Sistemas Operativos
Llamadas al sistema: Minishell
msh.2020a

(C) Francisco Rosales, frosal@fi.upm.es

27 de enero de 2020

1. Objetivo de la práctica

El alumno deberá diseñar y codificar, en lenguaje C y sobre sistema operativo Unix, un programa que actúe como intérprete de mandatos. El programa deberá seguir estrictamente las especificaciones y requisitos contenidos en este documento.

Con la realización de este programa el alumno adquirirá valiosos conocimientos de programación en entorno Unix. Tanto en el uso de las llamadas al sistema operativo (`FORK`, `EXEC`, `SIGNAL`, `PIPE`, `DUP`, etc), como en el manejo de herramientas como el visualizador de páginas de manual (`man`), el compilador de C (`gcc`), el regenerador de programas (`make`), etc.

NOTA: Durante la lectura de este documento encontrará la notación "man -s# xxxxx" que sugiere usar el mandato `man` para obtener información sobre el apartado xxxxx de la sección # del manual. Haga caso de las recomendaciones.

2. Descripción de la práctica

El minishell utiliza la entrada estándar (descriptor de fichero 0), para leer las líneas de mandatos que interpreta y ejecuta. Utiliza la salida estándar (descriptor de fichero 1) para presentar el resultado de los mandatos internos. Y utiliza el estándar error (descriptor de fichero 2) para mostrar las variables especiales `prompt` y `bgpid` (vea el epígrafe **Variables especiales**) así como para notificar los errores que se puedan dar. Si ocurre un error en alguna llamada al sistema, se utiliza para notificarlo la función de librería `perror`.

Blanco Es un carácter tabulador o espacio.

Separador Es un carácter con significado especial (`|<>&`), el fin de línea o el fin de fichero (por teclado Ctrl-D).

Texto Es cualquier secuencia de caracteres delimitada por blanco o separador.

Mandato Es una secuencia de textos separados por blancos. El primer texto especifica el nombre del mandato a ejecutar. Las restantes son los argumentos del mandato invocado.

El nombre del mandato se pasa como argumento 0 (man execvp). Cada mandato se ejecuta como un proceso hijo directo del minishell (man fork). El **valor** de un mandato es su estado de terminación (man -s2 wait). Si la ejecución falla se notifica el error (por el estándar error).

Secuencia Es una secuencia de dos o más mandatos separados por el caracter '`|`'. La salida estándar de cada mandato se conecta por una tubería (man pipe) a la entrada estándar del siguiente. El minishell normalmente espera la terminación del último mandato de la secuencia antes de solicitar la siguiente línea de entrada. El **valor** de una secuencia es el valor del último mandato de la misma.

Redirección La entrada y/o la salida de un mandato o secuencia puede ser redirigida añadiendo tras él la siguiente notación. En caso de cualquier error durante las redirecciones, se notifica (por el estándar error) y se suspende la ejecución de la línea.

- `<` **fichero** Usa **fichero** como entrada estándar abriéndolo para lectura (man open).
- `>` **fichero** Usa **fichero** como salida estándar. Si el fichero no existe se crea, si existe se trunca (man creat), modo de creación 0666.
- `>&` **fichero** Usa **fichero** como estándar error. Si el fichero no existe se crea, si existe se trunca (man creat), modo de creación 0666.

Background Un mandato o secuencia terminado en '`&`' supone la ejecución asíncrona en segundo plano del mismo, esto es, el minishell no queda bloqueado esperando su terminación. En cambio, actualiza el valor de la variable `bgpid` como el identificador del proceso por el que habría esperado, y muestra su valor con el formato `"[%d]\n"`.

Señales Ni el minishell ni los mandatos lanzados en background, deben morir por señales generadas desde teclado (`SIGINT`, `SIGQUIT`) (man signal). Por contra, los mandatos lanzados en primer plano deben morir si le llegan estas señales, por lo tanto mantienen la acción por defecto.

Mandato Interno Es aquél que bien se corresponde directamente con una llamada al sistema o bien es un complemento que ofrece el propio minishell. Para que su efecto sea permanente, ha de ser implementado y ejecutado dentro del propio minishell. Será ejecutado en un subshell (man fork) sólo si se invoca en background o aparece en una secuencia y no es el último.

Todo mandato interno comprueba el número de argumentos con que se le invoca y si encuentra este o cualquier otro error, lo notifica (por el estándar error) y termina con **valor** distinto de cero.

Los mandatos internos del minishell son:

- **cd** [**Directorio**] Cambia el directorio por defecto (man -s2 chdir). Si aparece **Directorio** debe cambiar al mismo. Si no aparece, cambia al directorio especificado en la variable de entorno `HOME`. Presenta (por la salida estándar) como resultado el camino absoluto al directorio actual de trabajo (man getcwd) con el formato: `"%s\n"`.

umask [**Valor**] Cambia la máscara de creación de ficheros (man -s2 umask). Presenta (por la salida estándar) como resultado el valor de la actual máscara con el formato: `"%o\n"`. Además, si aparece **Valor** (dado en octal, man strtol), cambia la máscara a dicho valor indicado.

limit [**Recurso** [**Máximo**]] Establece límites máximos de consumo de recursos para el proceso actual y procesos derivados de él (man setrlimit). Si no aparece **Máximo** se presenta (por la salida estándar) el límite actual del **Recurso** con el formato `"%s\t%d\n"`. Si tampoco aparece éste, se presentan (por la salida estándar) todos los límites, uno por línea. **Recurso** puede tomar los siguientes valores:

cpu Tiempo de CPU por proceso.

fsize Tamaño por fichero.

data Tamaño de segmento de datos por proceso.

stack Tamaño de segmento de pila por proceso.

core Tamaño de fichero core.

nofile Número de descriptores de fichero.

