Tuan Nguyen

**Lab2 contains:**

-main.c : main source code

-makefile : to compile

-readme: help text file to know how to use the shell

-two folder: folder and file to test command and i/o redirect

-in.txt, out.txt, in2.txt, out2.txt: to test each command, i\o redirect , piping

-batch.txt, batch2.txt : batch file to execute sequence of command

**Program description**

**-**The shell support 8 commands, i\o redirect, background, piping ex:

- cd: change directory

-clr : clear the screen

-dir<directory>: list all contents under <directory>

-environ : list all environment variables

-echo<comment> : display <comment> in the screen

-help : display the user manual of use of program

-pause : pause the shell until user hit Enter to continues

-quit : quit the shell

-i/o : redirect input and output

-& : background execution

-| : piping two commands

First the program asks user input a shell command then parse the input into tokens: for example cd folder : input[0] input[1].

After parse input, it will pass each input into argument[], then it call run\_command fuction( pass in argument[]) to execute command.

**if user enter cd**: execute command to display current directory

**cd folder** : to change current directory to new directory

**cd <not folder name>** : print error message “ no file or directory found”

**if user enter dir**: list the content of current directory

**dir > in.txt file** : write list the content of current directory to in.txt file

**dir >> in.txt file** : append list the content of current directory to in.txt file

**dir filename:** list the content of filename directory to the screen

**dir filename >> in.txt file**: append the list of contents of filename directory to in.txt file

**dir filename > in.txt file:** write the list of contents of filename directory to in.txt file

**if user enter ls:** list the content of ls

**ls –l , ls –a, ls -r**

**if user enter clear**: clear the screen

**if user enter environ:** list all the environment into screen

**environ > in.txt file**: write all the environment to in.txt file

**environ >> in.txt file**: append all the environment to in.txt file

**if user enter help:** display help page into screen

**help > in.txt** : write help page to in.txt

**help >> in.txt** : append help page to in.txt

**if user enter echo “hello”:** display hello to screen

**echo hello > in.txt** : write hello to in.txt file

**echo hello >> in.txt** : append hello to in.txt file

**if user enter pause :** pause user input until they hit enter to continue

**if user enter external command &:** run external command in background andreturn to command line prompt immediately after run

ex: ls &

**if user enter exit:** the program will exit.

The program also support I/O redirect and Piping

I/O redirect can be reading input from a file and output to a file, or take input from a program.c and output to screen or a text file.

ex: hello < in.txt > out.txt: in.txt take input "hello" and write to out.txt

ex: cat < in.txt: print content of in.txt to screen

ex: hello < in.txt >> out.txt: in.txt take input "hello" and append to out.txt

Piping can be take command1 | command2 and strung them together

-ls –l | more : list the content of ls

-cat in.txt | wc –c: print the word count in in.txt to screen

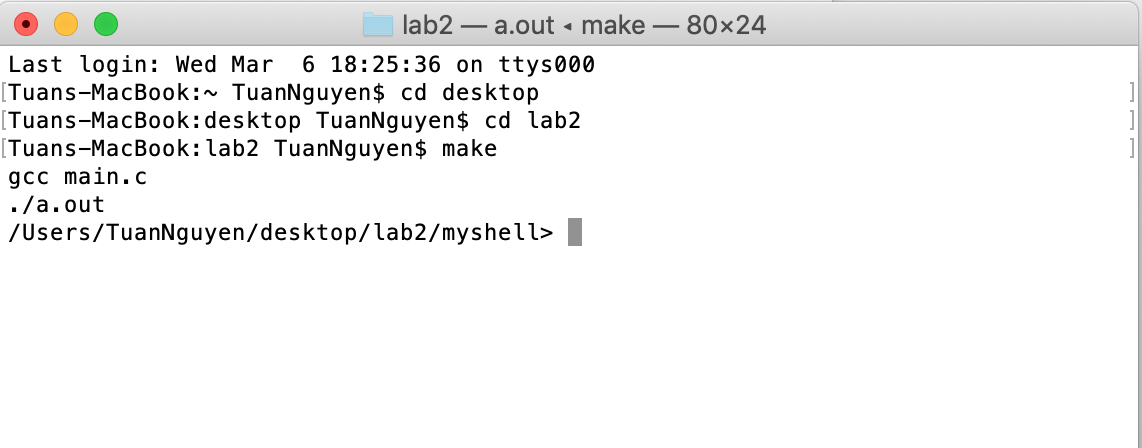
The program can run a batch file to execute sequence of command

-user enter : ./myshell batch.txt

To compile:

