



INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y
CIENCIAS SOCIALES Y ADMINISTRATIVAS

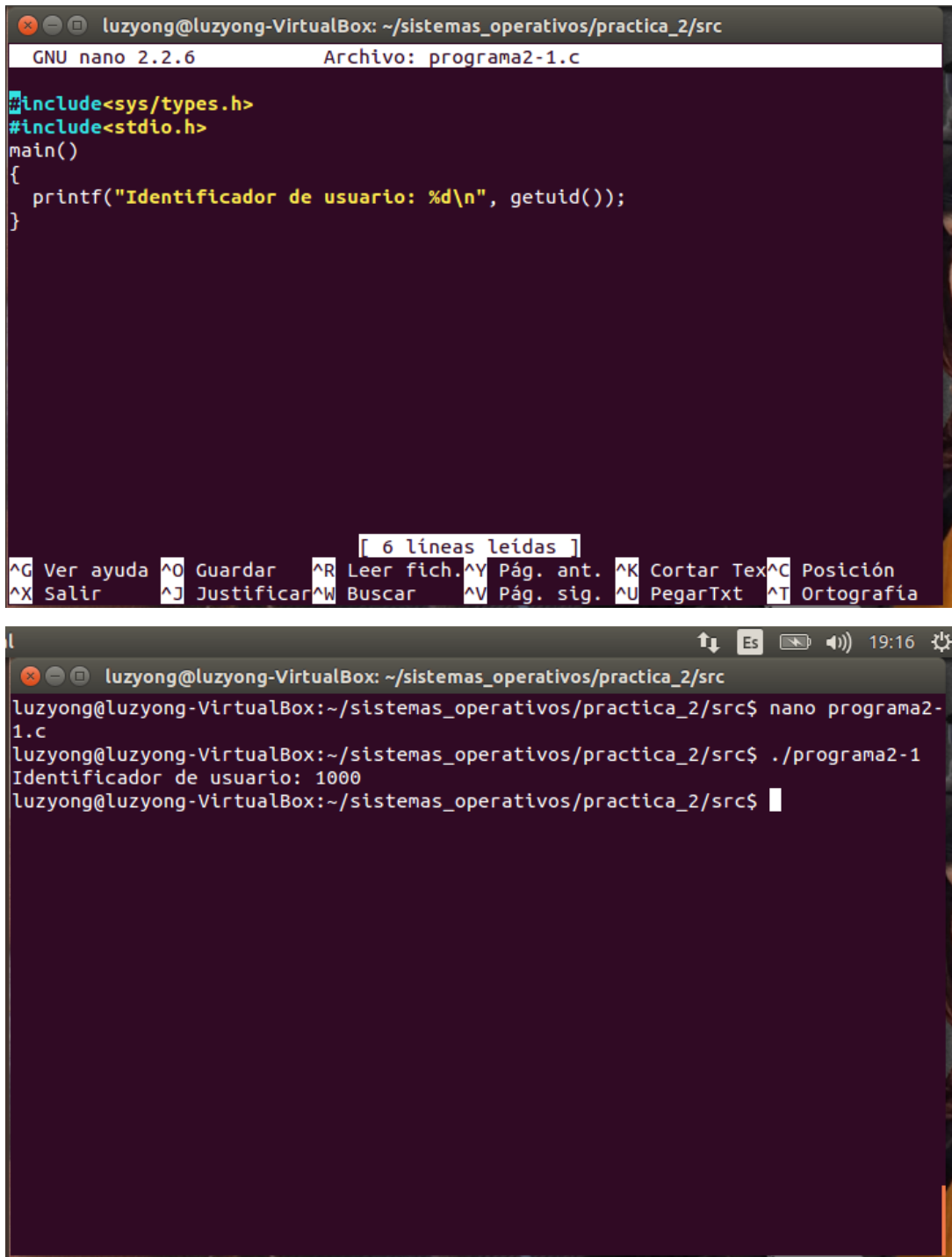


Práctica 2: PID, UID y variables de ambiente.

Yong Rodríguez Luz María

Programa 2.1:

En este programa obtenemos el UID o identificador de usuario por medio de `getuid()`.