Los tamaños se consideran expresados en bytes y el tiempo en segundos. Un **Máximo** de -1 supone un valor infinito.

set [**Variable** [**Valor...**]] Da valor a variables de entorno (man putenv). El **Valor** de una variable es una lista de palabras separadas por blancos. Si no aparece **Valor** presenta (por la salida estándar) el valor actual de **Variable** con el formato `"%s=%s\n"`. Si tampoco aparece ésta, presenta (por la salida estándar) el valor de todas las variables (man -s5 environ).

Metacaracteres El minishell interpreta determinados caracteres de forma especial. Todo texto que comienza por `~` o que contiene `$` sufre la sustitución de la expresión de la que forma parte este carácter por su correspondiente valor (man realloc).

`~`[**Usuario**] Un nombre válido de **Usuario** tiene el formato `"%[_a-zA-Z0-9]"` (man sscanf). Si aparece **Usuario**, se sustituye por el directorio `home` de dicho usuario, según se indica en su entrada *passwd* (man getpwnam). Si no, se sustituye por el valor de la variable de entorno `HOME`.

`$`**Variable** Un nombre válido de **Variable** tiene el formato `"%[_a-zA-Z0-9]"` (man sscanf). Se sustituye por el valor de la variable de entorno **Variable** (man getenv).

Variables especiales El minishell conoce y trata de forma especial algunas variables de entorno (man putenv getenv).

prompt Mensaje de apremio antes de leer cada línea. Por defecto será `"msh> "`.

mypid Identificador de proceso del propio minishell.

bgpid Identificador de proceso del último proceso arrancado en background.

status Valor de terminación del último mandato o pipeline ejecutado en foreground.

Expansión de nombres de fichero Antes de la ejecución de cada orden, cada texto es examinado en busca de caracteres comodín. Si aparece algún comodín, el texto será expandido a la lista de nombres de fichero que casen con el patrón establecido. Si no aparece ningún fichero que case con el patrón, el texto no será alterado. El carácter conocido por comodín es:

? Casa con cualquier carácter individual.

Se establece la siguiente restricción. Si en el texto aparece también algún carácter `/`, no se tratarán los comodines. Esto reduce la tarea de casar con el patrón a repasar el contenido del directorio actual de trabajo (man -s3 glob opendir readdir closedir).

3. Código fuente de apoyo

Para facilitar la realización de esta práctica se dispone del fichero `msh.2020a.tgz` que contiene código fuente de apoyo. Al extraer su contenido desde el directorio `home` de tu cuenta, se crea el directorio `DATSI/so/msh.2020a`, donde se debe desarrollar la práctica. Dentro de este directorio se habrán incluido los siguientes ficheros:

Makefile Fichero fuente para la herramienta `make`. NO debe ser modificado. Con él se consigue la recompilación automática sólo de los ficheros fuente que se modifiquen.

scanner.l Fichero fuente para la herramienta `lex`. NO debe ser modificado. Con él se genera automáticamente código C que implementa un analizador lexicográfico (*scanner*) que permite reconocer el token **TXT**, considerando los posibles separadores (`\t␣|<>&\n`).

parser.y Fichero fuente para la herramienta `yacc`. NO debe ser modificado. Con él se genera automáticamente código C que implementa un analizador gramatical (*parser*) que permite reconocer sentencias correctas de la gramática de entrada del minishell.

main.c Fichero fuente de C que muestra como usar el *parser*. Este fichero es el que se DEBE MODIFICAR. Se recomienda estudiar detalladamente para la correcta comprensión del uso de la función de interfaz, `obtain_order`. La versión que se ofrece hace eco de las líneas tecleadas que sean sintácticamente correctas. Esta funcionalidad debe ser eliminada y sustituida por la ejecución de las líneas tecleadas.

4. Recomendaciones generales

Desarrolle el minishell por etapas, complicándolo progresivamente. Comience implementando una versión elemental. Continúe introduciendo las funcionalidades en el orden en que se describen en el apartado **Descripción de la Práctica**.

Para ello lea detenidamente este documento y sea estricto con la información en él contenida, en concreto con los formatos de presentación de los mandatos. Lea asimismo las páginas de manual a las que se hace referencia. Cuando tenga una idea clara de cómo implementar lo que se le pide, codifíquelo, compile y pruebe su práctica.

Para probar su práctica habrá de entregar la práctica. Se le contestará con un correo electrónico. La traza resultante puede darle información valiosa sobre las causas de los posibles errores.

5. Documentación a entregar

LA ÚLTIMA VERSIÓN REGISTRADA DE SU PRÁCTICA ES LA ÚNICA CUYA VALORACIÓN IMPORTA, ES LA ÚNICA VÁLIDA Y DEFINITIVA.

TODA ENTREGA PREVIA SE CONSIDERA QUE ES SÓLO PARA DEPURAR.

Según vaya añadiendo a su práctica nuevas funcionalidades, deberá entregarla usando el mandato `entrega.so msh.2020a`. Este mandato realizará la recolección de los siguientes ficheros:

autores.txt Fichero con los datos del autor.

bitacora.txt Es conveniente llevar una memoria del desarrollo de la práctica.

main.c Código fuente del minishell, implementando todas la funcionalidades que se requieren.

Referencias

[Bou 83] "The UNIX System", S.R. Bourne
Addison-Wesley, 1983.

[Roc 85] "Advanced Unix Programming", M.J. Rochkind
Prentice-Hall, 1985.

[Sun 90] "Programming Utilities and Libraries", SUN Mycrosystems
Sun Mycrosystems, 1990.