Lenguaje C ¿?
Nombre: Pichardo Rico Cristian Eduardo **Computación I** **Grupo: 9080**

Práctica 3

(1) Uso de las operaciones elementales.

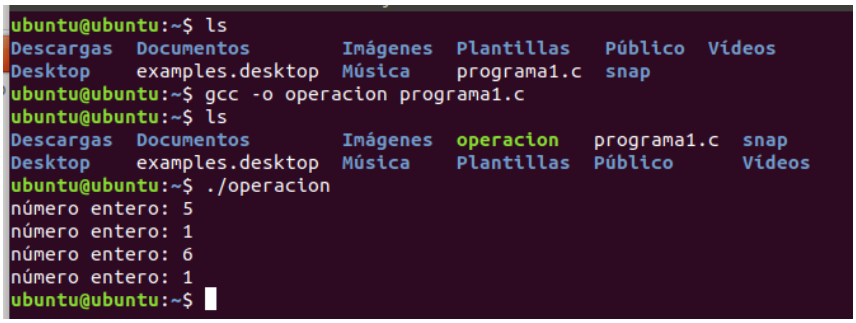
Ejercicio 1. Escriba el programa (programa1) y compílelo. De acuerdo al resultado ¿Cómo esperaría el resultado de la división? ¿Qué resultado le dio?

En el código declaramos las variables enteras *suma, resta, multi y div* y después las definimos como "*suma = 2+3*" la cual identifica la suma de dos enteros, en este caso la suma de "2+3" que nos devuelve un entero, pues la suma de enteros es cerrada en los enteros. En este caso obtenemos "2+3=5". Igualmente definimos "*resta= 3-2*" la cual es una resta de enteros, la cual da como resultado otro entero, pues la resta de enteros se puede ver como la suma de un entero con un inverso de un entero (es decir, n-m = n+(-m) si n y m son enteros) y la suma de enteros es cerrada en los enteros, por lo que su resta también nos devuelve un entero. En este caso obtenemos "3-2=1" Después definimos "*multi=2*3*" la cual es un producto de enteros que nos devuelve un entero ya que también cumple con la propiedad de cerradura. Y en este caso obtenemos "2*3=6".
Hasta aquí, todo es sencillo puesto que las operaciones están definidas en los números enteros y tanto la suma, resta y producto de enteros cumplen con la propiedad de cerradura en los enteros por lo que el resultado es un entero, sin embargo, la propiedad de cerradura no se cumple para la división de enteros.
Definimos "*div=3/2*" pero tenemos un problema, declaramos la variable "*div*" como entero y el resultado de "3/2" da un número racional (o en dado caso un real) por lo que la división se hará de la siguiente forma: La división entre dos enteros puede verse como dividendo=(cociente x divisor)+ residuo.
Lo que haría nuestro código en el momento de realizar "*div=3/2*" sería tomar unicamente el cociente sin tomar su residuo. Es decir, como "3/2=1,5" puede escribirse como "3=2(1)+1"donde 3 es el dividendo, 2 es el cociente, el primer 1 que multiplica al 2 es el divisor y el ultimo 1 que suma al producto es el residuo, por lo que "*div=1*".
Y finalmente, ponemos la función *printf* para que nos imprima los resultados de las variables que definimos anteriormente. Guardamos el archivo (con extensión .c), hacemos el ejecutable con *gcc* y lo ejecutamos para ver el resultado.

Código:

```c
#include <stdio.h>
// Programa 1: opearitmetic.c
//Este esun ejemplo de una suma entera, no se requiere nada

int suma;
int resta;
int multi;
int div;

int main(void){

     suma = 2+3;
     resta = 3-2;
     multi = 2*3;
     div = 3/2;

     printf("número entero: %i \n", suma);
     printf("número entero: %i \n", resta);
     printf("número entero: %i \n", multi);
     printf("número entero: %i \n", div);
     return 0;
}
```

Programa ejecutado:



Tal y como lo describimos, el resultado de "*suma=3+2*" es 5, el resultado de "*resta=3-2*" es 1, el resultado de "*multi=2*3*" es 6 y, finalmente, el resultado de "*div=3/2*" es 1.

