Integrantes:
- Jesús Tirado
- Alberto Montalban

Monitor de Shell

Introducción:

Para cumplir con los requerimientos del proyecto hemos pensado en el diseño de un programa que nos permita hacer la función de un intérprete de comandos y que al mismo tiempo de ir realizando las instrucciones tecleadas vaya generando un reporte dentro de un archivo de texto.
El programa ha sido desarrollado sobre el sistema operativo Ubuntu y programado usando el lenguaje C que incluye únicamente librerías estándar.

¿Cómo funciona?

Como ya sabemos un intérprete de comandos tiene la particularidad de permitirnos generar instrucciones para el sistema operativo, sabemos también que una vez abierto un intérprete lo que hace es esperar a que se teclee una instrucción, posterior a eso realiza la tarea deseada y regresa al prompt para seguir esperando otro comando.
Es por ello que nuestro sistema utiliza procesos (fork) en cada una de las instrucciones tecleadas. Es decir, por cada nueva instrucción se genera un proceso el cual tiene como condición terminar para que el proceso padre pueda volver a mostrar el prompt del shell esperando una nueva instrucción.
El reporte del sistema se maneja con procesos suaves (hilos). La implementación consta de generar un hilo que crea un archivo el cual va almacenando cada una de las instrucciones tecleadas reportando la hora y fecha de su ejecución.

Otras funciones:

Señales:
Se usa la implementación de la función “signal” establecida para que en la ejecución del programa se pueda captar una señal “^C” que en uso normal esta señal daría fin al programa en ejecución, pero para nuestro proyecto tiene como objetivo modificar el prompt impreso en el shell.

Tuberías:
En un intérprete de comandos es muy común poder ligar las salidas y entradas de los comandos haciendo uso de tuberías (pipes o |), por ello también se ha simulado esta implementación con la limitante de que solo podemos unir dos comandos. Pensamos en esto ya que es una maneja de ejemplificar un grado más profundo de manejo de procesos (fork) porque al momento de solicitar la respuesta de un proceso, necesariamente el primero tiene que esperar a que termine el hijo para que pueda obtener sus parámetros de entrada.
Cabe mencionar que para cumplir con esta función nos dimos a la tarea de poder manipular los direccionamientos de entrada, salida y error, haciendo uso de las correspondientes funciones dup y close.

Ejecución:

Para poder compilar necesitamos teclear –pthread ya que se implementan hilos en el programa.

```
$ gcc monitor.c -o monitor -pthread
```

El programa lanza el prompt que estará esperando instrucciones desde teclado.

```
monitor@sistop:~$
```

Integrantes:
- Jesús Tirado
- Alberto Montalban

Un ejemplo de la manipulación de señales es la siguiente:
El programa no se cierra al presionar el juego de teclas Ctrl + C sino que muestra un nuevo prompt que definimos en el código.

```
monitor@sistop:~$^C
monitor@segundoUsuario:~$
```

Todas las instrucciones tecleadas son procesos que hacen una llamada a la función del sistema operativo y están dirigidas para poderse visualizar dentro del programa. Ejemplo:

```
monitor@sistop:~$ls -a
.  ..  ._.DS_Store  .DS_Store  ejemplo.txt  historial.txt  monitor  monitor.c  proyShell.c
monitor@sistop:~$
```

Para re direccionar la salida de un comando y volcarla a un archivo bastaría con ejecutar:

ls > jemplo.txt

También podemos, por ejemplo, contar las líneas que tiene un archivo re direccionando la entrada estándar de *wc* hacia un archivo de texto. Así:

wc <ejemplo.txt

Para re direccionar el error podríamos teclear algún comando no reconocido y enviarlo a un archivo de texto, por ejemplo:

dfsfds 2> ejemplo.txt

```
monitor@sistop:~$ls > ejemplo.txt
monitor@sistop:~$ls
ejemplo.txt  historial.txt  monitor  monitor.c  proyShell.c
monitor@sistop:~$cat ejemplo.txt
ejemplo.txt
historial.txt
monitor
monitor.c
proyShell.c
monitor@sistop:~$wc < ejemplo.txt
 5  5 56
monitor@sistop:~$dfsfds 2> ejemplo.txt
monitor@sistop:~$cat ejemplo.txt
no se encontro la orden: No such file or directory
ll.c
monitor@sistop:~$
```

Sistemas Operativos, segundo proyecto.

