AWK programming

Christine Gaspin
Christophe Klopp
http://bioinfo.genotoul.fr/
http://www.sigenae.org/

Course organization

- 9:00 AM to 5:00 PM
- Lunch break 12:00 AM till 2:00 PM
- Five minutes breaks every hour 10:00 AM to 10:05 AM...
- BBB includes :
  - Chat public or private
  - Shared notes (location of the course notes and solutions)
  - Polls
- The idea is to adapt to all participants. Therefore feel free to ask question, to give other solutions, to interact.
- We will learn from your questions and our errors.

Do you have command line acces?

- Use your local machine or a distant server
  - Local :
    - *nix or mac open a terminal
    - MS-Windows open mobaxterm
  - Distant :

    https://mobaxterm.mobatek.net/
    - *nix or mac open a terminal and connect to server using ssh
    - MS-Windows open putty or mobaxterm and connect to server
- Paste the result of the date command in the shared notes followed by you name.

Self presentation

- Name
- Laboratory
- Research field or object
- What are you expecting from this day?

What you are going to learn about awk?

- How to use build in variables.
- How to create set and delete a variable.
- How to create and use loops.
- How to create and use tests.
- How to create and populate an array.

Why should you use AWK?

1) Because it enables to make reproducible file processing.
2) Because it is quite simple and can process large files.
3) Because it is installed on close to all Unix servers.
4) Because it exists in mobaXterm on MS-Windows.
5) Because it has a lot of useful functions.

Write an AWK script in vi

To create a new file or edit a existing :
vi myfile.awk **or vi and then in vi** "w myfile.awk"

Once in vi :
i to insert
A to append
[ESC] to stop to insert
d to delete
yy to copy and p to paste a line
[ESC] : nu to present the line numbers
[ESC] : 1,$s/before/after/
/mysearch to search in the file
[ESC] : wq to write and quit
[ESC] : q! to force to quit

invoking AWK

```
awk ‘$2 == 5’ myfile.txt
cat myfile.txt | awk ‘$2 == 5’
awk ‘$2 == 5’ < myfile.txt
```

If the file filter.awk contains ‘$2 == 5’ then

```
cat myfile.txt | awk -f filter.awk
awk -f filter.awk myfile.txt
awk -f filter.awk < myfile.txt
```

If your programme is longer than the line or has a complicated structure use an awk script file.