Ejercicio 2. Cree otro fuente con base al anterior (opearitmetic1.c) pero ahora con datos reales, no enteros y con salida de 4 dígitos de precisión.

Esto fue sencillo, solamente declaramos las variables *suma*, *resta*, *multi* y *div* como variables de tipo flotante (*float*) y después las definí de la siguiente manera: primero definí "suma = 2.004525+3.256414", después "resta = 2.254-3.25687", después "multi = 2.2*3.5" y, por ultimo, "div = 3.3698/2.2587". Nótese que utilice números al azar y también varié la cantidad de decimales entre cada numero para probar que coloqué los 4 dígitos de precisión.
Después, usando la función *printf* hice que se imprimiera el resultado de cada variable y aquí fue donde indiqué los dígitos de precisión. Utilice %f para imprimir un número flotante (en este caso, el resultado de cada variable las cuales declaré en un inicio como float) y le agregue el número "0.4" es decir, indique en la instrucción que debía imprimir un número de tipo flotante asociada la variable correspondiente con 4 cifras decimales de precisión. El número de las cifras enteras coloque "0" pero no importa, debido a que si indicamos menos cifras enteras que las necesarias, al momento de ejecutar simplemente se ignorara esto y se imprimirán la cantidad de cifras que sea necesario usar.
Por, lo que al final, para imprimir los resultados de las variables *suma*, *resta*, *multi* y *div* con 4 cifras decimales de precisión fue:

```
printf("número real de la suma: %0.4f \n", suma);
 printf("número real de la resta: %0.4f \n", resta);
 printf("número real de la multiplicación: %0.4f \n", multi);
 printf("número real de la división: %0.4f \n", div);
```

Tal y como se ve en el código del programa:

```c
#include <stdio.h>
// Programa 2: opearitmetic2.c
//Este esun ejemplo de una suma entera, no se requiere nada

float suma;
float resta;
float multi;
float div;

int main(void){

     suma = 2.004525+3.256414;
     resta = 2.254-3.25687;
     multi = 2.2*3.5;
     div = 3.3698/2.2587;

     printf("número real de la suma: %0.4f \n", suma);
     printf("número real de la resta: %0.4f \n", resta);
     printf("número real de la multiplicación: %0.4f \n", multi);
     printf("número real de la división: %0.4f \n", div);
     return 0;
}
```

Después guardé el código en un documento con terminación .c e hice el ejecutable con *gcc*. Después ejecuté el programa dándonos el resultado correcto de las operaciones que definimos en las variables pero con unicamente 4 cifras significativas en todas ellas.

```
ubuntu@ubuntu:~$ ls
Descargas  Documentos        Imágenes  Plantillas   Público  Vídeos
Desktop    examples.desktop  Música    programa2.c  snap
ubuntu@ubuntu:~$ gcc -o A programa2.c
ubuntu@ubuntu:~$ ls
A          Desktop     examples.desktop  Música      programa2.c  snap
Descargas  Documentos  Imágenes          Plantillas  Público      Vídeos
ubuntu@ubuntu:~$ ./A
número real de la suma: 5.2609
número real de la resta: -1.0029
número real de la multiplicación: 7.7000
número real de la división: 1.4919
ubuntu@ubuntu:~$
```

El resultado de “suma = 2.004525+3.256414” fue 5.2609 (el valor con mas cifras significativas era 5,260939), el resultado de “resta = 2.254-3.25687” fue -1.0029 (el valor con mas cifras significativas era -1.00287, aquí se nota como se redondea la ultima cifra significativa), el resultado de “multi = 2.2*3.5” fue 7.7000 (el valor sin los ceros decimales que no valen nada es 7.7, aquì se nota que cuando no hay suficientes cifras decimales se agregan ceros a la derecha) y, por ultimo, el resultado de “div = 3.3698/2.2587” es 1.4919 (el valor con mas cifras significativas era 1.4919201310488…).

Ejercicio 3. En el siguiente ejercicio (programa2.c) veamos algunos ejemplos de el formato de printf (Escribalo y compilelo).