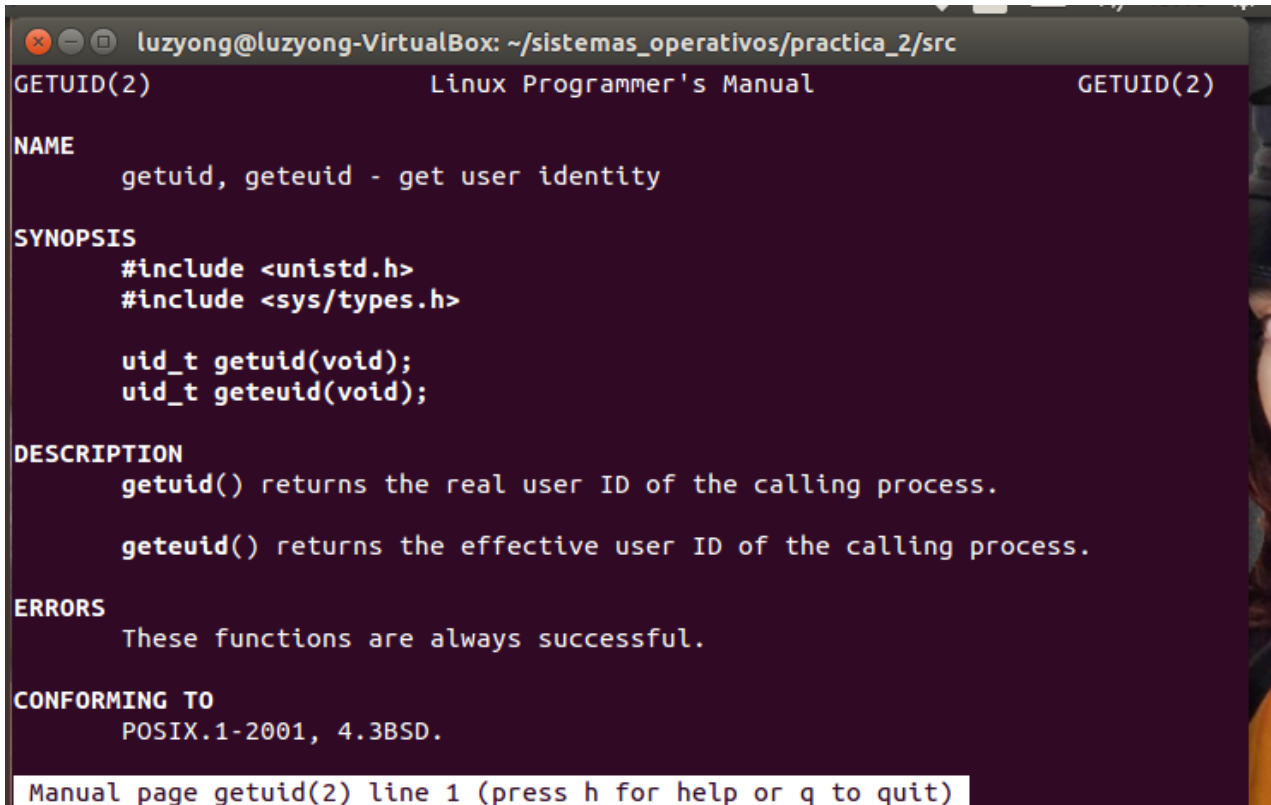
```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GNU nano 2.2.6 Archivo: programa2-1.c

#include<sys/types.h>
#include<stdio.h>
main()
{
    printf("Identificador de usuario: %d\n", getuid());
}

[ 6 líneas leídas ]
^G Ver ayuda ^O Guardar ^R Leer fich.^Y Pág. ant. ^K Cortar Tex^C Posición
^X Salir ^J Justificar ^W Buscar ^V Pág. sig. ^U PegarTxt ^T Ortografía

luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ nano programa2-1.c
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./programa2-1
Identificador de usuario: 1000
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$
```

- Pregunta 2.1: Investigue la función de `getuid()`.
Muestra el identificador del usuario.
La función `getuid()` retorna el valor del ID del usuario real que está ejecutando el proceso. La función `geteuid()` retorna el valor del ID del usuario efectivo que está ejecutando el proceso.



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GETUID(2)                                Linux Programmer's Manual                                GETUID(2)

NAME
    getuid, geteuid - get user identity

SYNOPSIS
    #include <unistd.h>
    #include <sys/types.h>

    uid_t getuid(void);
    uid_t geteuid(void);

DESCRIPTION
    getuid() returns the real user ID of the calling process.

    geteuid() returns the effective user ID of the calling process.

ERRORS
    These functions are always successful.

