INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

LABORATORIO: Sistemas Operativos

Práctica 11

Cooperación entre procesos. Programación concurrente.

LosDos

Integrantes:

Amanda Velasco Gallardo - 154415
Carlos Octavio Ordaz Bernal - 158525

Fecha(s) de elaboración de la práctica:

12 de abril de 2019

Introducción

El `shell` es un intérprete de comandos. Es el encargado de validar las instrucciones introducidas por el usuario y cuenta con variables de ambiente que permiten personalizar el entorno de trabajo. Una característica que distingue a los `shell` del resto de los intérpretes es que éste es un lenguaje de programación que cuenta con instrucciones para la ejecución selectiva y repetitiva de grupos de comandos, cada programa escrito en un `shell` se le conoce como guión (*script*).

La principal ventaja de utilizar un *script* de `shell` es la capacidad de ejecutar instrucciones secuenciales, en vez de tener que estar ingresando una a una dentro de la línea de texto del intérprete de comandos. Además, facilita las tareas que se deben realizar ya que se pueden programar un conjunto de instrucciones de una sola vez.

Las variables de `shell` son aquellas que permiten personalizar el ambiente de trabajo. Existe un conjunto de variables predefinidas y el usuario es capaz de definir sus propias variables. Existen dos tipos de variables de `shell`: las booleanas y las que pueden recibir un valor. Además, existe otro tipo de variables conocidas como variables de entorno o globales que son heredadas a cualquier `subshell` creado. Si una variable de entorno y una variable de `shell` tienen el mismo nombre, la de mayor precedencia es la segunda.

La principal ventaja de utilizar `shell` es la personalización, cada intérprete puede contener variables definidas por el usuario dependiendo de las tareas que se vayan a realizar. Es posible decir que los `shell` vienen en varios sabores por lo diferentes que pueden ser en términos de la forma en que trabajan y sus variables definidas.

Desarrollo

Rehacer el ejercicio `Prac07LabSO19`, pero ahora `EdoCtaClientes.java` y `MovimientosCliente.java` deberán ser scripts de `Linux` en `tcsh`.
Al igual que en el ejercicio de la práctica 7, se programaron dos archivos: uno para los movimientos bancarios de cada cliente, y otro para generar los estados de cuenta individuales por cliente. Los archivos conservaron el mismo nombre, con la diferencia de ser *scripts* para el `shell tcsh`.

En primer lugar, el archivo `MovimientosCliente` genera por cada cliente un archivo de texto con su nombre, las claves de las transacciones realizadas,

el monto por operación, el número total de operaciones, y el monto total. Para ello, se leyó del archivo `movimientos.txt` cada una de las líneas, se comparó la clave del cliente actual con la clave del cliente de la operación, y si coincidían se acumulaba el monto, aumentaba el número de operaciones, y se añadía la operación al archivo de texto del cliente. Se utilizó la función `set` para fijar valores de variables, `echo` (con redirectores >>) para concatenar cadenas de texto al archivo de salida, y *pipes* para poder realizar operaciones aritméticas con `bc` pues sus valores no son enteros.

En segundo lugar, dentro del archivo `EdoCtaClientes` se mandaba a ejecutar en segundo plano, una instancia del programa `MovimientosCliente` por cada cliente que se encontrara al momento de la lectura del archivo `clientes.txt`. Además, finalizada su ejecución, se guardaba el valor que arroja como salida (`exit`) obtenido con `$status` para comprobar si se había creado o no el estado de cuenta de un determinado cliente. Se utilizó la función `set` para fijar valores de variables, `echo` (con redirectores >>) para concatenar cadenas de texto al archivo de salida, `wait` para esperar la terminación de todos los programas hijos, y `date` para obtener mediciones del tiempo en distintos formatos. Finalmente, se realizaba una impresión, en la misma línea de comandos de terminal, que informaba si algún estado de cuenta no fue creado y el tiempo de ejecución total.

Conclusiones

La presente práctica nos permitió conocer más a fondo distintos aspectos de `shell` y de *scripts* que se habían visto superficialmente con anterioridad. En particular, aprendimos cómo, desde el shell, crear procesos hijos que corran paralelamente (comando `&`), esperar a que finalicen su ejecución (`wait`), y revisar su *exit status* (`status`). Otro aspecto que ejercitamos en esta práctica es la estructura de control de flujo `foreach`, la cual utilizamos para leer líneas de un archivo de texto. Finalmente, un punto novedoso en esta práctica fue la necesidad de operar con números de punto flotante. Dichas operaciones no están soportadas en la aritmética básica del `shell`, por lo que fue necesario utilizar la calculadora `bc`.

Referencias

- Ríos, J. (2019). Notas del curso de Sistemas Operativos. Recuperado el 03 de abril de 2019, del sitio web: Comunidad ITAM.