Unix Input-Output Redirection

Computer Science Department
CS2211a: Software Tools & Systems Programming
Fall 2013
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Standard Input/Output

- When you give the shell a command, it tried to find an executable program with the same name as the command
  - If it find it, it executes the program
  - If it does not find it, it tells that it can not find the program
- During execution, program's input/output/errors are directed from/to
  - *Standard Input* (stdin)
    - place from which programs read (by default terminal keyboard)
  - *Standard Output* (stdout)
    - place to which programs write (by default terminal screen)
  - *Standard Error* (stderr)
    - place where errors are reported (by default terminal screen)
- To demonstrate the standard I/O, cat (concatenate) command will be utilized
  - cat without arguments
    - Echoes line-by-line everything typed at the standard input to standard output
    - Quits when you press Ctrl-d at a new line -- (EOF)

Standard Input/Output

- *Example:*

```
obelix[11]% cat
This is my first standard I/O example
This is my first standard I/O example
The example includes three lines
The example includes three lines
This is the third and last line
This is the third and last line
obelix[12]%
```

Enter has been hit

Enter has been hit

Enter has been hit

Enter has been hit

Ctrl-d at a new line has been hit

- Standard input, standard output and standard error
  - Can be *redirected by user* to a *file* or a *device* other than the terminal

Redirecting Standard Output

- The ***redirect output*** symbol, >, instructs the shell to redirect a command's output to the specified file instead of to the screen

```
command [options] [arguments] > filename
```

- Use caution when redirecting outputs, since the shell may overwrite existing filename

```
obelix[12]% cat > file1
This is my first standard I/O example
The example includes three lines
This is the third and last line
obelix[13]% ls -l file1
-rw-------  1 elsakka  faculty      103 Sep 21 15:08 file1
obelix[14]%
```

Enter has been hit

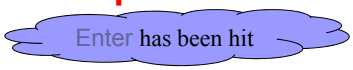
Enter has been hit

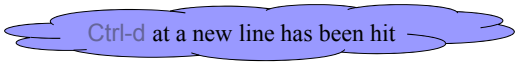
Enter has been hit

Ctrl-d at a new line has been hit

Redirecting Standard Output

```
obelix[14]% cat > file2
This is my second example
It is just two lines
obelix[15]% ls -l file[12]
-rw-------  1 elsakka  faculty     103 Sep 21 15:08 file1
-rw-------  1 elsakka  faculty      47 Sep 21 15:20 file2
obelix[16]% cat file1 file2 > file3
obelix[17]% ls -l file[312]
-rw-------  1 elsakka  faculty     103 Sep 21 15:08 file1
-rw-------  1 elsakka  faculty      47 Sep 21 15:20 file2
-rw-------  1 elsakka  faculty     150 Sep 21 15:22 file3
obelix[18]%
```





Redirecting Standard Output

- The ***append output*** symbol, >>, instructs the shell to add the new data to the end of a file, accumulating on any data that was already there

```
command [options] [arguments] >> filename
```

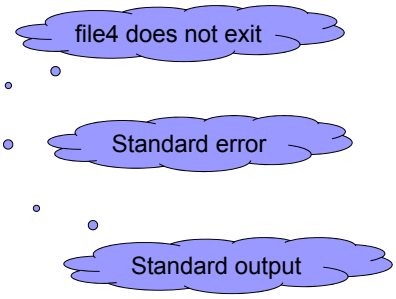
  - filename is created if not there
  - filename is appended if it is there

```
obelix[18]% ls -l file[1-3]
-rw-------  1 elsakka  faculty     103 Sep 21 15:08 file1
-rw-------  1 elsakka  faculty      47 Sep 21 15:20 file2
-rw-------  1 elsakka  faculty     150 Sep 21 15:22 file3
obelix[19]% cat file2 >> file3
obelix[20]% ls -l file[1-3]
-rw-------  1 elsakka  faculty     103 Sep 21 15:08 file1
-rw-------  1 elsakka  faculty      47 Sep 21 15:20 file2
-rw-------  1 elsakka  faculty     197 Sep 21 15:58 file3
obelix[21]%
```