Esto me hubiese ayudado muchísimo en algunos ejercicios anteriores, tanto de esta practica como de la anterior, en donde se nos pedía imprimir unicamente 3 o 4 cifras significativas.

De forma sencilla es lo que expliqué en el ejercicio 2 de esta práctica y en el ultimo ejercicio de la práctica anterior. Lo que estamos haciendo es definiendo cuantas cifras enteras queremos y cuantas cifras decimales queremos de precisión. En este caso, colocamos que queremos 4 cifras enteras y 4, y cifras decimales respectivamente. Lo delas cifras enteras aún no me queda claro como funciona porque si las cifras enteras son menores o iguales al numero que especificamos todo funciona perfectamente pero si el numero de cifras enterases mayor, entonces se ignora por completo el numero que especificamos y se imprimen todas las cifras enteras que el numero contiene.

Ejemplo, si tenemos “float x=145.256” y lo imprimimos con la función *printf* y con %1.2f se imprimiría en pantalla “*154,26*” es decir, tiene dos cifras significativas en los decimales y por ello redondea el ultimo decimal pero ignora totalmente el numero de cifras enteras que hemos colocado.

Volviendo al ejercicio, el código quedaría así:

```c
#include <stdio.h>
// En este caso, el especificador de formato contiene
// una anchura de 4 caracteres como minimo
// y un numeromaximo de 4 cracteres a imprimir
// el resultado sera 10.6789

int main(){

     float suma;
     suma = 4+6.6789;

     printf("El especificador es: 4.4\n");
     printf("La suma es: %4.4f", suma);
     printf("\n\nEl especificador es: 4.5\n");
     printf("La suma es: %4.5f", suma);
     printf("\n\nEl especificador es: 4.6\n");
     printf("La suma es: %4.6f\n", suma);
     return 0;
}
```

Es decir, declaramos primero “suma” como una variable flotante, después definimos “*suma=4+6.6789*” es decir que *suma=10.6789*. Y después usamos la función *printf* varias veces para imprimir el valor de la suma con 4, 5 y 6 cifras significativas respectivamente.

Después de guardar el archivo como un código de C y sacar el ejecutable con *gcc*, ejecute el programa y el resultado fue el siguiente:

```
ubuntu@ubuntu:~$ ls
Descargas  Documentos        Imágenes  Plantillas   Público  Vídeos
Desktop    examples.desktop  Música    programa2.c  snap
ubuntu@ubuntu:~$ gcc programa2.c
ubuntu@ubuntu:~$ ls
a.out      Desktop     examples.desktop  Música      programa2.c  snap
Descargas  Documentos  Imágenes          Plantillas  Público      Vídeos
ubuntu@ubuntu:~$ ./a.out
El especificador es: 4.4
La suma es: 10.6789

El especificador es: 4.5
La suma es: 10.67890

El especificador es: 4.6
La suma es: 10.678900
ubuntu@ubuntu:~$
```

Y lo que nos imprime el programa son los resultados dela suma (la cual daba 1*0.6789*) con 4, 5 y 6 cifras decimales, redondeando la cifra decimal final. Y como la variable suma solo tiene 4 cifras decimales para 5 y 6 cifras decimales les agrega uno y dos ceros a la derecha respectivamente pata mostrar las cifras decimales especificadas.

Ejercicio 4. Escriba y compile el siguiente programa (programa3).
Aquí hay muchas cosas que explicar, primero que nada, este programa suma los primeros "*m*" números naturales, siendo "*m*" el número natural que nosotros ingresemos.

Entonces en el programa primero declaramos a la variable entera (*int*) "*m*" y la definimos como "*m=0*" y luego con *printf* mandamos a imprimir "*Introduce el número de vueltas: "* aunque en mi opinión habría quedado mejor poner alguna oración que indicara que se iba a hacer la suma de los números naturales hasta el que nosotros hagamos ingresado, después haciendo uso de la función *scanf (que nos permite introducir cualquier combinación de valores númericos, caracteres sueltos y cadenas de caracteres a través del teclado)* y dentro de la función definimos la cadena de control como "*%2d*" que significa que solo se permitirá introducir números enteros de hasta dos dígitos y después declaramos la lista de argumentos (los cuales representan los datos) y en este caso es nuestra variable *m* por lo cual queda como "*&m*".

