

Benemérita Universidad Autónoma de Puebla  
Facultad de Ciencias de la Computación

# SISTEMAS OPERATIVOS I

---

PRACTICA 5: PROCESOS



Docente:

Prof. Marcos González Flores

Alumno:

Jesús Huerta Aguilar

Alex Abdiel Ruano Flores

Matricula:

202041509

202075025

NRC: 46152

Sección: 003

QUINTO SEMESTRE

Puebla, Pue.

28/02/2022

Practica No.5

Nombre: Procesos

Objetivo: Aprender a crear procesos sincronizarlos e identificarlos

Desarrollo:

EJERCICIO 1:

```
#include <stdio.h>
#include <unistd.h>

int var=50;

int main() {
    pid_t pidC;
    printf("** Proceso PID = %d comienza **\n", getpid());
    pidC = fork();
    printf("Proceso PID = %d, pidC = %d ejecutándose\n",getpid(),pidC);
    if(pidC > 0) {
        var = 60;
    }
    else if(pidC == 0){
        var = 40;
    }
    while(1) {
        sleep(2);
        printf("Procedimiento PID %d, var = %d ejecutándose\n",getpid(),var);
    }
}
```

TRANSCRITO + EJECUCIÓN:



```
alumno@alumno-VirtualBox: ~
GNU nano 6.2 proceso.c
#include <stdio.h>
#include <unistd.h>

int var=50;

int main(){
    pid_t pidC;
    printf("** Proceso PID = %d comienza **\n", getpid());
    pidC = fork();
    printf("Proceso PID = %d, pidC = %d ejecutándose\n",getpid(),pidC);
    if(pidC > 0){
        var = 60;
    }
    else if(pidC == 0){
        var = 40;
    }
    while(1){
        sleep(2);
        printf("Procedimiento PID %d, var = %d ejecutándose\n",getpid(),var);
    }
}
```

22 líneas leídas

^G Ayuda	^O Guardar	^W Buscar	^K Cortar	^T Ejecutar	^C Ubicación	M-U Deshacer
^X Salir	^R Leer fich.	^E Reemplazar	^U Pegar	^J Justificar	^_/ Ir a línea	M-E Rehacer

