

[Software Development]

# ***Linux Tool/Set (part A)***

*Davide Balzarotti*

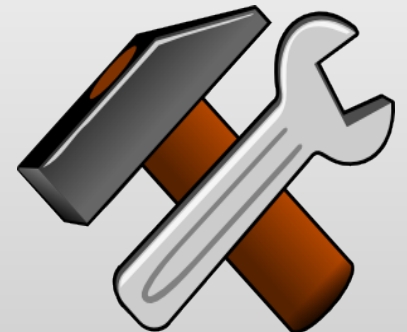
Eurecom – Sophia Antipolis, France

# If Unix can Do It, You Shouldn't

- As a general rule:

*Don't implement your own code unless you have a **solid** reason why common tools won't solve the problem*

- Common tools are capable of performing many tasks
- Most of the hard problems have already been solved (by very smart people) for you...
  - ... you just need to know where to look
  - ... and choose the right tool for your job



# Getting Help

- Any Linux distribution comes with thousands of pre-installed tools
  - Knowing every command is difficult
  - Knowing every option of each command is practically impossible
- The most useful skill is to know where to look for help
  - Google knows all the answers  
(if you know how to pose the right questions)
  - `help` – documentation for the shell builtin commands
  - `cmd --help` – most of the programs give you some help if you ask

# The Manual

- Unix man pages were introduced in 1971 to have an online documentation of all the system commands
- The pages are organized in 8 sections:
  1. general commands
  2. system calls
  3. C functions
  4. special files
  5. file formats
  6. games
  7. misc
  8. admin commands

```
CAL(1)                                BSD General Commands Manual                                CAL(1)

NAME
    cal, ncal - displays a calendar and the date of easter

SYNOPSIS
    cal [-3jmy] [[month] year]
    ncal [-jJpwy] [-s country_code] [[month] year]
    ncal [-Jeo] [year]

DESCRIPTION
    The cal utility displays a simple calendar in traditional format and
    ncal offers an alternative layout, more options and the date of easter
```

# Other forms of Help

- `info` – access the Info pages
  - Official documentation of the GNU project
  - Simple hypertext
- `apropos` – searches for keywords in the header lines of all the manpages

```
balzarot:~> apropos calendar
cal (1)          - displays a calendar and the date of easter
calendar (1)     - reminder service
gcal (1)         - a program for calculating and printing calendars.
gcalcli (1)      - Google Calendar Command Line Interface
konsolekalendar (1) - Command line interface to KDE calendars
ncal (1)         - displays a calendar and the date of easter
```

# Builtins

- Some commands (`kill`, `echo`) exist in two versions:  
as builtin and as external tool
- **`type [-t] [-a] name`**  
tells you how `name` would be interpreted if used as a command
  - alias
  - keyword (i.e., a shell reserved word)
  - function (i.e., a shell function)
  - builtin
  - file

```
balzarot:~> type -a ls
ls is aliased to `ls --color -FX -rt'
ls is /bin/ls
```

