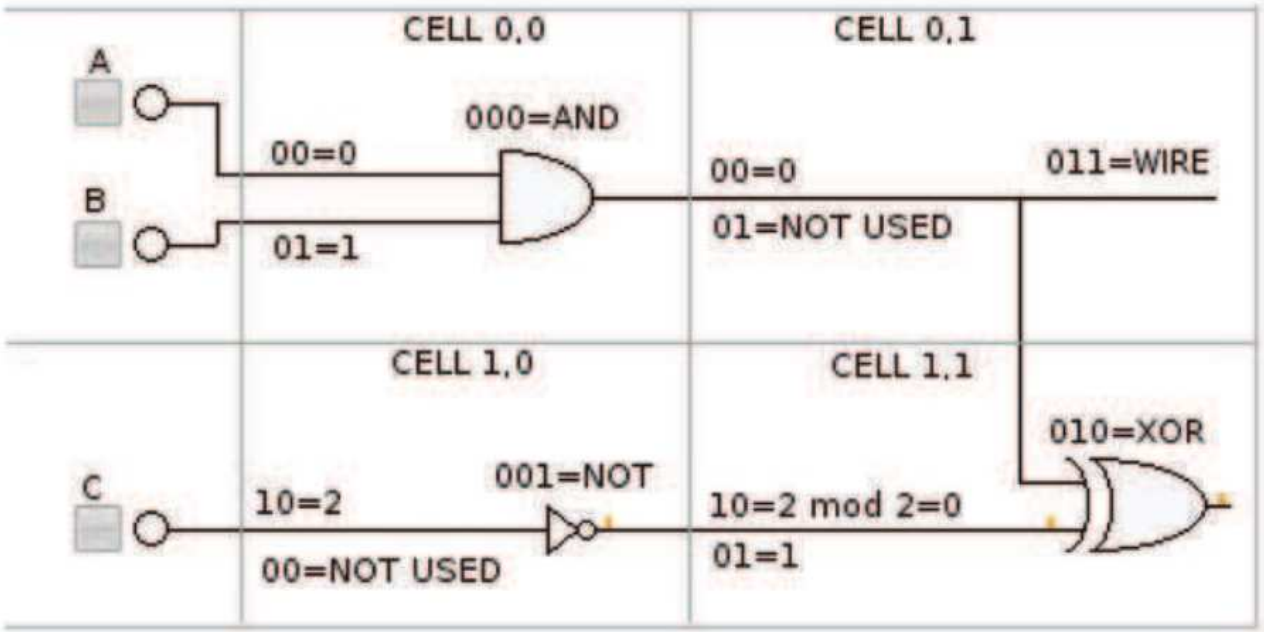


Figura 1. Esquema cinemático de un mecanismo plano de cuatro barras.

Proyecto 2,descripción

Se tiene un circuito combinatorio. Este circuito puede ser representado por un arreglo. Donde cada elemento del arreglo tiene 3 elementos: a) la compuerta que se usa en el circuito y las conexiones del mismo.



	CELL 0,0	CELL 1,0	CELL 0,1	CELL 1,1
STRING=	0001000	1000001	0001011	1001010

Proyecto 2

- Se tiene una tabla de verdad, las entradas A, B, C, y D, indican los valores que entran a las primeras compuertas del circuito. Los valores S_0 y S_1, son los valores de salida del circuito.

Proyecto 2

- Se tiene una tabla de verdad, las entradas A, B, C, y D, indican los valores que entran a las primeras compuertas del circuito. Los valores S_0 y S_1, son los valores de salida del circuito.
- Es posible que dado un circuito y una tabla de verdad, los valores no se cumplan, entonces la tabla es lo equivalente a los puntos de precisión del mecanismo, y la salida del circuito es lo equivalente al punto C.

Experiment 4					
A	**B**	**C**	**D**	**S_0**	**S_1**
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	1	0	1

Proyecto 1

1. Proponer una forma de almacenar y leer 1 mecanismo de n barras rigidas, que solo tiene artículaciones fijas, motrices, o con rotación en el plano. Es decir, deben de poder leer desde un archivo (de texto o binario) el mecanismo y *armarlo* en gnuplot o en lo que quieran, mostrando cuales son articulaciones fijas, cuales rotan y cuales son motrices (no es necesario simular el movimiento, solo leer un mecanismo cualquiera del archivo y representarlo).

Proyecto 1

1. Proponer una forma de almacenar y leer 1 mecanismo de n barras rigidas, que solo tiene artículaciones fijas, motrices, o con rotación en el plano. Es decir, deben de poder leer desde un archivo (de texto o binario) el mecanismo y *armarlo* en gnuplot o en lo que quieran, mostrando cuales son articulaciones fijas, cuales rotan y cuales son motrices (no es necesario simular el movimiento, solo leer un mecanismo cualquiera del archivo y representarlo).
2. Para el mecanismo particular que se muestra aqui, datos en la diapositiva 4, se debe de poder simular la posición del punto C para un periodo de tiempo rango del ángulo θ_2, incrementando el valor inicial por $\delta\theta_2$ hasta llegar al valor final.