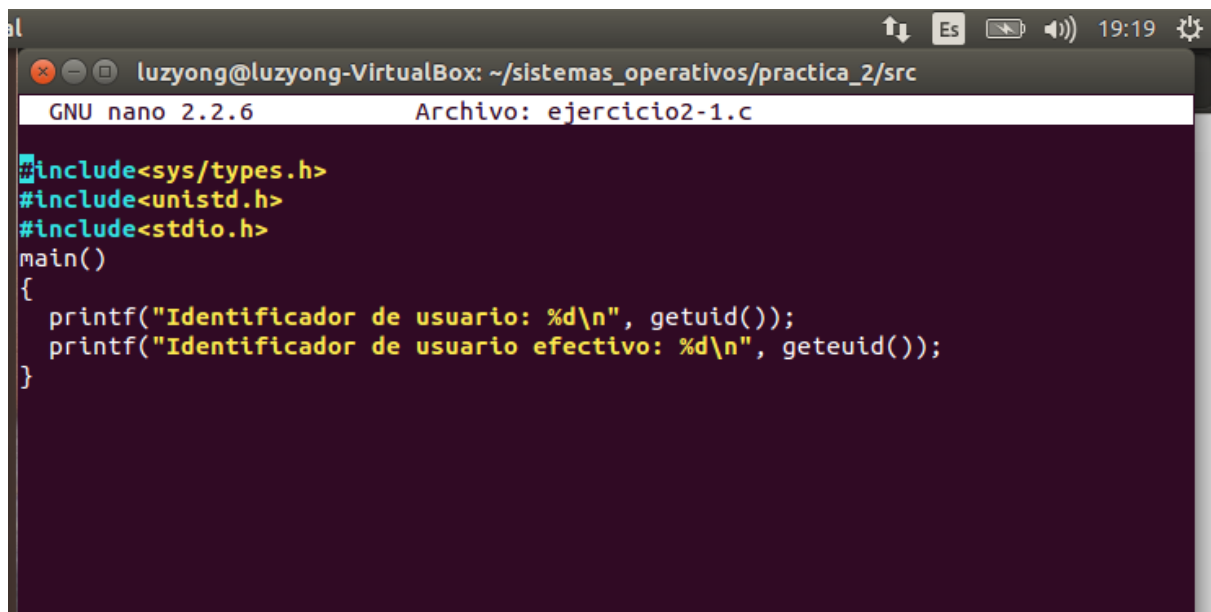
CONFORMING TO
    POSIX.1-2001, 4.3BSD.

Manual page getuid(2) line 1 (press h for help or q to quit)
```

Ejercicio 2.1:

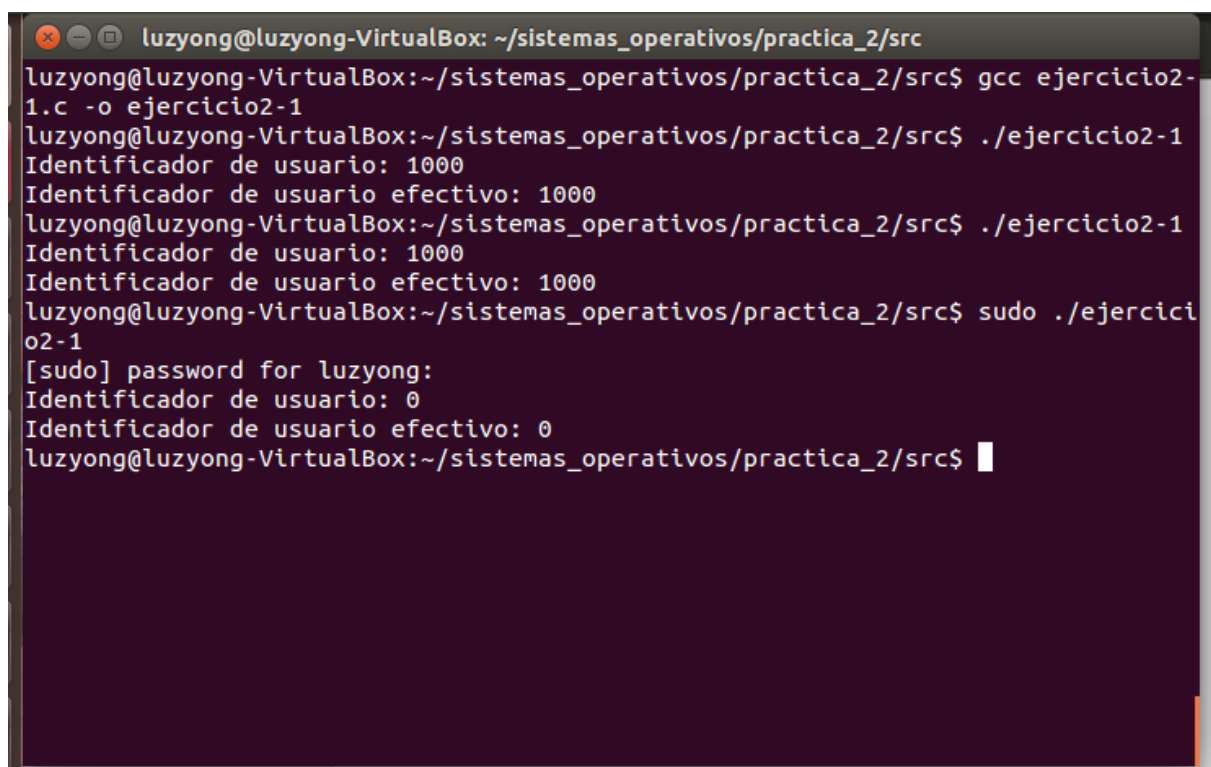
Modifique el código para que también muestre el usuario efectivo. ¿En qué condiciones la impresión del usuario real y efectivo serán distintas?

El usuario real es aquel que haya ejecutado el proceso. El usuario efectivo es el dueño o creador del programa que crea ese proceso. Serán iguales los valores de ambos siempre y cuando no se especifique quién es el dueño del proceso. Esto se hace activando el bit SUID del archivo o carpeta que se va a ejecutar. Al hacer esto y después ejecutarlo por un usuario distinto al que creó ese archivo, el valor del usuario efectivo será el UID del dueño del archivo y el valor del usuario real será el UID del que lo ejecutó. Esto sucede porque al activar este bit, el sistema permite al usuario ejecutar el proceso como si fuera el dueño de este.



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GNU nano 2.2.6 Archivo: ejercicio2-1.c

#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>
main()
{
    printf("Identificador de usuario: %d\n", getuid());
    printf("Identificador de usuario efectivo: %d\n", geteuid());
}
```



