Operating Systems
System calls: Minishell
msh.2020b

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

September 4, 2020

1. Objective of the practice

The student should design and code, in C language and on Unix operating system, a program that acts as a command interpreter. The program shall strictly follow the specifications and requirements contained in this document.

With the completion of this program the student will acquire valuable knowledge of Unix programming. Both in the use of operating system calls (FORK, EXEC, SIGNAL, PIPE, DUP, etc), as well as in the use of tools such as the manual page viewer (man), the C compiler (gcc), the program regenerator (make), etc.

NOTE: While reading this document you will find the note "man -s# xxxxx" which suggests using the man command to obtain information about the xxxxx section of the # section of the manual. Heed the recommendations.

2. Practice description

The minishell uses the standard input (file descriptor 0), to read the command lines that it interprets and executes. It uses the standard output(filedescriptor1) to present the result of the internal commands. And it uses the error standard (file descriptor 2) to display the special variables prompt and bgpid (see Special Variables section) as well as to report any errors that may occur. If an error occurs in any system call, the perror library function is used to notify it.

Blank Is a tab character or space.

Separator It is a character with special meaning (|<>&), the end of line or the end of file (by keyboard Ctrl-D).

Text Is any sequence of characters delimited by a blank or separator.

Mandate It is a sequence of texts separated by blanks. The first text specifies the name of the command to be executed. The remaining ones are the arguments of the invoked mandate.

The command name is passed as argument0 (man execvp). Each command is executed as a direct child process of the minishell (man fork). The value of a command is its termination status (man -s2 wait). If the execution fails, the error is reported (by the standard error).

Sequence A sequence of two or more commands separated by the '|' character. The standard output of each mandate is connected by a man pipe to the standard input of the next mandate. The minishell normally waits for the completion of the last command in the sequence before requesting the next input line. The value of a sequence is the value of the last command in the sequence.

Redirection The input and/or output of a command or sequence can be redirected by adding the following note after it. In case of any error during the redirections, it is notified (by the error status) and the execution of the line is suspended.

> file Uses file as input to open it for reading (man open).

> file Uses file as standard output. If the file does not exist it is created, if it does exist it is truncated (man creat), creation mode 0666.

> & file Use file as this error. If the file does not exist it is created, if it does exist it is truncated (man creat), creation mode 0666.

Background A command or sequence ending in '&' means that the minishell is executed synchronously in the same plane, that is, the minishell is not blocked waiting for its termination. Instead, it updates the value of the bgpid variable as the identifier of the process for which it would have waited, and displays its value in the format "[%d]ın".

Signals Neither the minishell nor the commands launched in background, must be killed by signals generated from the keyboard (SIGINT, SIGQUIT) (man signal). By contrast, commands launched in the foreground must die if these signals reach it, so they keep the default action.

Internal Command This is a command that either corresponds directly to a system call or is an add-on provided by minishell itself. For its effect to be permanente, it has to be implemented and executed within the minishell itself. It will be executed in a subshell (man fork) only if it is invoked in the background or appears in a sequence and is not the last one.

Every internal command checks the number of arguments with which it is invoked and if it finds this or any other error, it notifies it (by the error status) and ends with a non-zero value.

The internal commands of the minishell are:

cd [Directory] Changes the default directory (man -s2 chdir). If Directory appears, you must change to Directory. If it does not appear, change to the directory specified in the HOME environment variable. It displays (by the current output) as result the absolute path to the current working directory (man getcwd) with the format: "%s".

umask [Value] Changes the file creation mask (man -s2 umask). It presents (by the standard output) as a result the value of the current mask with the format: "%or". In addition, if Value (given in octal, man strtol) appears, it changes the mask to that value.

time [Command] Consumption of execution times (man -s2 times). If Mandato appears, it executes it and reports its consumption. Without arguments, it reports on the consumption of minishell and its children. It displays (by the standard output) user mode time, system mode time and real time, in seconds and milliseconds, with the format: "%d.%03du %d.%03ds %d.%03dr".

read Variable [Variable...] Gives value to the specified environment variables (at least one) (man putenv). It reads a line from the standard input and considering space and tab separators (man strtok),

assigns the first word to the first environment variable, the second to the second, etc., and the rest of the line to the last variable. If there are fewer words than variables, the latter will not modify their value.

