

# Sistemas Operativos

2º Curso I. T. Informática (Sistemas y Gestión) e I. Informática

Práctica 3 – Llamadas al sistema en Linux

Convocatoria de Junio – Curso 2010/2011

En la documentación adjunta se describe un *shell* básico capaz de ejecutar órdenes en primer y segundo plano (`smallsh`). Asimismo, en la página web de la asignatura se puede encontrar el código de este intérprete de órdenes. El objetivo de la práctica es ampliar y añadir nuevas características al `smallsh`. Existen una serie de mejoras obligatorias y algunas opcionales. Implementar las características opcionales servirá para subir nota. A continuación se listan las mejoras obligatorias. Será necesario implementar **todas** las características obligatorias para aprobar la práctica:

1. Los procesos en **segundo plano** no se deben ver afectados por las señales `SIGINT` ni `SIGQUIT`. El propio *shell* tampoco se debe ver afectado.
2. Añadir la orden `cd [directorio]`. El directorio es opcional y si no se especifica debemos regresar al directorio de **inicio de sesión**.
3. Al salirse del *shell* con la orden `exit`, todos los procesos lanzados en segundo plano desde este shell y que todavía se encuentren activos tendrán que ser terminados forzosamente, en caso de haber alguno. Para todos ellos deberá mostrarse un mensaje de finalización análogo al del apartado 8, con el mensaje de texto correspondiente indicando la finalización por terminación del shell.
4. El **prompt del shell** debe mostrar siempre el número de procesos activos en segundo plano (ver apartado 8, control de procesos en segundo plano), la fecha, la hora actual, el usuario y el directorio actual con el formato `[#procesos fecha hora usuario@directorio]`.
5. Implementar los **separadores de órdenes** “&&” y “| |”, con la misma semántica que en el bash estándar (ejecución condicional de órdenes, dependiendo del estado de salida de la orden anterior en la lista). Por ejemplo:

```
$ false || echo si # (se ejecutará el echo)
```

```
$ true || echo si # (no se ejecutará el echo)
```

```
$ false && echo si # (no se ejecutará el echo)
```

```
$ true && echo si # (se ejecutará el echo)
```

Además, el shell deberá proporcionar en todo momento acceso a la variable `$?`, con el valor de retorno del último comando ejecutado en primer plano, tal y como lo hace el bash estándar (por ejemplo, debe funcionar como se espera una orden como `echo $?`).

6. Soporte de los **caracteres comodín** estándares del bash (“\*”, “?”, “{ }”, “[ ]”, ...). Por ejemplo, si se ejecuta “`ls a*b?`”, se debe expandir dicha línea para que a la orden `ls` se le pasen, como parámetros, todos los nombres de ficheros (incluidos directorios) que empiecen por “a”, y terminen en “b” seguido de cualquier otro carácter. Los comodines “\*” y “?” nunca sustituyen a un punto “.” inicial, es decir, “\*a” representa a todos los ficheros que terminan en “a”, sin incluir aquellos ficheros cuyo nombre empiece por “.”. Para ello se tendría que escribir “. \*a”. También se deben permitir expresiones con caminos completos, como “`ls /bin/a*b?`”, o que incluyan rangos de caracteres y/o cadenas opcionales, como “`ls /bin/[A-M]*{.cad,.CAD}`”<sup>1</sup>.

---

<sup>1</sup>Para ello, se recomienda utilizar las funciones `glob` y `globfree`, que realizan todo el trabajo, aunque también podría resultar útil la función alternativa `fnmatch`.

7. Añadir como orden interna el comando `mymerge` que compare dos ficheros línea a línea. Cada vez que dos líneas sean diferentes se deberá insertar en un fichero especificado por parámetro el número de línea y el contenido de la línea de cada uno de los ficheros. Si los ficheros son exactamente iguales, se mostrará en pantalla un mensaje informando de la igualdad de los archivos. Si los ficheros son diferentes, se mostrará en pantalla el contenido del archivo de diferencias.
8. Implementar un **nuevo comando interno** `bgproc` que muestre una lista de todos los procesos que se lanzaron en segundo plano en la sesión actual y que aún continúan activos. Para cada proceso se deberá mostrar, por columnas, la siguiente información:
  - Nombre del comando.
  - Fecha y hora exacta en que se lanzó el proceso (en formato `dd/mm/yy hh:mm:ss`).
  - Tiempo (en segundos) que lleva el proceso activo (corresponde a la fecha actual menos la fecha del punto anterior).

Cada vez que termine un proceso en segundo plano, deberá mostrarse un mensaje de finalización donde conste su nombre, su PID, su código de retorno y un texto indicando si su finalización fue normal o por una señal del `kill` (y, en ese caso, deberá mostrarse el número concreto de la señal).