# Builtins

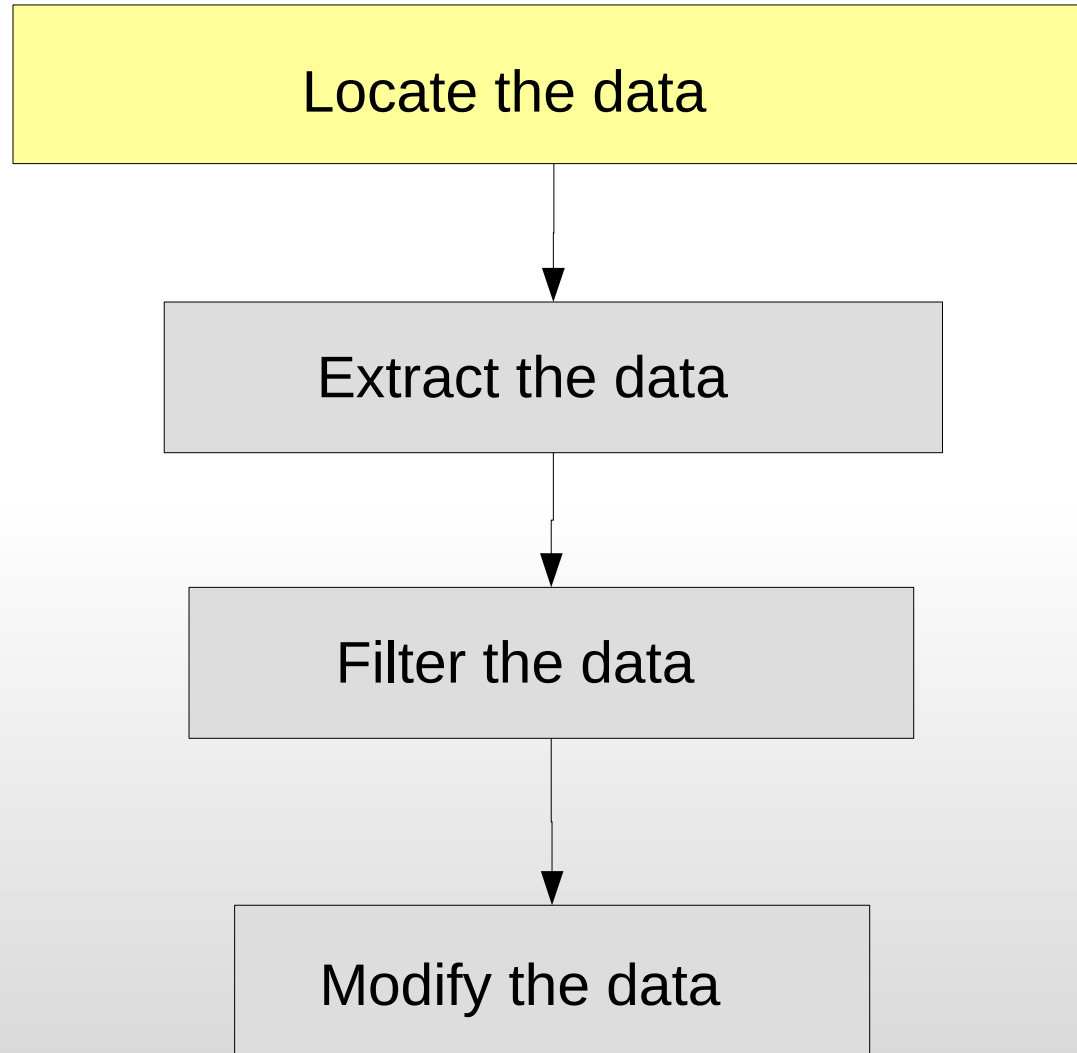
- Directory management:  
`cd`, `pwd`, `pushd`, `popd`
- Shell management:  
`history`, `enable`, `alias`, `export`
- Getting help:  
`help`, `type`
- Printing messages:  
`echo`, `printf`
- Job control:  
`fg`, `bg`, `jobs`, `kill`

# Job Control

- Job control refers to the ability to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point
- When a shell runs a program, it usually waits until the program terminates before returning control to the user (i.e., accepting another command)
  - If a command ends with a “&” it is instead executed in background
- Useful commands:
  - `jobs` – list the status of the jobs
  - `ps` – list all the system processes
  - `bg` – move a job to background
  - `fg` – move a job to foreground
  - `kill` – send a signal to a process or a job  
(9 is kill, check the manpage for the other ones)
  - `CTRL-z` – suspend the current job



# Different Tools for Different Jobs



# Finding Files

```
find path [options] [test] [action]
```

- It looks for files starting from the directory path and performs the action for each file that matches test

# Finding Files

```
find path [options] [test] [action]
```

- It looks for files starting from the directory path and performs the action for each file that matches test
- Options:
  - maxdepth n – descend no more than n directory levels