Proyecto 1

1. Proponer una forma de almacenar y leer 1 mecanismo de n barras rigidas, que solo tiene artículaciones fijas, motrices, o con rotación en el plano. Es decir, deben de poder leer desde un archivo (de texto o binario) el mecanismo y *armarlo* en gnuplot o en lo que quieran, mostrando cuales son articulaciones fijas, cuales rotan y cuales son motrices (no es necesario simular el movimiento, solo leer un mecanismo cualquiera del archivo y representarlo).
2. Para el mecanismo particular que se muestra aqui, datos en la diapositiva 4, se debe de poder simular la posición del punto C para un periodo de tiempo rango del ángulo θ_2, incrementando el valor inicial por $\delta\theta_2$ hasta llegar al valor final.
3. A partir de la simulación se puede medir la diferencia entre los puntos de la simulación y el mas cercano de los puntos de precisión, sin repetir el mismo punto de precisión ni el mismo punto de la trayectoria.

Proyecto 1

1. Proponer una forma de almacenar y leer 1 mecanismo de n barras rigidas, que solo tiene artículaciones fijas, motrices, o con rotación en el plano. Es decir, deben de poder leer desde un archivo (de texto o binario) el mecanismo y *armarlo* en gnuplot o en lo que quieran, mostrando cuales son articulaciones fijas, cuales rotan y cuales son motrices (no es necesario simular el movimiento, solo leer un mecanismo cualquiera del archivo y representarlo).
2. Para el mecanismo particular que se muestra aqui, datos en la diapositiva 4, se debe de poder simular la posición del punto C para un periodo de tiempo rango del ángulo θ_2, incrementando el valor inicial por $\delta\theta_2$ hasta llegar al valor final.
3. A partir de la simulación se puede medir la diferencia entre los puntos de la simulación y el mas cercano de los puntos de precisión, sin repetir el mismo punto de precisión ni el mismo punto de la trayectoria.
4. La suma de los valores absolutos de las diferencias (entre simulación y puntos de precisión) será una medida de que tan bien el mecanismo cumple con su función.

Proyecto 1

5 Realizar las tareas anteriores. Generar 10000 parámetros aleatorios (parámetros de la diapositiva 4) y devolver el mejor (en un archivo de salida). Ejecutar lo anterior para los casos de prueba de: Cabreara, Simon and Prado (2002), Optimal Synthesis of Mechanisms with Genetic Algorithms. Mechanism and Machine Theory, 37, 1165-1177.

Proyecto 1

5 Realizar las tareas anteriores. Generar 10000 parámetros aleatorios (parámetros de la diapositiva 4) y devolver el mejor (en un archivo de salida). Ejecutar lo anterior para los casos de prueba de: Cabreara, Simon and Prado (2002), Optimal Synthesis of Mechanisms with Genetic Algorithms. Mechanism and Machine Theory, 37, 1165-1177.

6 *NOTA: Pueden entregar mas de un programa, y usar una versión que no grafique sino que sólo calcule y otra versión que muestre el mejor mecanismo encontrado.*

Proyecto 2

1. Proponer una forma de almacenar y leer un circuito combinatorio.

Proyecto 2

1. Proponer una forma de almacenar y leer un circuito combinatorio.
2. Simular el comportamiento de un circuito dado por un archivo con el formato que hayan definido para leerlo y mostrarlo en una gráfica (gnuplot).

Proyecto 2

1. Proponer una forma de almacenar y leer un circuito combinatorio.
2. Simular el comportamiento de un circuito dado por un archivo con el formato que hayan definido para leerlo y mostrarlo en una gráfica (gnuplot).
3. A partir de la simulación se puede calcular la diferencia entre una tabla de verdad (dada en un archivo), y la simulación.

Proyecto 2

1. Proponer una forma de almacenar y leer un circuito combinatorio.
2. Simular el comportamiento de un circuito dado por un archivo con el formato que hayan definido para leerlo y mostrarlo en una gráfica (gnuplot).
3. A partir de la simulación se puede calcular la diferencia entre una tabla de verdad (dada en un archivo), y la simulación.
4. Para 10000 circuitos generados aleatoriamente calcule la diferencia entre la tabla de verdad y la simulación, almacene el mejor circuito. Usar 3 casos de prueba (al menos 1 con mas de 1 salida) de: Use of evolutionary Techniques to Automate the design of combinational Circuits. Coello, Christiansen, Hernández.
5. *NOTA: Pueden entregar mas de un programa, y usar una versión que no grafique sino que sólo calcule y otra versión que muestre el mejor circuito encontrado.*

procesos y forks

Procesos

- Cada instancia que se ejecuta de un programa se le conoce como un proceso.
- Un proceso tiene un ID único en el sistema (PID).
- Cada proceso tiene su propio espacion de memoria: stack, data, code, y **heap**. (las direcciones de memoria pueden o no ser referenciadas igual, es decir la dirección 0x008 para el proceso 1 puede ser diferente a lo que es la dirección 0x008 para el proceso 2).

