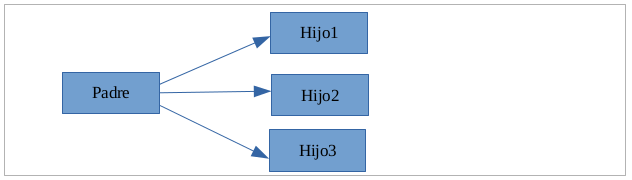
**ENUNCIADO**

**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:**

//actividad4.c

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main () {

pid\_t pid1, pid2, pid3;

//Creo el hijo 1

pid1 = fork();

if (pid1 == -1){

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

exit (-1);

}else if(pid1 == 0){

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

}else{

pid2 = fork();

if (pid2 == -1){

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

exit (-1);

}else if(pid2 == 0){

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

}else{

pid3 = fork();

if (pid3 == -1){

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

exit (-1);

}else if(pid3 == 0){

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

}else{

pid3 = wait(NULL);

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

exit(0);

}

}

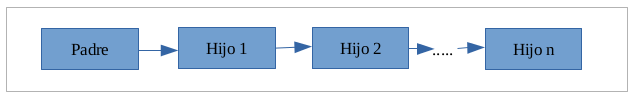
}

}

**Resolución de problemas:**

**Actividad 5**

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



Solución:

//actividad5.c

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main () {

pid\_t pid;

for(int i = 0; i<5; i++){

pid = fork();

if (pid == -1){//Error al crear el proceso

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

exit (-1);

}else if(pid == 0){//En el proceso hijo

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

}else{//En el proceso padre

pid = wait(NULL);

exit(0);

}

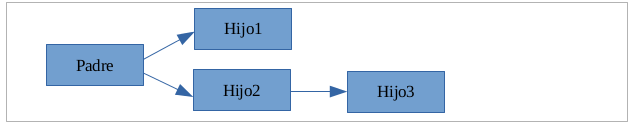
}

}

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:

//actividad6.c

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main () {

pid\_t pid1, pid2, pid3;

//Creo el hijo 1

pid1 = fork();

if (pid1 == -1){//Error al crear el proceso

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

exit (-1);

}else if(pid1 == 0){//En el proceso hijo

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

}else{//En el proceso pade

wait(NULL);

pid2 = fork();

if (pid2 == -1){//Error al crear el proceso

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

exit (-1);

}else if(pid2 == 0){//En el proceso hijo

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

//Creo el hijo 3

wait(NULL);

pid3 = fork();

if (pid3 == -1){//Error al crear el proceso

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

exit (-1);

}else if(pid3 == 0){//En el proceso hijo

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

}else{

wait(NULL);

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:

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main (){

pid\_t pid;

int i = 6;

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

pid = fork();

if (pid == -1){//Error al crear el proceso

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

exit (-1);

}else if(pid == 0){//En el proceso hijo

i -= 5;

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

}else{//En el proceso padre

i += 5;

pid = wait(NULL);

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

}

}

Resolución de problemas:

**REQUISITOS**

1. Entregar la solución de las actividades en los siguientes formatos: un único archivo de extensión .odt o .docx basado en el enunciado de las actividades y tantos archivos de código fuente como se hayan generado.
2. Comprimir todos los archivos de las soluciones en un único fichero, cuyo nombre estará formado por el nombre original de la actividad, seguido de un guión bajo y del primer apellido del alumno.

**CRITERIOS DE CORRECCIÓN**

1. Nivel de cumplimiento de los Requisitos (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%).