# Finding Files

```
find path [options] [test] [action]
```

- Tests specify what to look for:
  - Time values: `amin`, `atime`, `cmin`, `ctime`, `mmin`, `mtime`
  - File owner: `group`, `user`
  - File name: `name`, `iname`, `wholename`
  - Other file characteristics: `perm`, `size`, `type`
  - A `+` in front of a number means greater than, a `-` lesser than

# Finding Files

```
find path [options] [test] [action]
```

- Actions specify what to do when you find the files
  - By default, it prints the relative file path (`-print`)
  - Execute a command: **-exec**
    - `{}` is a placeholder for the name of the file
    - The command must be terminated by `\;` or by `+`
  - Print the file: `-print`, `-printf`, `-ls`, `-print0`

# Finding Files

```
find path [options] [test] [action]
```

- Combining together multiple expressions:
  - Two or more tests are implicitly in AND
  - Combining more expressions in OR:
    - `\( expr1 -o expr2 \)`
    - `expr` can be just a test or a sequence of a test and an action

# Examples

```
# Find files bigger than 10 megabytes
```

```
balzarot> find . -size +10M
```

```
# Find files modified in the last 30 minutes
```

```
balzarot> find . -mmin -30
```

```
# Find ps and pdf files
```

```
balzarot> find . \( -iname "*.pdf" -o -iname "*.ps" \) -print
```

```
# Find temporary files older than a week and print name and size
```

```
balzarot> find ~/tmp /tmp /var/tmp -mtime +7 -printf "%f %s\n"
```

```
# Change permissions of all the .h files
```

```
balzarot> find . -name "*.h" -exec chmod ug+r {} \;
```

```
# Remove all the tmp file in the home
```

```
balzarot> find ~ -name "*.tmp" -type f -print | xargs /bin/rm -f
```

```
# Better version (works also with filename containing spaces)
```

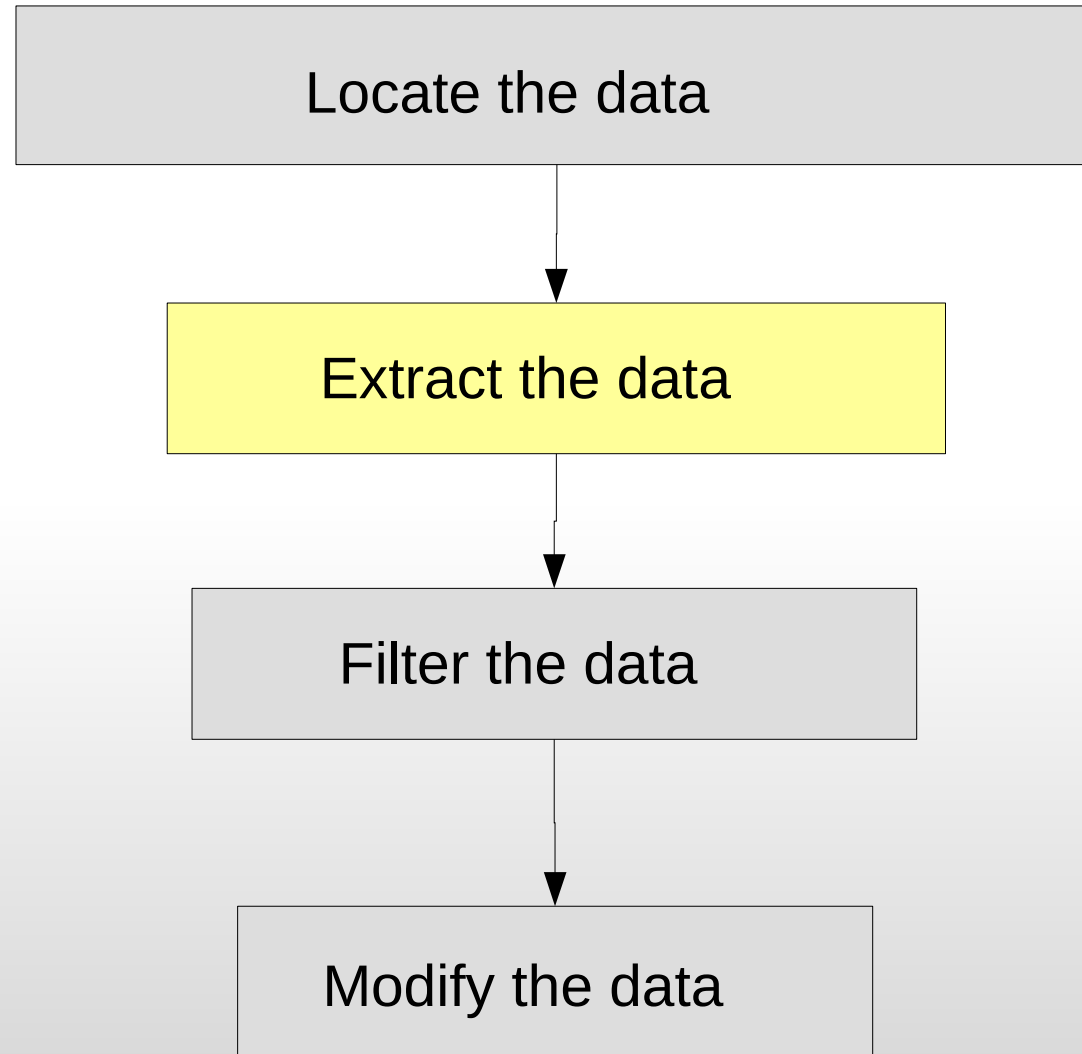
```
balzarot> find ~ -name "*.tmp" -type f -print0 | xargs -0 /bin/rm -f
```