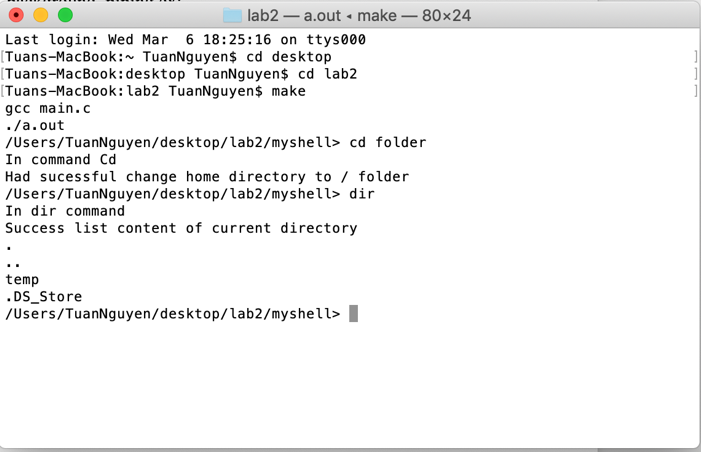
-cd lab2

-make



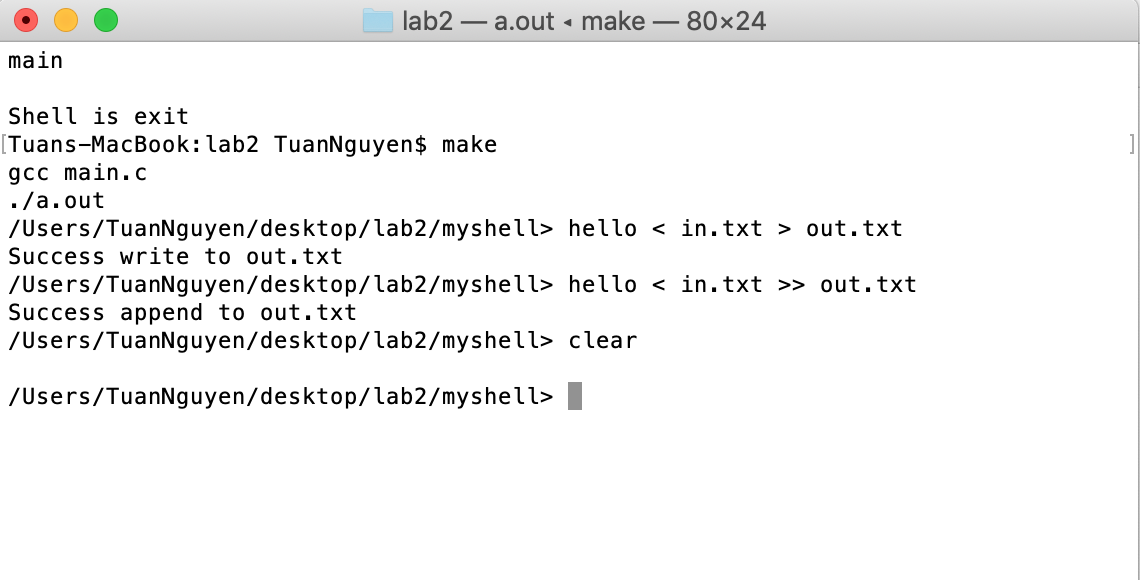
1) cd <dirname>: to change current directory to new directory

ex: cd folder , cd file



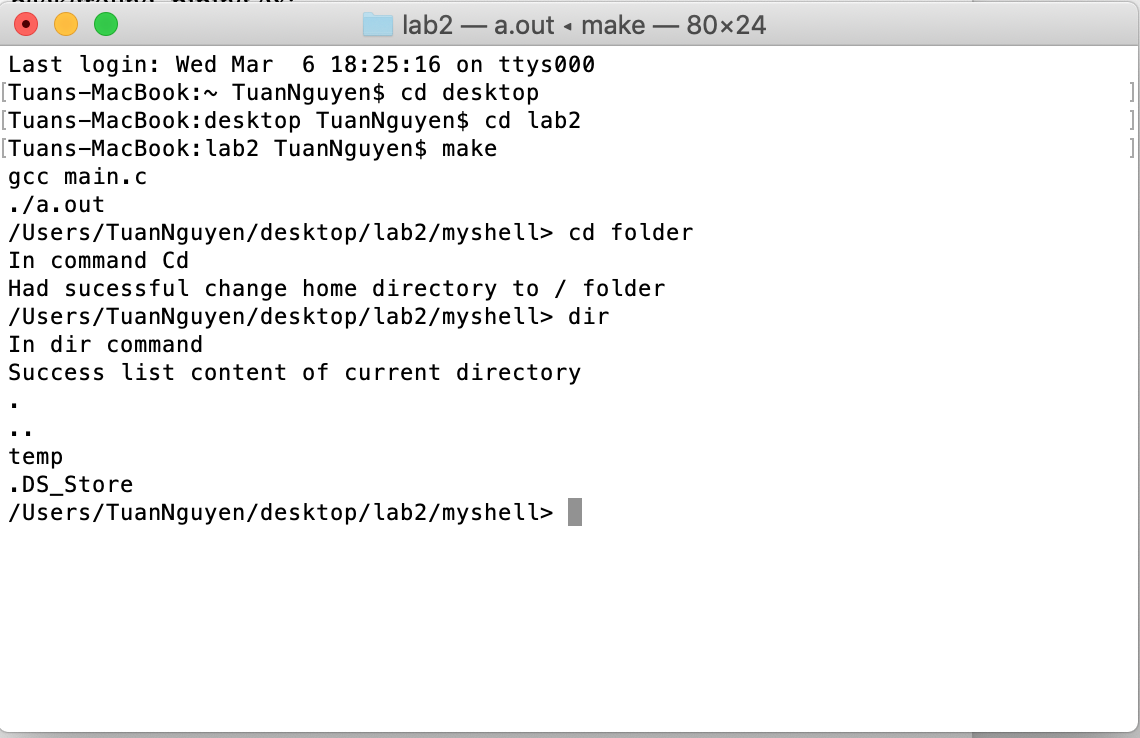
2) clear: Clear the screen with white space

