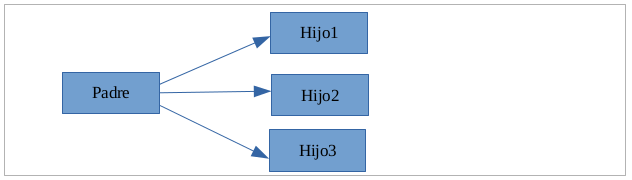
**ENUNCIADO**

**Actividad 4**

Haz un programa C que genere una estructura de procesos con un padre y 3 hijos. Visualiza por cada hijo su PID y el del padre. Visualiza también el PID del padre de todos.



**Resultado esperado:**

$./actividad4

Soy el hijo 3, Mi padre es 2186 y mi PID es 3077

Soy el hijo 2, Mi padre es 2186 y mi PID es 3076

Soy el hijo 1, Mi padre es 2186 y mi PID es 3075

Proceso padre 2186

O bien:

$./actividad4

Soy el hijo 1, Mi padre es 2186 y mi PID es 3075

Soy el hijo 2, Mi padre es 2186 y mi PID es 3076

Soy el hijo 3, Mi padre es 2186 y mi PID es 3077

Proceso padre 2186

**Solución:**

// ejercicio4

// Joseba Martinez

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main ()

{

pid\_t pid, pid2, pid3;

int status1, status2, status3;

// No se crea hijo 1

if ((pid=fork())== -1){

printf("No se ha podido crear el primer proceso hijo...");

exit (-1);

// Primer hijo

} else if (pid == 0) {

printf ("Soy el hijo 1, mi PID es %d, El PID de mi padre es: %d.\n", getpid(), getppid());

} else {

// Segundo hijo

if ((pid2 = fork()) == -1){

printf("No se ha podido crear el primer proceso hijo...");

exit (-1);

} else if (pid2 == 0){

printf ("Soy el hijo 2, mi PID es %d, El PID de mi padre es: %d.\n", getpid(), getppid());

}else{

// Tercer hijo

if ((pid3 = fork()) == -1){

printf("No se ha podido crear el primer proceso hijo...");

exit (-1);

}else if (pid3==0){

printf ("Soy el hijo 3, mi PID es %d, El PID de mi padre es: %d.\n", getpid(), getppid());

}else{

// Padre: espera a que sus tres hijos termine

waitpid(pid,&status1,0);

waitpid(pid2,&status2,0);

waitpid(pid3,&status3,0);

printf ("Soy el proceso padre %d. \n", getpid());

}

}

}

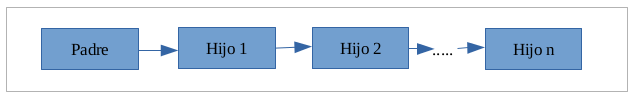
exit(0);

}

**Resolución de problemas:**

**Actividad 5**

Haz un programa C que genere la siguiente estructura de procesos:



**Solución:**

// ejercicio5

// Joseba Martinez

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main (){

pid\_t pid;

int status;

int n=3; //Numero de procesos

for (int i=0;i<n;i++){

if ((pid=fork()) == -1){

exit(-1);

}else if (pid==0){

printf("Soy el hijo %d, mi padre es %d \n",getpid(),getppid());

}else{

waitpid(pid,&status,0);

exit(0);

}

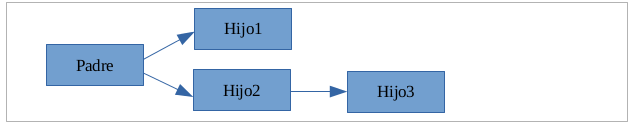
}

}

**Resolución de problemas:**

**Actividad 6**

Haz un programa C que genere la siguiente estructura de procesos:



Resultado esperado:

$./actividad6

Yo soy el hijo 2, mi padre es PID= 3410, yo soy PID= 3412  
Yo soy el hijo 1, mi padre es PID= 3410, yo soy PID= 3411  
Yo soy el hijo 3, mi padre es PID= 3412, yo soy PID= 3413

**Solución:**

// ejercicio6

// Joseba Martinez

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main (){

pid\_t pid,pid2;

int status;

//Primer Hijo

if ((pid=fork()) == -1){

exit(-1);

}else if (pid==0){

printf("Soy el hijo 1, mi padre es PID=%d, yo soy PID=%d\n",getppid(),getpid());

}else{

// Segundo Hijo

if ((pid2=fork()) == -1){

exit(-1);

}else if(pid2==0){

printf("Soy el hijo 2, mi padre es PID=%d, yo soy PID=%d\n",getppid(),getpid());

// Tercer Hijo

if((pid=fork()) == -1){

exit(-1);

}else if (pid == 0){

printf("Soy el hijo 3, mi padre es PID=%d, yo soy PID=%d\n",getppid(),getpid());

}else{

}

}else{

waitpid(pid,&status,0);

}

}

exit(0);

}

**Resolución de problemas:**

**Actividad 7**

Realiza un programa en C que cree un proceso (tendremos un proceso padre y otro hijo). El programa definirá una variable entera y le dará un valor 6. El proceso padre incrementará dicho valor en 5 y el hijo restará 5.

Resultado esperado:

$./actividad7

Valor inicial de la variable: 6

Variable en Proceso Hijo: 1

Variable en Proceso Padre: 11

**Solución:**

// ejercicio7

// Joseba Martinez

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

int num=6;

int status;

pid\_t pid, Hijo\_pid;

void main (){

printf("Valor inicial de la variable: %d\n",num);

pid=fork();

// Hijo

if (pid == -1){

exit(-1);

}else if(pid == 0){

num=num-5;

printf("Variable en Proceso Hijo: %d\n",num);

}else{

// Padre

waitpid(pid,&status,0);

num=num+5;

printf("Variable en Proceso Padre: %d\n",num);

}

exit(0);

}

**Resolución de problemas:**

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Correcto funcionamiento de los programas desarrollados en C (35%).
3. Ajuste a los resultados esperados (35%).
4. Claridad del código C (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida).
6. Avisos o errores de compilación (-15%).
7. Nivel de comentarios en el código fuente (-15%).