# Locating Files

```
locate [options] pattern
```

- Locate search for files matching a given pattern in a system database
  - The database is created (and updated) by calling `updatedb`
  - By default, `locate` looks for “\*pattern\*”
- Options:
  - `e` check if the file still exists
  - `i` case insensitive
  - `0` separate entries with null bytes (for `xargs`)





# View the File Content

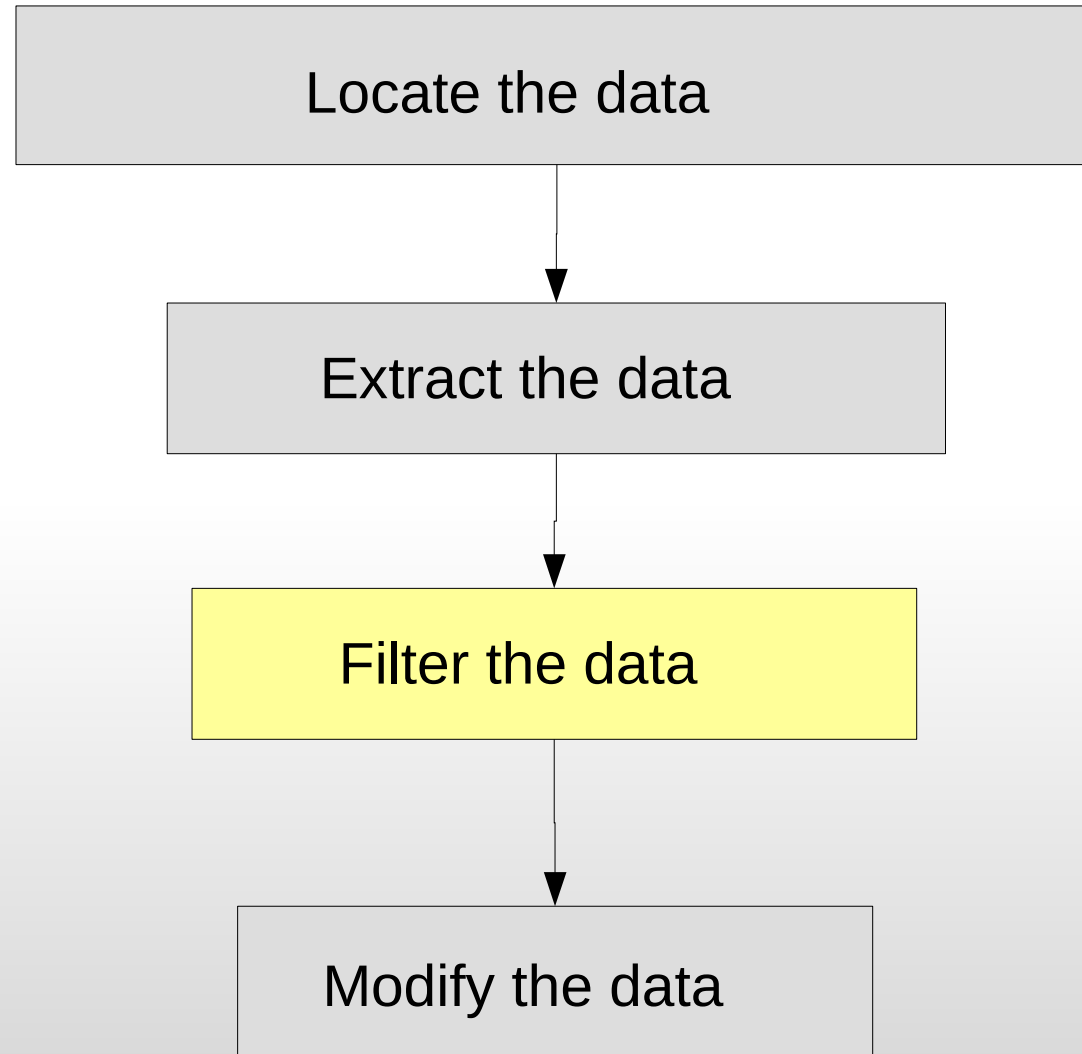
- `cat` – print (concatenate) one or more files
- If the output must be read by a human and it is too long, it can be run through a pager
  - `more` – shows the output one page at the time
  - `less` – shows the output allowing the user to freely more up and down
- Head or Tail
  - `head -n x` : shows the first x lines
  - `tail -n x` : shows the last x lines
  - `tail -f` : shows the last 10 lines and keep printing the new data as the file grows
- If the input is binary, `hexdump -C` will do the job

# Info about the File Content

- `wc` prints some useful statistics about a text stream
  - `wc -l` : print the number of lines
  - `wc -c` : print the number of bytes
  - `wc -w` : print the number of words
  - `wc -L` : print the length of the longest line
- `extract` extracts and prints the file metadata
  - The metadata depends on the file type
    - For mp3, it extracts the ID3 tags
    - For photos, it extracts the EXIF tags
    - For MS Word Documents, it extracts authors, last modification...
    - ....

# Working with Other File Formats

- Many tools exist to convert other file formats to plain text
  - `html2text` – convert html to text
  - `pdftotext` – convert pdf to text
  - `ps2ascii` – convert postscript to text
  - `antiword` – convert Microsoft Word documents to text
  - `unrtf` – convert rtf to text



# Search and Filter Text

```
grep [options] pattern [file list]
```

- GREP (Global Regular Expression and Print) allows to search (or filter) a file for known patterns
  - One of the most useful (and used) command-line tools
    - ... to find patterns in a file
    - ... to check which file contains a certain info
    - ... to filter out the lines you are not interested

# Search and Filter Text

```
grep [options] pattern [file list]
```

- By default, it prints the lines that match the pattern
  - Comes in three flavors
    - `grep` – the original
    - `egrep` – the extended version (matches extended regex)
    - `fgrep` – the fixed version: matches a list of strings (no regex)