fork

La función **fork** de la libreria posix stdio.h se utiliza para crear un nuevo proceso duplicando el proceso existente (desde donde es creado el nuevo proceso).

- Al proceso que inicial se le llama proceso padre, y al generado proceso hijo.
- Cada uno tiene su propio PID.
- Los medidores del uso del CPU (función clock) se reinician en el proceso hijo.
- El hijo tiene copia de *todos* los datos del padre, hasta antes de la llamada al fork(). Muchas veces, el sistema habilita la copia hasta que se requiere (que es modificada por ambos).

fork, ejemplo 1

La función devuelve un tipo *pid_t* y está declarada en unistd.h. Es -1 en el padre cuando falla, el PID del hijo cuando es exitosa y 0 en el hijo cuando es exitosa.

```
1  #include <stdio.h>
   #include <unistd.h>
3  int main(int argc, char*argv[]){
       int pidP, pidH;
5      pidH = fork();
       pidP=getpid();
7      if(pidH==0){
   printf(" Proceso hijo con PID=%d\n", pidP);
9      }else if(pidH > 0){
           printf("Proceso padre con PID %d creo hijo con
      PID= %d\n", pidP, pidH);
11     }
   }
```

Salida:

```
Proceso padre con PID 17528 creo hijo con PID= 17529
Proceso hijo con PID=17529
```

fork, ejemplo 2


```c
#include <stdio.h>
#include <unistd.h>
int main(int argc, char*argv[])
{
    int pidP, pidH;
    int valor=153982;
    pidH = fork();
    pidP=getpid();
    if(pidH==0){
printf(" Proceso hijo dir valor=%p\n", &valor);
    }else if(pidH > 0){
        printf("Proceso padre dir valor=%p\n", &valor);
valor=5;
    }
    usleep(100);
    if (pidH==0)
      printf(" Proceso hijo valor=%d\n", valor);
return 0;
}
```

fork, ejemplo 2


```
1  #include <stdio.h>
   #include <unistd.h>
3  int main(int argc, char*argv[])
   {
5      int pidP, pidH;
       int valor=153982;
7      pidH = fork();
       pidP=getpid();
9      if(pidH==0){
printf(" Proceso hijo dir valor=%p\n", &valor);
11     }else if(pidH > 0){
           printf("Proceso padre dir valor=%p\n", &valor);
13 valor=5;
       }
15     usleep(100);
       if (pidH==0)
17       printf(" Proceso hijo valor=%d\n", valor);
   return 0;
19 }
```

Salida:

```
Proceso padre dir valor=0x7fff867769e4
Proceso hijo dir valor=0x7fff867769e4
Proceso hijo valor=153982
```

fork, ejemplo 3


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char*argv[]){
 int pidP, pidH;
 int *valor=malloc(3*sizeof(int));
 for (int i=0; i<3; i++) valor[i]=i;
 pidP=getpid();
 pidH = fork();
 if(pidH==0){
   for (int i=0; i<3; i++)
    printf(" Proceso hijo dir valor=%d\n", valor[i]);
   printf("Dir valor=%p\n",valor);
  free(valor);
 }else if(pidH > 0){
   for (int i=0; i<3; i++)
    printf("Proceso padre dir valor=%d\n", valor[i]);
   printf("Dir valor padre=%p\n",valor);
   free(valor);
 }
 if (pidH==0){
     printf("Proceso padre=%d %d\n",getppid(),pidP);
 }
return 0;
}
```

fork, ejemplo 3

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(){
  int fd[2];  int pids[10];
  char str[128]={"Hablo"};
  pipe(fd);
  pids[0]=fork();
  if(pids[0]!=0){
    close(fd[0]);
    write(fd[1],str,6);
    strcpy(str,"Escucho");
    write(fd[1],str,8);

  }else{
    close(fd[1]);
    read(fd[0],str,6);
    printf("%s\n",str);
    read(fd[0],str,8);
    printf("%s\n",str);
    exit(0);
  }
 return 0;
}
```

