42

MINISHELL

Creating you own shell

## Minishell

### 1.1. Por onde começar: Subject, Githubs, Vídeos, Testar Comandos

Antes de começar o minishell, devemos estudar para saber o que temos que fazer.

O processo passa por compreender o projeto na sua globalidade, indo seguidamente ao detalhe.

Para tal, começamos por ler o subject (por alto – dps lemos novamente mais profundamente) e folha avaliação do Walter, estudar o github de alguns colegas (Balsa e Walter), ver o vídeo sobre sinais do parágrafo seguinte, ver os vídeos que o Balsa tem no Github dele (após os vídeos do github do Balsa, ver os vídeos que filmei das explicações do Balsa sobre o projeto), ver apresentação do Balsa e Adam, reler o subjetc e ver a página de avaliação do Walter, e testar os comandos para entender como funciona a Shell bash.

Alguns projetos anteriores como o Pipex ou o Minitalk irão ser usados no Minishell, pelo que a sua compreensão é crucial. Pipex tem a ver com pipes, minitalk tem a ver com sinais...que pode ser útil pós Sinais da minishell, que são formas do sistema operativo comunicar com os processos. Por exemplo quando pressionas ctrl c o sistema operativo envia o sinal sigint e o processo decide o que fazer, geralmente é usado para interromper o programa - <https://youtu.be/83M5-NPDeWs?si=GyZIZhmCKjowfVsn>

**Ordem de estudo:**

1. **Ler subject** para saber o que é pedido (por alto – dps lemos novamente mais profundamente) e ler a **folha de avaliação do Walter:**

* <https://cdn.intra.42.fr/pdf/pdf/116143/en.subject.pdf>
* <https://github.com/waltergcc/private-evaluation_pages/blob/main/minishell_evaluation_page.pdf>

1. Ver github Balsa e ver github do Walter:

* <https://github.com/LuisBalsa/Minishell>
* <https://github.com/waltergcc/42-minishell>

1. Ver o **vídeo sobre sinais**: <https://youtu.be/83M5-NPDeWs?si=GyZIZhmCKjowfVsn>
2. **Ver vídeos** que estão aconselhados no Github do Balsa (e as explicações/resumos nas folhas de papel que eu fiz) e mais o que eu juntei.

* <https://www.youtube.com/watch?v=ubt-UjcQUYg>
* <https://www.youtube.com/watch?v=ZjzMdsTWF0U&t=1140s>

E estes eu juntei este: <https://www.youtube.com/watch?v=SToUyjAsaFk>

1. **Ver o código do Luís** e fazer (e rever) os **vídeos da explicação** do Minishell do **Balsa** (guardados no disco duro externo e/ou no google drive)
2. Ver a **apresentação que Balsa e o Adam** fizeram: <https://docs.google.com/presentation/d/1B9C5RdX7990AD5i_BUcHaxHaU3B_f7qfwdOLw0jdsAI/edit#slide=id.p>
3. Ver **esquema do Adam:** <https://whimsical.com/42-minishell-CksZXguS9A2gGJRLfifAjk@or4CdLRbgroTMouPNJkxnWQ6GgduV53armjKdXDG8>
4. Reler o subject, ver a página de avaliação do Walter e **Testar comandos** ao mesmo tempo, de forma a entender o que cada comando faz. Os comandos a testar estão no github do Balsa, no Github do Walter, e aqui: <https://docs.google.com/spreadsheets/u/0/d/1TDwyd-S0WBAXehgkrKQtBJ6zquQ4p6k7JfE5g3jICNA/htmlview#gid=0>
5. Entender o VPATH para automatizar o MAKEFILE
6. Como último ponto (e caso necessário), ver slack do rank3 para ver info que possa existir de jeito: <https://app.slack.com/client/T039P7U66/C048MP7L230>

-------------------------------

#### Explicações do Balsa – Github Balsa e vídeos sugeridos por ele no Github:

O Balsa colocou o minishell na página principal dele: <https://github.com/LuisBalsa>

Aqui está o link po minishell:

