SISTEMAS OPERATIVOS

PRÁCTICA 2: MINISHELL

ANA M JURADO CRESPO
am.juradoc@alumnos.urjc.es

SISTEMAS OPERATIVOS - GIS - URJC - *5 DIC 2019*

CONTENIDOS

AUTORES

Tanto este documento como todos los archivos .c **practica2.zip** han sido realizados por:

Ana María Jurado Crespo

am.juradoc@alumnos.urjc.es

El código está bajo versión de controles en github en un repositorio privado:

https://github.com/nukyma/minishell/invitations

Los dos colaboradores del repositorio, tanto **nukyma** como **nukyma314** son mis alias. Depende del equipo y del momento del comit, he tenido la necesidad de usar uno u otro. Más información en **Problemas Encontrados**.

DESCRIPCIÓN DEL CÓDIGO

He ido desarrollando la práctica siguiendo los objetivos parciales ya que me parecía una buena forma de ir avanzando en el desarrollo de la práctica.

CÓDIGO

FUNCIONES

Las funciones de las que consta la práctica son:

```
// FUNCIONES
int exec_one_command(tline *line, char *buffer);
int input_output(tline *line, int changeInpunt, int changeOutput, int changeOutputError);
int exec_commands(tline *line, char *buffer);
int signal_control(int process);
int command_cd(int argc, char *argv[]);
```

Las funciones están comentadas dentro del código. En el presente documento sólo voy a citar brevemente lo que hacen y explicar el funcionamiento general del programa (función main)

int exec_one_command(tline *line, char *buffer)

Función que ejecuta un sólo comando.

int input_output(tline *line, int changeInpunt, int changeOutput, int changeOutputError)

Redirige la salida hacía o la entrada desde ficheros según se indique

int exec_commands(tline *line, char *buffer)

Función que ejecuta varios comandos

int signal_control(int process)

Controla las señales SIGINT y SIGQUIT por defecto

int command_cd(int argc, char *argv[])

Función que ejecuta el comando cd haciendo uso de *chdir*

MAIN

Función principal del programa.

VARIABLES

```
  char buf[1024];     // Buffer chars de lectura del input por teclado
  tline * line;       // Puntero a variable tipo tline

  printf("msh> ");    // Simbolo prompt
```

Tenemos:

- Un buffer de lectura que recoge los caracteres introducidos por teclado.
- Un puntero de tipo tline a una línea
- El símbolo que vamos a imprimir como símbolo del prompt

A continuación desactivamos el control de señales por defecto en el programa principal myshell.

```
// Mientras tengamos un input...
while (fgets(buf, 1024, stdin)) {

    line = tokenize(buf);   // linea introducida en el "shell"

    if (line==NULL) {
        continue;
    }

    // Debemos comprobar el número de comandos ya que en caso de ser un único comando
    // es imprescindible diferenciar los comandos exit y cd del resto
    // Linea introducida en shell tiene un comando
    if(line->ncommands == 1){       // Linea introducida en shell tiene un comando
        if(strcmp(line->commands[0].argv[0], "exit") == 0){     // Comando exit
            return 0;
        } else if(strcmp(line->commands[0].argv[0], "cd") == 0){    // Comando cd
            command_cd(line->commands[0].argc, line->commands[0].argv);
        } else {
            exec_one_command(line, buf);        // Ejecutar un único comando
        }
    } else {    // Linea introducida en shell tiene varios comandos
        exec_commands(line, buf);   //Ejecutar varios comandos
    }

    printf("msh> ");    // Al acabar de ejecutar un mandato debemos imprimir de nuevo el simbolo
}
```

Mientras tengamos algún tipo de input vamos a ejecutar el bucle. “Tokenizamos” la línea introducida en el myshell, si la línea es nula, da igual, continuamos. Miramos el número de argumentos. Debemos comprobar el número de comandos ya que en caso de ser un único comando es imprescindible diferenciar los comandos exit y cd del resto. Si el comando es exit o cd, se tratan de forma diferente y si el comando es distinto hacemos una llamada a la función ***exec_one_command.*** Si tenemos más de un comando, se hace una llamada a la función ***exec_commands.*** Volvemos a printar el símbolo del prompt y quedamos a la espera de que el usuario introduzca una nueva línea en la shell.

OBJETIVOS

A continuación se encuentran los objetivos y algunos de los mandatos que he empleado para testear el código según iba desarrollándolo.

Objetivo 1

Ser capaz de reconocer y ejecutar en foreground líneas con un solo mandato y 0 o más argumentos.

