PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE INGENIERÍA

SECCIÓN DE INGENIERÍA INFORMÁTICA
SISTEMAS OPERATIVOS

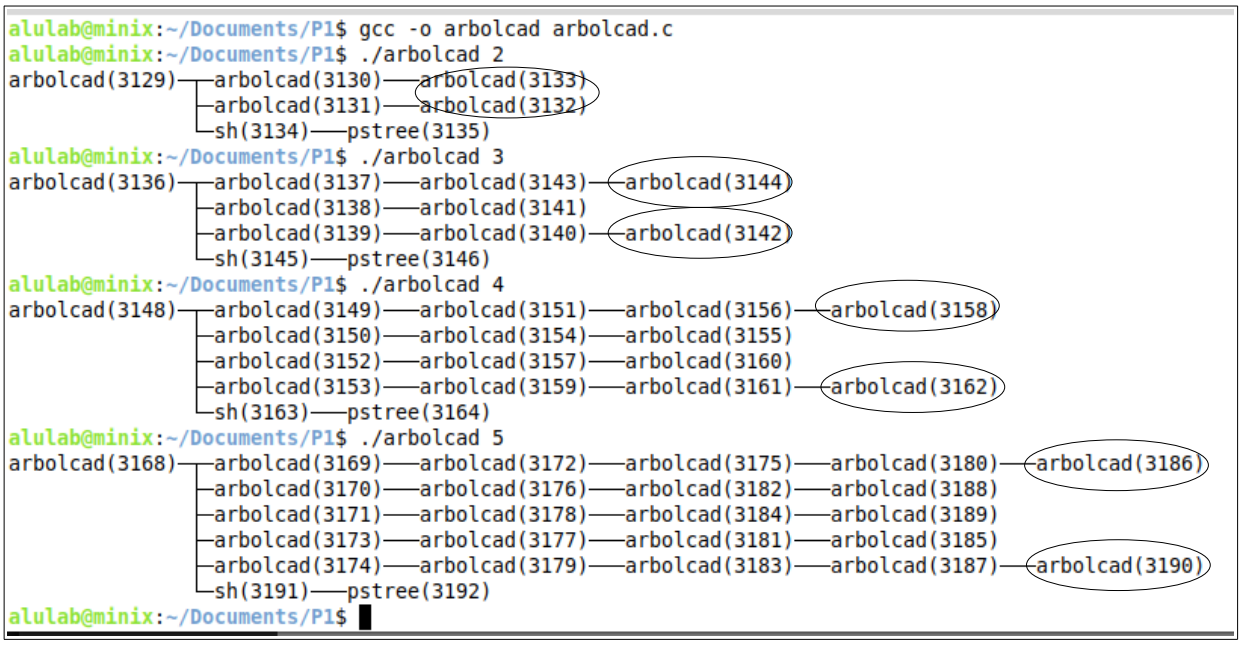
Practica tipo b
Semestre 2020-2

1.- (10 puntos) En esta pregunta usaremos el paquete `pstree`. En la distribución de Linux Mint ya viene instalado. Si usted tiene otra distribución derivada de Debian, es probable que también se encuentre instalada. Si no ocurre así, usted puede instalarla desde una terminal, escribiendo la siguiente orden:

```
sudo apt install pstree
```

La instalación le tomará menos de un minuto.

Compile y ejecute el programa *arbolcad.c* (proporcionado junto con este archivo). El programa necesita un número como argumento. Proporcione números pequeños (máximo 5), para ahorrar los recursos del sistema.



Como puede apreciar este programa crea un árbol de procesos con un patrón determinado.

Se le solicita modificar el programa para lograr que se comunique los procesos que están señalados con una elipse en la figura de arriba. La modificación debe ser tal que se mantenga la estructura del árbol. Para demostrar que la comunicación se ha llevado de forma correcta los procesos en cuestión deben intercambiar sus *pids* y luego imprimen su *pid* y entre paréntesis el *pid* de su pariente lejano.