# Searching for Text

- Options:
  - n : show the line number
  - i : ignore case
  - v : display the lines that do NOT match
  - H : show the file name too
  - -color : emphasize the match
  - x : try to match the entire line
  - B n ( -A n ) : prints also the n lines before (after) the match
  - o : prints only the matching string (not the full line)
  - l : prints only the name of the files that contain the pattern
- In order to use grep you need to understand regular expressions (regex)



# Regular Expressions 101

- Regular expressions are a powerful language to express patterns
- Characters matching
  - [ ]** any characters between the brackets
  - [^ ]** any character excluding the ones between the brackets
  - .** any character (except the end of line)
- Repetition operators
  - ?** zero or one time (extended regex only)
  - \*** zero or more times
  - +** one or more times (extended regex only)
  - {n}** exactly n times
  - {n, }** n or more times
  - {n1, n2}** between n1 and n2 times

# RegExp 101

- Anchors
  - ^ beginning of the line
  - \$ end of the line
- Grouping (extended regex only)
  - (**pattern**) group together pattern
  - \k back-reference to the k-th pattern
  - pattern1** | **pattern2** one or the other
- Note: there is no way to put two patterns in AND
  - Either you pipe two `grep` commands, or you merge the patterns in one

# Examples

```
# Which header file defines the function setDistance
```

```
balzarot> grep setDistance *.h
```

```
# Which lines contains Linux or Unix
```

```
balzarot> egrep -in "(Linux)|(Unix)" file.txt
```

```
# Working on some crosswords?
```

```
balzarot> grep -i "^c..a.r..$" /etc/dictionaries-common/words
```

```
# Dictionary's word without the apostrophe
```

```
balzarot> grep -v "'" /etc/dictionaries-common/words
```

```
# Four digit years
```

```
balzarot> egrep -o "[12][0-9]{3}"
```

```
# Match IP addresses
```

```
balzarot> egrep '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|  
[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|  
[01]?[0-9][0-9]?)' file.txt
```

# Sorting

- `sort` sorts the input alphabetically (and prints it to `stdout`)
  - **r** reverse order
  - **n** sort numerically
- `uniq` discards repeated consecutive lines
  - **c** prefix the lines with the number of occurrences
  - **d** print the duplicated lines, not the ones that appears only once
  - **u** only prints the unique lines

Green A  
Red  
Yellow  
Blue

Blue B  
White  
Red

---

sort A B

---

Blue  
Blue  
Green  
Red  
Red  
Yellow  
White



sort A B

sort A A B

Blue  
Blue  
Green  
Red  
Red  
Yellow  
White

Blue  
Blue  
Blue  
Green  
Green  
Red  
Red  
Red  
Red  
Yellow  
Yellow  
White

Green  
Red  
Yellow  
Blue

A

Blue  
White  
Red

B

sort A B

Blue

Blue

Green

Red

Red

Yellow

White

sort A A B

Blue

Blue

Blue

Green

Green

Red

Red

Red

Yellow

Yellow

White

sort A B B

Blue

Blue

Blue

Green

Red

Red

Red

Yellow

White

White

# Set Operations

- `sort` and `uniq` are simple tools, but they can be combined together to accomplish powerful tasks