AWK command structure

```
awk 'awk command string' filename.txt
```

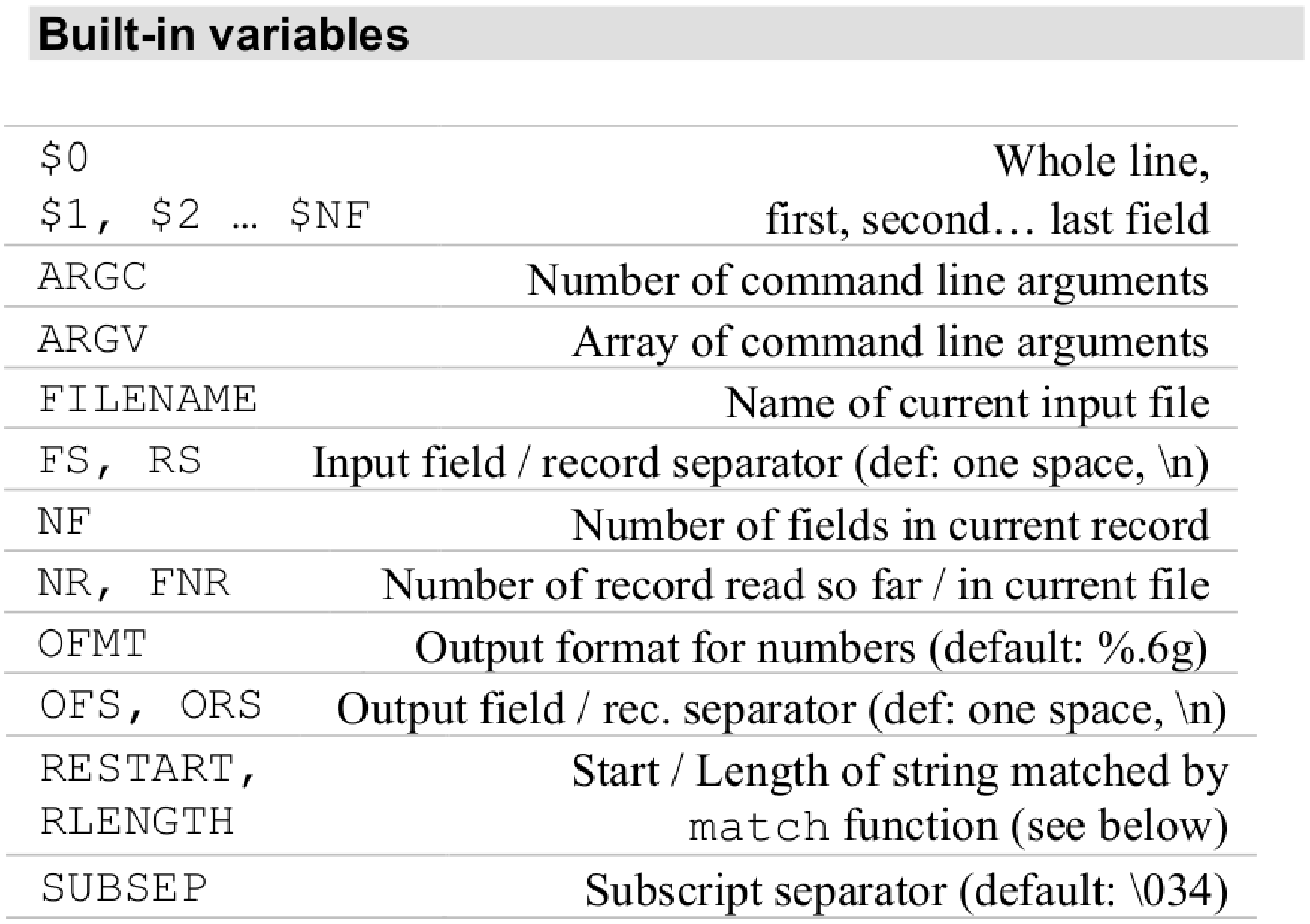
<u>awk command string structure :</u>

'**BEGIN**{statements} *FILTER*{statements}**END**{ statements}'

- **BEGIN** runs before reading the input file
- *FILTER* RE to select the processed lines
- **END** runs after reading the input file
- Statements are separated by « ; »

Use simple or double quotes at command ends.

AWK built-in variables



Built-in variables

`$0`
`$1, $2 … $NF`
	Whole line,
	first, second… last field

`ARGC`	Number of command line arguments
`ARGV`	Array of command line arguments
`FILENAME`	Name of current input file
`FS, RS`	Input field / record separator (def: one space, \n)
`NF`	Number of fields in current record
`NR, FNR`	Number of record read so far / in current file
`OFMT`	Output format for numbers (default: %.6g)
`OFS, ORS`	Output field / rec. separator (def: one space, \n)

`RESTART,`
`RLENGTH`
	Start / Length of string matched by
	`match` function (see below)

`SUBSEP`	Subscript separator (default: \034)

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Data files used for the exercises

- Fasta : sequence file
- GFF3 : gene transfer format
- VCF : variant call format
- Expression measurements : text file with counts