Su salida debe ser análoga a la siguiente:

```
alulab@minix:~/Documents/P1$ ./arbolcad_sol 2
3343(3342)
3342(3343)
arbolcad_sol(3339)─┬─arbolcad_sol(3340)───arbolcad_sol(3342)
                   ├─arbolcad_sol(3341)───arbolcad_sol(3343)
                   └─sh(3344)───pstree(3345)
alulab@minix:~/Documents/P1$ ./arbolcad_sol 3
3353(3354)
3354(3353)
arbolcad_sol(3346)─┬─arbolcad_sol(3347)───arbolcad_sol(3350)───arbolcad_sol(3353)
                   ├─arbolcad_sol(3348)───arbolcad_sol(3351)
                   ├─arbolcad_sol(3349)───arbolcad_sol(3352)───arbolcad_sol(3354)
                   └─sh(3355)───pstree(3356)
alulab@minix:~/Documents/P1$ ./arbolcad_sol 4
3371(3370)
3370(3371)
arbolcad_sol(3357)─┬─arbolcad_sol(3358)───arbolcad_sol(3361)───arbolcad_sol(3363)───arbolcad_sol(3370)
                   ├─arbolcad_sol(3359)───arbolcad_sol(3366)───arbolcad_sol(3369)
                   ├─arbolcad_sol(3360)───arbolcad_sol(3365)───arbolcad_sol(3368)
                   ├─arbolcad_sol(3362)───arbolcad_sol(3364)───arbolcad_sol(3367)───arbolcad_sol(3371)
                   └─sh(3372)───pstree(3373)
alulab@minix:~/Documents/P1$ ./arbolcad_sol 5
3395(3392)
3392(3395)
arbolcad_sol(3374)─┬─arbolcad_sol(3375)───arbolcad_sol(3378)───arbolcad_sol(3381)───arbolcad_sol(3386)───arbolcad_sol(3392)
                   ├─arbolcad_sol(3376)───arbolcad_sol(3382)───arbolcad_sol(3388)───arbolcad_sol(3394)
                   ├─arbolcad_sol(3377)───arbolcad_sol(3384)───arbolcad_sol(3390)───arbolcad_sol(3396)
                   ├─arbolcad_sol(3379)───arbolcad_sol(3383)───arbolcad_sol(3387)───arbolcad_sol(3391)
                   ├─arbolcad_sol(3380)───arbolcad_sol(3385)───arbolcad_sol(3389)───arbolcad_sol(3393)───arbolcad_sol(3395)
                   └─sh(3397)───pstree(3398)
alulab@minix:~/Documents/P1$
```

2.- (10 puntos) En esta pregunta se le solicita escribir un programa cliente y un programa servidor.

a) (8 puntos) La descripción de cada uno de ellos se proporciona a continuación:

Cliente (*fifo_lfibo_client.c*): este programa solicita al servidor que le envíe *n* números de *Fibonacci*. El número *n* es un número aleatorio entre 1 y 20. Luego lee del *fifo* cliente lo enviado por el servidor y los imprime a la pantalla (use error estándar)

Servidor (*fifo_lfibo_server.c*) : este programa toma el pedido del cliente y crea un hijo. El hijo atiende la solicitud, generando los *n* números de *Fibonacci* y enviándolos al *fifo* del cliente. Mientras que el padre vuelve a tomar otra solicitud del *fifo* servidor si hubiera.

Usted debe de ignorar la señal que los hijos envían al padre cuando éstos terminan. Esto se logra con la siguiente instrucción:

```
if (signal(SIGCHLD, SIG_IGN) == SIG_ERR)  perror("signal");
```

```
  GNU nano 4.8                    randFibo.c
#include <time.h>

int main(void)
{   int n, x,y, i,t;

    srand(time(NULL));
    n = random() % 20;
    printf("%d numeros de Fibonacci\n",n);
    x=0;
    y=1;
    printf("%d ",x);
    for(i=0;i < n-1; i++) {
         t = x;
         x = y;
         y = y+t;
         printf("%d ",x);
    }
    printf("\n");
    exit(0);
}
```

Le puede servir el programa *randFibo.c* (izquierda) que devuelve un cantidad aleatorio de números de Fibonacci, cada vez que se ejecuta.

La salida debe de ser análoga a la siguiente:

```
alulab@minix:~/Documents/P2$ ./fifo_lfibo_server &
[1] 3556
alulab@minix:~/Documents/P2$ ./fifo_lfibo_client 2>salida1&
[2] 3557
alulab@minix:~/Documents/P2$ ./fifo_lfibo_client 2>salida2&
[3] 3559
[2]   Done                    ./fifo_lfibo_client 2> salida1
alulab@minix:~/Documents/P2$ ./fifo_lfibo_client 2>salida3&
[4] 3561
[3]   Done                    ./fifo_lfibo_client 2> salida2
alulab@minix:~/Documents/P2$ ./fifo_lfibo_client 2>salida4&
[5] 3563
[4]   Done                    ./fifo_lfibo_client 2> salida3
alulab@minix:~/Documents/P2$
[5]+  Done                    ./fifo_lfibo_client 2> salida4
alulab@minix:~/Documents/P2$
```

```
alulab@minix:~/Documents/P2$ cat salida1 salida2 salida3 salida4
Server sent 19 fibonacci numbers
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
Server sent 14 fibonacci numbers
0
1
1
2
3
5
8
13
21
34
55
89
144
233
```

b) (2 puntos) Modifique el programa servidor para que se ejecute como un demonio. Use la función de biblioteca *daemon()*.

Porf. Alejandro T. Bello Ruiz