MEMORIA PRÁCTICA 2

FECHA DE ENTREGA: DEL 27 DE MARZO AL 31 DE MARZO

DANIEL DENEB MECHA MARTÍN Y PEDRO JOSÉ ALEMAÑ QUILES

Para poder crear el sistema de votación multiproceso, ha sido esencial avanzar de modo que pudiéramos solucionar algunos problemas y luego otros, pero nunca abordar demasiados a la vez. Comenzamos asegurándonos de que la comunicación y la gestión de señales entre procesos era la adecuada, puesto que si el setup de los handlers o las reacciones efectivas no fueran las adecuadas, tendríamos una cascada de problemas más adelante.

Después gestionamos la escritura y lectura de los archivos que documentaban las votaciones. Tuvimos, durante el proceso, dudas acerca de cómo formatear la información de modo que pudiéramos parsearla adecuadamente. Que los votantes pudieran encontrar su casilla correspondiente y el candidato, devolver 'vaciar las urnas' tras el recuento. La mayoría de problemas que con que nos fuimos encontrando los resolvimos gracias a strtok y fseek.

Para poder randomizar los votos, necesitábamos que que cada proceso tuviera una semilla distinta y esto nos confundió por un tiempo. Llamábamos a srand(time()), pero al empezar estos prácticamente a la vez, recibían todos la misma semilla. De modo que llamamos a srand(time()+getpid)) que, al ser srand una función caótica (desconocemos su implementación, pero puede ser hash), aun recibiendo valores de entrada similares, puede producir evaluaciones muy distintas.

El mayor problema que hemos tenido, con diferencia, es el bloqueado de procesos por señales que ya llegaron previamente. Esto sucede porque los programas multiproceso tienen una secuencia de ejecución a menudo impredecible (conforme son más complejas las interacciones y la comunicación entre estos pasa a ser más complicada la buena organización de semáforos y demás elementos de gestión de tráfico). Tanto es así que cuando un votante recibe una señal y espera a recibir la siguiente hay una ventana de tiempo en que la puede recibir antes de ejecutar la suspensión (espera inactiva) que solo se activará cuando llegue. De modo, que conforme se ejecuta más veces el programa en una misma iteración, más aumentan las probabilidades de que esto suceda.

De hecho, haciendo pruebas teníamos muchos 'procesos fantasma'. Procesos que logeaban sus resultados en ejecuciones a las que no pertenecen, pero que, al seguir a la espera de recibir determinadas señales, nunca se desbloqueaban antes de la finalización de su ejecución. Hemos podido resolver muchos de estos problemas mediante el uso de procmask, pero consideramos que sigue siendo un asunto muy complejo el que todas las interacciones sucedan siempre de la manera adecuada.