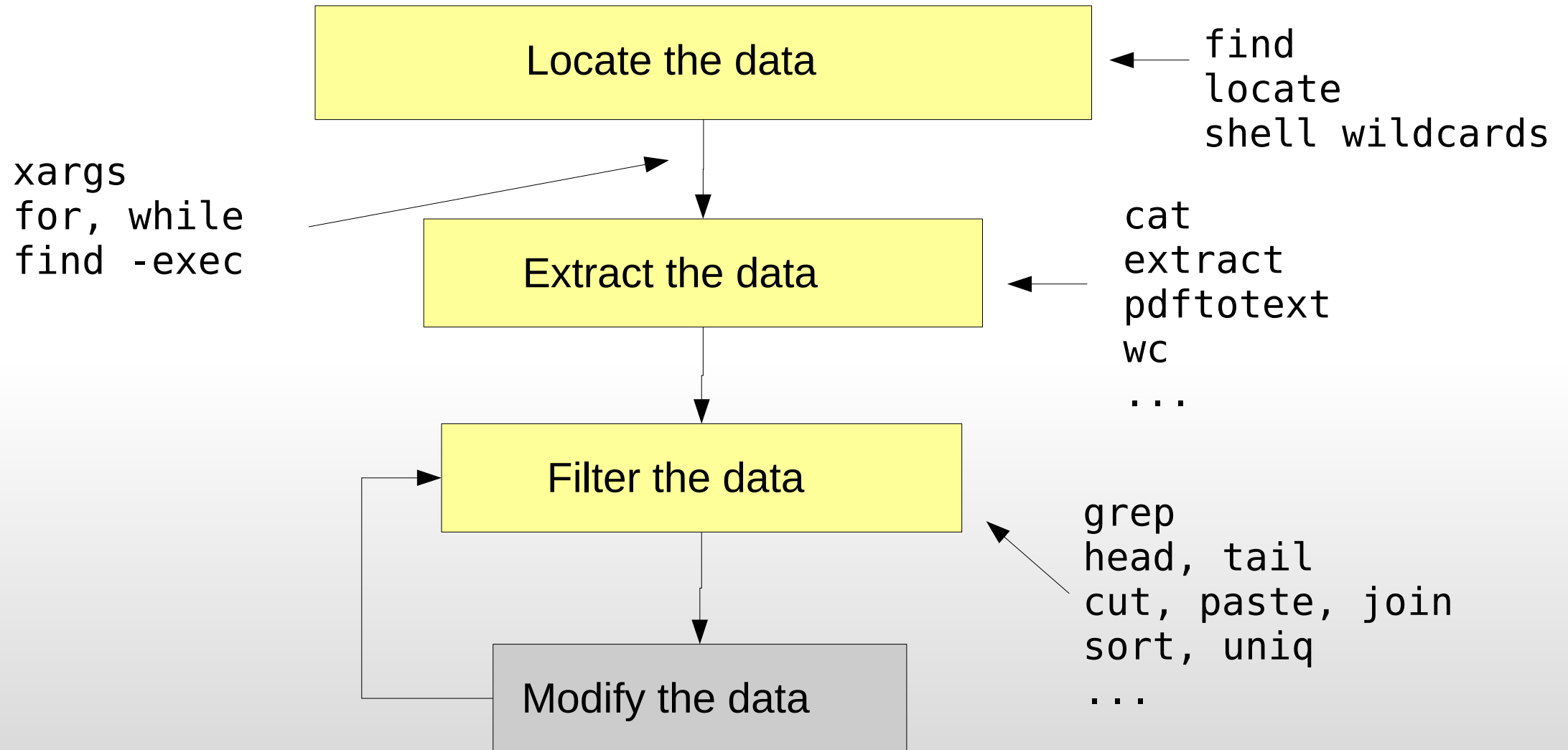
SetA and SetB (intersection)	<code>sort setA setB   uniq -d</code>
SetA or SetB (union)	<code>sort setA setB   uniq</code>
SetA but not in SetB (complement)	<code>sort setA setB setB   uniq -u</code>
SetB but not in SetA (complement)	<code>sort setA setA setB   uniq -u</code>
SetA or SetB but not in both (xor)	<code>sort set1 set2   uniq -u</code>



# Cut & Paste

- `cut` selects only certain columns of a text
  - `-d c` use `c` as field delimiter
  - `-f x` select only the field `x`
    - `-f x1-x2` select from field `x1` to `x2`
    - `-f x1-` select from field `x1` to the end of the line
  - `--complement` complement the selection
- `paste` add the content of one file as last column of another (put them side to side)
- `join` is like `paste`, but merge the files only when the first field is the same
  - `-1 x` for the first file, use field `x`
  - `-2 x` for the second file, use the field `x`

# Different Tools for Different Jobs



***Let's Practice....***