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ gcc ejercicio2-1.c -o ejercicio2-1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./ejercicio2-1
Identificador de usuario: 1000
Identificador de usuario efectivo: 1000
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./ejercicio2-1
Identificador de usuario: 1000
Identificador de usuario efectivo: 1000
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ sudo ./ejercicio2-1
[sudo] password for luzyong:
Identificador de usuario: 0
Identificador de usuario efectivo: 1000
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$
```

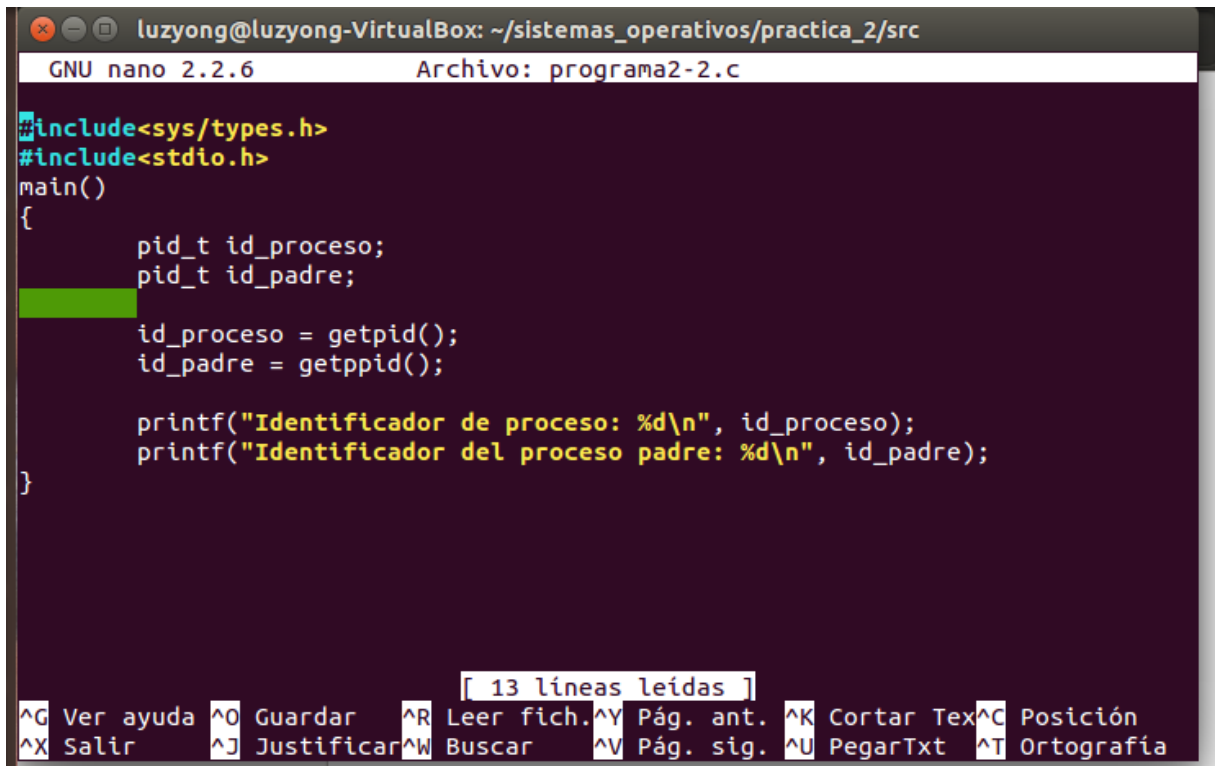
```
luz2@luz2-VirtualBox: ~/sistemas_operativos/practica_2/src
s /bin/bash luz2
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$ sudo passwd luz2
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$ ls -l
total 24
-rwxrwxr-x 1 luz2 luz2 7386 oct 17 19:18 ejercicio2-1
-rw-rw-r-- 1 luz2 luz2 188 oct 17 19:18 ejercicio2-1.c
-rwxrwxr-x 1 luz2 luz2 7346 oct 17 19:15 programa2-1
-rw-rw-r-- 1 luz2 luz2 105 oct 17 19:15 programa2-1.c
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$ chmod 4775 ejercicio2-1
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$ ls -l
total 24
-rwsrwxr-x 1 luz2 luz2 7386 oct 17 19:18 ejercicio2-1
-rw-rw-r-- 1 luz2 luz2 188 oct 17 19:18 ejercicio2-1.c
-rwxrwxr-x 1 luz2 luz2 7346 oct 17 19:15 programa2-1
-rw-rw-r-- 1 luz2 luz2 105 oct 17 19:15 programa2-1.c
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$ ./ejercicio2-1
Identificador de usuario: 1000
Identificador de usuario efectivo: 1000
luz2@luz2-VirtualBox:~/sistemas_operativos/practica_2/src$
```

1.-Activando SUID en el archivo ejercicio2-1