<https://github.com/LuisBalsa/Minishell>

[13:29, 26/03/2024] Luís Balsa 42 School: Vê o Readme do meu github

[13:30, 26/03/2024] Luís Balsa 42 School: https://github.com/LuisBalsa/minishell

[13:30, 26/03/2024] Luís Balsa 42 School: Tenho aí explicado

[13:30, 26/03/2024] Luís Balsa 42 School: Tenho aí dois vídeos

[13:31, 26/03/2024] Luís Balsa 42 School: Com esses dois vídeos ficas a entender como funciona esse algoritmo que falo na apresentação e consequentemente ficas a entender o que faço no meu. Depois é só tratar de pormenores específicos da minishell

Vídeos:

<https://www.youtube.com/watch?v=ubt-UjcQUYg>

<https://www.youtube.com/watch?v=ZjzMdsTWF0U&t=1140s>

E estes eu juntei este: <https://www.youtube.com/watch?v=SToUyjAsaFk>

Msg de mim para Caio: E o Luís Balsa vai fazer uma apresentação do minishell nos próximos dias (aos novos alunos que vão entrar), e aí explica como fez o minishell baseado numa binary tree. Podes ver aqui: <https://docs.google.com/presentation/d/1B9C5RdX7990AD5i_BUcHaxHaU3B_f7qfwdOLw0jdsAI/edit>

----------------------------------

#### Explicações do Walter – Github e página de avaliação do Walter:

Msg de mim para Caio: Mas vês esse vídeo sobre sinais, lês o ficheiro READ\_ME do minishell do Luís, lês o READ\_ME do Walter (<https://github.com/waltergcc/42-minishell>) e a página de avaliação do projeto (no repo do Walter: https://github.com/waltergcc/private-evaluation\_pages; <https://github.com/waltergcc/private-evaluation_pages/blob/main/minishell_evaluation_page.pdf>) e ficas com uma ideia muito clara sobre o que é para fazer no minishell.

### Esquema do Adam sobre Overview do Minishell

Ficheiro passado pelo Adam:

<https://whimsical.com/42-minishell-CksZXguS9A2gGJRLfifAjk@or4CdLRbgroTMouPNJkxnWQ6GgduV53armjKdXDG8>

### Comandos para testar Minishell

Testa estes e se encontrares outros comandos guarda numa lista tua para testares depois. [22:21, 09/04/2024] Luís Balsa 42 School: No meu github e do Walter tens mais

[22:21, 09/04/2024] Luís Balsa 42 School: A ideia é testares isto na bash, não sei se me estou a fazer entender

[22:22, 09/04/2024] Luís Balsa 42 School: E quando estiveres a fazer a tua shell já vais saber como ela se tem de comportar:   
<https://docs.google.com/spreadsheets/u/0/d/1TDwyd-S0WBAXehgkrKQtBJ6zquQ4p6k7JfE5g3jICNA/htmlview#gid=0>

**1 - Lista comandos Global:**

<https://docs.google.com/spreadsheets/u/0/d/1TDwyd-S0WBAXehgkrKQtBJ6zquQ4p6k7JfE5g3jICNA/htmlview#gid=0>

**2 - Lista Comandos no Github do Balsa**

https://github.com/LuisBalsa/Minishell

Command List for Tests

echo ~

Print the home directory of the current user

cat <<e | grep hello | grep hello | grep world | grep -o l | wc -l

A complex pipeline of text processing commands

cat /dev/random | head -c 100 | wc -c

Generate random data and count the number of characters

env -i ./minishell

Run the minishell with a clean environment

/dev/urandom

You should see: minishell: /dev/urandom: Permission denied

sleep 5 | sleep 5 | sleep 5

Should return in 5 seconds

"ls > 1 > 2 > 3"

When you don't have the right permissions on file 2

ls -l | wc -l

Then run ls -l | wc -l > aqui (the second command creates a file called 'aqui', and this file counts)

cat < input.txt | wc

If input.txt does not exist You should see:

minishell: input.txt: No such file or directory

0 0 0

cat << $USER $USER

> $USER

You should see: cat: luide-so: No such file or directory

echo " ' $ ' " don't expand

The text within single quotes is treated literally

notcmd

Then run exit, echo $? should return 127

notexists < input.txt | notexists2 | notexists3

If input.txt does not exist

notcmd1 && (notcmd2 || echo $?)

vs.

notcmd1 && notcmd2 || echo $?