ex : clear

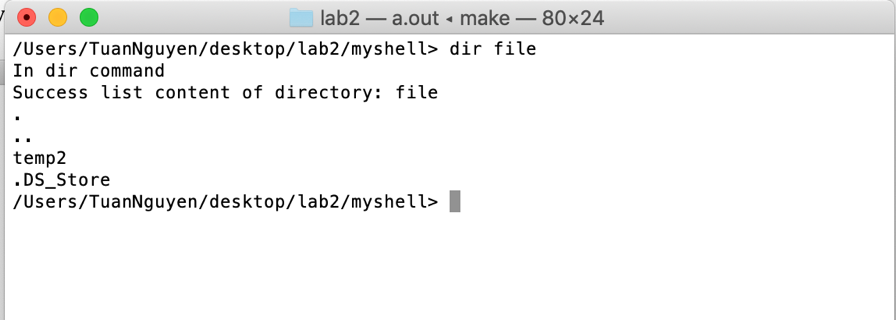


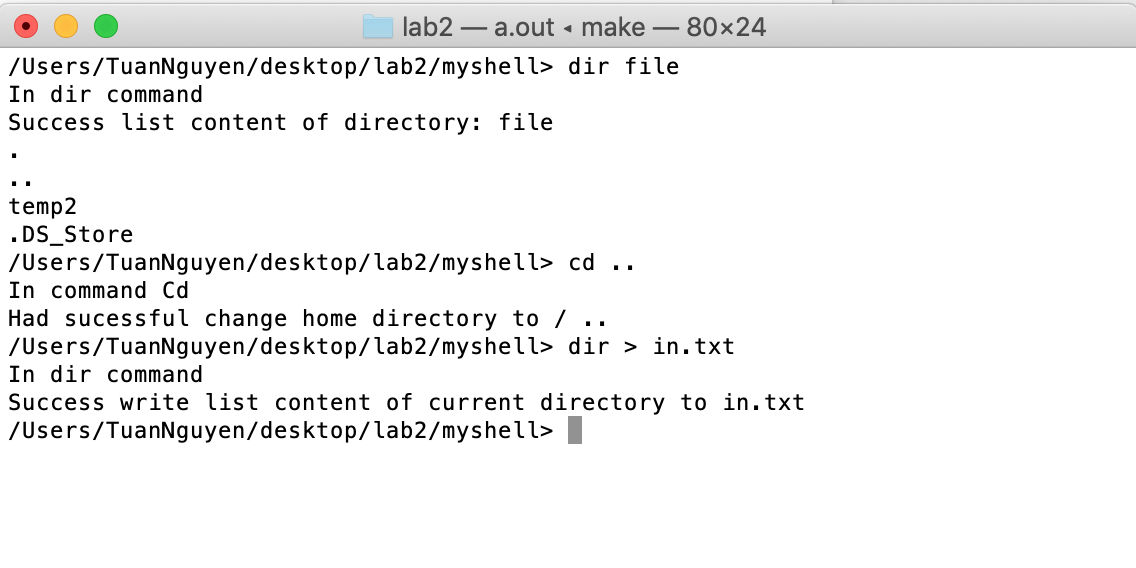
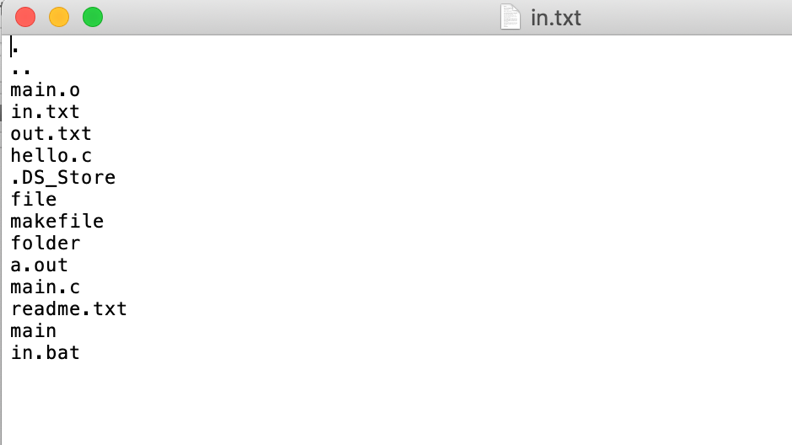
3) dir <dirname>: list the content of directory or write the content into a .txt file

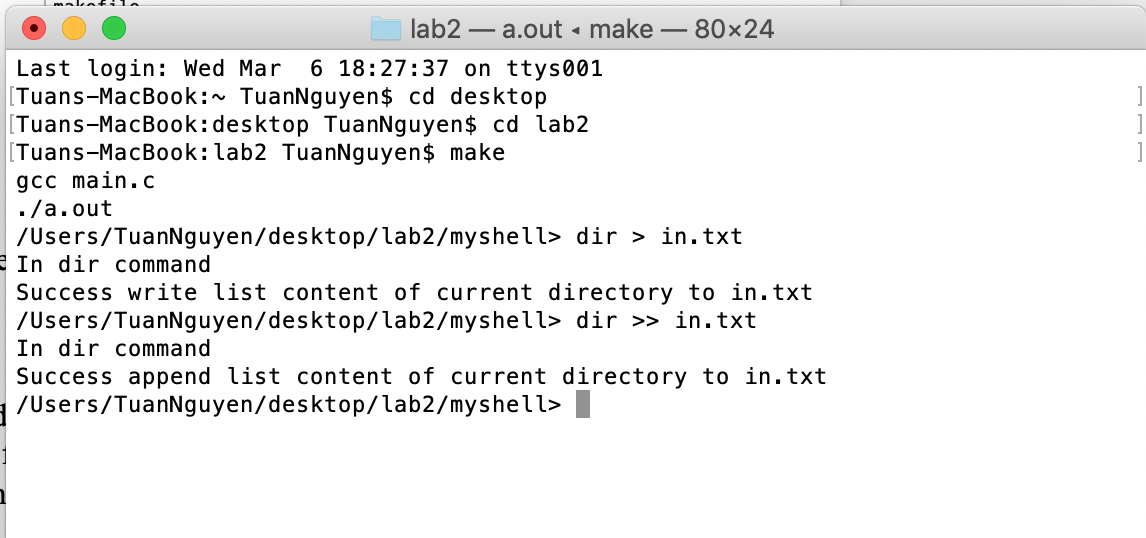
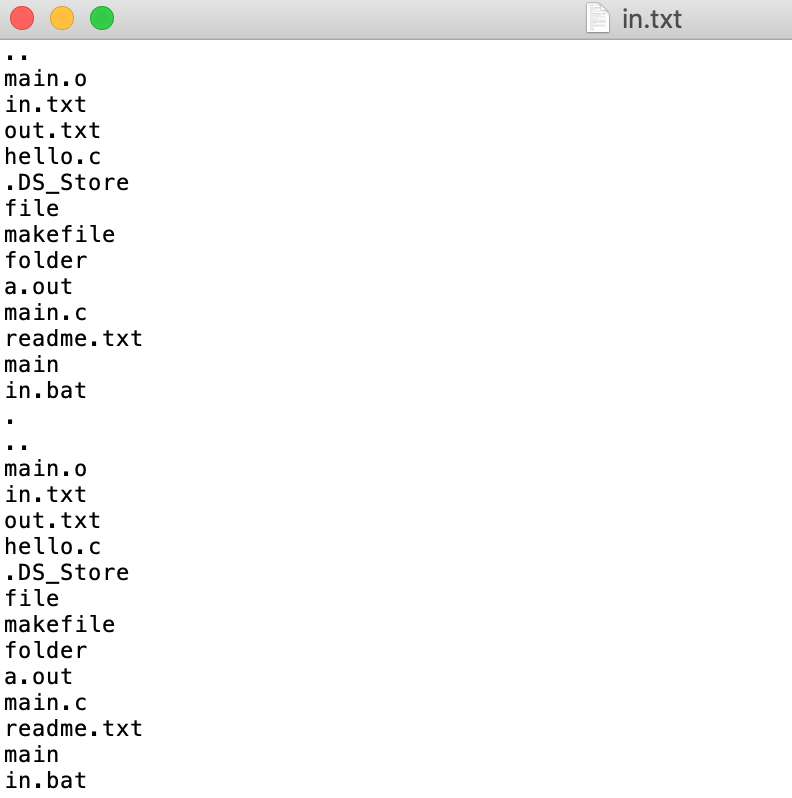
ex: dir folder : list the content of folder



