Tarea Sistemas Operativos: MiniShell

Diego Rodriguez & Matías Medina & Cristóbal Donoso
Facultad de Ingeniería - Universidad de Concepción, Chile

Sistemas Operativos - Cecilia Hernandez
Martes 5 de abril del 2016

1 Introducción

Una terminal o shell es una de las formas que permite al usuario interactuar con la máquina por medio del Sistema Operativo (también están las GUI). Con la terminal el usuario tiene mayor control sobre una aplicación producto de las funciones y configuraciones que ésta entrega.
Usualmente no nos preguntamos cómo el sistema operativo maneja la ejecución de los comandos, o que hace internamente para administrar los procesos.
En el desarrollo de esta tarea implementaremos una MiniShell haciendo uso del lenguaje de bajo nivel C y librerias propias tales como: unistd.h o errno.h. El programa ejecuta comandos con un pipe y hasta tres argumentos. A continuación se explicará al lector los fundamentos y funciones de la MiniShell

2 Desarrollo

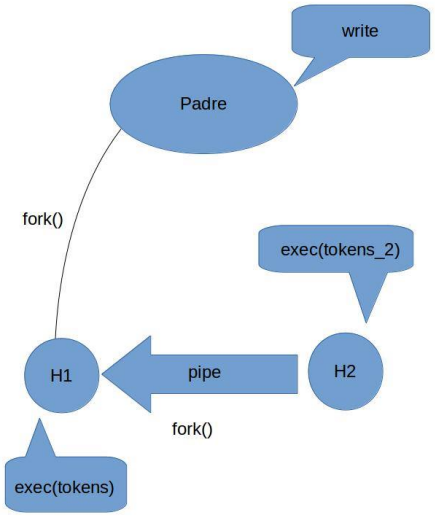
Lo primero que la minishell hace es parsear la entrada. Esto es, separar los comandos y sus argumentos en *tokens*, que posteriormente serán tratados en procesos correspondientes.
Cada comando ingresado es un conjunto de procesos (padre, hijo), cada proceso es generado con la llamada a sistema fork, por consiguiente, la presencia de dos comandos requerirá un trato especial a la hora de ejecutar sus acciones.

2.1 Sin pipe

Si el comando ejecutado no posee pipe, se crearán dos procesos (padre e hijo) iguales, sin embargo, será el hijo quien sobreescriba sus registros con la llamada a sistema exec, la cual ejecutará los comandos. Durante la actuación del proceso hijo el padre espera la salida del proceso para comenzar a ejecutar sus funciones.
La minishell genera un archivo de extensión .log la cual será utilizada más adelante. El proceso padre es quien guarda el tiempo de ejecución del comando ejecutado. Así mismo copia la salida en un archivo temporal oculto y posteriormente lo escribe en el archivo .log definitivo e imprime su contenido en la minishell.

2.2 Con pipe

Es necesario comunicar ambos lados del pipe. En UNIX la llamada a sistema pipe() nos permite conectar dos procesos con sus respectivos descriptores de archivos. Así, cuando un proceso A escribe, el proceso B puede leer esos datos desde su descriptor de archivo.



Por otro lado debemos hacer uso de la llamada a sistema fork() dos veces y redireccionar la salida de un proceso hacia la entrada del otro. Finalmente el *padre absoluto* termina la ejecución almacenando el tiempo, la salida, los comandos y argumentos en el archivo .log (similar al caso sin pipe).

2.3 Otras funciones de la shell

Hemos mencionado cómo se ejecutan los distintos comandos ingresados en esta shell, pero además de ejecutar comandos la minishell posee otras propiedades. Ingresando el comando "LIST_COMMAND" podemos ver todos los comandos ingresados en una sesión de la minishell.

Ingresando el comando "CHOOSE_COMMAND" podemos ver los comandos ingresados en una lista y elejir el que queramos para reejecutarlo.

La ejecución de estos comandos conciste en que guardamos en un log temporal sólamente los comandos ingresados, sin sus salidas. Luego leyendo ese archivo se pueden desplegar los distintos comandos ingresados en la sesión.

Para la reejecución de los comandos se parsea la entrada, en este caso el comando almacenado en el log y se ejecuta siguiendo el procedimiento mencionado anteriormente dependiendo si el comando utiliza pipe o no.

Ingresando el comando "SEARCH_COMMAND" buscamos un comando ingresado previamente y desplegamos todas sus instancias dentro de la sesión, es decir las distintas salidas que entregó ese comando al ejecutarse. Para ello, buscamos en el log los comandos ejecutados, que están marcados con un carácter "@" al inicio. Luego las siguientes lineas del log que corresponden a la salida entregada por el comando, son mostradas en pantalla.

Adicionalmente podemos ingresar "DELETE_LOG" para borrar el archivo log y "QUIT" para cerrar la minishell.

3 Conclusiones

La ejecución de comandos o programas no es algo trivial. El sistema operativo debe comunicarse con el usuario y viceversa de manera eficiente para comodidad de este último. Cuando ejecutamos un comando en una shell, ocurren distintos procesos internos que es fundamental conocer a la hora de desarrollar aplicaciones.

La creación y el manejo de procesos, usando llamadas a sistema como fork y pipe, y el redireccionamiento de entrada/salida usando las llamadas dup2, read y wite son herramientas importantes para la programación en bajo nivel y se hacen necesarias a la hora de programar estas aplicaciones.

Dimos a conocer al lector conceptos elementales con los que el sistema operativo lleva a cabo los procesos. Sin embargo, la ejecución de procesos se ve influida por más factores(estructuras de control, planificadores, etc) que en la actualidad se han resuelto para mejorar el rendimiento de las computadoras.