Check the difference in the logic of these commands

cat <file

You should see: minishell: file: No such file or directory

touch file

Create a file

chmod 000 file

Remove all permissions from the file

cat <file

You should see: minishell: file: Permission denied

asdsad

You should see: minishell: asdsad: command not found

$EMPTYVAR

Show a new prompt line

echo $?

Should return 0

sadasdads | $EMPTYVAR

You should see: minishell: sadasdads: command not found

echo $?

Should return 0

ls | cat -e", setup="touch a b c d

You should see: cat: invalid option -- ','

echo bonjour | | cat -e

You should see: minishell: syntax error near unexpected token '|'

asdf | echo bonjour

You should see: bonjour and then minishell: asdf: command not found

cat <$USE\*

You should see: minishell: $USE\*: ambiguous redirect

echo Hello && echo OK Error: $? && echoHello || echo Not OK Error: $?

You should see:

Hello

OK Error: 0

minishell: echoHello: command not found

Not OK Error: 127

echo -"n" hello

You should see: hello

echo -"n -nnnn" -n hello

You should see: -n -nnnn -n hello

echo "|$123USER|" "|$\_123USER123|" '"|$\_123USER123|"' "|$\_123USER?123|"

You should see: |23USER| || "|$\_123USER123|" |?123|

**3 - Lista Comandos no Github do Walter**

https://github.com/waltergcc/42-minishell

Tests

Check -Wall -Wextra -Werror

make -n

gcc -Wall -Wextra -Werror -c src/main.c -o obj/main.o

Simple Command

/bin/ls

include libft Makefile minishell minishell\_tester src

date

Wed 7 Jul 2021 11:00:00 AM -03

ifconfig

network information

who

wcorrea- tty2 2023-06-20 10:46 (tty2)

[empty]

print a new line

[space]

print a new line

[tab]

do nothing

Command with Arguments

touch 1 2 3

create 3 files

/bin/ls -l

list files with details

rm 1 2 3

remove 3 files

cat Makefile

print the content of the Makefile

cat -e Makefile

print the content of the Makefile with $ at the end of each line

wc -l Makefile

55 Makefile

echo

echo

print a new line

echo This is a test

This is a test

echo -n Hello World

Hello World/home/wcorrea-$

exit

exit

exit the minishell

exit 1

exit the minishell with return code 1

exit 42

exit the minishell with return code 42

exit -42

exit the minishell with return code 214

exit 42 10

minishell: exit: too many arguments

exit 42scholl

minishell: exit: numeric argument required; then exit with error code 2

Return value of a process

echo $?

0

/bin/ls

then run echo $?; the return code is 0

ls notexist

then run echo $?; the return code is 2

/bin/notexist

then run echo $?; the return code is 127

expr $? + $?

then run echo $?; the return code is 2

Signals

ctrl-C on empty prompt

display a new line with a new prompt

echo $? must display 130

ctrl-\ on empty prompt

do nothing

ctrl-D on empty prompt

quit minishell

echo $? must display 0

ctrl-C after write some stuff

display a new line with a new prompt

echo $? must display 130

ctrl-\ after write some stuff

do nothing

ctrl-D after write some stuff

do nothing

ctrl-C in cat without arguments

^C

echo $? must display 130

ctrl-\ in cat without arguments

\Quit (core dumped)

echo $? must display 131

ctrl-D in cat without arguments

close interactive mode

echo $? must display 0

Double Quotes

"/bin/ls"

execute ls

""ls""

execute ls

""ls -l""

execute ls with details

"""ls -l"""

minishell: ls -l: command not found

"echo Hello World"

minishell: echo Hello World: command not found

echo "Hello World"

Hello World

echo "with spaces"

Hello World

echo "test with %specials \*chars"

test with %specials \*chars

echo "cat lol.c cat > Iol.c"

cat lol.c cat > Iol.c

echo "

minishell: unclosed quotes

Single Quotes

'/bin/ls'

execute ls

''ls''

execute ls

''ls -l''

execute ls with details

'''ls -l'''

minishell: ls -l: command not found

'echo Hello World'

minishell: echo Hello World: command not found

echo 'Hello World'

Hello World

echo 'with spaces'

Hello World

echo '$USER'

$USER

echo '

minishell: unclosed quotes

echo '$HOME > home.txt'

$HOME > home.txt

echo '$?'