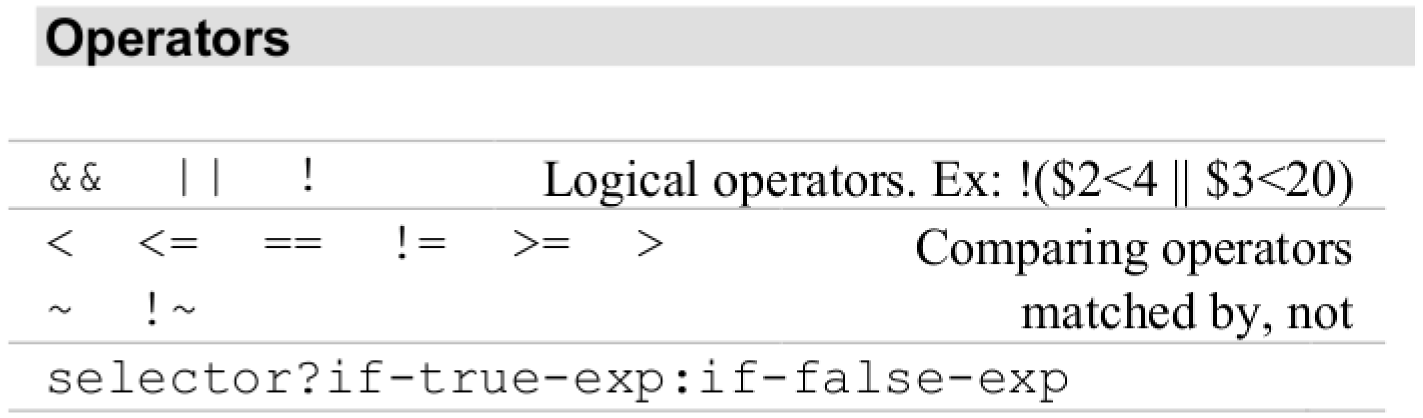
- Files have headers and content
- File can be structured in columns

- http://genoweb.toulouse.inra.fr/~klopp/SedAwk2021/Data/

Exercises : using built-in variables

- Extract header lines from the gtf file
- Extract the second column from the expression measurement file
- Print input file name
- Print number of blocks in each row of the expression measurement file

AWK operators



Operators

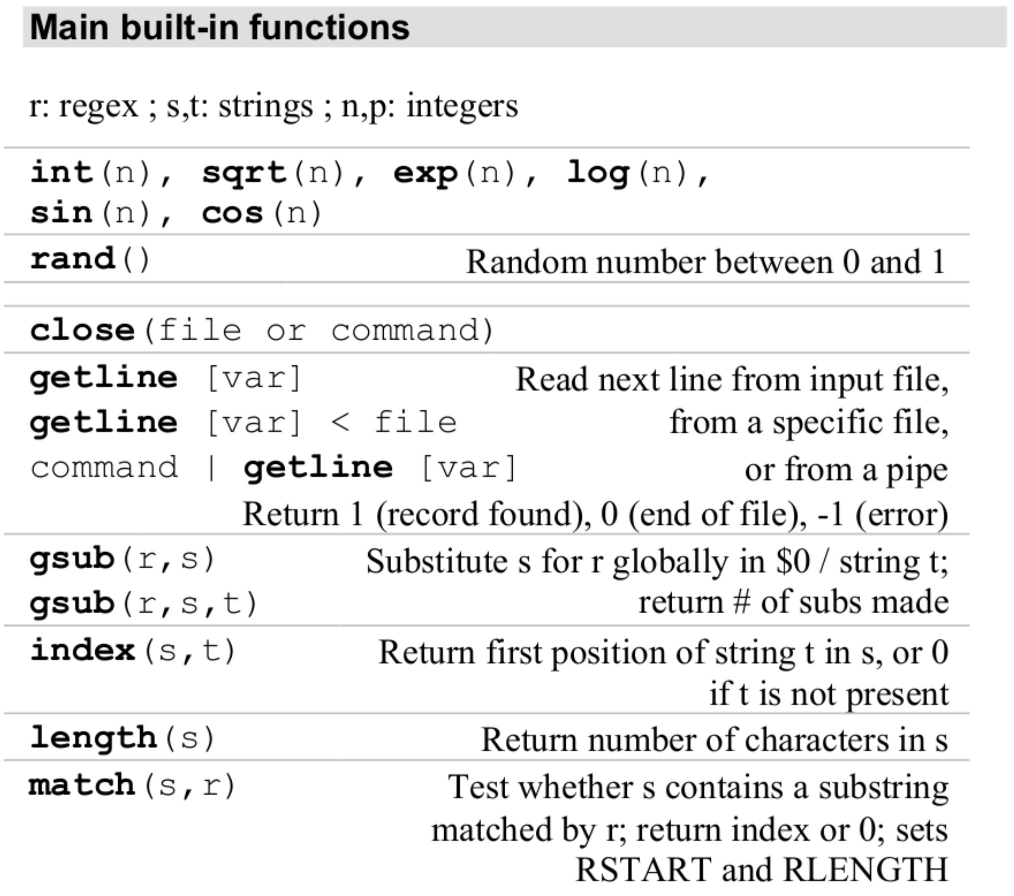
`&& || !`	Logical operators. Ex: !($2<4 || $3<20)
`< <= == != >= >`	Comparing operators
`~ !~`	matched by, not
`selector?if-true-exp:if-false-exp`	