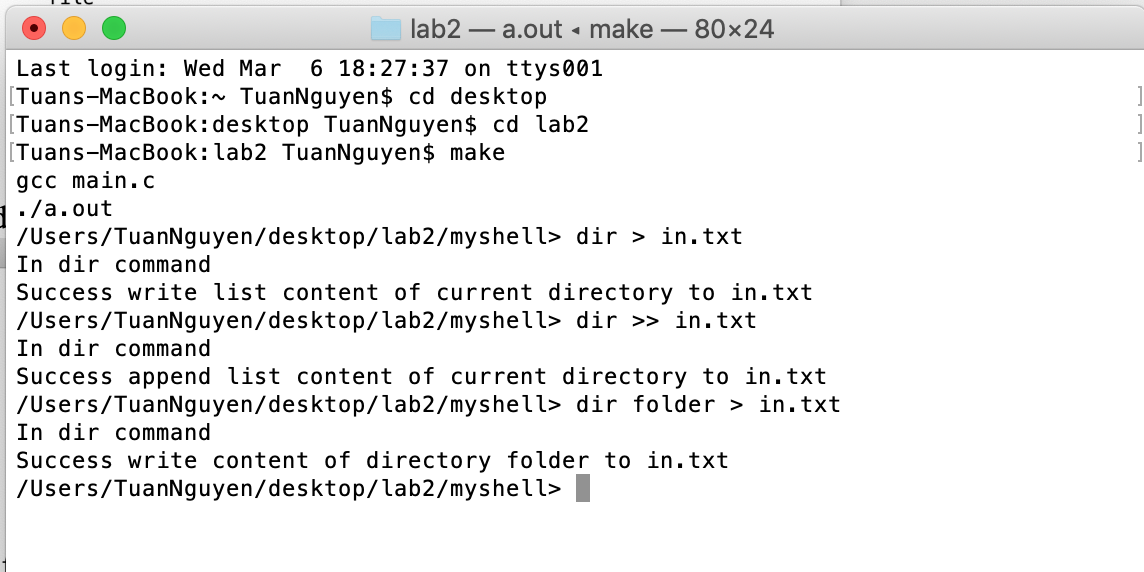
ex: dir file : list the content of file

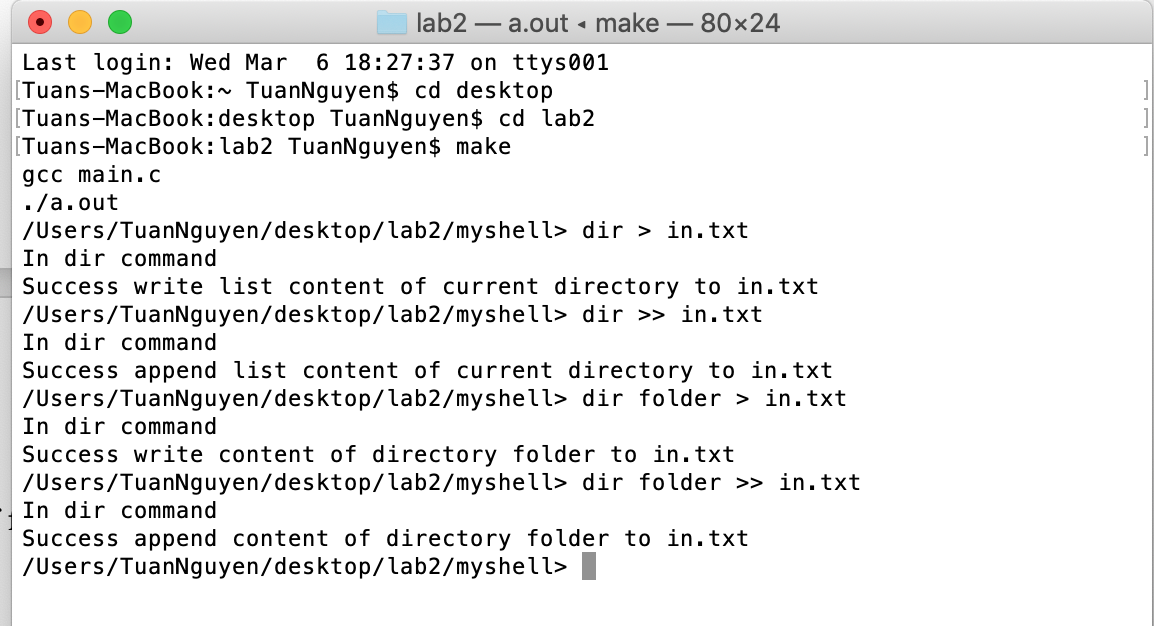
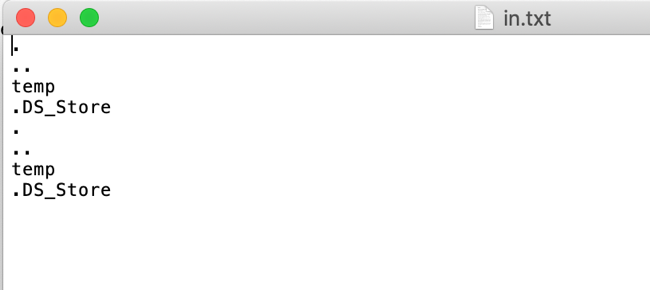


ex: dir > in.txt: write the content of current directory into in.txt

ex: dir >> in.txt: append the content of current directory into in.txt

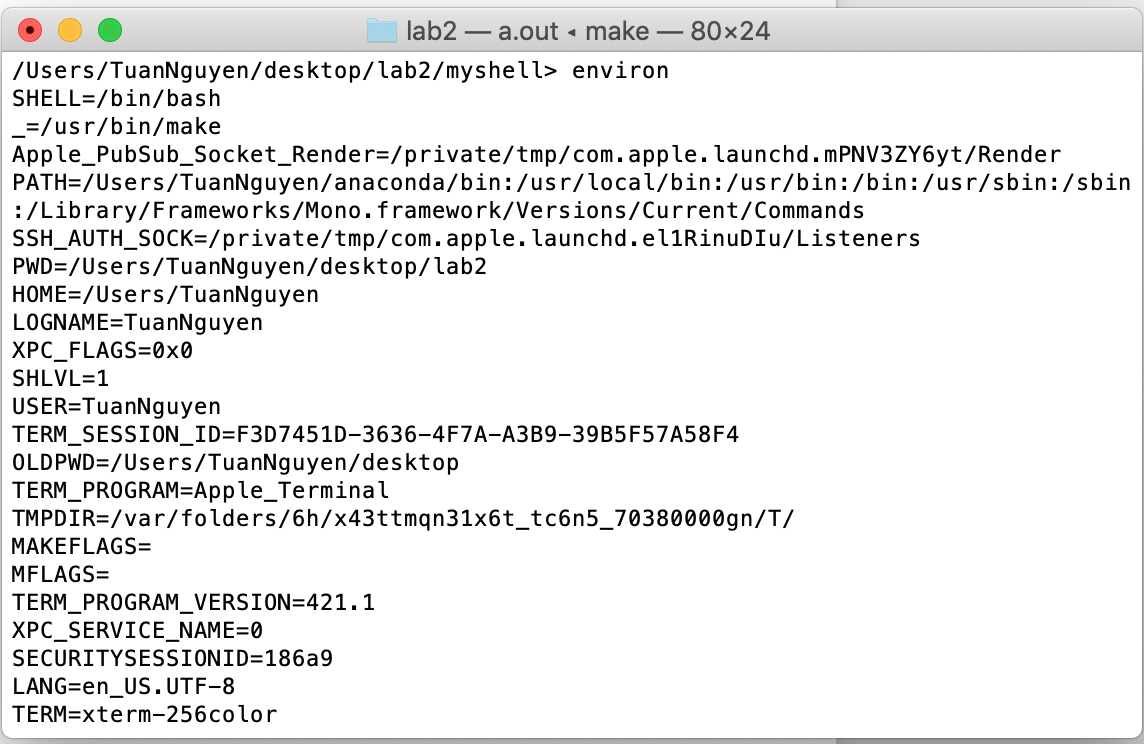
ex: dir folder > in.txt : write list the content of folder into in.txt