Después declaramos la variable entera sin signo (*unsigned int: 16 bytes: Rango=0-65535*) "*i*" y "*s*" y las definimos como "*s=0*" e "*i=1*" y luego comenzamos con el ciclo *while.* Primero empezamos declarando el ciclo while y la condición que este seguirá la cual es "*i<=m*" lo cual significa que el ciclo while seguirá ejecutándose siempre que la variable "*i*" sea menor o igual que la variable "*m*" y luego definimos el cuerpo del ciclo while la cual, en este caso, depende de dos operaciones, la primera es "*s=s+1*" esto significa que la variable "*s*" tomará los valores de "*s+i*" esto puede hacerse porque hemos definido en un inicio a "*s=0*" y "*i=1*" por lo que lo primero que pasará será la operación "s=0+1". Y también definimos la operación "i=i+1". Aquí lo que pasará es que el ciclo iniciará con los valores iniciales "*s=0*" e "*i=1*" y el numero "*m*" que definamos en la primera parte siempre y cuando que *m>0 (porque si m=0 entonces la suma de los primeros 0 números es 0)* por lo que la condición de while es cierta pues "i=1<=*m*" entonces se ejecutaran las operaciones, primero "s=0+1" por lo que "s=1" y "i=1+1" por lo que "i=2", entonces se regresa a la condición principal del ciclo while, por lo que si "*i=2<=m*" se repetirá el ciclo while, es decir, primero "*s=1+2*" (*s=3*) y después "*i=2+1*" (*i=3*) y así hasta que la condición principal de while sea falso, es decir: "*i>m*" por lo que el ciclo while se detiene y se pasa a la siguiente función/instrucción.

Y después usando *printf* imprimimos "Después de %i vueltas ",m; y "la suma es %d\n",s; donde lo primero que se imprime es el numero de vueltas *m* que introducimos al inicio y después la suma de los primeros *m* números naturales. Y ahí terminaría el programa.

El código quedó así:

```c
#include <stdio.h>
//Programa 3: Programa simple que suma los primeros n números naturales

int main(void){

     int m=0;

     printf("Introduce el número de vueltas: ");
     scanf("%2d", &m);
     unsigned int s=0, i=1;

     while (i<=m){ //comienza el cilo while
       s=s+i;
       i=i+1;
    } //termina el ciclo while

    printf("Después de %i vueltas ", m);
    printf("la suma es %d\n", s);
     return 0;
} //fin del main
```

Y el resultado es lo esperado, se imprime “Introduce el número de vueltas: ” luego se introduce un número entero y después se imprime “Después de *m*” siendo *m* el numero que hemos introducido y “la suma es *s*” siendo *s* la suma de los primeros *m* números.

```
ubuntu@ubuntu:~$ ls
Descargas  Documentos        Imágenes  Plantillas   Público
Desktop    examples.desktop  Música    programa3.c  Vídeos
ubuntu@ubuntu:~$ gcc programa3.c
ubuntu@ubuntu:~$ ls
a.out      Desktop     examples.desktop  Música      programa3.c  Vídeos
Descargas  Documentos  Imágenes          Plantillas  Público
ubuntu@ubuntu:~$ ./a.out
Introduce el número de vueltas: 5
Después de 5 vueltas la suma es 15
ubuntu@ubuntu:~$ ./a.out
Introduce el número de vueltas: 10
Después de 10 vueltas la suma es 55
ubuntu@ubuntu:~$
```

Ejercicio 5. Haga el mismo programa pero ahora con el ciclo for.
A pesar de que el profesor no explicó el ciclo *for,* no fue un problema porque el ciclo *for* es ligeramente mas sencillo. Lo único que modifiqué fue el ciclo *while* que lo cambié por el ciclo *for* y ya no definí la variable *i* antes de empezar a trabajar con el ciclo *for,* es decir, declaré la variable “*i*” y la variable “s” como enteros sin signo y solo definí la variable “*s*” como “*s=0*”.

