Memoria Practica (III)

Programación de Servicios y Procesos | Practica_02

Que

Escribe una clase llamada Mayúsculas que haga lo siguiente:

Cree un proceso hijo.

El proceso padre y el proceso hijo se comunicarán de forma bidireccional utilizando streams.

El proceso padre leerá líneas de su entrada estándar y las enviará a la entrada estándar del hijo (utilizando el OutputSream del hijo).

El proceso hijo leerá el texto por su entrada estándar, lo transformará todo a letras mayúsculas y lo imprimirá por su salida estándar. Para realizar el programa hijo, se puede utilizar cualquier lenguaje de programación generando un ejecutable.

El padre imprimirá en pantalla lo que recibe del hijo a través del InputStream del mismo.

Para que

Para poder abordar la práctica tres de una manera más cómoda o, al menos, para planificarla de mejor manera.

Pseudocodigo

opcion.cpp

```
Incluir bibliotecas necesarias

Generar una semilla de inicio para la función de generación de números aleatorios
usando la hora actual (std::time) y convertirla a un entero sin signo (unsigned int)

Usar la semilla generada para inicializar el generador de números aleatorios
(std::srand)

Generar un número aleatorio entre 0 y 2 (inclusive) para representar la elección de
la máquina

Si el número aleatorio es 0, asignar "piedra" a la variable "opcion"
Si el número aleatorio es 1, asignar "papel" a la variable "opcion"
Si el número aleatorio es 2, asignar "tijeras" a la variable "opcion"

Mostrar en la salida estándar "Maquina: " seguido de la opción elegida por la máquina

Devolver el número aleatorio que representa la elección de la máquina
```

app_02.cpp

```
Clase Juego
  Función jugar
    Crear una tubería (pipe) llamada "fd"
    Crear un arreglo de opciones ("piedra", "papel", "tijeras")
    Crear un búfer "buf" de tamaño 1024
    Obtener el PID del proceso actual

    Si hay un error al crear la tubería
      Mostrar "Error: fallo en el pipe"
    Fin Si

    Si el PID es -1
      Mostrar "Error: fallo en el fork()"
    Sino si el PID es igual a 0 (proceso hijo)
      Cerrar el extremo de lectura de "fd"
      Ejecutar el comando "./opcion" y obtener el resultado
      Traducir el resultado a una de las opciones y escribirlo en la tubería
      Cerrar el extremo de escritura de "fd"
    De lo contrario (proceso padre)
      Cerrar el extremo de escritura de "fd"
      Leer la opción elegida por el usuario
      Leer la opción seleccionada por la máquina desde la tubería
      Cerrar el extremo de lectura de "fd"
      Calcular el resultado del juego
      Devolver el resultado
    Fin Si
  Fin Función

  Función esGanador
    Comparar las opciones seleccionadas por el usuario y la máquina
    Devolver 0 si el usuario gana, 2 si la máquina gana o 1 en caso de empate
  Fin Función
Fin Clase

Función principal main
  Crear una instancia de la clase Juego llamada "ej"
  Inicializar puntajes para el usuario (pUser) y la máquina (pCPU)

  Repetir 3 veces
    Jugar una ronda y obtener el resultado
    Actualizar puntajes según el resultado
    Mostrar el resultado de la ronda

  Determinar el ganador general basado en los puntajes
  Mostrar el resultado del juego (usuario gana, máquina gana o empate)
  Retornar 0
Fin Función
```

Como

Para empezar, en esta práctica he creado dos archivos: "opcion.cpp" y "app_02.cpp".

"opcion.cpp" es un programa que imprime una opción aleatoria (piedra, papel o tijeras) y devuelve su índice. Primero se declara la semilla de la función "srand" y luego se genera un número entero aleatorio entre 0 y 2. Luego, mediante una estructura "if", se asigna un valor a una cadena de caracteres que será impresa, dependiendo del valor generado. Finalmente, se devuelve ese valor.

"app_02.cpp" contiene una clase llamada "Juego" que tiene dos funciones, "jugar" (pública) y "esGanador" (privada). "esGanador()" devuelve un índice para determinar si es ganador (0), empate (1) o perdedor (2).

En la función "jugar", se declara un "pid_t" para el pipe, un array de dos posiciones que hace referencia a un file descriptor para el pipe, un array de caracteres que es el buffer y otro array de caracteres que almacena la opción del usuario. Primero, se crea el pipe y se maneja cualquier posible error. Luego, se llama a la función "fork" para crear procesos hijos, y se utiliza una estructura "if" anidada para calcular si ha ocurrido un error o si el proceso actual es el padre o el hijo.

Si se está en el proceso hijo, se cierra la lectura en el pipe, se ejecuta el programa "opcion" con la ayuda de la función "system", se divide su retorno entre 256 (esto se hace porque la función "system" devuelve el valor multiplicado por 256, ya que también incluye el estado de finalización del proceso en bits), y se guarda en una variable. Luego, se crea un array de caracteres donde se almacena una opción de entre las opciones disponibles, se escribe en el pipe y se cierra la escritura en el pipe.

Si se está en el proceso padre, se cierra la escritura en el pipe, se solicita al usuario que elija una opción, se almacena en una cadena, se lee desde el pipe, se cierra la lectura y se devuelve el valor que retorna la función "esGanador" con el buffer y la opción del usuario como parámetros.

En la función principal, se declara un objeto de tipo "Juego" y dos valores enteros que representan los puntos de la máquina o del usuario. Luego se crea un bucle con 3 iteraciones, en el cual se llama a la función "jugar" del objeto. Según el resultado, mediante estructuras "if", se determina quién ha ganado y se muestra el mensaje correspondiente, sumando un punto al ganador. En caso de empate, se muestra el mensaje correspondiente.

Después del bucle, se revisan los resultados y se muestra el mensaje correspondiente.

Conclusión

Esta práctica me ha resultado útil para realizar la práctica 3, además de comprender un poco mejor cómo se generan y funcionan los procesos en C