```
luz2@luz2-VirtualBox: ~
luz2@luz2-VirtualBox:~$ /home/luz2/sistemas_operativos/practica_2/src/ejercicio2-1
Identificador de usuario: 1002
Identificador de usuario efectivo: 1002
luz2@luz2-VirtualBox:~$ /home/luz2/sistemas_operativos/practica_2/src/ejercicio2-1
Identificador de usuario: 1002
Identificador de usuario efectivo: 1000
luz2@luz2-VirtualBox:~$
```

2.-Ejecutando el archivo ejercicio2-1 con otro usuario

Programa 2.2: Este programa nos muestra el PID y el PPID del proceso que se ejecuta (en este caso, el programa 2.2)



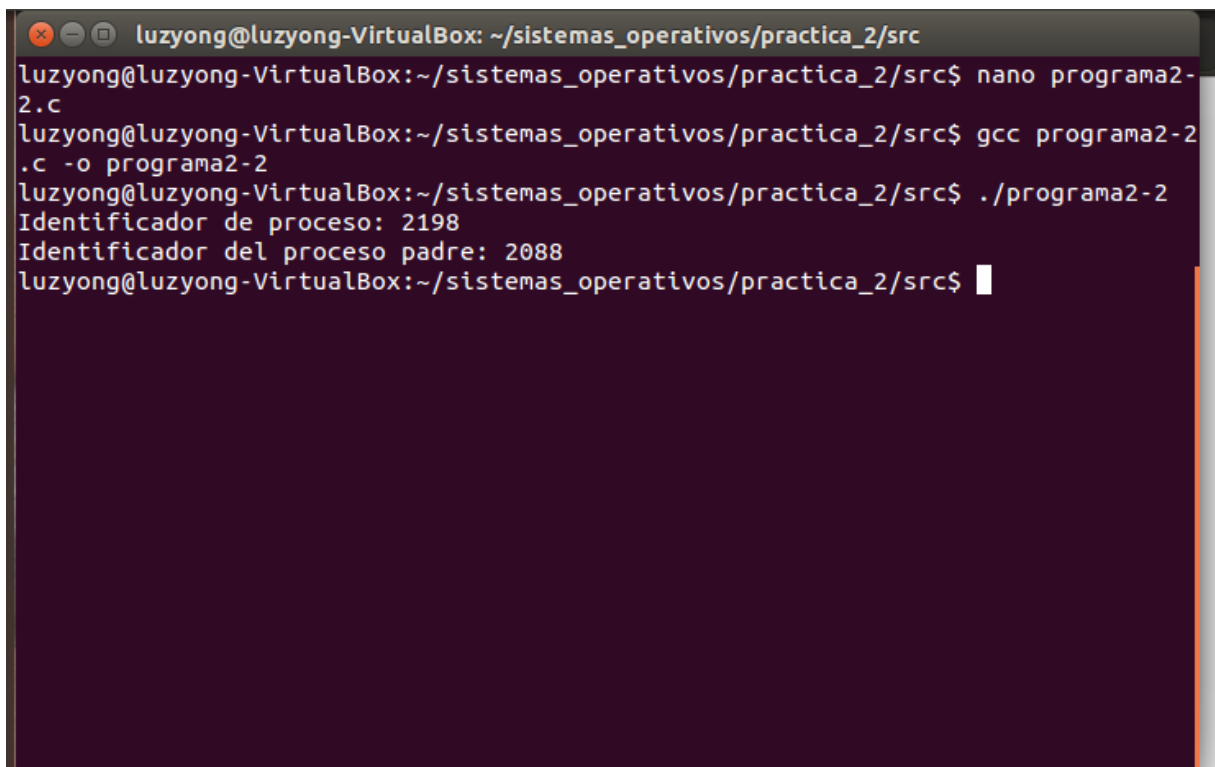
```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GNU nano 2.2.6 Archivo: programa2-2.c