- ls
- date
- whoami
- pwd
- htop
- man ls
- man echo
- cal 11 2019
- ps -la
- ps -le
- ls -a
- ls -la
- du -h

Ejemplos de ejecuciones:

```
msh> cal
   December 2019
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

msh> whoami
ana
msh> pwd
/home/ana/sop2/minishell/folder
```

```
msh> ps -la
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  5934 11953  0  80   0 - 14851 -      pts/1    00:00:00 vim
0 S  1002 29629 29348  0  80   0 -   273 wait   pts/2    00:00:00 m5c
0 R  1002 29675 29629  0  80   0 -  7688 -      pts/2    00:00:00 ps
```

Objetivo 2

Ser capaz de reconocer y ejecutar en foreground líneas con un solo mandato y 0 o más argumentos, redirección de entrada estándar desde archivo y redirección de salida a archivo.

Redirección de **salida**:

- whoami > output
    - cat output
- date > output
    - cat output

- cal 11 2019 > calendario
    - cat calendario
- ps -le > procesos
    - cat procesos
    - head procesos
- head procesos > procesos_head
    - cat procesos_head

Redirección de **entrada:**

Creamos un fichero de texto con un poema presente en “Alice In Wonderland”:

- touch poem
    - Edit lines (+ paste poem)
- sort -r < poem
- rev < poem
- head -1 poem
- grep bat < poem

Redirección **entrada y salida** simultáneas:

- sort -r < poem > poem_ordered.txt
    - cat poem_ordered.txt
- rev < poem > poem_backward.txt
    - cat poem_backward.txt
- head -1 < poem > title
    - cat title

Redirección **salida error:**

- bash ana >& salida_error
    - cat salida_error

Ejemplos de ejecuciones:

```
msh> ps -le > procesos
msh> head procesos
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 56360 -      ?        00:11:58 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:01 kthreadd
1 I     0     4     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 I     0     6     2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0     7     2  0  80   0 -     0 -      ?        00:02:02 ksoftirqd/0
1 I     0     8     2  0  80   0 -     0 -      ?        00:07:58 rcu_sched
1 I     0     9     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    10     2  0 -40   - -     0 -      ?        00:00:00 migration/0
5 S     0    11     2  0 -40   - -     0 -      ?        00:01:02 watchdog/0
msh> sort -r < poem
Up above the world you fly,
Twinkle, Twinkle Little Bat
Twinkle, twinkle, little bat!
Like a tea tray in the sky.
How I wonder what you're at!
How I wonder what you're at!
msh> rev < poem > poem_backwards.txt
msh> cat poem_backwards.txt
taB elttiL elkniwT ,elkniwT
!ta er'uoy tahw rednow I woH
,ylf uoy dlrow eht evoba pU
.yks eht ni yart aet a ekiL
!tab elttil ,elkniwt ,elkniwT
!ta er'uoy tahw rednow I woH
msh> bash ana >& salida_error
msh> cat salida_error
bash: ana: No such file or directory
```

Objetivo 3

Ser capaz de reconocer y ejecutar en foreground líneas con dos mandatos con sus respectivos argumentos, enlazados con '|', y posible redirección de entrada estándar desde archivo y redirección de salida a archivo:

Dos mandatos **enlazados con |**:

- ls -la | grep poema
  - Mostrar archivos cuyo nombre contenga "poema"
- ls -la | grep -B2 parser.h
  - Mostrar línea que contiene parser.h y las dos anteriores
- ls -la | grep -A2 -B2 parser.h
  - Mostrar línea que contiene parser.h y las dos anteriores y posteriores
- ps -la | sort -r
  - Ordena los procesos activos alfabéticamente de Z a A
- du -h | grep logs
  - Muestra el uso en disco human readable y te muestra las líneas que contienen "logs"

Dos mandatos **enlazados con |** y redirección **salida**:

- ps -la | sort -r > act_proc
  - En act_proc guardamos lista de procesos activos ordenados alfabéticamente en orden inverso
- calendar | grep born > cumpleaños
  - Lista con personas que cumplen años hoy

Dos mandatos **enlazados con |** y redirección **entrada**:

- cat < lines | grep wonder
  - Muestra las líneas del poema que contienen "wonder"
- head -1 < lines | grep Bat
  - Título del poema, contienen la palabra "Bat"

Dos mandatos **enlazados con |** y redirección **entrada y salida**:

