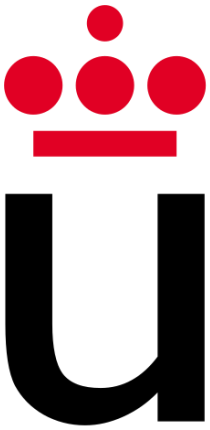

Universidad
Rey Juan Carlos

Práctica 2: Minishell

Realizado por:

-Sergio Fernández Gómez

-Alejandro Morales Martín

Índice

1.Introducción

En esta práctica el objetivo es implementar una minishell que ejecute mandatos de forma correcta, así como lo hace el terminal de Ubuntu con la que trabajamos en clase.

Trabajaremos con la implementación de comandos en *background* y *foreground* y comandos enlazados mediante el uso de *pipes,* además haremos más hincapié en los comandos *cd, jobs y fg.*

2.Comentarios acerca del código

El código contará con unas funciones básicas (aparte de la función principal *main)* para la funcionalidad correcta de la shell:

- *void ejecutarComando(tline * comando)* esta función ejecutará el comando solicitado en caso de ser uno solo y diferenciará si el comando se solicita ejecutar en *foreground o background*.

-*void ejecutarComandos(tline* comando*): esta función trabajará cuando haya dos o más comandos, trabajará con *pipes* (como hemos visto en clase) para comunicar la información entre procesos, además diferenciará si el comando que se quiere ejecutar es en *background o foreground.*

-*void cd (tline* comando)*: esta función ejecuta el comando cd de igual forma que en una shell normal, en caso de no tener argumentos, nos enviará al directorio HOME, en caso de tener argumentos, comprobará si el directorio existe y en caso afirmativo nos cambiará a ese directorio especificando la ruta completa.

-*void fg()*: esta función devolverá el proceso a su ejecución en *foreground, si no acabó en background,* trabajará con funciones como *tcsetgrp()* que se encarga de enviar el proceso al foreground.

-*void jobs():* esta función se encargará de la ejecución del comando *jobs* de forma similar a la shell de Ubuntu. En esta función trabajaremos con un tipo de dato definido por una struct que hemos declarado llamada "job", este struct será esencial para poder implementar el comando *jobs*.

El struct contará con los siguientes atributos: *pid_t* pidProcesoBackground que será el identificador del proceso, *tline** comandoBackground que devolverá lo analizado de la cadena de caracteres que se le pasa (struct facilitada en el enunciado de la práctica), *char** lineaComando guardará el nombre del comando y *struct _job** punteroSiguienteNodo guardará el puntero al siguiente objeto de tipo *job,* esto hace que que guardemos este tipo de objeto en una especie de lista enlazada para que sea más fácil acceder entre ellos.

-*void addJob(pid_t ppb, tline *c):* con esta función añadimos un nuevo objeto de tipo *job* a la lista enlazada.

-*void borrarJob()*: esta función se encargará de eliminar un objeto de tipo job de la lista enlazada mediante el uso de *free ()* ya que al añadir un *job* a la lista reservamos memoria mediante *malloc.*

-*int main(int argc, char* argv[]):* El main primeramente hará una comprobación del número de argumentos pasados por parámetro, siendo uno el único valor aceptado ya que tiene que ser únicamente el nombre de la minishell el parámetro que reciba.

Después haremos que las señales SIGQUIT y SIGINT las ignore como indica en la práctica. Y finalmente se probarán todas las funciones declaradas anteriormente.

3. Comentarios personales

La implementación de fg y jobs, desde nuestro punto de vista, no está al nivel de la asignatura, se necesitan más conocimientos para implementarlo correctamente, hemos buscado formas y hemos escogido y adaptado la que mejor nos parecía. El resto de la práctica corresponde perfectamente a la complejidad de una práctica sobre procesos. Respecto a la cantidad de tiempo invertida, habremos invertido un par de tardes más o menos.

Buena práctica si te quieres pelear con ella y aprender bien para sacar la mejor nota posible, pero limita un poco el hecho de que solo puedas sacar un 8 teniendo el resto de las cosas perfectas.