Metacharacters The minishell interprets certain characters in a special way. Any text beginning with ~ or containing $ suffers the substitution of the expression of which this character is part by its corresponding value (man realloc).

~A valid User name has the format "%[_a-zA-Z0-9]" (man ss- canf). If User appears, it is replaced by the home directory of that user, as indicated in his passwd entry (man getpwnam). If not, it is replaced by the value of the HOME environment variable.

A valid Variable name has the format "%[_a-zA-Z0-9]" (man ss- canf). It is replaced by the value of the environment variable Variable (man getenv).

Special variables The minishell knows and treats in a special way some variables fromaround (man putenv getenv).

prompt Prompt before reading each line. The default is "msh> ".

mypid Process identifier of the minishell itself.

bgpid Process identifier of the last process started in background.

status Termination value of the last command or pipeline executed in foreground.

File name expansion Prior to the execution of each command, each text is examined for wildcard characters. If any wildcard appears, the text will be expanded to the list of filenames matching the set pattern. If no file matching the pattern appears, the text will not be altered. The character known by commodity is:

? House with any individual car'acter.

The following restriction is established. If any character / also appears in the text, wildcards are not treated. This reduces the task of matching the pattern to going through the contents of the current working directory (man -s3 glob opendir readdir closedir).

3. Support source code

The file msh.2020b.tgz, which contains support source code, is available to facilitate the realization of this practice. By extracting its contents from the home directory of your account, the directory DATSI/so/msh.2020b is created, where the practice should be developed. The following files would have been included in this directory:

Makefile Source file for the make tool. It should NOT be modified. With it, automatic recompilation is achieved only for the source files that are modified.

scanner.l Source file for the lex tool. It should NOT be modified. It automatically generates C code that implements a lexicographic analyzer (scanner) that allows to recognize the TXT token, considering the possible separators (e.g.

u|<>&).

parser.y Source file for the yacc tool. It should NOT be modified. It automatically generates C code that

implements a parser to recognize correct sentences from the minishell input grammar.

main.c C source file showing how to use the parser. This file is the one that MUST BE MODIFIED. It is recommended to study in detail for the correct understanding of the use of the interface function, obtain order. The version provided echoes the syntactically correct typed lines. This functionality should be eliminated and replaced by the execution of the typed lines.

4. General recommendations

Develop the minishell in stages, complicating it progressively. Start by implementing an elementary version. Continue introducing the functionalities in the order described in the section Description of the Practice.

To do so, read this document carefully and be strict with the information contained in it, in particular with the formats for the presentation of the mandates. Also read the manual pages referred to. When you have a clear idea of how to implement what you are asked to do, code it, compile and test your practice.

To prove your practice you will have to submit the practice. You will receive a reply by e-mail. The resulting trace can give you valuable information about the causes of the possible errors..

5. Documentation to be submitted

, , , / . . . . ..
THE LATEST RECORDED VERSION OF ITS PRACTICE IS THE ONLY ONE WHOSE
THE ASSESSMENT MATTERS, IT IS THE ONLY VALID AND DEFINITIVE ONE.
. . . . . , , .
ALL PREVIOUS DELIVERIES ARE CONSIDERED TO BE ONLY FOR DEBUGGING PURPOSES.

As you add new features to yourpractice, you should deliver it using the delivery.so msh.2020b command. This mandate will perform the collection of the following fichers:

autores.txt File with the author's data.

logbook.txt It is convenient to keep a memory of the development of the practice.

main.c Minishell source code, implementing all the required functionality.

References

[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.