#include<sys/types.h>
#include<stdio.h>
main()
{
    pid_t id_proceso;
    pid_t id_padre;

    id_proceso = getpid();
    id_padre = getppid();

    printf("Identificador de proceso: %d\n", id_proceso);
    printf("Identificador del proceso padre: %d\n", id_padre);
}

[ 13 líneas leídas ]
^G Ver ayuda ^O Guardar ^R Leer fich.^Y Pág. ant. ^K Cortar Tex^C Posición
^X Salir ^J Justificar ^W Buscar ^V Pág. sig. ^U PegarTxt ^T Ortografía
```



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ nano programa2-2.c
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ gcc programa2-2.c -o programa2-2
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./programa2-2
Identificador de proceso: 2198
Identificador del proceso padre: 2088
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$
```

- **Pregunta 2.2:** Investigue la utilidad de la función getpid y la de getppid.
Getpid nos da la identificación del proceso. La función getpid() nos devuelve el ID del proceso que se está ejecutando y getppid() nos devuelve el ID del proceso padre del proceso que se está ejecutando.
- **Pregunta 2.3:** Si se ejecuta varias veces el programa ¿por qué el identificador del padre es siempre el mismo?
Porque el proceso padre es el que inicia ese programa, es decir, ese proceso y al terminarse el programa, se termina el proceso x. Al volverlo a ejecutar, aunque sea el mismo programa, se considera como un nuevo proceso. El proceso padre en este caso es la terminal, que es la que está llamando al programa y ejecutando el proceso. Cuando se termina de ejecutar el programa, la terminal sigue abierta, por lo tanto, el proceso padre no cambia.

Ejercicio 2.2: Modificamos el programa anterior para que no terminara, con la ayuda de sleep(). Al mismo tiempo que se ejecutaba, ejecutamos en otra terminal el comando ps con el proceso more utilizando una tubería.

- ¿Qué nombres les asigna el sistema al proceso y al proceso padre y cuál es el significado de cada uno de los nombres?
CMD bash y ejercicio2-2, PID 2088 y 2335 y TTY pts/12 para ambos.
CMD es el nombre del ejecutable, PID es el identificador de proceso y TTY es el número de dispositivo de la terminal.

```

luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GNU nano 2.2.6 Archivo: ejercicio2-2.c
#include<sys/types.h>
#include<stdio.h>
main()
{
    pid_t id_proceso;
    pid_t id_padre;

    id_proceso = getpid();
    id_padre = getppid();

    printf("Identificador de proceso: %d\n", id_proceso);
    printf("Identificador del proceso padre: %d\n", id_padre);
    sleep();
}
[ 14 líneas leídas ]
^G Ver ayuda ^O Guardar ^R Leer fich.^Y Pág. ant.^K Cortar Tex^C Posición
^X Salir ^J Justificar ^W Buscar ^V Pág. sig.^U PegarTxt ^T Ortografía