Para el ciclo *for* primero coloqué la función del ciclo *for* y después definí sus condiciones, primeramente “i=*1*” es decir, tomé la variable *i* que no había tomado ningún valor en especifico y lo definí con el valor *1*. Después la condición es que “*i<=m*” y que cada vez que el ciclo *for* haga una vuelta la variable “*i*” sume uno, lo que se escribe como “*i++*” es decir, que *i* incrementará o se le sumará 1.
Y después definimos el cuerpo del ciclo *for* como “*s=s+i*” es decir, la primera suma que pusimos en el ciclo while. Y termina el ciclo.
¿Qué sucederá? Que la variable “*i*” tomará el valor de 1, después si “*i<=m*” entonces se ejecutará la suma “*s=s+i*” e “*i*” se le sumará 1. Si la condición “*i<=m*” se sigue cumpliendo, entonces el ciclo *for* se seguirá ejecutando.

El código quedó así:

```c
#include <stdio.h>
//Programa 3 con for: Programa simple que suma los primeros n números naturales

int main(void){

     int m=0;

     printf("Introduce el número de vueltas: ");
     scanf("%2d", &m);
     unsigned int s=0, i;

      for(i=1;i<=m;i++){ //comienzaciclo for
            s=s+i;
      } //termina ciclo for

    printf("Después de %i vueltas ", m);
    printf("la suma es %d\n", s);
     return 0;
} //fin del main
```

Y el resultado es lo esperado, se imprime “Introduce el número de vueltas: ” luego se introduce un número entero y después se imprime “Después de *m*” siendo *m* el numero que hemos introducido y “la suma es *s*” siendo *s* la suma de los primeros *m* números. Utilice los mismos ejemplos que el ejercicio anterior para comparar resultados.

```
ubuntu@ubuntu:~$ ls
ciclofor.c  Desktop     examples.desktop  Música      Público
Descargas   Documentos  Imágenes          Plantillas  Vídeos
ubuntu@ubuntu:~$ gcc ciclofor.c
ubuntu@ubuntu:~$ ls
a.out       Descargas  Documentos        Imágenes  Plantillas  Vídeos
ciclofor.c  Desktop    examples.desktop  Música    Público
ubuntu@ubuntu:~$ ./a.out
Introduce el número de vueltas: 5
Después de 5 vueltas la suma es 15
ubuntu@ubuntu:~$ ./a.out
Introduce el número de vueltas: 10
Después de 10 vueltas la suma es 55
ubuntu@ubuntu:~$
```

Ejercicio 6. Escriba el siguiente ejemplo para verificar lo anterior (programa4)
Este ejercicio no lo pude terminar con éxito con el código que colocó el profesor en la práctica porque me imprime cosas que no deberían imprimirse y aunque lo he intentado resolver, no he podido solucionarlo en este momento.

Primero se declara la variable entera “*termina*” y la definimos como “termina=9” después definimos el ciclo *while* bajo la condición que si la variable *termina* es diferente de cero entonces el ciclo *while* se ejecuta y si *termina=0* entonces el ciclo while termina.
Dentro del ciclo establecimos una condición más mediante *if*, la condición es que si la variable *termina* es diferente de cero entonces se imprime “*Haz decidido continuar*” y si *termina=0* entonces se imprimirá “*Haz decidido terminar*”.

El código queda así:

```c
#include <stdio.h>

int main(void){

     int termina='9';

     while(termina != '0'){
       printf("Introduce un número: ");
       termina=getchar();

       if(termina != '0') {
         printf("Haz escogido continuar\n");
       }
       else {
         printf("Haz escogido terminar\n");
       }
     }

     return 0;
}
```

Sin embargo, al momento de ejecutar el programa pasa esto:

```
ubuntu@ubuntu:~$ ls
Descargas  Documentos        Imágenes  Plantillas   Público
Desktop    examples.desktop  Música    programa4.c  Vídeos
ubuntu@ubuntu:~$ gcc programa4.c
ubuntu@ubuntu:~$ ls
a.out      Desktop     examples.desktop  Música      programa4.c  Vídeos
Descargas  Documentos  Imágenes          Plantillas  Público
ubuntu@ubuntu:~$ ./a.out
Introduce un número: 5
Haz escogido continuar
Introduce un número: Haz escogido continuar
Introduce un número: 0
Haz escogido terminar
ubuntu@ubuntu:~$
```

Como se observa en la ultima captura al introducir cualquier numero diferente de cero se imprime "*Haz escogido continuar"*. Hasta aquí todo bien, sin embargo, después se imprime "*Introduce un número: Haz escogido continuar*" seguido de "Introduce un número: ".