- df -h | tee disk_usage.txt

Ejemplos de ejecuciones:

```
msh> du -h | grep logs
8.0K    ./.git/logs/refs/remotes/origin
12K     ./.git/logs/refs/remotes
8.0K    ./.git/logs/refs/heads
24K     ./.git/logs/refs
32K     ./.git/logs
msh> cat < lines | grep wonder
lines: Error. No se encuentra el fichero
msh> cat < poem | grep wonder
How I wonder what you're at!
How I wonder what you're at!
msh> df -h | tee disk_usage.txt
Filesystem      Size  Used Avail Use% Mounted on
udev            463M     0  463M   0% /dev
tmpfs            99M  6.7M   92M   7% /run
/dev/sda         20G  9.0G  9.5G  49% /
tmpfs           493M     0  493M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           493M     0  493M   0% /sys/fs/cgroup
tmpfs            99M     0   99M   0% /run/user/1000
tmpfs            99M     0   99M   0% /run/user/1002
msh> head -3 disk_usage.txt
Filesystem      Size  Used Avail Use% Mounted on
udev            463M     0  463M   0% /dev
tmpfs            99M  6.7M   92M   7% /run
```

Objetivo 4

Ser capaz de reconocer y ejecutar en foreground líneas con más de dos mandatos con sus respectivos argumentos, enlazados con '|', y posible redirección de entrada estándar desde archivo y redirección de salida a archivo:

Más de dos mandatos **enlazados con |:**

- du -h | grep logs | sort
- cal -A2 -B2 | grep Fr | rev | wc -w

Más de dos mandatos **enlazados con |** y redirección **salida:**

- cal | cut -c 1-3,19-22 | sort
- cal | cut -c 1-3,19-21 | sort | rev > text

Más de dos mandatos **enlazados con |** y redirección **entrada:**

- cat < poem | grep wonder | wc -l
    - Número de líneas que contienen "wonder" en el archivo poema

Más de dos mandatos **enlazados con |** y redirección **entrada y salida:**

- cat < poem | grep wonder | wc -l > lines_wonder.txt

Más de dos mandatos **enlazados con |** y redirección de **salida con tee:**

- df -h | sort | tee output > /dev/null

Ejemplos de ejecuciones:

```
msh> cal | cut -c 1-3,19-21 | sort | rev > text
msh> cat text


 12 51
6   1
 82 22
    92
 41 8
 aS uS
msh> cat < poem | grep wonder | wc -l > lines_wonder.txt
msh> cat lines_wonder.txt
2
msh> df -h | sort | tee output > /dev/null
msh> cat output
/dev/sda         20G  9.0G  9.5G  49% /
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           493M     0  493M   0% /dev/shm
tmpfs           493M     0  493M   0% /sys/fs/cgroup
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            99M     0   99M   0% /run/user/1000
tmpfs            99M     0   99M   0% /run/user/1002
tmpfs            99M  6.7M   92M   7% /run
udev            463M     0  463M   0% /dev
msh>
```

Objetivo 5

Ser capaz de ejecutar el comando *cd*. El comando *cd* debe permitir tanto el acceso a través de rutas absolutas como relativas, además de la posibilidad de acceder al directorio especificado en la variable HOME si no recibe ningún argumento, escribiendo la ruta absoluta del nuevo directorio actual de trabajo.

Ejemplos de ejecuciones:

```
msh> pwd
/home/ana/sop2/minishell
msh> cd folder
msh> pwd
/home/ana/sop2/minishell/folder
msh> ls -la
total 8
drwxrwxr-x 2 ana ana 4096 Dec  3 18:15 .
drwxrwxr-x 4 ana ana 4096 Dec  3 18:15 ..
msh> cd casa
Directorio no válido
```

Objetivo 6

Evitar que los comandos lanzados en background y el minishell mueran al enviar las señales desde el teclado SIGINT y SIGQUIT, mientras los procesos en foreground respondan ante ellas.

Ejemplos de ejecuciones:

No he podido sacar ejemplos de esto ya que no se puede sacar un pantallazo de como no cambia minishell al presionar ctrl + c, por ejemplo.

