

LPIC-1 101-500 – Lesson 4

103.4 Use streams, pipes, redirects



Introduction

- Linux treats input/output as a stream of data
- Because “**Everything is a File**” we can use the same methods to interact between commands, files, devices, etc



Standard I/O stream

- **Standard Input (stdin)**

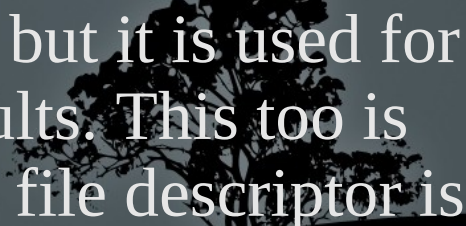
This is the stream that feeds input data and it is attached in the keyboard by default. The file descriptor for **stdin** is **0**

- **Standard Output (stdout)**

This is the stream used by commands for the output of their computations. It is attached to the terminal by default. Its file descriptor is **1**

- **Standard Error (stderr)**

This, just like stdout, is an output stream but it is used for errors and warnings instead of output results. This too is attached to the terminal by default and its file descriptor is **2**



Pipes

- Pipes allow us to redirect the **stdout** (or **stderr** even) from one command to the **stdin** of another. The operator for pipes is the vertical bar “|” e.g.:

```
$ find /usr/share | xargs file | grep -i png | less
```



Pipes

- `$ cut -d: -f4 /etc/passwd | less #`
stream the results from the 4th column of `/etc/passwd` to `less`
- `$ cut -d: -f4 /etc/passwd | sort #`
sort output data from `cut` alphabetically
- `$ cut -d: -f4 /etc/passwd | sort -n #`
sort output data from `cut` numerically
- `$ cut -d: -f4 /etc/passwd | sort -n |`
`uniq #` filter duplicated entries from the previous command

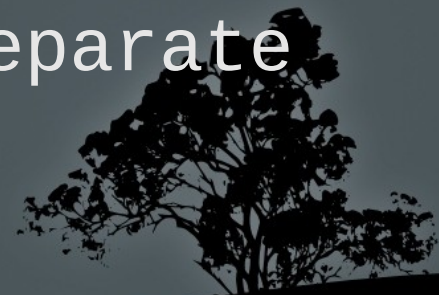
Pipes

- `$ find /usr/share -type f | wc -l`
find total number of files
/usr/share
- `$ echo "1 + 1" | bc # equals 2! :)`



Reuse output as arguments with ``xargs``

- `xargs` accepts some text from `stdin` and reuses that as arguments to the command that follows `xargs`
- `$ find /etc | xargs #` if used without arguments it just print the parameters
- `$ cat /etc/passwd | xargs -d: #` uses ":" as delimiter. Each line in `passwd` is accepted as a series of arguments
- `$ cat /etc/passwd | xargs -d: -n1 #` uses ":" as delimiter. Each line in `passwd` is accepted as a separate argument



Reuse output as arguments with ``xargs``


- `$ find ~ -mtime 365 | xargs file #` executes the `file` command in all the files of the home directory, older than a year
- `$ find / -name "*.ttmp" | xargs rm -f #` delete all files ending in `.tmp` This command will not play correctly if there are spaces or other special characters in the file name
- `$ find / -name "*.tmp" -print0 | xargs -0 rm -f #` same as above but works better with spaces and special characters in filenames. That's because it uses 'Null' as the argument delimiter instead of space.



Reuse output as arguments with ``xargs``

- `$ cut -d: -f1 /etc/passwd | xargs -n 1 id`
all the users in the system pass as a separate argument to the `id` command

Options:

- `-n 2` # 2 arguments per line
 - `-0` # special characters are treated literally. Combined with `-print0` from `find`
 - `-d` # Set delimiter
 - `-p` # interactive operation of `xargs`
- 

Redirections

- Redirection is the ability to detach Standard Streams (stdin, stdout, stderr) from their default devices (keyboard, terminal) and redirect them to system files e.g.:

```
$ cat file1 file2 file3 > merged-file #  
redirect the contents of file1, file2,  
file3 to a new file, merged-file.
```



Redirections

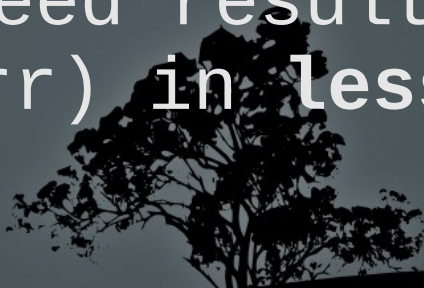
Redirection Operator	Effect
> or 1>	Create a new file from stdout data. If the file exists the previous content is discarded. '1' is the file descriptor for stdout
>> or 1>>	Append stdout data to an existing file. If the file does not exist a new one will be created.
2>	Create a new file from stderr data. If the file exists the previous content is discarded. '2' is the file descriptor for stderr
2>>	Append stderr data to an existing file. If the file does not exist a new one will be created.
&>	Create a new file from stdout and stderr data. If the file exists the content is discarded
&>>	Append data from stdout and stderr to an existing file. If the file does not exist a new one is created
2>&1	Redirect stderr to stdout
<	Capture data from a file and redirect to a command's stdin
<<	Send the following text to stdin
<>	A file can be used both as stdin and as stdout at the same time