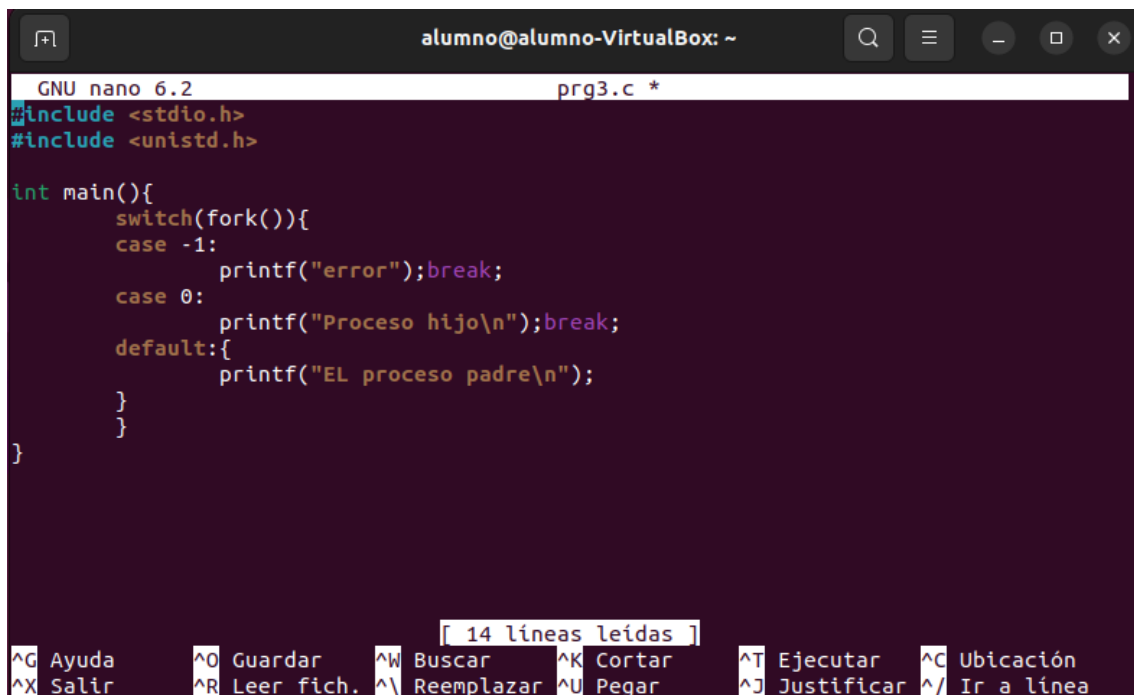


EJERCICIO 2:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    switch(fork()){
        case -1:
            printf("error"); break;
        case 0:
            printf("Proceso hijo\n"); break;
        default:{
            printf("EL proceso padre\n");
        }
    }
}
```

TRANSCRITO + EJECUCIÓN:

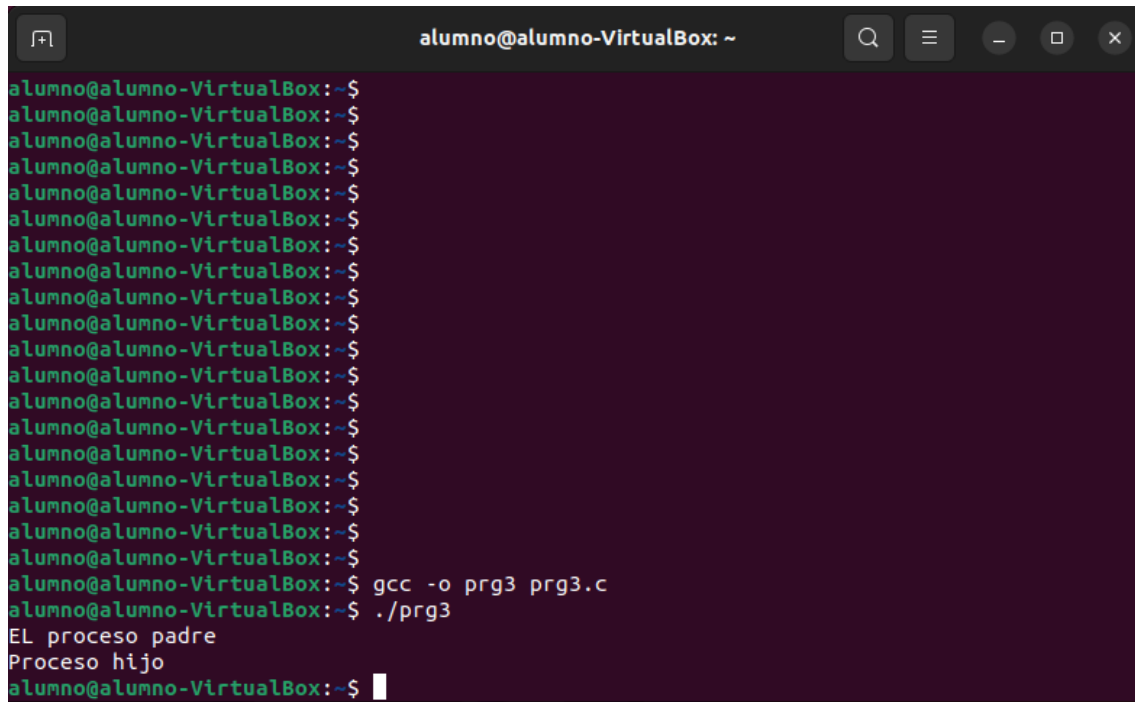


The screenshot shows a terminal window titled 'alumno@alumno-VirtualBox: ~'. Inside, the GNU nano 6.2 editor is open, editing a file named 'prg3.c'. The code in the editor is the same as in the previous block. The terminal output shows the program's execution: 'Proceso hijo' followed by a newline, and then 'EL proceso padre' followed by a newline. At the bottom of the terminal, a status bar indicates '14 líneas leídas' and provides keyboard shortcuts for various editor functions.

```
GNU nano 6.2 prg3.c *
#include <stdio.h>
#include <unistd.h>

int main(){
    switch(fork()){
        case -1:
            printf("error");break;
        case 0:
            printf("Proceso hijo\n");break;
        default:{
            printf("EL proceso padre\n");
        }
    }
}

[ 14 líneas leídas ]
^G Ayuda      ^O Guardar    ^W Buscar     ^K Cortar     ^T Ejecutar   ^C Ubicación
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar      ^J Justificar ^_ Ir a línea
```