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Exercises : using operators

- Extract lines from the expression measurement file where the third block value is between 5 and 7
- Extract lines from the expression measurement file where the fourth block value is like ‘5...’

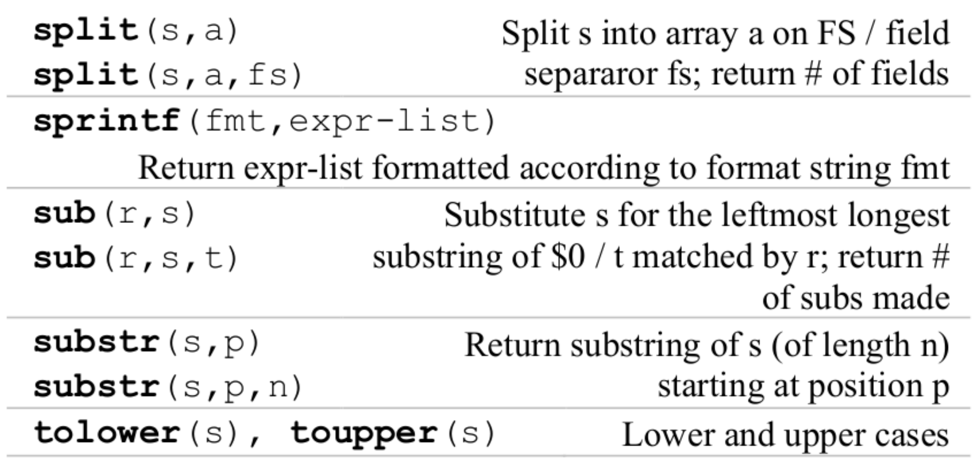
AWK functions



Main built-in functions

r: regex ; s,t: strings ; n,p: integers

int(n), **sqrt**(n), **exp**(n), **log**(n), **sin**(n), **cos**(n)	
rand()	Random number between 0 and 1
close(file or command)	
getline [var] **getline** [var] < file command | **getline** [var]	Read next line from input file, from a specific file, or from a pipe Return 1 (record found), 0 (end of file), -1 (error)
gsub(r,s) **gsub**(r,s,t)	Substitute s for r globally in $0 / string t; return # of subs made
index(s,t)	Return first position of string t in s, or 0 if t is not present
length(s)	Return number of characters in s
match(s,r)	Test whether s contains a substring matched by r; return index or 0; sets RSTART and RLENGTH



split(s,a) **split**(s,a,fs)	Split s into array a on FS / field separaror fs; return # of fields
sprintf(fmt,expr-list)	Return expr-list formatted according to format string fmt
sub(r,s) **sub**(r,s,t)	Substitute s for the leftmost longest substring of $0 / t matched by r; return # of subs made
substr(s,p) **substr**(s,p,n)	Return substring of s (of length n) starting at position p
tolower(s), **toupper**(s)	Lower and upper cases

`srand()` to set the random number generation

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Exercises : using functions

- Print a random value
- Print length of "ABCDEFGHIJ" string
- Print substring of "ABCDEFGHIJ" from position 3 to 7
- Split string "ABCDEFGHIJ" in characters and print second cell