Redirections

- `$ cut -d: -f1 /etc/passwd > user-list.txt`
create a list of users from passwd
- `$ echo "# $(date +%F)" >> ~/.history #`
Append, as comment, of the current date at the end of the command history file
- `$ find /etc > results.txt 2> errors.txt #`
save results in results.txt end errors in errors.txt



Redirections

- `$ find /etc &> everything.txt # send results and errors in everything.txt`
 - `$ find /var &>> everything.txt # append results and errors in everything.txt`
 - `$ find /var >> everything.txt 2>&1 # identical to the command above`
 - `$ find /etc | less # feed results (stdout) in less`
 - `$ find /etc 2>&1 | less # feed results and errors (stdout + stderr) in less`
- 

Redirections

- `$ mail -s "Welcome to LPI!" theo < message.txt #`
this command will send the contents of `message.txt` as an email to `theo` with subject "Welcome to LPI!"
- `$ cat > song.txt # This will create a new file`
`This is the end! # with this content`
`Ctrl+c # Press Ctrl+c to terminate it`
- `$ cat > song2.txt << EOF # create a new file`
`This is the end! # with this contents`
`EOF # terminated with EOF on a new`
`# line`



Redirections

- `$ tr a-z A-Z < /etc/fstab > fstab.caps`
convert `fstab` content to upper-case and save to a new file
- `$ grep important file1 > file2 ;`
`mv file2 file1` # This will keep only the lines containing the string 'important' in `file1`

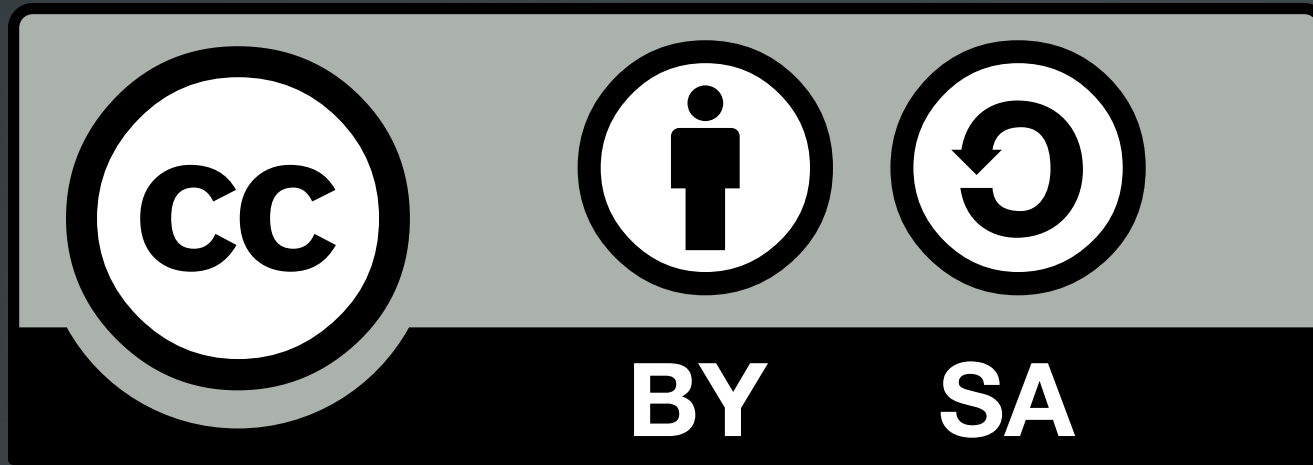
Examples to avoid!

- `$ grep important file1 > file1` # never do this! You will end up with an empty file! Redirections have precedence over commands.

Output to terminal and files with `tee`

- `$ find /etc | tee results.txt #` display results and errors in terminal, save results only in `results.txt`. If the file exists, old content is discarded
- `$ find /etc 2> /dev/null | tee results.txt #` display results only in terminal, save results only in `results.txt`. If the file exists, old content is discarded
- `$ find /etc 2>&1 | tee all.txt #` display results and errors in terminal, save results and errors in `all.txt`. If the file exists, old content is discarded
- `$ find /var 2>&1 | tee -a all.txt #` display results and errors in terminal, append results and errors in `all.txt`. If the file does not exist, a new one is created

License



The work titled "LPIC-1 101-500 – Lesson 4" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