ex: dir folder >> in.txt : append list the content of folder into in.txt

4) environ - list all the environment into screen

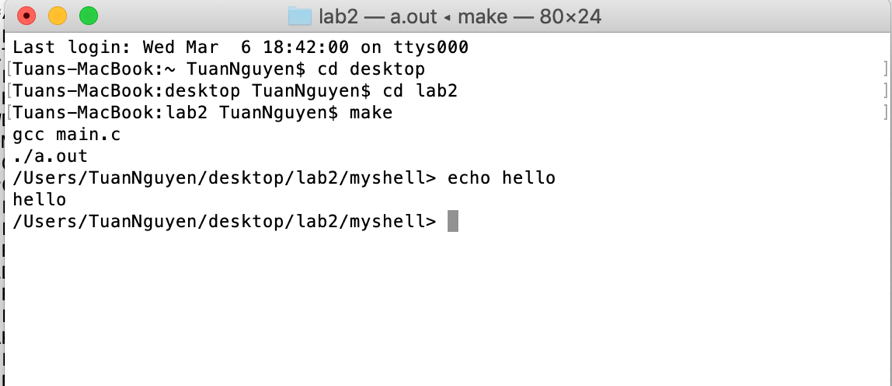
ex: environ



5) echo <argument> - display the <argument> follow by new line.