Modifiqué el código para que en lugar de utilizar *getchar()* se utilice la función *scanf*, de esta forma establezco de mediante el teclado se introduzca un número entero y se guarde en la variable *termina.* Y de esta manera el programa funciona a la perfección.

El código quedó así:

```c
#include <stdio.h>

int main(void){

     int termina='9';

     while(termina != 0){
       printf("Introduce un número: ");
       scanf("%i", &termina);

       if(termina != 0) {
         printf("Haz escogido continuar\n");
       }
       else {
         printf("Haz escogido terminar\n");
       }
     }

     return 0;
}
```

Lo que dice el código es que en un principio la variable entera *termina* esta definida como "*9*" por lo que se inicia el ciclo *while* porque "*termina=9*" cumple con la condición del ciclo *while* "(*termina != 0*)" es decir, que si la variable *termina* es diferente de cero se ejecutará el ciclo *while. Después* de ello, se imprime en la pantalla "*Introduce un número: "* (obviamente de tipo entero), ahí se podrá introducir un número entero y al presionar *enter* comenzará a ejecutarse la función *if* . Es decir, si el número que nosotros introducimos es distinto de cero entonces se imprimirá en pantalla "*Haz decidido continuar*" y como la variable *termina* es distinto de cero se volverá a ejecutar el ciclo *while* pues recordemos que la condición para que *while* se detenga (en este caso) es que la variable *termina* sea igual a cero. Y en el caso que nosotros introduzcamos el número cero entonces *if hará* que se imprima "*Haz escogido terminar*" y como la variable "*termina=0*" entonces *while* ya no se ejecuta.

El programa ejecutado es:

```
ubuntu@ubuntu:~$ ls
Descargas  Documentos        Imágenes  Plantillas   Público
Desktop    examples.desktop  Música    programa4.c  Vídeos
ubuntu@ubuntu:~$ gcc programa4.c
ubuntu@ubuntu:~$ ls
a.out      Desktop     examples.desktop  Música      programa4.c  Vídeos
Descargas  Documentos  Imágenes          Plantillas  Público
ubuntu@ubuntu:~$ ./a.out
Introduce un número: 4
Haz escogido continuar
Introduce un número: 5
Haz escogido continuar
Introduce un número: 0
Haz escogido terminar
ubuntu@ubuntu:~$
```

Aquí se observa que si introducimos un número que es distinto de cero entonces se imprime "*Haz escogido continuar*" y el ciclo *while* se sigue ejecutando y si el número que introducimos es cero entonces se imprime "*Haz escogido terminar*" y el ciclo *while* se detiene.

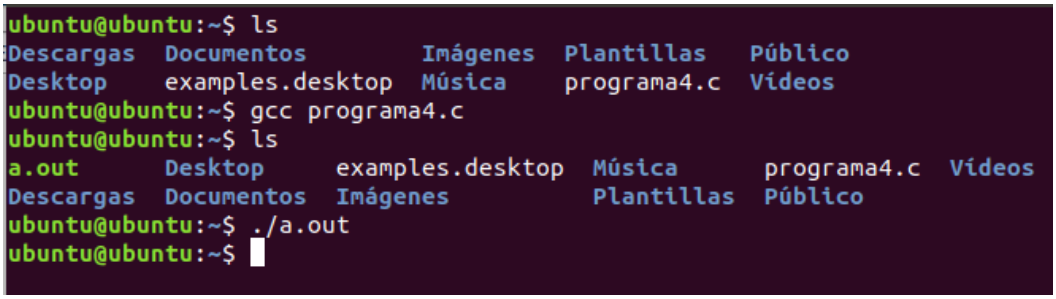
Ejercicio 7. Si en lugar del comando de prueba dentro del argumento del comando de while colocáramos un 0 (cero) ¿Qué sucedería? Escriba o copie el programa anterior y sustituya el comando de prueba por el cero ¿Describa ahora que sucede?

