Sistemas Operativos
Tarea: Procesos, IPC y llamadas al sistema

Objetivo General:
Conocer la creación de nuevos procesos, el uso de fork y exec, uso de librería Pthread y cuantificar las diferencias de tiempo en creación de tareas.

Entregable:
Deberá generar un .pdf que incluya una explicación de lo realizado. Adicionalmente deberá subir el código realizado de acuerdo a las siguientes actividades.

Actividad 1: Creación de procesos + IPC

- Escriba un programa que haga las veces de un **shell**, es decir:
  (1) Reciba por teclado un comando.
  (2) Lo ejecute.
  (3) Muestre la salida de su ejecución.
- El proceso **shell** debe de leer de la consola el nombre del programa a ejecutar que será el **comando**, y sus parámetros en el caso de tenerlos.
- Luego crear un proceso **comando** que cargue y ejecute el **comando**.
- Después de que se ejecute el proceso **comando**, el proceso **shell** será el que imprima el resultado.
- Al terminar el proceso **comando**, el proceso **shell** esperará por un nuevo ingreso por parte del usuario y la secuencia se repite.
- Si el usuario ingresa la palabra **exit** el proceso **shell** termina.
- El algoritmo se resume en la siguiente figura 1.

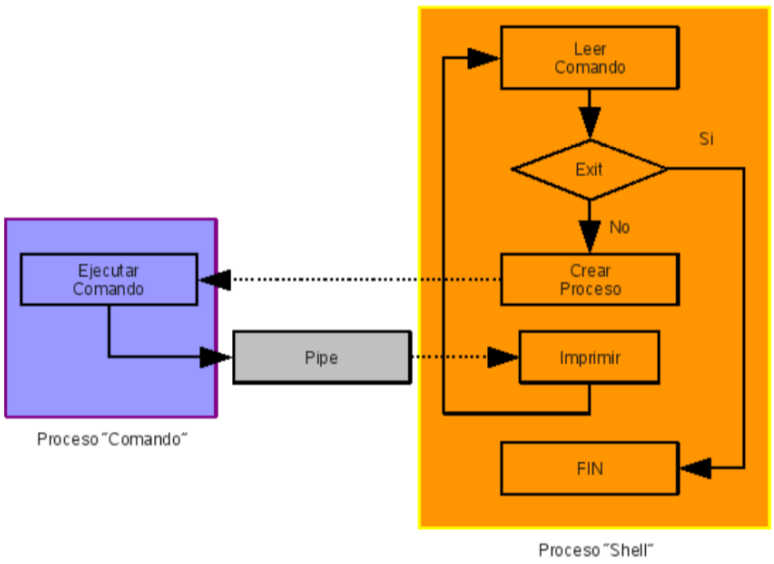

Figura 1. Esquema de una posible solución de la actividad 1.

Considere el uso de las llamadas al sistema: **fork**, **execvp**, **waitpid** para la manipulación de procesos. Recuerde que debe de hacer procesamiento de los parámetros para poder usar en execvp. Para la comunicación entre procesos **IPC** puede implementarse gestionando los descriptores de archivos correspondientes a **stdin** y **stdout** (Ver ejemplo de uso de **pipe** y **dup** en el repositorio de código del onedrive publicado en el aulavirtual. La carpeta unit2/use_pipes/demodup.c).

Actividad 2: Creación de procesos + Tiempos de ejecución

- Escriba dos programas que sirvan para mostrar las diferencias de tiempo entre la creación de tareas usando nuevos procesos o hilos.
- Los programas deberán medir el tiempo de cambio de contexto que transcurre desde que se hace la llamada para la creación de tarea (**fork/pthread_create**) hasta cuando la tarea se comienza a ejecutar.
- Se debe almacenar el tiempo antes de **fork/pthread_create** y restarlo del tiempo cuando se ejecute la primera instrucción del proceso **hijo/thread** correspondiente.
- Para generar los datos, el proceso inicial (**padre o primer hilo**) deberá poder generar múltiples tareas. El número de tareas a crear es un parámetro ingresado por el usuario.
- Luego que todos las tareas hayan terminado, deberá mostrarse el tiempo promedio en mili-segundos de los cambios de contexto.

Considere que en el caso de los hilos se puede usar una variable global para almacenar los tiempos. En el caso de los procesos hijos debe implementar un mecanismo de IPC (Un ejemplo puede ser los programas **pipe_unix.c** y **pipewfork.c** del unit2 del repositorio). Hay varias APIs que pueden usarse para medir tiempo por ejemplo: **gettimeofday**, **clock** y **getrusage**. Determine cuál es la mejor para resolver el problema y demuestre su uso. Qué se puede concluir de los resultados obtenidos:

1. Que **API** produce tiempos de cambio de contexto menores **fork** o **pthread create**?
2. Hay alguna diferencia si el número de **procesos** (N) aumenta?
3. Que **API** de las mencionadas arriba provee la mejor resolución de tiempo y porqué?

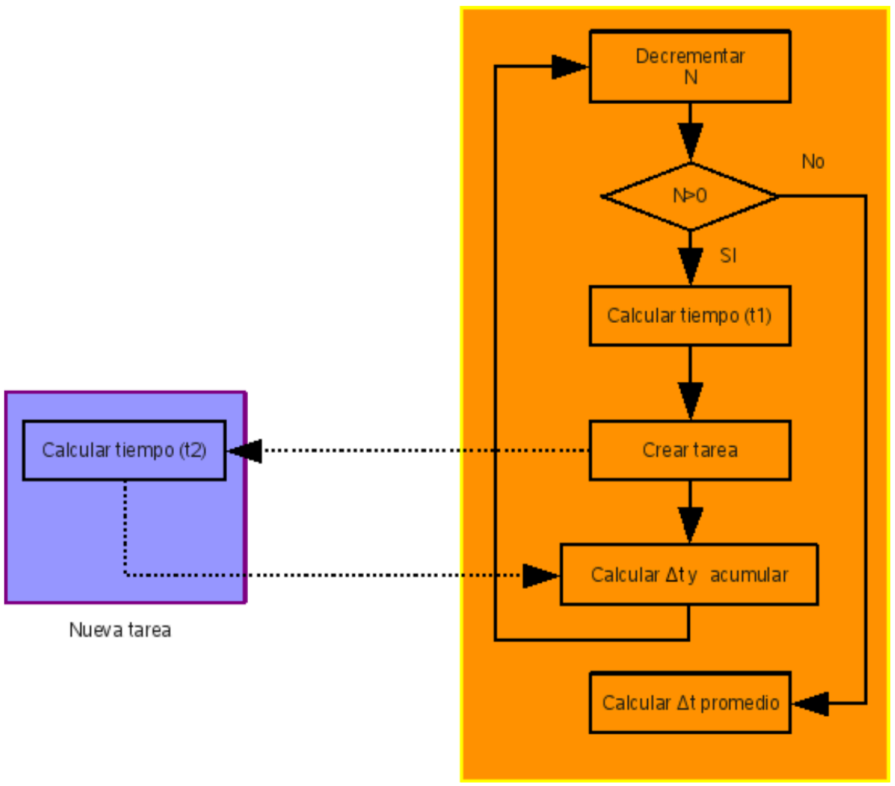

Figura 2. Esquema de una posible solución de la actividad 2.