COMENTARIOS PERSONALES

PROBLEMAS ENCONTRADOS

El principal problema que he tenido has sido poder contar con un entorno en el que compilar el código usando libparser.a y parser.h. Tras intentar hacerlo en mi máquina (ios), en el escritorio remoto de Ubuntu de my apps e intentar instalar virtualbox con linux en mi máquina he terminado usando un servidor con linux (neidio.com) que me ha prestado amablemente un amigo.

```
ana@neidio:~/practica2/minishell$ gcc myshell.c libparser.a -static -o myshell
ana@neidio:~/practica2/minishell$ pwd
/home/ana/practica2/minishell
ana@neidio:~/practica2/minishell$ ls -la
total 932
drwxrwxr-x 3 ana ana   4096 Dec  4 19:27 .
drwxrwxr-x 3 ana ana   4096 Dec  4 19:24 ..
drwxrwxr-x 8 ana ana   4096 Dec  4 19:24 .git
-rw-rw-r-- 1 ana ana     43 Dec  4 19:24 .gitignore
-rw-r--r-- 1 ana ana  20730 Dec  4 19:27 libparser.a
-rwxrwxr-x 1 ana ana 893336 Dec  4 19:28 myshell
-rw-rw-r-- 1 ana ana   8021 Dec  4 19:24 myshell.c
-rw-r--r-- 1 ana ana    267 Dec  4 19:27 parser.h
-rw-rw-r-- 1 ana ana     64 Dec  4 19:24 README.md
ana@neidio:~/practica2/minishell$
```

He intentado implementar el objetivo de ejecutar en background pero sinceramente, no sabía ni por dónde empezar. Creo que este objetivo es mucho más complicado que cualquier cosa que se ha explicado en clase.

CRÍTICAS Y MEJORAS

Por motivos de trabajo no puedo asistir presencialmente a casi ninguna clase y hecho mucho en falta que se comenten más cosas sobre las clases y las prácticas en el foro.

Todos los comentarios y recomendaciones que se hacen en clase me las pierdo. Si la opción de dispensa académica está aceptada para esta asignatura, los alumnos que la pedimos no deberíamos estar en inferioridad de condiciones.

En un lenguaje como C, echo de menos tener un entorno de desarrollo más completo donde poder debugear y donde poder ver de forma visual qué variables tengo, de qué tipo son, qué valores tienen, si apuntan a algún sitio a dónde y en qué posición de memoria están guardadas.

No sé debugear en C y creo que es algo que se debería enseñar. Y como entorno de desarrollo he usado Sublime, que no es más que un procesador de textos vitaminado. He tenido que poner y quitar demasiados **printf()** para averiguar lo que estaba haciendo.

TIEMPO DEDICADO

He dedicado casi más tiempo a conseguir un entorno en el que poder compilar el código que iba desarrollando que a desarrollarlo.

La práctica como tal la he desarrollado en distintas máquinas y en todos los ratos libres de los que he dispuesto en unos 10 días. Es difícil calcular el tiempo en horas dedicado ya que he trabajado en ella de forma intensa en los últimos 10 días, pero a grandes rasgos han sido unas 30 horas.

BIBLIOGRAFÍA

Compilación:

https://juncotic.com/compilacion-fuente-ejecutable/

https://stackoverflow.com/questions/20663645/ignoring-file-lib-a-file-was-built-for-archive-which-is-not-the-architecture-be?noredirect=1&lq=1

https://stackoverflow.com/questions/30948807/static-library-link-issue-with-mac-os-x-symbols-not-found-for-architecture-x8

Funciones de C:

https://www.geeksforgeeks.org/chdir-in-c-language-with-examples/

https://www.geeksforgeeks.org/constants-vs-variables-in-c-language/

https://www.programiz.com/c-programming/c-functions

https://www.gnu.org/software/libc/manual/pdf/libc.pdf

BASH comandos:

https://www.cheatography.com/davechild/cheat-sheets/linux-command-line/

https://codingornot.com/07-bash-argumentos-y-funciones-desde-linea-de-comandos

https://poesiabinaria.net/2016/04/como-procesar-multiples-argumentos-de-entrada-en-scripts-para-bash/

https://gist.github.com/LeCoupa/122b12050f5fb267e75f

Entrada / Salida:

https://www.atareao.es/tutorial/terminal/redirigir-entrada-y-salida-en-linux/

Guías de estilo en C:

C Code Style Guidelines: https://www.cs.swarthmore.edu/~newhall/unixhelp/c_codestyle.html

CS50: https://cs50.readthedocs.io/style/c/#comments

Nasa Style Guide:
https://pdfs.semanticscholar.org/4c0f/60983b227236f3a56332079f0f80086f7d00.pdf

POEM

Poema procedente del libro de Lewis Carrol “Alice’s Adventures in Wonderland”. Lo he usado para testear el código y para mostrar ejemplos de la ejecución del código.

Twinkle, Twinkle Little Bat

Twinkle, Twinkle Little Bat

How I wonder what you're at!

Up above the world you fly,

Like a tea tray in the sky.

Twinkle, twinkle, little bat!

How I wonder what you're at!