Pues si en un inicio declaramos a *termina* como cero entonces al momento de ejecutarse el ciclo *while* la condición no se cumplirá por lo que *while* no debería hacer nada y el programa directamente se cerraría.

Modificando el código quedaría así:

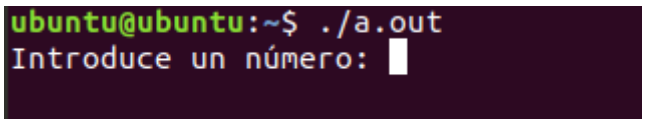
```c
#include <stdio.h>

int main(void){

     int termina=0;

     while(termina != 0){
       printf("Introduce un número: ");
       scanf("%i", &termina);

       if(termina != 0) {
         printf("Haz escogido continuar\n");
       }
       else {
         printf("Haz escogido terminar\n");
       }
     }

     return 0;
}
```

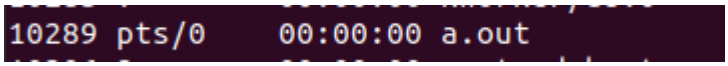
Y al ejecutarlo pasa exactamente lo que describí en un principio.



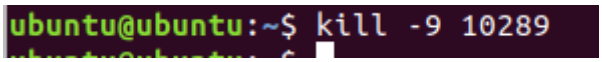
Ejercicio 8. Si un programa está corriendo y requiere terminarlo ¿Cuales son las formas de hacerlo? Explique.

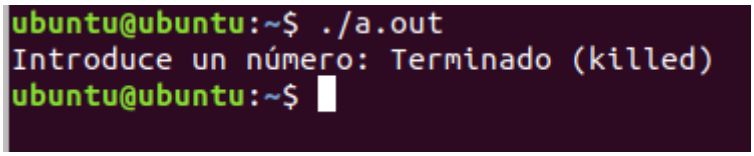
Se puede hacer mediante el comando *kill* en conjunto con el comando *ps.* La forma de hacerlo es utilizar *ps -A* para ver todos los procesos que se estén ejecutando, entre ellos nuestro programa. En mi caso mi programa se llama *a.out* y luego de ejecutarlo, mediante *ps -A* busque su ID asociado.





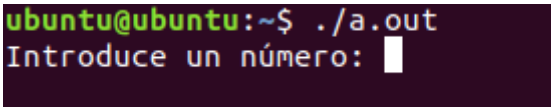
Y después utilizando *kill* junto con su nivel de señal y el ID asociado al programa que se está ejecutando puedo detener al programa.



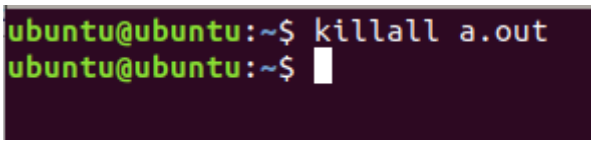


Y también podemos hacer uso de *killall* usando unicamente el nombre de nuestro proceso.
Es decir, volvemos a ejecutar un programa y lo dejamos abierto y con el comando *killall* y el nombre del proceso asociado a nuestro programa podemos detenerlo.

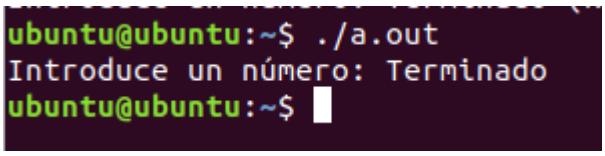
Primero ejecutamos el programa:



Después ejecutamos el comando "*killall a.out*":



Y comprobamos que nuestro programa se ha detenido:


ubuntu@ubuntu:~$ ./a.out
Introduce un número: Terminado
ubuntu@ubuntu:~$