9. Implementar el comando interno `mypslist`. La invocación del comando debe mostrar una lista con el histórico de todos los procesos que se han ido creando en el `smallsh`, incluyendo los procesos que hayan finalizado por cualquier causa, y la relación de precedencia. Para diferenciar el estado de los distintos procesos, al lado del nombre de cada proceso, debe aparecer, entre paréntesis, el estado que puede ser `activo (a)` o `terminado (t)`. Para establecer claramente la relación de precedencia, al lado del estado de los distintos procesos, debe aparecer el `pid` del proceso y el `pid` del proceso padre, separados por un guión y entre paréntesis. El proceso raíz o inicial de la lista es el propio `smallsh`.
10. Añadir como orden interna el comando `sincro dirorigen dirdestino` con el comportamiento de comprobar el contenido de dos directorios y actualizar la información del segundo con respecto al primero. La actualización de un fichero en el directorio destino implica que existe alguna diferencia en la fecha y/o en la hora del fichero correspondiente en el directorio origen. Se mostrará un mensaje por pantalla por cada uno de los ficheros analizados informando si se actualiza o no el archivo correspondiente.
11. Implementar un **nuevo comando interno** `asincro dirorigen dirdestino segs nveces` al que se le indica un directorio origen, un directorio destino, el número de segundos que transcurren entre cada una de las comprobaciones del comando y el número de veces que se ejecuta la orden. El efecto del comando es actualizar periódicamente el contenido del directorio destino respecto al directorio origen el número de veces indicado y tras la pauta de tiempo establecida.
12. Ampliar la funcionalidad del comando `asincro`, descrito en el apartado 11, para que el sistema pueda manejar varios directorios pendientes de ser actualizados, procediendo cada uno de ellos tal y como se describe en el apartado 11.
13. Implementar un nuevo comando interno `vasincro` que muestre una lista de todos los comandos `asincro` que se lanzaron en la sesión actual y que aún continúan activos. Para cada proceso, se deberá mostrar, por columnas, la siguiente información: nombre del comando, directorio origen, directorio destino, número de segundos que transcurren entre cada una de las comprobaciones del comando, número de veces totales que se ejecuta la orden, número de veces que quedan por ejecutar el `asincro` correspondiente y número de segundos que faltan para la siguiente ejecución del comando.
14. Implementar el **mecanismo de tuberías con nombre (named pipe)** entre procesos. Se debe permitir el entubamiento de cualquier número de órdenes que, además, pueden llevar opciones y parámetros. Se utilizará el símbolo `%` para indicar que se quiere crear una tubería con nombre entre dos procesos. Un ejemplo de utilización podría ser:

```
$ ps aux % grep alumno % tail -1
```

Si suponemos que los PIDs de estas tres ordenes son 1111, 2222 y 3333 respectivamente, durante la ejecución de esta línea de ordenes se debe crear la tubería con nombre `/tmp/fifo-sh-1111` donde escribirá la primera orden y de donde leerá la segunda, y la tubería con nombre `/tmp/fifo-sh-2222` donde escribirá la segunda y de donde leerá la tercera.

15. Implementar el **mecanismo de tuberías** (“|”) entre procesos. Se debe permitir el entubamiento de cualquier número de órdenes que, además, pueden llevar opciones y parámetros.

## Aportaciones voluntarias

Se recuerda que la nota máxima en esta práctica sólo se obtiene con originalidad y con aportaciones voluntarias. Algunas de dichas aportaciones pueden ser las siguientes:

1. Añadir al shell la posibilidad de un **historial de órdenes** manejado con las flechas del teclado, y que recuerde la historia de órdenes entre sesiones (de forma similar a como hace el bash original con el fichero `~/.bash_history`).
2. Permitir la **compleción de rutas y nombres de ficheros mediante el uso del tabulador**, buscando ficheros y/o directorios que se encuentren en la subruta absoluta o relativa hacia un fichero según la va tecleando el usuario (tomar como ejemplo el comportamiento del propio bash).
3. Añadir como órdenes internas diversos **filtros estándar de UNIX**, como `head`, `tail`, `cut`, o cualesquiera otros. Para cada orden se pueden implementar las opciones que se deseen. En particular, en este apartado se valorará muy positivamente el uso de las facilidades de UNIX para ficheros proyectados en memoria, y las funciones asociadas `mmap` y `munmap` (consultar las páginas del manual correspondientes).
4. Otra posibilidad es implementar otros **comandos estándar de manejo del sistema de ficheros**, como `ls`, `mv`, `cp`, `mkdir`, etc., siempre con el conjunto de opciones que se deseen.
5. Implementar la **funcionalidad de las variables de shell** (variables locales y globales, asignación, consulta y sustitución con `$`, `unset`, `export`, etc.).
6. Cualesquiera otras mejoras que el grupo considere interesantes. En cualquier caso, las mejoras se considerarán tanto más relevantes cuanto más variado sea el número de llamadas al sistema empleadas para implementar las mejoras en cuestión. Antes de trabajar en una mejora, sería también conveniente consultar la conveniencia y dificultad de la misma con el profesor.

## Observaciones

- Para el manejo de ficheros y directorios, podéis mirar los temas 3 y 4 del libro **“UNIX, Programación Avanzada” (3ª edición, ampliada y actualizada), de Fco. Manuel Márquez, ed. RA-MA**. Para los procesos, señales, y temas relacionados, consultar los temas 5, 6 y 7 del mismo libro. En general, todas las funciones que aparecen en dichos capítulos pueden resultar de utilidad, y hay total libertad para usar cualquiera de ellas.
- Se debe entregar una **memoria impresa con los listados y una explicación de los mismos** resaltando, por un lado, los puntos más importantes, y por otro, las mejoras introducidas. Los ficheros se deben entregar en **un disco que acompañe a la memoria impresa**.
- Es obligatorio entregar la **parrilla de corrección de la práctica, indicando el funcionamiento correcto de todos y cada uno de los apartados**. El funcionamiento incorrecto de cualquiera de estos apartados en la entrevista de prácticas, implica el suspenso de la práctica.

- La fecha tope de entrega para todas las ingenierías es el **2 de mayo de 2011 para la convocatoria de junio y el 2 de septiembre de 2011 para la convocatoria de septiembre (improrrogable)**.

Murcia, 22 de febrero de 2011.

Los profesores de prácticas de la asignatura.