Universidad de Concepción

Ingenieria Civil Informatica

Proyecto 1:
Construir una shell simple

Autores:
Pablo Venegas
Carlos Von Plessing

Docente:
Cecilia Hernández

Agosto 31, 2017

1 Introducción

Mediante el presente documento se da a conocer el desarrollo e implementación de crear un intérprete de comandos simple en Linux (shell) simple a partir del lenguaje de programación C, buscando fomentar el manejo de los procesos concurrentes de Unix, tanto como de ejecución, creación, y terminación usando las llamadas al sistema correspondientes para cada caso además de las necesarias para manipular y obtener información de recursos dentro de la maquina en la que se está trabajando.
Principalmente el proyecto está dividido en dos grandes partes, la primera de ellas corresponde a la creación de una shell que acepte los comandos básicos de una consola de Linux, esto quiere decir que debe recibir esperar a recibir los comandos, parsear esta entrada e identificar el comando y sus argumentos, para luego de manera interna ejecutar el comando de manera concurrente, dado que esta shell será usada de forma foreground, esta esperara por el termino de la ejecución para imprimir el promtp para luego esperar por el siguiente comando de entrada.
La segunda parte de este proyecto consiste en incluir la llamada a sistema mmap(), la cual permite designar una memoria compartida entre el proceso padre e hijo, permitiendo de esta forma que ambos procesos comparta este espacio para comunicarse entre sí, permitiendo de esta manera evitar problemas de consistencia o concurrencia, dado que cada vez que la shell ejecute un comando el hijo registre este y el tiempo antes de ejecutarlo en la memoria compartida, para luego el padre espere por la orden de que el proceso hijo a terminado, dado que este registrara en la memoria compartida ese momento, dejando también la duración que le tomo al proceso hijo realizar dicho comando.

2 Desarrollo

A continuación se detalla el código creado para la tarea:
Primero imprime ShellmMapv4: y espera hasta que el usuario le de una entrada.
Al recibir una entrada esta la parsea primero dividiendo esta por cada "|" y luego con cada espacio, para los comandos. Ejemplo:
" ls –l | grep mar | wc –l "
Primero divide los 3 comandos en:
Ls –l
Grep mar
Wc –l
Y luego los separa por espacio en un arreglo bidimensional.
Command[0][0]= ls
Command[0][1]= -l
Command[1][0]= grep
Command[1][1]= mar
Command[2][0]= wc
Command[2][1]= -l
Luego, con la entrada parseada, si hay mas de un pipe (pcount>1) se "pipean" todos los elementos excepto el ultimo, dejando a los hijos los comandos Si no hay pipes en la entrada, solo crea 1 hijo para que ejecute el único comando que hay. Al final de cada iteración libera espacio y borra el arreglo para la siguiente iteración.

3 Conclusiones

El trabajo desarrollado en este informe nos ha demostrado la facilidad y la rapidez con la que se pueden simplificar ciertas operaciones que muchas veces como programadores, preferimos hacer un programa en cualquier lenguaje que nos permita hacer lo mismo que podemos hacer simplemente con un par de comandos en consola, además de lo fuerte que son los pipes, que nos permiten "anidar" distintos comandos en una sola línea de la consola, de manera que las llamadas a sistema se van ejecutando a una dentro de la "tubería", además al crear nuestra propia shell, se nos presenta una nueva forma de poder entender y analizar lo que pasa dentro del sistema operativo, viendo cómo se van creando los procesos, como hacer para evitar la concurrencia y además hacer que la consistencia dentro de cada ejecución sea la correcta y la que necesitamos.