```
alumno@alumno-VirtualBox: ~  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$ gcc -o prg3 prg3.c  
alumno@alumno-VirtualBox:~$ ./prg3  
EL proceso padre  
Proceso hijo  
alumno@alumno-VirtualBox:~$
```

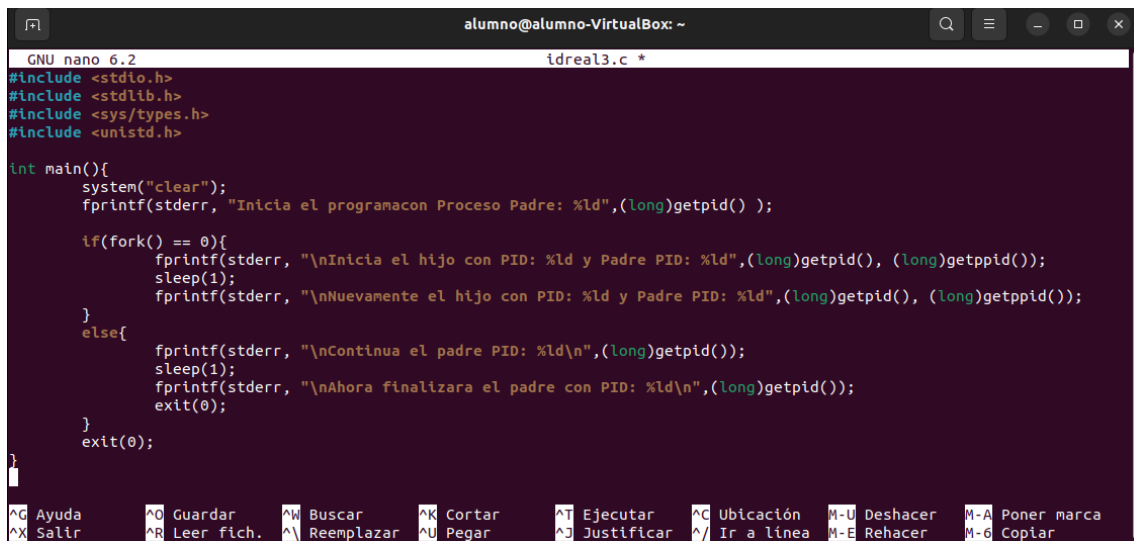
### EJERCICIO 3:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    system("clear");
    fprintf(stderr, "Inicia el programa con Proceso Padre: %ld", (long)
getpid());

    if(fork() == 0){
        fprintf(stderr, "\nInicia el hijo con PID: %ld y Padre PID:
%ld", (long)getpid(), (long)getppid());
        sleep (1);
        fprintf(stderr, "\nNuevamente el hijo con PID: %ld y Padre PID:
%ld", (long)getpid(), (long)getppid());
    }
    else{
        fprintf(stderr, "\nContinúa el padre PID:
%ld\n", (long)getpid());
        sleep (1);
        fprintf(stderr, "\nAhora finalizará el padre con PID:
%ld\n", (long)getpid());
        exit(0);
    }
    exit(0);
}
```