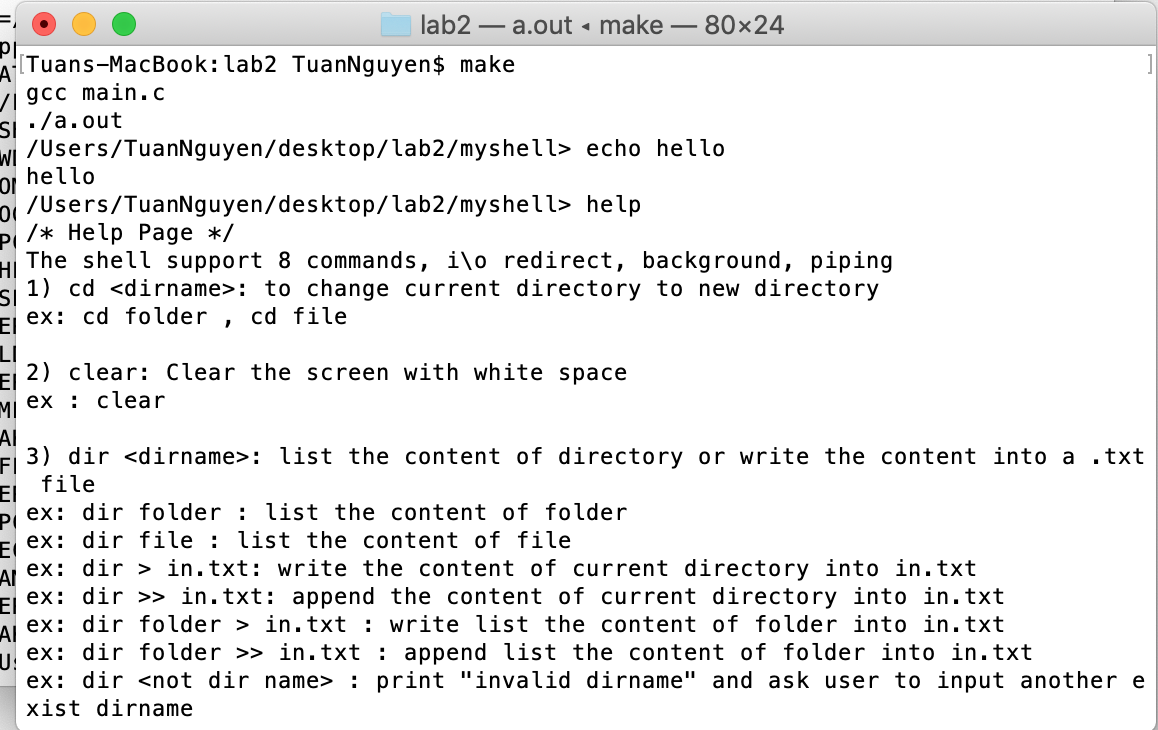
ex: echo hello

print: hello



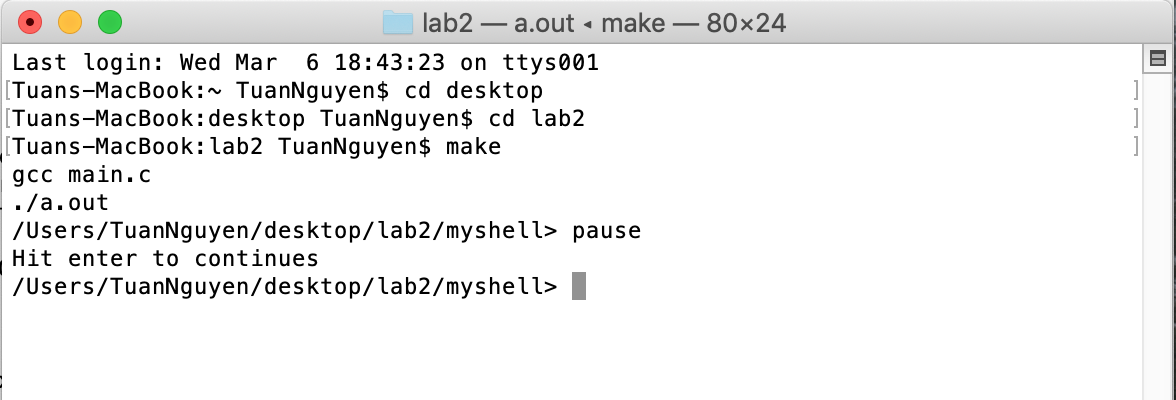
6) help : display help page into screen

ex: help



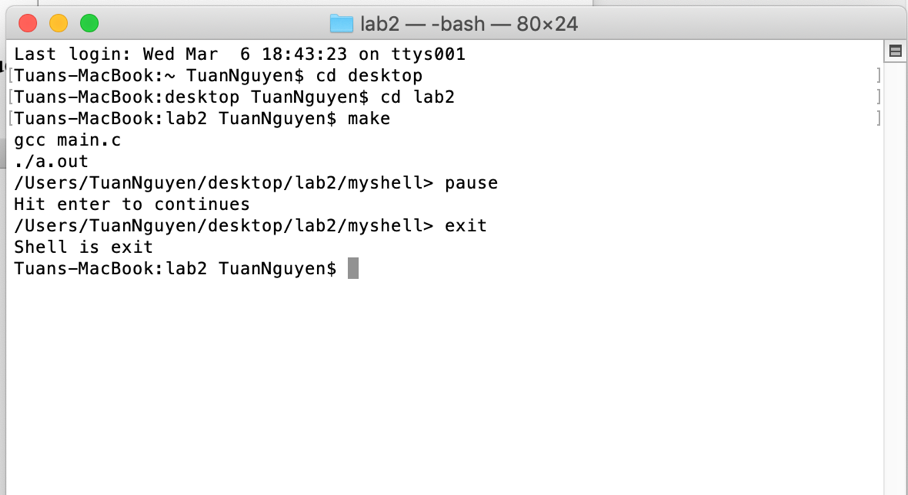
7) pause: pause user input until they hit enter to continue

ex: pause



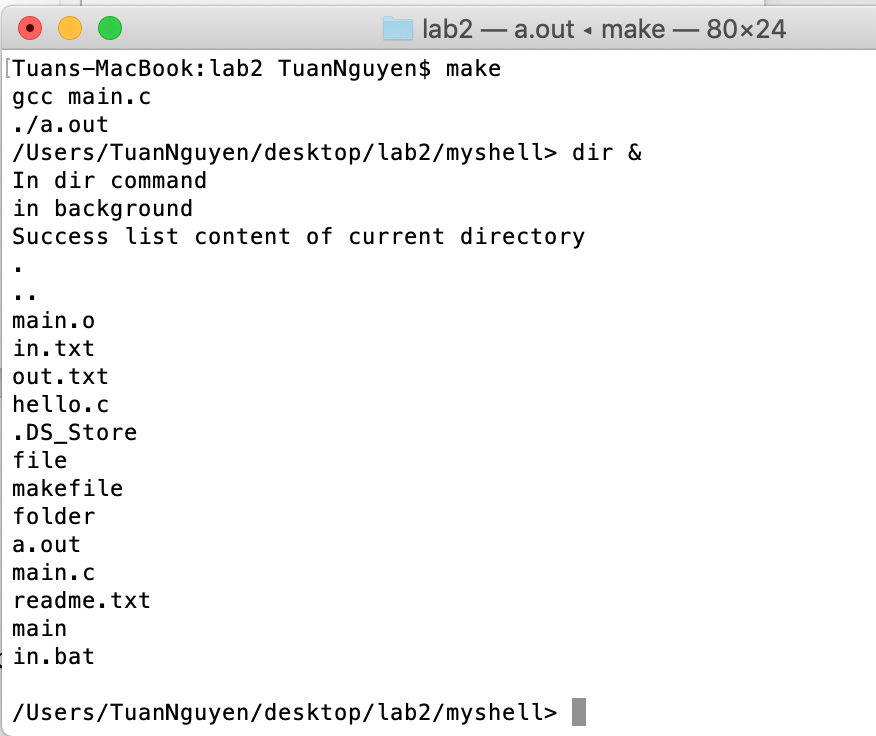
8) exit: quit the shell

ex: exit



Background command:

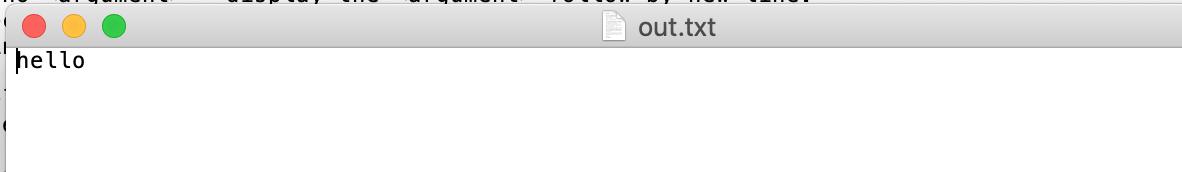
dir & : run dir in the background and return to command line prompt immediately after run dir



I/O redirect: redirection on either stdin and stout

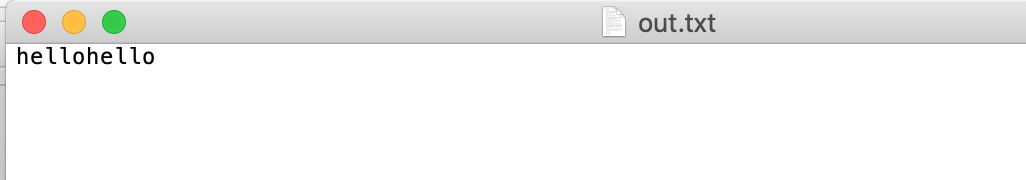
ex: hello < input.txt > out.txt: input.txt take input "hello" and write to out.txt



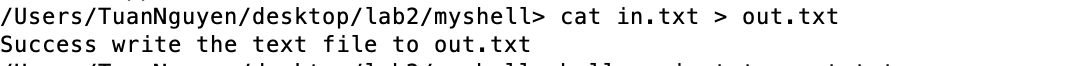


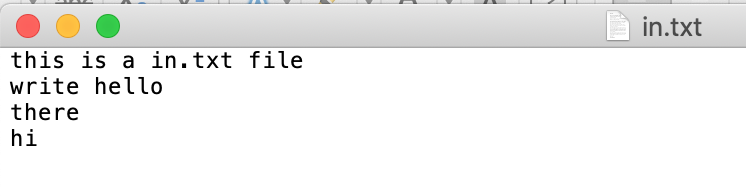
ex: hello < input.txt >> out.txt: input.txt take input "hello" and append to out.txt