AWK data structures : arrays and dictionaries

`Variable :` `a = 10; var = "var"; tt =1.36`

`Array :` `a[1]=1; a[2]=3; a[3]="eight"`

Quick way to produce an array : `split("123",a,"")`

`Hash table, dictionary :`

`h["one"]=1; h["two"]=3; h["three"]=”eight”`

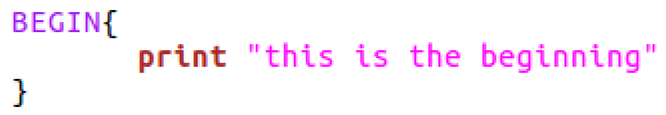
Delete function removes an element from an array or a hash table.
`Delete(a[1]); delete(h["two"])`

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Exercise 1 : follow up

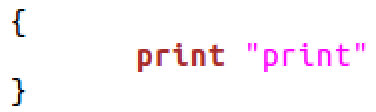
- What are these programs doing?

1)



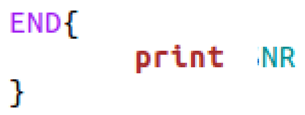
```
BEGIN{
        print "this is the beginning"
}
```

2)



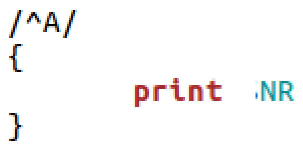
```
{
        print "print"
}
```

3)



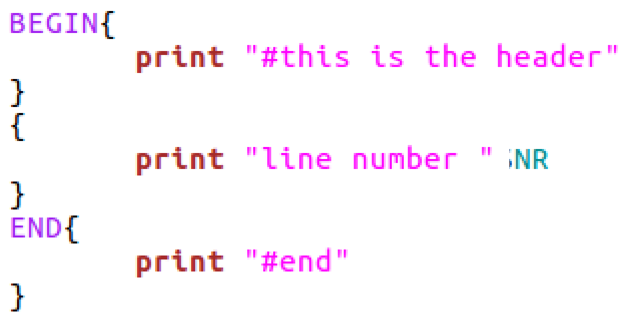
```
END{
        print NR
}
```

4)



```
/^A/
{
        print NR
}
```

5)



```
BEGIN{
        print "#this is the header"
}
{
        print "line number " NR
}
END{
        print "#end"
}
```

How does this program work?

- mel.awk file contains and uses a gff file as input

```
BEGIN{
    cnt = 0;
    totlen =0
}
$3 == "exon" {
    cnt = cnt + 1;          #cnt++;
    totlen = totlen + ($5-$4+1);  #totlen += ($5-$4+1);
}
END{
    print "cnt " cnt " ml " (totlen/cnt)
}
```

Exercises : using data structures

- Create variable a, attribute value 10 to a and print a
- Create array t, attribute value 10 to t[1] and value 24 to t[2] and print t[1] and t[2]
- Create dictionary d, attribute value 10 to d[“ten”] and value 24 to d[“twenty”] and print d[“ten”] and d[“twenty”]

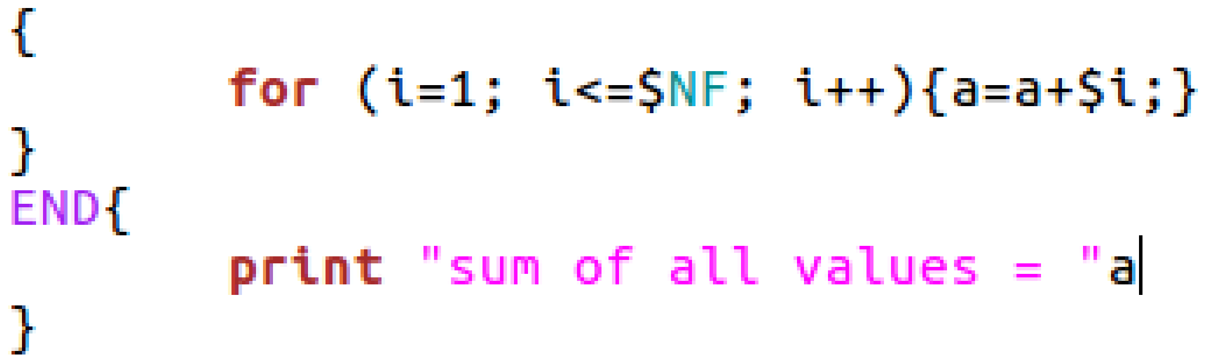
AWK control structure

if (expression) statement1 **else** statement2
while (expression) statement
for (expr1;expr2;expr3) statement
do statement **while** (expression)
break / **continue** : immediately leave / start next iteration of innermost enclosing loop
exit / **exit** expression : go immediately to the END action; if within the END action, exit program

