Fundamentos de los Sistemas Operativos
Ficha de entrega de práctica

**: campo obligatorio*
***IMPORTANTE:** esta ficha no debe superar las DOS PÁGINAS de extensión*

Grupo de prácticas*: 1 - 41
Miembro 1: Alejandro Vialard Santana
Miembro 2:

Número de la práctica*: 5	**Fecha de entrega*:** 19/05/2022

Descripción del trabajo realizado*
En esta práctica se realiza una consola de comandos, en la que el alumno debe de implementar métodos creando procesos para que realicen las diferentes instrucciones que se le pase por consola (en nuestro caso será una string de tamaño 1024). **NO ESTÁ PARA RETO, DEBIDO A QUE NO LOGRO QUE FUNCIONE CORRECTAMENTE.**
Para ello, en esta práctica he realizado los siguientes métodos:
-**checkError():** método que comprueba si ha ocurrido algún error en alguna parte.
-testArgs(): método que comprueba los argumentos pasados para inicializar la consola (en este casos sólo será uno ./shell).
-analizeLine(): método que comprueba la línea de comandos pasada por parámetro que se guardarán en una lista de argumentos (argList). Para ello se emplea la función **strtok()** que separa una string de caracteres mediante un delimitador, de tal manera que recorremos la string en un bucle donde se irán buscando los argumentos hasta encontrar un espacio o \n (nuestro delimitador), y cuando el carácter sea el de final de línea termina y habrá guardado todos los argumentos.
-analizeLinePipe(): (OJO, ESTE MÉTODO EN PRINCIPIO FUNCIONA, PERO COMO ES PARA EL RETO Y NO ESTÁ RESUELTO, NO ES NECESARIO MIRARLO) método similar al anterior pero también separa por "|" para detectar que es una pipe.
-lowerCase(): uno de los métodos que se tenía que cumplir. Este transforma los caracteres de la línea de comandos a minúscula para que funcione de manera "no case sensitive". Se recorre la longitud del argumento y se suma 32 para ponerlo minúscula.
-pipes(): (ESTE MÉTODO ES NECESARIO PARA EL RETO, PERO HE MIRADO INFORMACIÓN Y PROBADO MUCHAS VECES Y NO LOGRO QUE FUNCIONE, A NO SER QUE SEA OTRO MÉTODO DE TRATAMIENTO DE TUBERÍAS EL QUE NO FUNCIONA).
-execute(): método que sirve para ejecutar el comando. A este se le pasa la lista de argumentos a tratar, crea un nuevo proceso con fork() y se lo atribuye a un pid. Si es -1, lanza error; si no lo es, se ejecuta con excevp() el argumento. Luego se espera a que el proceso hijo termine con waitpid().
-main: función principal en la que se ejecutará todos los métodos necesarios para que funcione la consola. Primero se testea los argumentos para iniciar la consola. Luego usamos getlogin_r(), gethostname() para pillar el nombre de usuario y el host y poder imprimirlo para dar sensación de que estamos usando una consola. Para que funcione siempre, hacemos un bucle while(1) que terminará cuando escribamos "exit" por consola. Pillamos la cadena con fgets(), comprobamos si la línea de comandos está vacía, de manera que comprobamos si la string es igual a un \n, hacemos un continue y veremos que si le damos a enter en la consola, salta a la siguiente línea. La siguiente comprobación era para poder saber si hay una | y tratarla como tubería, pero al no funcionar el reto no es necesario, pero su funcionamiento sería que si detecta una barra vertical, se activa una condición a 1. Entonces, dependiendo del valor de la condición, se trata de una forma u otra, en este caso si es 1 se trata como tubería (que en este caso no funcionaría porque está comentado la función pipes(), que si quiere comprobar que ocurre solo debe descomentar)

Grupo de prácticas*: 1 - 41
Miembro 1: Alejandro Vialard Santana
Miembro 2:

Número de la práctica*: 5	**Fecha de entrega*:** 19/05/2022

```
if (condicionTuberia == 1) {
    analizeLinePipe(string, argList);

    lowercase(argList);

    // pipes(argList);

    condicionTuberia = 0;
}
```

Si la condición no es uno, quiere decir que no es una tubería y se realizará lo siguiente: se analiza la línea de comandos con analizeLine(), se pasa a minúscula por si hay alguna mayúscula con lowercase(), y ahora se tratan los otros comandos que se debían de implementar. Si el argumento es “exit”, sale del bucle con exit(0), y si el primer argumento es un “cd” se utiliza la función chdir con el argumento en la posición 1 para cambiar de directorio. Por último se utiliza la función execute() para ejecutar los procesos.

*Aclaración: Las pruebas están en el PDF llamado “Pruebas Práctica 5 -AVS”.

Horas de trabajo invertidas* Miembro 1: 6 horas aprox

Cómo probar el trabajo*
Solamente debe de entrar en el directorio en el que se encuentra el archivo con “cd Practica5 \– \AVS” y compila con “gcc shell.c -o shell” y ejecuta con “./shell”. Si todo sale bien debe de salir algo así:

```
[efso@localhost ~]$ cd Escritorio/Práctica5\ -\ AVS/
[efso@localhost Práctica5 - AVS]$ gcc shell.c -o shell
[efso@localhost Práctica5 - AVS]$ ./shell

SHELL CREADO POR ALEJANDRO VIALARD SANTANA
[AVS ~efso@localhost ~/home/efso/Escritorio/Práctica5 - AVS]$
```

Incidencias
(errores no resueltos, anomalías, cualquier cosa que se salga de lo normal)

La principal incidencia es que no logro hacer que funcionen las pipes, y no sé en qué fallo para ello. Por eso, pese a implementar código, he puesto que no cumplo el reto.

Comentarios