Salida:

```
Hablo
Escucho
```

fork, ejemplo 4


```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main()
{
 int fds[10][2];
 int pids[10]; pids[0]=100;
 pids[1]=100;
 char str1[128]={"process1"};
 char str2[128]={"process2"};
 pipe(fds[0]);
 pipe(fds[1]);
 pids[0]=fork();
 pids[1]=fork();
 if(pids[0]!=0 && pids[1]!=0)
    {
  close(fds[0][0]);
  close(fds[1][0]);
  write(fds[0][1],str1,9);
  write(fds[1][1],str2,9);
 }
```

```c
else if (pids[0]==0){
  close(fds[0][1]);
  read(fds[0][0],str2,9);
  printf("0pid0=%d pid1=%d
    mpid=%d\n",pids[0],pids
    [1],getpid());
  exit(0);
 }else if (pids[1]==0){
  close(fds[1][1]);
  read(fds[1][0],str1,9);
  printf("1pid0=%d pid1=%d
    mpid=%d\n",pids[0],pids
    [1],getpid());
  exit(0);
 }
return 0;
}
```

```
1pid0=4498 pid1=0 mpid=4499
0pid0=0 pid1=4500 mpid=4498
0pid0=0 pid1=0 mpid=4500
```

fork, ejemplo 5


```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(){
 int fds[10][2];
 int pids[10]; pids[0]=100;
 pids[1]=100;
 char str1[128]={"padre"};
 pipe(fds[0]);
 pipe(fds[1]);
 pids[0]=fork();
 if(pids[0]!=0){
  close(fds[0][0]);
  close(fds[1][1]);
  write(fds[0][1],str1,9);
  printf("Padre dice:%s\n",
    str1);
  read(fds[1][0],str1,9);
  printf("Padre recibe:%s\n",
    str1);
 }
 else if (pids[0]==0){
    close(fds[0][1]);
    close(fds[1][0]);
    read(fds[0][0],str1,9);
    printf("Hijo leyo= %s\n",
     str1);
    strcpy(str1,"algo");
    write(fds[1][1],str1,9);
    exit(0);
 }
 return 0;
}
```

```
Padre dice:padre
Padre recibe:  algo
Hijo leyo= padre
```

fork, ejemplo 6


```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(){
int fds[10][2];
int pids[10]; pids[0]=100;
pids[1]=100;
char str1[128]={"padre"};
pipe(fds[0]);
pipe(fds[1]);
pids[0]=fork();
if(pids[0]!=0){
 close(fds[0][0]);
 close(fds[1][1]);
 write(fds[0][1],str1,9);
 printf("Padre dice:%s\n",str1);
 if(read(fds[1][1],str1,9)<=0)
  perror("Read:");
 printf("Padre recibe:%s\n",str1);
}
```

```c
else if (pids[0]==0){
 close(fds[0][1]);
 close(fds[1][0]);
 read(fds[0][0],str1,9)
    ;
 printf("Hijo leyo= %s\
    n",str1);
 strcpy(str1,"algo");
 write(fds[1][1],str1
    ,9);
 exit(0);
}
 return 0;
}
```

```
Padre dice:padre
Hijo leyo= padre
Read:: Bad file
descriptor
Padre recibe:padre
```

fork, ejemplo 7


```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
int main(){
  int pids[10];
  for (int i=0; i<10; i++) pids[i]=i;
  pids[1]=fork();
  if (pids[1]!=0) pids[2]=fork();
  printf("Mi PID=%d\n",getpid());
  usleep(10000);
  if (pids[1]==0)
     printf("0 Hijo=%d Padre=%d\n",getpid(),getppid());
  if (pids[2]==0)
     printf("1 Hijo=%d Padre=%d\n",getpid(),getppid());
 return 0;
}
```

```
Mi PID=5656
Mi PID=5655
Mi PID=5657
0 Hijo=5656 Padre=5655
1 Hijo=5657 Padre=5655
```

Tarea 8

Escriba un programa que reciba como **argumento** desde la consola. El nombre de entrada de un archivo de texto, realice un fork para crear un proceso hijo, envie el contenido del archivo (sin importar su longitud, investigue la función *write* y *read*) al proceso hijo e imprima el contenido a la salida estándar. Si un error ocurre reportelo por medio de perror.

BONUS del proyecto

BONUS del proyecto (vale 2 BONUS de tarea).

- Levante n-1 procesos hijos (n procesos totales).
- Envie una 10000/n de las configuraciones propuestas a cada proceso por medio de un pipe, y por medio de otro pipe obtenga el resultado de la evaluación (un número por cada configuración).
- Tome el tiempo que tarda el programa en ejecutarse serial y paralelo (con los n procesos)

EXTRA BONUS(Un BONUS normal): Escriba un programa que haga lo mismo que el anterior pero el número de procesos es un **argumento** de entrada del proceso padre (suponga que el número máximo de procesos será igual o menor a 24).
FECHA de entrega de cual proyecto toman: 13 de octubre de 2014, a las 23:50.