Pruebas de funcionamiento básico del Shell

En este documento se os presentan una serie de ejemplos de pruebas que podéis pasarle a vuestro Shell para comprobar que no se producen una serie de errores típicos que suelen aparecer. **El que pase estas pruebas no significa que el Shell esté correcto, simplemente no se han detectado algunos de los problemas típicos que suelen aparecer**. Es tarea vuestra hacer pruebas más exhaustivas para probar el correcto funcionamiento del mismo.

Estas pruebas deben realizarse después de haber completado la implementación completa del Shell (las cuatro tareas) y después de haber comprobado que vuestro Shell puede lanzar tareas en primer plano y segundo plano sin problemas (se ejecutan correctamente, aparecen bien registradas en la lista de tareas, no se quedan procesos zombies, etc.).

Para las pruebas que os presentamos a continuación, debéis comprobar que los mensajes que os da vuestro Shell son similares (dan la misma información y el número de mensajes es el mismo) a los que aparecen en las capturas de las pruebas.

Pruebas básicas

Primero vamos a comprobar lo básico, que al ejecutar un comando incorrecto el Shell nos informa del error y por otro lado informa que el proceso ha terminado con exit (nos da su PID, su nombre, motivo Exited y en info el valor que vosotros hayáis puesto. En el ejemplo sale 255 que se corresponde a exit(-1)).

```
COMMAND->incorrecto
Error, command not found: incorrecto
Foreground pid: 6381, command: incorrecto, Exited, info: 255
COMMAND->
```

Ahora vamos a probar el comando interno “cd”, que funciona en todos los casos. Vemos el directorio donde estamos (pwd), cambiamos al directorio padre (cd ..), comprobamos que efectivamente ha cambiado al directorio padre (pwd), volvemos al directorio original y comprobamos de nuevo que ha cambiado con éxito. También deberíais comprobar que “cd” sin argumentos no provoca ningún error y “cd” a un directorio que no existe nos informa del error.

```
COMMAND->pwd
/home/tc/SO/Shell
Foreground pid: 6378, command: pwd, Exited, info: 0
COMMAND->cd ..
COMMAND->pwd
/home/tc/SO
Foreground pid: 6379, command: pwd, Exited, info: 0
COMMAND->cd Shell
COMMAND->pwd
/home/tc/SO/Shell
Foreground pid: 6380, command: pwd, Exited, info: 0
COMMAND->cd
COMMAND->
COMMAND->cd basura
No se puede cambiar al directorio basura
COMMAND->
```

Otra prueba muy importante es comprobar que se maneja bien la lista de procesos en background, para eso podemos lanzar 3 procesos en background, procurando que el primero que termine sea el segundo lanzado y comprobando que van terminando sin problema, van desapareciendo de la lista de tareas y al final no se queda ninguno zombie (es importante que la duración de los comandos “sleep” sea la indicada en el ejemplo, ya que puede desvelar un error típico en el manejo de la lista de trabajos).

```
COMMAND->sleep 20 &
Background job runnig... pid: 6382, command: sleep
COMMAND->sleep 10 &
Background job runnig... pid: 6383, command: sleep
COMMAND->sleep 30 &
Background job runnig... pid: 6384, command: sleep
COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6384, command: sleep, state: Background
 [2] pid: 6383, command: sleep, state: Background
 [3] pid: 6382, command: sleep, state: Background
COMMAND->Background pid: 6383, command: sleep, Exited, info: 0

COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6384, command: sleep, state: Background
 [2] pid: 6382, command: sleep, state: Background
COMMAND->Background pid: 6382, command: sleep, Exited, info: 0

COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6384, command: sleep, state: Background
COMMAND->Background pid: 6384, command: sleep, Exited, info: 0

COMMAND->jobs
Contents of Lista de trabajos:
COMMAND->
COMMAND->ps
  PID TTY          TIME CMD
 5467 pts/0    00:00:00 bash
 6393 pts/0    00:00:00 Shell
 6394 pts/0    00:00:00 ps
Foreground pid: 6394, command: ps, Exited, info: 0
COMMAND->
```

