**Ejercicio 1:** Estudia el código del fichero ejemploFork.c y responde a las siguientes preguntas:

¿Cuántos procesos se crean? Dibuja el árbol de procesos generado

Se crean 9 procesos.

¿Cuántos procesos hay como máximo simultáneamente activos?

Como máximo hay 7 procesos simultáneamente activos.

Durante la ejecución del código, ¿es posible que algún proceso quede en estado zombi? Intenta visualizar esa situación usando la herramienta top e introduciendo llamadas a sleep () en el código donde consideres oportuno.

En este caso no se queda ningún proceso en estado zombi.

¿Cómo cambia el comportamiento si la variable p\_heap no se emplaza en el heap mediante una llamada a malloc () sino que se declara como una variable global de tipo int?

No cambia porque los procesos no comparten memoria.

¿Cómo cambia el comportamiento si la llamada a open la realiza cada proceso en lugar de una sola vez el proceso original?

No hay duplicación.

En el código original, ¿es posible que alguno de los procesos creados acabe siendo hijo del proceso init (PID=1)? Intenta visualizar esa situación mediante top, modificando el código proporcionado si es preciso.

Solo es posible si muere el padre antes de que los hijos terminen.

**Ejemplo 1. Suma:** El código suma\_parcial1.c lanza dos hilos encargados de colaborar en el cálculo del siguiente sumatorio:

suma\_total (pdf)

Después de ejecutarlo varias veces observamos que no siempre ofrece el resultado correcto, ¿Por qué? En caso de no ser así, utiliza el ejemplo codificado en suma\_parcial2.c y observa que nunca obtenemos el resultado correcto. ¿Por qué?

Porque antes de que termine de hacer la suma se envía el proceso al final de la cola, los demás procesos toman el valor tmp = 0, ya que es una variable global y le suman j, después el primer proceso vuelve a tomar el control.