```

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ nano programa2-2.c
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ nano ejercicio2-2.c
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ gcc ejercicio2-2.c -o ejercicio2-2
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./ejercicio2-2
Identificador de proceso: 2335
Identificador del proceso padre: 2088

luzyong@luzyong-VirtualBox: ~
1993 ?      00:00:00 gvfsd-burn
1999 ?      00:00:00 gvfsd-metadata
2027 ?      00:00:00 telepathy-indic
2032 ?      00:00:00 mission-control
2049 ?      00:00:00 systemd-hostnam
2050 ?      00:00:00 zeitgeist-datah
2055 ?      00:00:00 zeitgeist-daemo
2061 ?      00:00:00 zeitgeist-fts
2069 ?      00:00:00 cat
2080 ?      00:00:04 gnome-terminal
2087 ?      00:00:00 gnome-pty-helpe
2088 pts/12  00:00:00 bash
2139 ?      00:00:00 update-notifier
2167 ?      00:00:01 eog
2174 ?      00:00:00 deja-dup-monito
2306 ?      00:00:00 kworker/u4:2
2335 pts/12  00:00:00 ejercicio2-2
2343 pts/0    00:00:00 bash
..'
```


Programa 2.3: En este programa imprimimos las variables de entorno con ayuda de environ.

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
GNU nano 2.2.6 Archivo: programa2-3.c

#include<stdlib.h>
#include<stdio.h>

extern char **environ;
int main(int argc, char *argv[])
{
    int j;
    printf("Las variables de entorno para %s son:\n", argv[0]);
    for(j=0; environ[j] != NULL; j++)
        printf("environ[%d] = %s\n", j, environ[j]);
    return(0);
}
```

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./programa2-3
Las variables de entorno para ./programa2-3 son:
environ[0] = XDG_VTNR=7
environ[1] = XDG_SESSION_ID=c2
environ[2] = CLUTTER_IM_MODULE=xim
environ[3] = SELINUX_INIT=YES
environ[4] = XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/luzyong
environ[5] = SESSION=ubuntu
environ[6] = GPG_AGENT_INFO=/run/user/1000/keyring-enQK7H/gpg:0:1
environ[7] = TERM=xterm
environ[8] = SHELL=/bin/bash
environ[9] = XDG_MENU_PREFIX=gnome-
environ[10] = VTE_VERSION=3409
environ[11] = WINDOWID=58720267
environ[12] = UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1543
environ[13] = GNOME_KEYRING_CONTROL=/run/user/1000/keyring-enQK7H
environ[14] = GTK_MODULES=overlay-scrollbar:unity-gtk-module
environ[15] = USER=luzyong
environ[16] = LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42
:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:
*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:
*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:
*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;
31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=
01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:
*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01
```

- Pregunta 2.4: ¿En qué archivo se encuentra definida la variable `environ`?
En el archivo o librería `unistd.h`
- Pregunta 2.5: ¿Qué significan las variables de entorno: `HOME`, `LOGNAME`, `PATH`, `TERM`, `PWD`?
`HOME`: Es el directorio del usuario logeado.
`LOGNAME`: Es el nombre del usuario logeado.
`PATH`: Son las ubicaciones de los directorios en donde se encuentran los archivos ejecutables como comandos o programas.
`TERM`: Tiene la información del diseño definido de la terminal.
`PWD`: Es el directorio de trabajo actual.

```

USER    The name of the logged-in user (used by some BSD-derived pro-
          grams).

LOGNAME
          The name of the logged-in user (used by some System-V derived
          programs).

HOME    A user's login directory, set by login(1) from the password file
          passwd(5).

LANG    The name of a locale to use for locale categories when not over-
          ridden by LC_ALL or more specific environment variables like
          LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC,
          LC_TIME, cf. locale(5).

PATH    The sequence of directory prefixes that sh(1) and many other
          programs apply in searching for a file known by an incomplete
          pathname. The prefixes are separated by ':'. (Similarly one
          has CDPATH used by some shells to find the target of a change
          directory command, MANPATH used by man(1) to find manual pages,
          and so on)

PWD     The current working directory. Set by some shells.

SHELL   The pathname of the user's login shell.