Hay que comprobar también que vuestro Shell detecta bien cuando se bloquea un proceso que está en segundo plano, como en el siguiente ejemplo, que se lanza “cat” en segundo plano y como intenta leer del terminal y no lo posee, recibe la señal SIGTTIN y se suspende. Vuestro Shell lo debe detectar y registrar en la lista de trabajos. Con “fg” lo ponemos en primer plano, comprobamos que funciona (escribiendo la cadena “abcd” y comprobando que la repite) y **terminamos normalmente “cat” pulsando “ctrl-d”**.

```
COMMAND->cat &
Background job runnig... pid: 6385, command: cat
COMMAND->Background pid: 6385, command: cat, Suspended, info: 21

COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6385, command: cat, state: Stopped
COMMAND->fg
abcd
abcd
Foreground pid: 6385, command: cat, Exited, info: 0
COMMAND->
```

También vamos a comprobar que vuestro Shell gestiona bien una serie de suspensiones distintas del comando “cat” en distintos escenarios, y el funcionamiento correcto de los comandos internos “fg” y “bg”.

Ejecutamos “cat” en primer plano y comprobamos que funciona, lo suspendemos con “ctrl-z” y comprobamos que informa de la suspensión y se registra correctamente en la lista de trabajos (jobs). Podemos comprobar también que efectivamente el proceso está parado ejecutando “ps u” y comprobando que en la columna STAT aparece como “T” (stopped).
Lo pasamos a primer plano con “fb” y comprobamos que sigue funcionando correctamente, para suspenderlo de nuevo (“ctrl-z”). De nuevo debe informar y registrarlo correctamente en la lista de trabajos.
A continuación, intentamos ponerlo en segundo plano con “bg” y deberá detectar que se ha suspendido inmediatamente, informar y registrarlo en la lista de trabajos.
Por último, lo pasamos de nuevo a primer plano (“fg”), comprobamos que funciona y lo matamos con “ctrl-c” y comprobamos que el Shell informa correctamente que ha muerto por una señal (“Signaled, info: 2”, la señal recibida es la 2, SIGINT).
Podemos comprobar que el proceso asociado a “cat” debe haber desaparecido con el comando “ps u”.

```
COMMAND->cat
abcd
abcd
^ZForeground pid: 6972, command: cat, Suspended, info: 20
COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6972, command: cat, state: Stopped
COMMAND->ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
tc        6952  0.9  0.4   6484  4620 pts/0    Ss   08:47   0:00 -bash
tc        6971  0.0  0.0   1816   320 pts/0    S    08:48   0:00 ./Shell
tc        6972  0.0  0.0   3552   364 pts/0    T    08:48   0:00 cat
tc        6973  0.0  0.2   4740  2104 pts/0    R+   08:48   0:00 ps u
Foreground pid: 6973, command: ps, Exited, info: 0
COMMAND->fg
abcd
abcd
^ZForeground pid: 6972, command: cat, Suspended, info: 20
COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6972, command: cat, state: Stopped
COMMAND->bg
Background pid: 6972, command: cat, Suspended, info: 21
Background job runnig... pid: 6972, command: cat
COMMAND->jobs
Contents of Lista de trabajos:
 [1] pid: 6972, command: cat, state: Stopped
COMMAND->fg
abcd
abcd
^CForeground pid: 6972, command: cat, Signaled, info: 2
COMMAND->jobs
Contents of Lista de trabajos:
COMMAND->ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
tc        6952  0.4  0.4   6484  4620 pts/0    Ss   08:47   0:00 -bash
tc        6971  0.0  0.0   1816   320 pts/0    S    08:48   0:00 ./Shell
tc        6974  0.0  0.2   4740  2112 pts/0    R+   08:49   0:00 ps u
Foreground pid: 6974, command: ps, Exited, info: 0
COMMAND->
```

No se muestran capturas de funcionamiento básico de “fg” y “bg”, pero deberíais comprobar que teniendo varios trabajos en background y stopped, podéis ponerlos en primer plano (fg) o segundo plano (bg) indicando el número que ocupan en la lista de tareas (jobs), es decir, probar “fg #num” y “bg #num”, que no aparecen en los ejemplos.

Por último, vamos a probar si cuando un comando parado (STOPPED) recibe una la señal de continuación (SIGCONT) vuestro Shell lo detecta y cambia su estado a BACKGROUND.

Para ello, lo primero que hacemos es ejecutar un “sleep 100” para suspenderlo inmediatamente con CTRL+Z. Comprobamos que efectivamente está parado (STOPPED) ejecutando el comando “jobs”. A continuación, le mandamos la señal SIGCONT con el comando externo “kill” al pid del proceso “sleep”. El Shell nos deberá informar de que el comando se ha reanudado, y si ejecutamos “jobs” de nuevo veremos que su estado ahora es BACKGROUND.