Integrantes:
- Jesús Tirado
- Alberto Montalban

Para el uso de tuberías podríamos ver los procesos que están corriendo en el sistema usando *ps* y le re direccionamos la salida a *sort* para que los ordene por PID. Este es un ejemplo básico porque nuestro programa solo está diseñado para hacer la unión de dos comandos mediante tuberías.

```
monitor@sistop:~$ps -a | sort
monitor@sistop:~$ 1089 tty1     00:00:11 ibus-daemon
 1097 tty1     00:00:00 ibus-dconf
 1099 tty1     00:00:00 ibus-x11
 1182 tty1     00:00:00 gsd-mouse
 1183 tty1     00:00:00 gsd-power
 1187 tty1     00:00:00 gsd-print-notif
 1188 tty1     00:00:00 gsd-rfkill
 1195 tty1     00:00:00 gsd-screensaver
 1197 tty1     00:00:00 gsd-sharing
 1201 tty1     00:00:00 gsd-smartcard
 1208 tty1     00:00:00 gsd-wacom
 1210 tty1     00:00:00 gsd-xsettings
 1211 tty1     00:00:00 gsd-sound
 1220 tty1     00:00:00 gsd-a11y-settin
 1221 tty1     00:00:00 gsd-clipboard
 1228 tty1     00:00:00 gsd-a11y-keyboa
 1231 tty1     00:00:00 gsd-datetime
 1232 tty1     00:00:00 gsd-housekeepin
 1233 tty1     00:00:05 gsd-color
 1237 tty1     00:00:00 gsd-media-keys
 1239 tty1     00:00:00 gsd-keyboard
 1253 tty1     00:00:00 gsd-printer
 1288 tty1     00:00:02 gnome-software
 1292 tty1     00:00:00 gsd-disk-utilit
 1295 tty1     00:00:00 kerneloops-appl
 1298 tty1     00:00:03 nautilus-deskto
 1390 tty1     00:00:02 ibus-engine-sim
 1531 tty1     00:00:00 update-notifier
 1601 tty1     00:00:00 deja-dup-monito
 6978 tty1     00:04:50 firefox
 7030 tty1     00:10:03 Web Content
 7257 tty1     00:00:36 Web Content
 7331 tty1     00:11:52 Web Content
 7378 tty1     00:11:05 Web Content
  753 tty1     00:00:00 gnome-session-b
 7628 tty1     00:00:00 oosplash
 7662 tty1     00:01:00 soffice.bin
  835 tty1     00:40:27 gnome-shell
 8583 tty1     00:00:00 sd_generic
 8586 tty1     00:00:00 sd_espeak-ng
 8592 tty1     00:00:00 sd_dummy
  867 tty1     00:01:23 Xwayland
 9188 pts/0    00:00:00 monitor
 9310 pts/0    00:00:00 monitor
 9350 pts/0    00:00:00 ps
 9351 pts/0    00:00:00 monitor
  PID TTY          TIME CMD
```

Sistemas Operativos, segundo proyecto.

Integrantes:
- Jesús Tirado
- Alberto Montalban

El reporte del sistema que ha sido implementado mediante hilos genera un archivo de texto dentro del mismo directorio y guarda todos los comandos tecleados imprimiendo también hora y fecha de su ejecución. Aquí se muestra el archivo generado con los comandos utilizados para hacer los ejemplos anteriores:

```
monitor@sistop:~$ls
ejemplo.txt  historial.txt  monitor  monitor.c  proyShell.c
monitor@sistop:~$cat historial.txt
24/11/17 12:29:02 >> sudo dmidecode -s system-manufacturer
24/11/17 12:29:29 >> sudo dmidecode
24/11/17 12:30:09 >> grep 'vendor_id' /proc/cpuinfo ; grep 'model name' /proc/cpuinfo ; grep 'cpu MHz' /pro
c/cpuinfo
24/11/17 12:30:28 >> free -o -m
24/11/17 12:30:46 >> free -m
24/11/17 12:31:19 >> echo $SHELL
24/11/17 12:31:37 >> echo $USER
24/11/17 12:33:28 >> uname -m
24/11/17 12:33:52 >> clear
24/11/17 12:34:06 >> celar
24/11/17 12:34:09 >> clear
24/11/17 12:34:13 >> sudo dmidecode
24/11/17 12:34:52 >> sudo dmidecode -s system-version
24/11/17 12:35:09 >> sudo dmidecode -s system-manufacturer
24/11/17 12:35:41 >> proc/cpuinfo ; grep 'model name' /proc/cpuinfo ; grep 'cpu MHz' /proc/cpuinfo
24/11/17 12:35:58 >> grep 'vendor_id' /proc/cpuinfo ; grep 'model name' /proc/cpuinfo
24/11/17 12:36:11 >> grep 'vendor_id' /proc/cpuinfo ; grep 'model name' /proc/cpuinfo ; grep 'cpu MHz' /pro
c/cpuinfo
24/11/17 12:41:24 >> dmesg | less
24/11/17 12:41:41 >> clear
24/11/17 12:41:47 >> ls -a
24/11/17 12:44:37 >> ls -la ~ > archivo.txt
24/11/17 12:45:01 >> ls
24/11/17 12:45:16 >> cat archivo.txt
24/11/17 12:47:07 >> wc
24/11/17 12:50:59 >> wc < ejemplo.txt
24/11/17 12:54:15 >> ls > ejemplo.txt
24/11/17 12:54:18 >> ls
24/11/17 12:54:44 >> cat ejemplo.txt
24/11/17 12:55:51 >> clear
24/11/17 12:56:07 >> ls > ejemplo.txt
24/11/17 12:56:19 >> ls
24/11/17 12:56:28 >> cat ejemplo.txt
24/11/17 12:56:44 >> wc < ejemplo.txt
24/11/17 12:58:04 >> dfsfds 2> ejemplo.txt
24/11/17 12:58:19 >> cat ejemplo.txt
24/11/17 13:00:22 >> ps -a | sort
24/11/17 13:01:33 >> clear
24/11/17 13:01:43 >> ls
monitor@sistop:~$salir
jesustirado@jesustirado-VirtualBox:~/Escritorio/shell$
```

Por ultimo hemos definido la instrucción “salir” para poder finalizar la ejecución de nuestro Shell.