Redirecting Standard Error

- Standard error is
  - another output
  - typically used to output error messages
- Unless standard error is redirected, the shell sends standard error to the terminal screen

```
obelix[21]% cat file4 file2
cat: cannot open file4
This is my second example
It is just two lines
obelix[22]%
```



Redirecting Standard Error

- Standard output can be redirected, *independent* from standard error

```
obelix[22]% cat file4 file2  > output_file
cat: cannot open file4
obelix[23]%
obelix[23]% cat output_file
This is my second example
It is just two lines
obelix[24]%
```

Redirecting Standard Error

- In Bourne-style shells (e.g., sh and ksh)
  - the ***standard error*** symbol is 2>
  - the ***standard output*** symbol is 1>, or simply >

    cat file4 file2 > output_file 2> error_file

  - Standard output and standard error *can be* redirected *to the same file* using 2>&1 notation
    - For example: cat x y > xy.txt 2>&1
  - To ***append***, use
    - ***2>> instead of 2>***
    - ***1>> instead of 1>***
    - ***>> instead of >***
  - ***Do not put any space between 1>, 1>>, 2>, 2>>, 2>&1***

Redirecting Standard Error

- In C-style shells (e.g., csh and tcsh)
  - the ***standard error*** and ***standard output*** can be redirected *together* to a file using >&

    cat file4 file2 >& both_output_and_error_file

  - the ***standard error*** and ***standard output*** can be *independently* redirected to two files as follows

    (cat file4 file2 > output_file) >& error_file

  - To ***append***, use
    - ***>>& instead of >&***
    - ***>> instead of >***
  - ***Do not put any space between >&, >>&, >>***

Redirecting Standard Input

- The ***redirect input*** symbol, <, instructs the shell to redirect a command's input from the specified file instead of from the keyboard

```
command [options] [arguments] < filename
```

```
obelix[24]% cat < file1
This is my first standard I/O example
The example includes three lines
This is the third and last line
obelix[25]%
```


Redirecting Standard Input

- Using cat with input redirection from a file yields *almost* the same results as giving by a cat command with the file name as argument.

  The difference here is in who will open the file, the ***shell*** or the cat

```
obelix[25]% rm file1
obelix[26]% cat < file1
file1 : No such file or directory
obelix[27]% cat file1
cat: cannot open file1
obelix[28]%
```

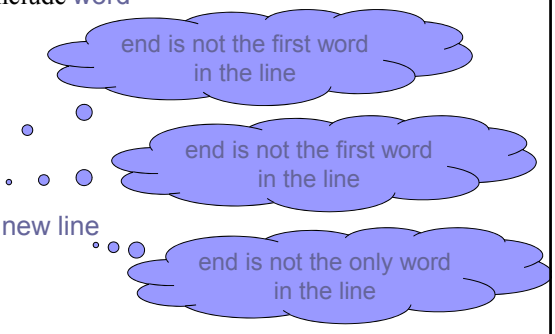
The error message came from the shell

The error message came from the cat command

Redirecting Standard Input

- Standard input can also be redirected from the text to be written immediately after the command using << word notation
  - Standard input data starts from the line following the command
  - Standard input data continues till the occurrence of a line that *start* with word
  - The word line must only include that word followed by new line
  - Standard input data will not include word
- Example

```
cat > out_file  << end
This is some sample text.
Here is a line with end in it.
Here is another line with end.
end but does not followed by new line
No more lines with that word.
end
```



Redirecting Standard Input

- Any word can be used, even something like + would work fine
- Example 2

```
cat > out_file_2  << +
This is some sample text.
Here is a line with end in it.
Here is another line with end.
end but does not followed by new line
No more lines with that word.
+
```

tr

- tr means *translate*
- tr [options] string1 string2
  - reads from standard input
  - modifies (i.e., *substitute* or *delete)* selected characters in the read input
  - writes the results to standard output
- The options specified, as well as the string1 and string2 operands, control the translations that occur while copying characters
- Example :

```
obelix[25]% tr  a-z  A-Z  <  file2
THIS IS MY SECOND EXAMPLE
IT IS JUST TWO LINES
Obelix[26]%
```