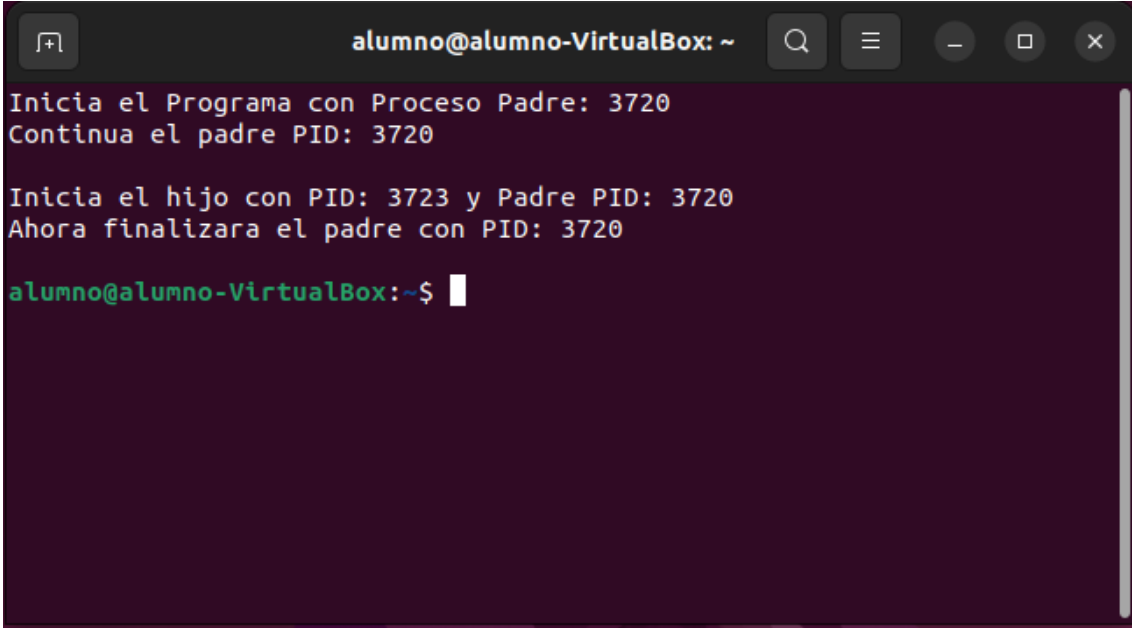
### TRANSCRITO + EJECUCIÓN:



```
alumno@alumno-VirtualBox: ~
GNU nano 6.2 idreal3.c *
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
    system("clear");
    fprintf(stderr, "Inicia el programa con Proceso Padre: %ld", (long) getpid() );

    if(fork() == 0){
        fprintf(stderr, "\nInicia el hijo con PID: %ld y Padre PID: %ld", (long) getpid(), (long) getppid());
        sleep(1);
        fprintf(stderr, "\nNuevamente el hijo con PID: %ld y Padre PID: %ld", (long) getpid(), (long) getppid());
    }
    else{
        fprintf(stderr, "\nContinúa el padre PID: %ld\n", (long) getpid());
        sleep(1);
        fprintf(stderr, "\nAhora finalizará el padre con PID: %ld\n", (long) getpid());
        exit(0);
    }
    exit(0);
}
```

A terminal window titled 'alumno@alumno-VirtualBox: ~' with standard window controls. The terminal output shows a program starting with parent PID 3720, spawning a child with PID 3723, and then terminating the parent process. The prompt is 'alumno@alumno-VirtualBox:~\$' with a cursor.

```
alumno@alumno-VirtualBox: ~  
Inicia el Programa con Proceso Padre: 3720  
Continua el padre PID: 3720  
  
Inicia el hijo con PID: 3723 y Padre PID: 3720  
Ahora finalizara el padre con PID: 3720  
  
alumno@alumno-VirtualBox:~$
```

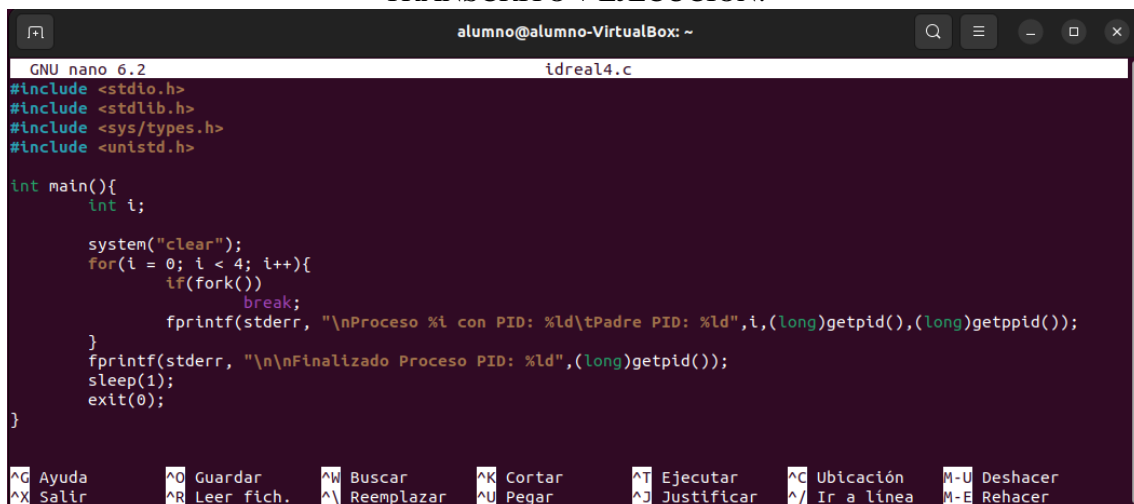
EJERCICIO 4:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    int i;