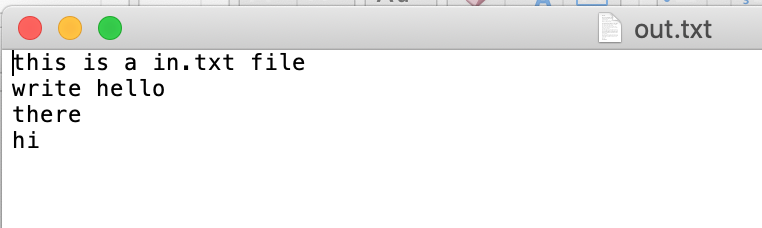




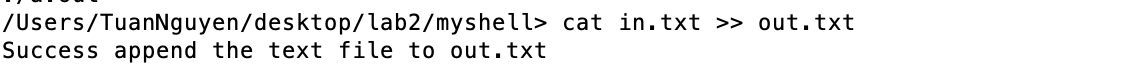
ex: cat input.txt > out.txt : write everything in input.txt to out.txt

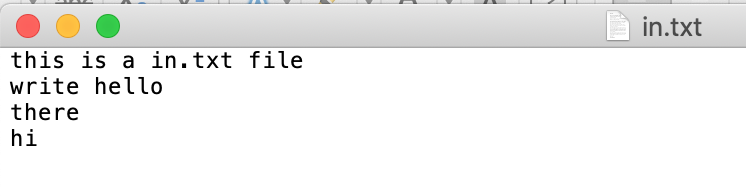


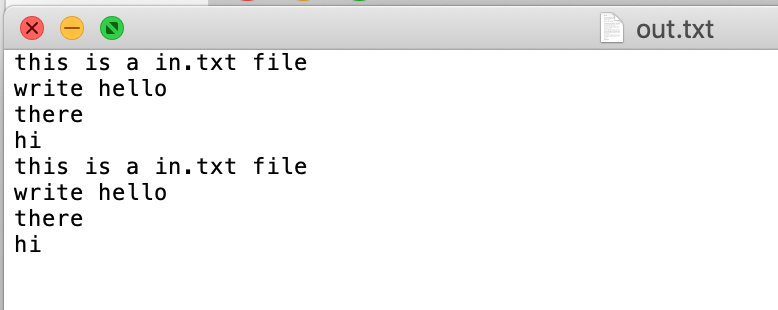




ex: cat input.txt >> out.txt : append everything in input.txt to out.txt





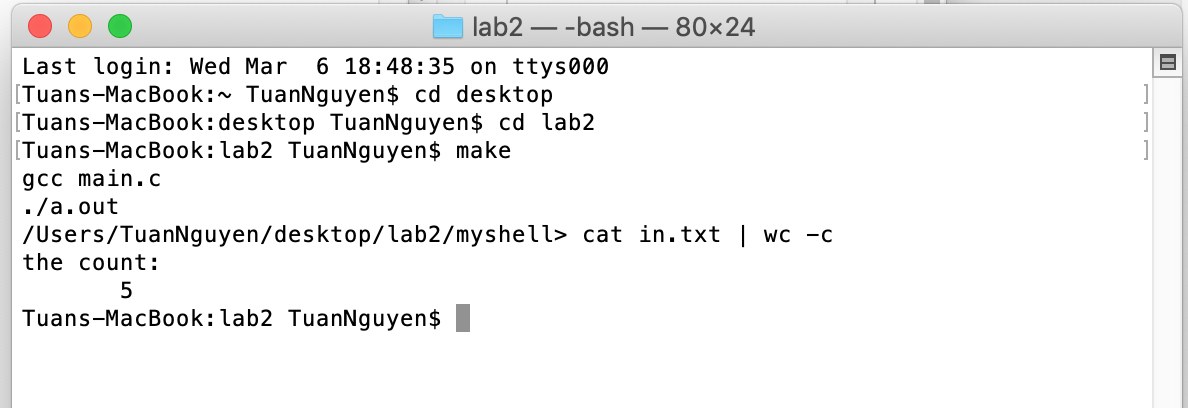


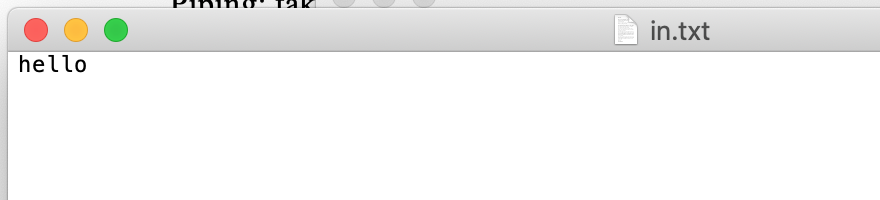
Piping: take command1 | command2 and strung them together

ex: cat in.txt | wc -c

-read the word inside out.txt and return the count.

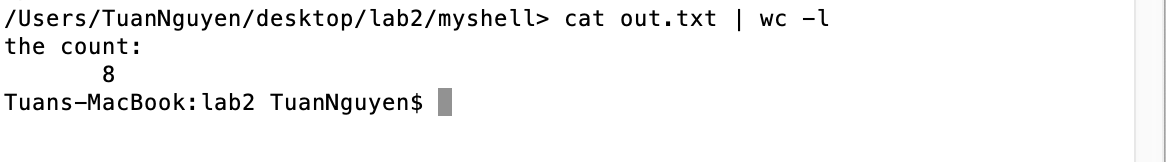
ex: word inside in.txt is "hello", and the count is 5.