- *How can we translate a content of a file to lower case letter?*

grep

- grep means *global regular expression and print*
- grep [options] pattern_string
  - Reads standard input and
  - writes lines containing pattern_string to standard output
- Example :

```
grep "unix is easy"  < myfile1  > myfile2
```

  - Write all lines in myfile1 that containing phrase unix is easy to myfile2

wc

- wc means *words count*
  - Prints the number of ***lines***, ***words***, and ***bytes*** in a files
- Example :

```
obelix[26]% wc < file1
3      19     103 file1

obelix[27]% wc  file1 file2
3      19     103 file1
2      10      47 file2
5      29     150 total
obelix[28]%
```


sort

- sort
  - Sort all input lines in alphabetical order
  - Write the result to the standard output

```
obelix[28]% cat > file4 <<+
This is one line
The example includes three lines
This is one line
+
obelix[29]% sort file4 > file5
obelix[30]% cat file5
The example includes three lines
This is one line
This is one line
obelix[31]%
```

uniq

- uniq
  - Remove duplicated lines from a sorted file
  - write the result to the standard output

```
obelix[31]% uniq file5
The example includes three lines
This is one line
obelix[32]%
```

- For more info about sort and uniq, see Chapter 19

Pipes

- A pipe:
  - Connects *standard output of one program* to *standard input of another*
  - General form:

    command1 | command2
    - standard output of command1 used as standard input for command2
- Example 1:

```
cat readme.txt | grep unix | wc -l
```

An alternative way (not efficient) is to:

```
grep unix < readme.txt > tmp
wc -l < tmp
rm tmp
```

- Example 2:

```
ls -l | more
```

Redirecting and Pipes

- The following commands are equivalent

```
cat readme.txt | grep unix | wc -l > output_file
cat readme.txt | grep unix | > output_file wc -l

grep unix < readme.txt | wc -l > output_file
< readme.txt grep unix | wc -l > output_file
grep unix < readme.txt | > output_file wc -l
< readme.txt grep unix | > output_file wc -l

grep unix readme.txt | wc -l > output_file
grep unix readme.txt | > output_file wc -l
grep -c unix readme.txt > output_file
> output_file grep -c unix readme.txt
```


Redirecting and Pipes

- Correct

```
cat readme.txt | grep unix | wc -l > output_file
cat readme.txt | grep unix | > output_file wc -l
```

- Wrong!! Why?

```
cat readme.txt | grep unix | output_file > wc -l
```

- Correct

```
grep unix < readme.txt | wc -l > output_file
< readme.txt grep unix | wc -l > output_file
```

- Wrong!! Why?

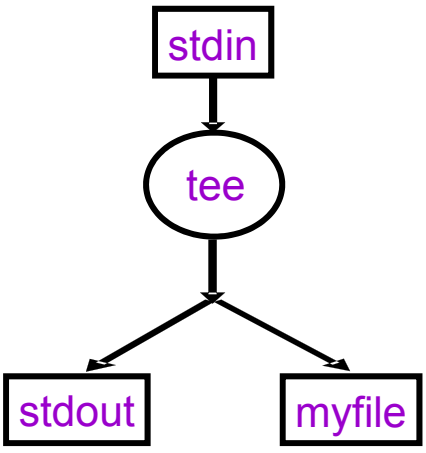
```
readme.txt > grep unix | wc -l > output_file
```

The tee Command

- cat readme.txt
  sent the content of readme.txt to the standard output
- cat readme.txt > myfile
  store the content of readme.txt in myfile
- How about if I want to do these two operations at once.

The tee Command

- tee - ***replicates*** the standard output (like a T connection)
  cat readme.txt | tee myfile



/dev/null

- /dev/null
  - A virtual file that is *always* empty
  - Redirecting standard error to /dev/null means *discard errors*
    - Bourne-style shells

      ls -l > recordfile 2> /dev/null
    - C-style shells

      (ls -l > recordfile) >& /dev/null
  - Copying from /dev/null will lead to an *empty file*
    - cp /dev/null myfile
    - This is equivalent to

      rm myfile

      touch myfile