Laboratorio 2 - Llamadas al Sistema

Nota: pueden escribir las respuestas para las preguntas que se piden en algunos de los ejercicio en un archivo de texto con nombre `respuestas_ejX.txt`, donde *X* es el número de ejercicio. Por ejemplo: `respuestas_ej2.txt`.

Ejercicio 1

El programa `hola.c` imprime el mensaje `hola mundo` en la *salida estándar*, utilizando la función de biblioteca `printf()`. Compilarlo con el comando `make hola`. Vamos a ejecutarlo utilizando el comando `strace`, que nos permite ver que llamadas al sistema utiliza el programa:

```
$ strace bin/hola
```

Si el comando `strace` no esta instalado en el sistema, utilizar el administrador de paquetes de la distribución para instalarlo. Por ejemplo, en Ubuntu el comando es `sudo apt install strace`.

Responder:

1. Identificar las llamadas al sistema utilizadas por las funciones de biblioteca empleadas en el programa.
2. ¿Cual es la llamada al sistema que se encarga de imprimir el mensaje en la *salida estándar*?

Ejercicio 2

Completar el programa `ej2.c`, de manera que lea el texto introducido desde la entrada estandar, haga un eco en la salida estándar y lo guarde en el archivo indicado en la linea de comandos. Si el archivo indicado no existe, se debe crearlo. De existir, se lo debe sobreescribir. Utilizar unicamente las siguientes llamadas al sistema: `read()`, `open()`, `write()` y `close()`.

Ejercicio 3

Completar el programa `ej3.c` para que cree *n* procesos hijos, utilizando la llamada al sistema `fork()`. El número *n* debe ser indicado mediante un parámetro en la línea de comandos. Cada proceso hijo debe imprimir por la salida estándar su *identificador de proceso* (PID) y finalizar, mediante la llamada al sistema `exit()`. Para obtener el PID emplear la llamada al sistema `getpid()`. El proceso padre debe esperar a que todos sus procesos hijos finalicen, y luego imprimir un mensaje. Utilizar la llamada al sistema `waitpid()` para esperar a que los procesos hijos terminen.

Por ejemplo, si se ejecuta el programa indicando que se creen 3 hijos, debe obtenerse una salida similar a la siguiente:

```
$ ej1 3
Hijo 3431
Hijo 3434
Hijo 3432
```

```
Fin ej1
$
```

Ejercicio 4

En esta parte del laboratorio se implementarán varias funcionalidades al `shell 6.828`, del curso *6.828 Operating Systems Engineering* del MIT. El código del intérprete lo pueden encontrar en el archivo `sh.c`.

Ejecución de comandos

Implementar la ejecución de comandos, por ejemplo `ls`. El parser genera una estructura `execcmd` que contiene el comando a ejecutar y los parámetros que se le hayan indicado. Deben completar el caso `' '` en la función `runcmd`. Para ejecutar el comando, utilizar la llamada a sistema `execv()`. Se debe imprimir un mensaje de error si `execv()` falla, utilizando la función `perror()`.

Redirección de E/S

Implementar redirección de E/S mediante los operadores `<` y `>`, de manera que el shell permita ejecutar comandos como:

```
$ echo "sistemas operativos" > x.txt
$ cat < x.txt
sistemas operativos
$
```

El parser implementado en el shell ya reconoce estos operadores, y genera una estructura `redircmd` con los datos necesarios para implementar la redirección. Deben completar el código necesario en la función `runcmd()`. Consultar las llamadas al sistema `open()` y `close()`. Imprimir un mensaje de error si alguna de las llamadas al sistema empleadas falla con `perror()`. Verificar los permisos con los que se crea el archivo.

Tuberías (pipes)

Implementar soporte de tuberías, de manera que se pueda ejecutar un comando como:

```
$ ls | wc
1   1   10
$
```

El parser ya reconoce el operador `|`, y guarda todos los datos requeridos para implementar la tubería en una estructura llamada `pipecmd`. Deben agregar el código necesario en la función `runcmd()`. Las llamadas al sistema `pipe()`, `fork()`, `close()` y `dup()` serán útiles. Una vez implementado, verificar que se pueda ejecutar el comando de ejemplo anterior (puede ser necesario pasar el path completo a los comandos, dependiendo de cómo se haya implementado la ejecución de comandos).

Opcional:

Agregar al shell una de las siguientes funcionalidades:

- Listas de comandos (cada comando se separa con un punto y coma).
- Historial de comandos.
- Ejecución en segundo plano (agregando el operador & al final del comando).

¡Fin del Laboratorio 2!