UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

MEMORIA DE LA PRÁCTICA 2

SISTEMAS OPERATIVOS

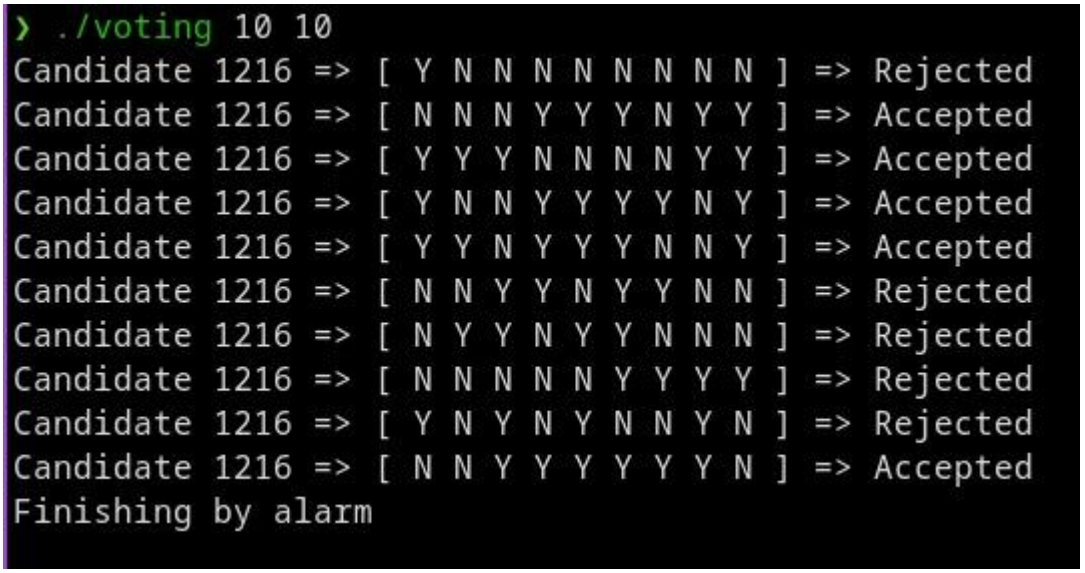
Sergio Larriba Moreno
Miguel Lozano Alonso

Requisitos:

- ☑ Creación de múltiples hijos Votante para realizar la votación.
- ☑ Recepción correcta de la señal SIGINT sobre el proceso Principal.
- ☑ Envío correcto de la señal SIGTERM hacia los hijos Votante finalizando así su ejecución.
- ☑ Recepción correcta de la señal SIGUSR1 sobre los procesos Votante.
- ☑ Gestión correcta del proceso Candidato (aquel que llegue primero).
- ☑ Votación correcta de los procesos Votante sobre el Candidato.
- ☑ Envío adecuado de la señal SIGUSR2 a los procesos Votante una vez finalizada la votación.
- ☑ Generación aleatoria de los votos.
- ☑ Buena gestión de las múltiples rondas de votación.
- ☑ Implementación correcta de sigsuspend.
- ☑ Temporización implementada adecuadamente.
- ☑ Correcta sincronización y liberación de semáforos.

Pruebas realizadas:

- ☑ Ejecución del programa para valores incorrectos para comprobar la funcionalidad de los diferentes controles de errores y su liberación de recursos
- ☑ Ejecución del programa con diferentes valores de entrada. Uno de los muchos resultados es el siguiente:



Nota: En la ejecución del código superior hemos cambiado la frecuencia de los votos a un voto por segundo para que la salida cupiera entera en la screenshot.

Explicación de las funciones desarrolladas:

- Archivo voting.c:
  - ➔ void handlerSIGINT(): manejador de la señal SIGINT.

- ➔ void handlerSIGALARM(): manejador de la señal SIGALRM
- ➔ main():
    1. Preparación de las señales correspondientes
    2. Creación de un array dinámico para almacenar los pid de los procesos hijo
    3. Inicialización de los semáforos correspondientes
    4. Creación de los procesos hijo, cuyos datos estarán almacenados en ficheros.
    5. Envío y recepción de señales.
    6. Finalización de los procesos hijo.
    7. Liberación y cierre de los recursos

- Archivo voter.c:
    - ➔ void sigusr1_handler(): manejador de la señal SIGUSR1.
    - ➔ void sigusr2_handler(): manejador de la señal SIGUSR2.
        1. Realización de la votación de los múltiples procesos, de uno en uno gracias a los semáforos.
    - ➔ void voter():
        1. Recepción del proceso Candidato
        2. Comienzo de la votación mediante la señal SIGUSR2
        3. Recuento de votos
        4. Liberación de recursos

Análisis de la ejecución:

¿Se produce algún problema de concurrencia en el sistema descrito?

- Sí, es posible que se produzcan problemas de concurrencia en el sistema descrito debido a que varios procesos Votante pueden competir por recursos compartidos al mismo tiempo.
- En particular, puede haber una condición de carrera cuando varios procesos intentan convertirse en el proceso Candidato al mismo tiempo, lo que puede provocar una inconsistencia en el estado del sistema.
- Además, el acceso al fichero compartido para registrar votos puede provocar problemas de concurrencia si varios procesos intentan escribir en el mismo fichero al mismo tiempo. Esto puede dar lugar a resultados incorrectos o inconsistentes.
- Sin embargo, gracias a los semáforos y señales podemos evitar los problemas de concurrencia descritos.

¿Es sencillo, o siquiera factible, organizar un sistema de este tipo usando únicamente señales?

- Organizar un sistema de este tipo usando únicamente señales no es sencillo, ya que las señales tienen algunas limitaciones. Por ejemplo, no se garantiza que una señal sea entregada al proceso destinatario en el momento preciso en que fue enviada, y algunos sistemas operativos tienen restricciones sobre las señales que pueden ser enviadas o recibidas por un proceso.

- Sin embargo, es factible implementar el sistema utilizando señales. Para hacerlo, se necesitará un diseño cuidadoso que tome en cuenta las limitaciones de las señales y garantice que los procesos puedan comunicarse y coordinarse correctamente.
- Por ejemplo, para coordinar la competición por ser el proceso Candidato, podría utilizarse una señal de sincronización, como la señal SIGUSR1, para indicar que se ha iniciado una ronda de votación y que los procesos deben competir por ser el candidato. Los procesos podrían enviar señales a un proceso maestro que coordine la elección del candidato y almacene los votos.