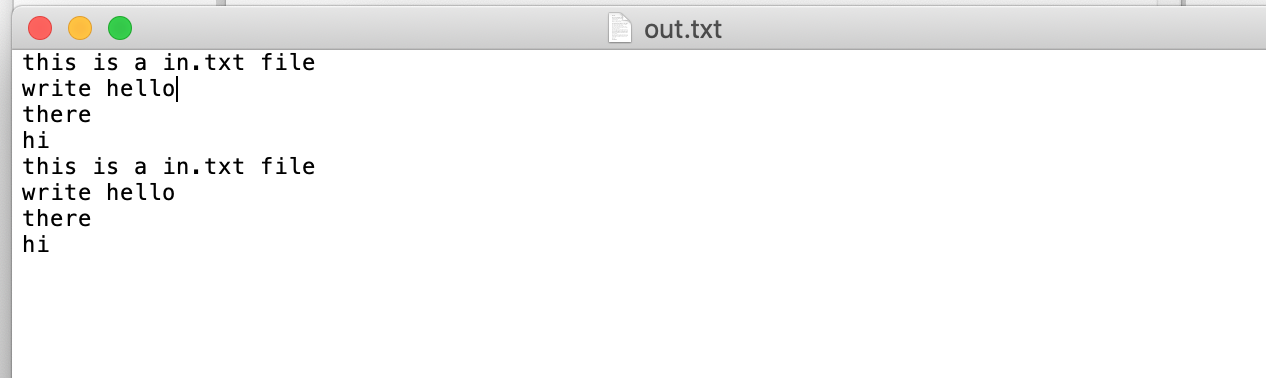


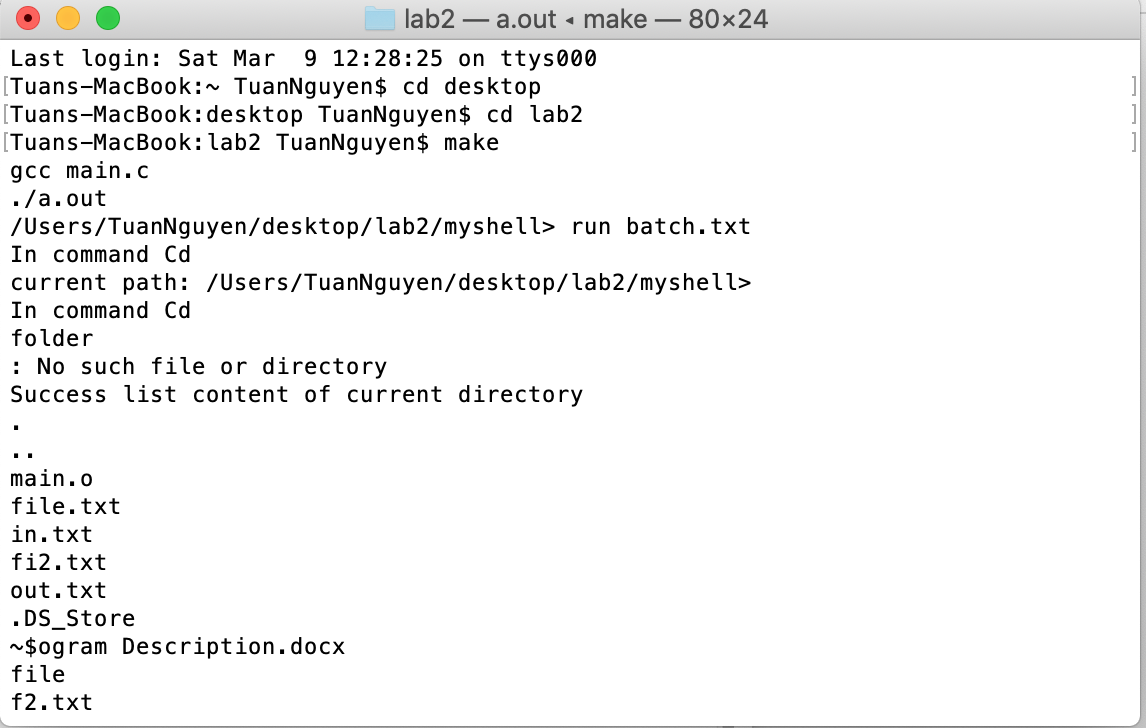
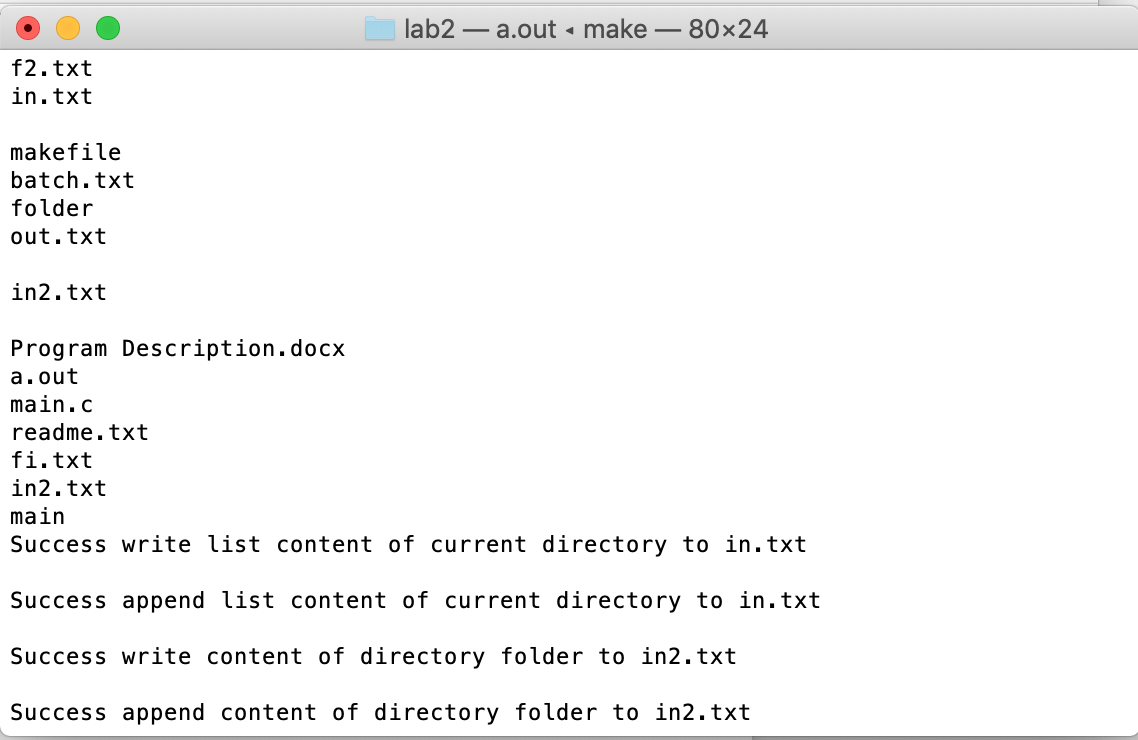


ex: cat out.txt | wc -l

-count how many lines inside out.txt , and return the count





Run batch file: type run batch.txt