    system("clear");
    for (i = 0; i < 4; i++){
        if(fork())
            break;
        fprintf(stderr, "\nProceso %i con PID: %ld\tPadre PID: %ld", i, (long) getpid(), (long) getppid());
    }
    fprintf(stderr, "\n\nFinalizado Proceso PID: %ld", (long) getpid());
    sleep(1);
    exit(0);
}
```

TRANSCRITO + EJECUCIÓN:



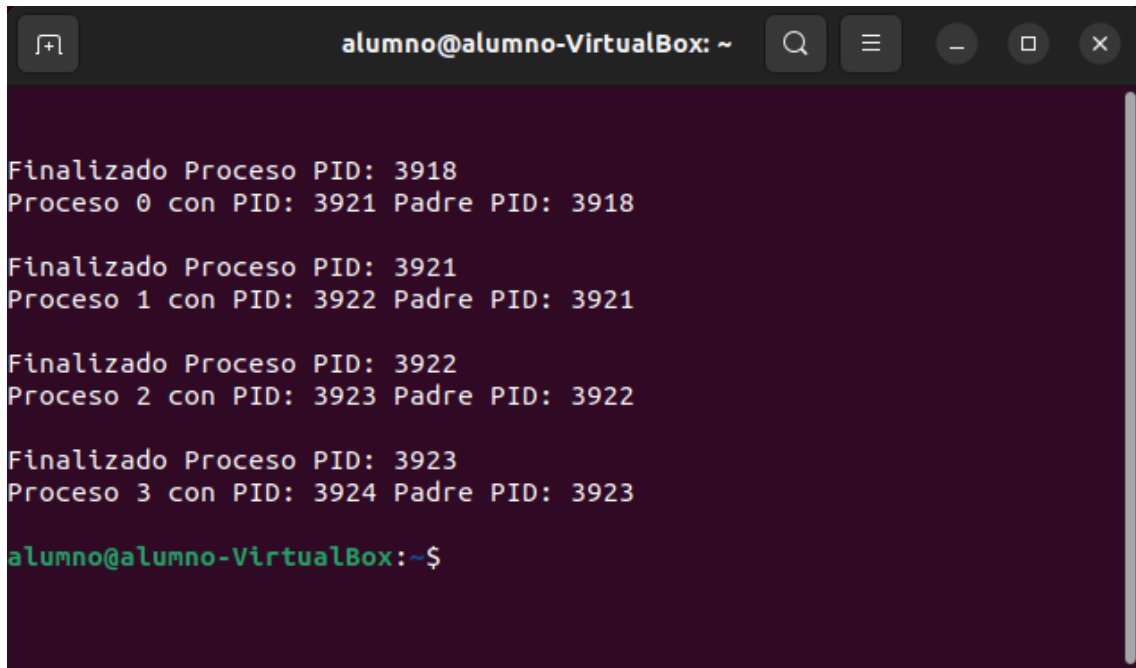
```
alumno@alumno-VirtualBox: ~
GNU nano 6.2 idreal4.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
    int i;

    system("clear");
    for(i = 0; i < 4; i++){
        if(fork())
            break;
        fprintf(stderr, "\nProceso %i con PID: %ld\tPadre PID: %ld", i, (long) getpid(), (long) getppid());
    }
    fprintf(stderr, "\n\nFinalizado Proceso PID: %ld", (long) getpid());
    sleep(1);
    exit(0);
}

^G Ayuda      ^O Guardar    ^W Buscar     ^K Cortar     ^T Ejecutar   ^C Ubicación  M-U Deshacer
^X Salir      ^R Leer fich. ^_ Reemplazar ^U Pegar      ^J Justificar ^_/ Ir a línea   M-E Rehacer
```



A terminal window titled 'alumno@alumno-VirtualBox: ~' with standard window controls. The terminal output shows four sequential process completions. Each entry consists of two lines: 'Finalizado Proceso PID: [value]' and 'Proceso [id] con PID: [value] Padre PID: [value]'. The values for PID and parent PID increase by 1 for each subsequent process. The terminal ends with a green prompt 'alumno@alumno-VirtualBox:~\$'.

```
alumno@alumno-VirtualBox: ~  
Finalizado Proceso PID: 3918  
Proceso 0 con PID: 3921 Padre PID: 3918  
  
Finalizado Proceso PID: 3921  
Proceso 1 con PID: 3922 Padre PID: 3921  
  
Finalizado Proceso PID: 3922  
Proceso 2 con PID: 3923 Padre PID: 3922  
  
Finalizado Proceso PID: 3923  
Proceso 3 con PID: 3924 Padre PID: 3923  
  
alumno@alumno-VirtualBox:~$
```