TERM    The terminal type for which output is to be prepared.

```

Ejercicio 2.3: Usando la función `getenv()` modificamos el programa 2.3 para que muestre el valor de la variable `HOME`.



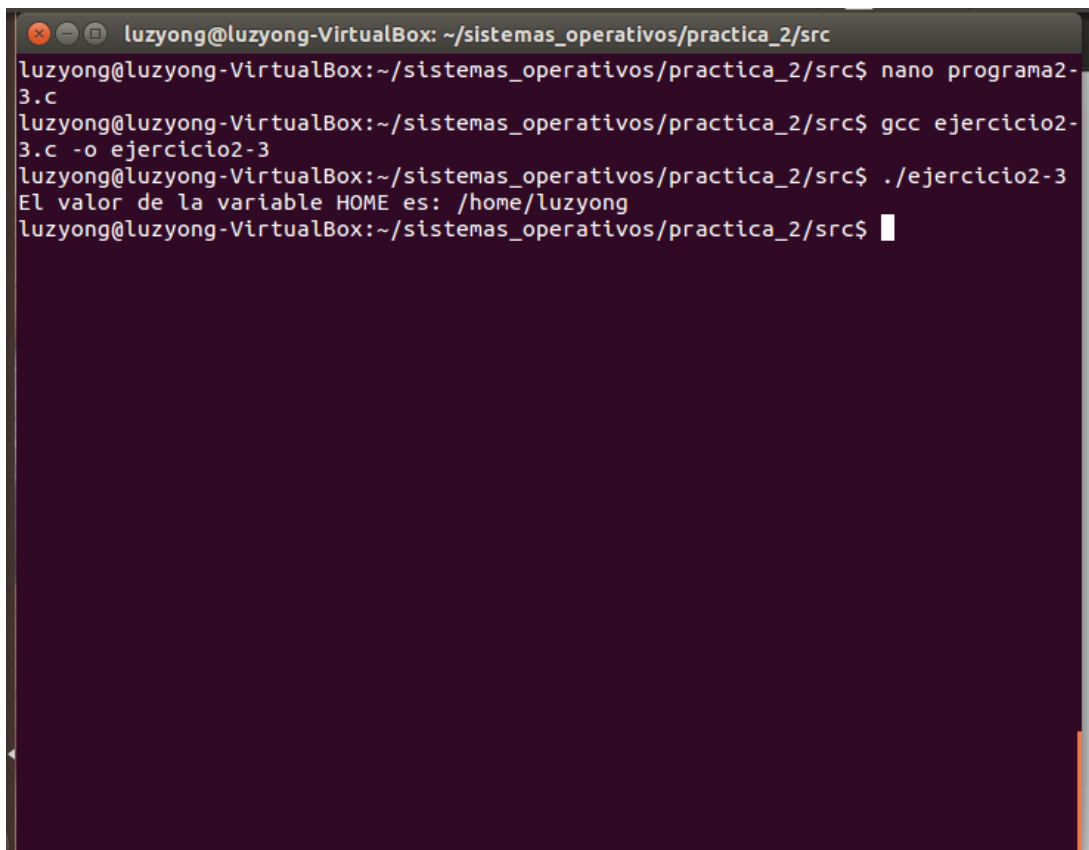
```
GNU nano 2.2.6 Archivo: ejercicio2-3.c

#include<stdlib.h>
#include<stdio.h>

extern char **environ;
int main()
{
    printf("El valor de la variable HOME es: %s\n", getenv("HOME"));
    return(0);
}
```

[9 líneas leídas]

^G Ver ayuda ^O Guardar ^R Leer fich.^Y Pág. ant. ^K Cortar Tex ^C Posición
^X Salir ^J Justificar ^W Buscar ^V Pág. sig. ^U PegarTxt ^T Ortografía



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_2/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ nano programa2-3.c
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ gcc ejercicio2-3.c -o ejercicio2-3
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$ ./ejercicio2-3
El valor de la variable HOME es: /home/luzyong
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_2/src$
```

Conclusiones.

En esta práctica aprendí la diferencia entre usuario real y usuario efectivo.

También visualicé las variables de ambiente y sus papeles en el sistema. Vi las diferentes ID que asigna al sistema de acuerdo con el rol que se desempeña (usuario o proceso). Reforcé conceptos como padre e hijo en procesos y comprendí la forma en que actúan de manera práctica.

Investigué un poco más de lo que pedía la práctica y me encontré con un concepto nuevo llamado SUID, aunque todavía me quedan algunas dudas, pude comprender a grandes rasgos su funcionalidad y relacionarla con los ejercicios de esta práctica.