$?

env

env

print the environment variables

env PATH

minishell: env: Arguments and options aren't supported

export

export

print the exported environment variables

export NEW\_VAR=42

add NEW\_VAR with content 42

export NEW\_VAR="A little change"

replace the value of NEW\_VAR

export NEW\_VAR=

replace the content of NEW\_VAR to empty

export NEW\_VAR=$USER

replace the value of NEW\_VAR

export NEW\_VAR="with unclosed quotes

minishell: unclosed quotes

export -p NEW\_VAR

minishell: export: options aren't supported

export NEW\_VAR=" -l |"

replace the content of NEW\_VAR to ' -l |'

unset

unset NEW\_VAR

remove the environment variable

unset -n NEW\_VAR

minishell: unset: options aren't supported

cd

cd .

stay in the same directory

cd . ls

minishell: cd: too many arguments

cd ..

go to the parent directory

cd /usr/bin

go to the /usr/bin directory

cd

go to the home directory

cd notexist

minishell: cd: notexist: No such file or directory

pwd

pwd

/home/wcorrea-/workplace/common\_core/minishell/minishell

pwd -L

minishell: pwd: options aren't supported

Relative Path

./minishell

execute the minishell inside minishell

../../../../../../bin/ls

include libft Makefile minishell minishell\_tester src

Environment path

echo $PATH

/home/wcorrea-/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin

unset PATH

remove the PATH environment variable

ls

minishell: ls: command not found

export PATH=/bin:/usr/bin

set the PATH environment variable with the basic path again

Redirection

ls -l > test

create a file with the files list inside

cat Makefile >> test

append the content of the Makefile to the file

wc -l < test

63

wc -c < test > test\_out

test\_out receive the number of characters in the file test

cat << EOF

start the here document

cat << EOF | wc -l

print the number of lines in the here document

Pipes

cat Makefile | grep NAME | wc -l

5

cat notexist | wc -l

cat: notexist: No such file or directory 0

< Makefile grep NAME | wc -l > out

put the result in out

Go Crazy and history

up down to navigate in the history

dsfdfdfd

minishell: command not found

wc -l < /etc/passwd > file1 > file2 > file3 > file4

write the result in the last redirection

cat | cat | ls

print list files and open the interactive mode

Environment variables

echo $USER

wcorrea-

echo $notexist

print an empty line

echo "$HOME"

/home/wcorrea-

echo ~

/home/wcorrea-

echo ~test

~test

echo test~

test~

echo ~/test

/home/wcorrea-/test

Trouble Cases

$EMPTYVAR

show a new prompt line

echo -n -n -n -n -n Hello World

Hello World/home/wcorrea-$

echo -nnnnnnnnn Hello World

Hello World/home/wcorrea-$

echo $HOME.test

/home/wcorrea-.test

echo $HOME/test

/home/wcorrea-/test

;

minishell: no support for operator `;'

\*

minishell: no support for operator `\*'

>

minishell: syntax error near unexpected token `newline'

<

minishell: syntax error near unexpected token `newline'

< .

minishell: syntax error near unexpected token `newline'

>>

minishell: syntax error near unexpected token `newline'

<<

minishell: syntax error near unexpected token `newline'

|

minishell: syntax error near unexpected token `|'

d|

minishell: no support for pipe prompt

ls | wc |

minishell: no support for pipe prompt

ls | wc >

minishell: syntax error near unexpected token `newline'

ls | wc <

minishell: syntax error near unexpected token `newline'

ls | wc <<

minishell: syntax error near unexpected token `newline'

ls | wc >>

minishell: syntax error near unexpected token `newline'

ls | cat << > | wc

minishell: syntax error near unexpected token `<< '

