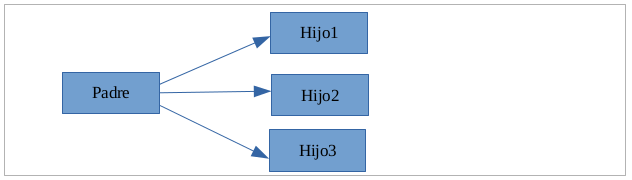
**ENUNCIADO**

**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:**

#include <stdio.h>

#include <unistd.h>

int main() {

pid\_t pid\_padre = getpid(); // Obtenemos el PID del padre antes de crear los hijos

pid\_t pid1, pid2, pid3;

// Crear el primer hijo

pid1 = fork();

if (pid1 == 0) {

// Código del primer hijo

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

} else if (pid1 > 0) {

// Crear el segundo hijo

pid2 = fork();

if (pid2 == 0) {

// Código del segundo hijo

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

} else if (pid2 > 0) {

// Crear el tercer hijo

pid3 = fork();

if (pid3 == 0) {

// Código del tercer hijo

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

} else if (pid3 > 0) {

// Código del padre

printf("Proceso padre %d\n", pid\_padre);

} else {

fprintf(stderr, "Error al crear el tercer hijo\n");

}

} else {

fprintf(stderr, "Error al crear el segundo hijo\n");

}

} else {

fprintf(stderr, "Error al crear el primer hijo\n");

}

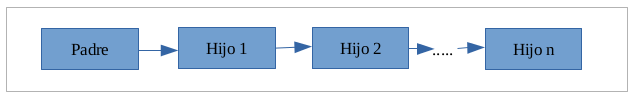
return 0;

}

**Resolución de problemas:**

**Actividad 5**

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



**Solución:**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

void crearHijos(int n, pid\_t padre) {

if (n <= 0) {

return;

}

pid\_t pid = fork();

if (pid == 0) {

printf("Yo soy el hijo %d, mi padre es PID= %d, yo soy PID= %d\n", 6 - n, padre, getpid());

crearHijos(n - 1, getpid());

\_exit(0);

} else if (pid < 0) {

fprintf(stderr, "Error al crear el hijo %d\n", 6 - n);

} else {

wait(NULL);

}

}

int main() {

int n = 6; // Cambia este valor para ajustar el número de hijos

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

crearHijos(n, getpid());

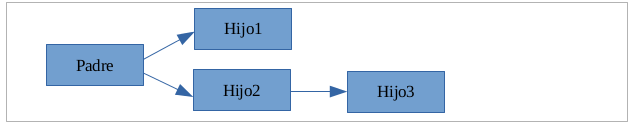
return 0;

}

**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:**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

pid\_t pid1, pid2, pid3;

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

pid1 = fork();

if (pid1 == 0) {

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

pid3 = fork();

if (pid3 == 0) {

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

}

wait(NULL);

} else {

pid2 = fork();

if (pid2 == 0) {

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

}

wait(NULL);

wait(NULL);

}

return 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 <stdio.h>

#include <unistd.h>

int main() {

int valor = 6;

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

pid\_t pid = fork();

if (pid == 0) {

valor -= 5;

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

} else if (pid > 0) {

valor += 5;

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

} else {

fprintf(stderr, "Error al crear el proceso hijo.\n");

}

return 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%).