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Loop exercise

- What is this program doing?



```
{
        for (i=1; i<=$NF; i++){a=a+$i;}
}
END{
        print "sum of all values = "a
}
```

awk parameters passing

- To pass an awk command or an awk script generic we can transmit paramerter.
- Two parameters
  - ARGC (argument count) and ARGV (argument array)
  - -v option

```
awk -f myscript.awk file ”A” ”B”
awk -f myscript.awk file A B

awk -v param1=”A” -v param2=”B” -f myscript.awk file
```

Exercises : using control structures

- Create a loop from 1 to 10 with step 1 and print the result
- Create a loop from 1 to 10 with step 1 and print the result only if the values are 4 or 8
- Create a loop from 1 to 10 with step 1 and print the result until 6 and leave the loop

What is programming?

- Starting from an idea or a question
- Write code which produces the expected result
- Before writing the code
  - Define your **inputs and outputs**
  - Define your **program structure** (loops, tests, functions...)
  - Decide what **data structure** you need (variable, array,…)
- Add tests to your inputs to make your code more **robust**
- Add documentation to make your code **maintainable** (document inputs and outputs, functions,...)

Exercise 2 : creating a sample file

- With AWK,
  - Create a file of 100 lines,
  - Each line contains one random number between 0 and 1
- Transform your program to print random characters instead of numbers
- Transform you program to print 10 random characters per line
- Transform you program to print 3 blocks of numeric values and 3 blocks of characters per line

Creating novel function in awk

- Awk has built-in functions but you can extend these by creating your own functions.
- Functions are used to replace parts of codes which are repeated in the program or to make the program more readable.
- Functions can be grouped in a function file which is loaded before using the functions. Loading is performed with @include "functions.awk".
- To create a function you use the ‘function’ function and define inputs and outputs.

```
function print_concatenate(str1, str2){
    print str1“_“str2
}

Function square(val){
    return(val * val)
}
```

Commenting an awk program

- To make you programs more readable for others or yourself you should add comments.
- Comment are lines which do not change the program behavior.
- Comment are lines starting with a hashtag ‘#’

```
# print_concatenate concatenates two strings with “_”
function print_concatenate(str1, str2){
    print str1“_“str2
}
```

Running awk scripts

- You can run an awk scripts without invoking “awk -f” .
- For this you have to add a **shebang** at the first line of you script.
- A **shebang** is a line starting with #! indicating the location of the awk interpreter. For example `/usr/bin/awk`.
  You can get it using the `which awk` command.
- Then you render your script executable with `chmod +x myscript.awk`

```
#!/usr/bin/awk -f

{
    print NR“\t“$0
}
```

Exercise 3 : follow up

- Change your program in order to run if from the command line without calling “awk -f myprogram.awk” but “myprogram.awk”
- Create a function in your program to build the character strings. Function input = number of character, function return = string
- Move you function to a function file and include this function file in a new program which only runs the main part of the code.
- Add comments to the function and the program to help maintenance

Exercise 4 : processing the file

- With AWK,
  - For each column of the file : to_be_processed.txt
    http://genoweb.toulouse.inra.fr/~klopp/awk/to_be_processed.txt
  - Count the number of columns per line.

- For each numerical column find minimum and maximum values.

- Count the number of distinct elements per column.

- Count the number of occurrences of each element in the fourth column for all lines of the file.

Exercise 5 : join files

- With AWK
  - Join both files :
    http://genoweb.toulouse.inra.fr/~klopp/awk/to_be_joined1.txt
    http://genoweb.toulouse.inra.fr/~klopp/awk/to_be_joined2.txt

    Using the first column