export =

minishell: export: `=': not a valid identifier

export =test

minishell: export: `=test': not a valid identifier

export TEST.=test

minishell: export: `TEST.=test': not a valid identifier

export 1TEST=test

minishell: export: `1TEST=test': not a valid identifier

export TEST=ABCD

create a var TEST with content ABCD

export $TEST=1234

create a var ABCD with content 1234

export $ABCD=more

minishell: export: `1234=more': not a valid identifier

export a\_test b\_test

create tmp vars without content

unset a\_test

remove the tmp var a\_test

export b\_test=1234

transform the tmp var b\_test in a environment var with content 1234

unset b\_test

remove the environment var b\_test

export TEST1="withdouble" TEST2 TEST3=wihoutquotes TEST4="with spaces and expand $USER" TEST5="ok, but it has more" TEST6= TEST7='with single $PATH' = TEST8 9TEST9 TEST10="" TEST11=too=many=equals TEST.12=" " TEST13='withsingle' TEST14/

This test will try to create environment vars and tmp vars with different cases. Check the difference with env and export commands

unset TEST1 TEST2 TEST3 TEST4 TEST5 TEST6 TEST7 TEST8 9TEST9 TEST10 TEST11 TEST.12 TEST13 TEST14/

unset all these variables that were valid

Not required but done

cd -

go to the previous directory

<< EOF

start the here document

> file

create a file

>> file

append to a file

### VPATH para automatizar MAKEFILE

Só fica a faltar uma cena que o Balsa falou, que é o **VPATH:**

[11:38, 26/03/2024] João Silva: O VPATH é uma variável no Makefile que define os diretórios onde o Make procura por arquivos-fonte (dependências) quando tenta construir um alvo específico. O uso do VPATH é útil quando seus arquivos-fonte estão localizados em diretórios diferentes dos diretórios de construção e você deseja que o Make os localize automaticamente.

[11:38, 26/03/2024] João Silva: Organizei o minishell em pastas e depois usei VPATH para ir buscar tudo

[11:38, 26/03/2024] João Silva: VPATH = $(SRCS) $(addprefix $(SRCS)/, $(\_SUBFOLDERS)) bonus/$(SRCS) $(addprefix bonus/$(SRCS)/, $(\_SUBFOLDERS))

VPATH pode ser útil nisto:   
  
Se você adicionou uma subpasta a um projeto do GitHub que já tinha seu próprio repositório Git (um submódulo), isso pode causar problemas e pode não ser o que você deseja. Se você deseja incorporar a subpasta ao repositório pai, siga estas etapas:

1. \*Remover o submódulo\*:

Primeiro, você deve remover a subpasta que havia sido adicionada como um submódulo. Use o comando git rm com a opção --cached para remover o submódulo sem excluir os arquivos da subpasta localmente:

shell

git rm --cached NOME\_DA\_SUBPASTA

2. \*Adicionar e commitar as alterações\*:

Após remover o submódulo, faça um commit para confirmar a remoção:

shell

git commit -m "Remover submódulo NOME\_DA\_SUBPASTA"

3. \*Adicionar a subpasta ao repositório principal\*:

Agora que a subpasta não é mais tratada como um submódulo, adicione-a ao seu repositório principal como qualquer outro diretório. Use git add para adicionar a subpasta e, em seguida, faça um novo commit:

shell

git add NOME\_DA\_SUBPASTA

git commit -m "Adicionar NOME\_DA\_SUBPASTA ao repositório principal"

4. \*Push para o GitHub\*:

Por fim, faça um push das alterações para o GitHub:

shell

git push origin NOME\_DO\_BRANCH

Substitua NOME\_DA\_SUBPASTA pelo nome da subpasta que você deseja adicionar ao repositório principal e NOME\_DO\_BRANCH pelo nome do branch que você está usando.

Isso deve incorporar a subpasta ao repositório principal e enviá-la para o GitHub. Lembre-se de que, ao fazer isso, você está adicionando permanentemente a subpasta ao repositório principal, e ela não será mais um submódulo separado. Certifique-se de que isso é o que deseja antes de prosseguir.

### Slack do Rank3 (Minishell & Philosophers)

<https://app.slack.com/client/T039P7U66/C048MP7L230>