Exercises

1. Use `find` to display the names of all files in the `/usr/common/classes/beerli` subdirectory tree.What do the errors mean (if any)? what to do about the flood of lines?
2. Use `find` to display the names of all files in the same directory that are bigger than 1MB. Hint: use man find (the option -size)
3. Create a list of users presently logged in in the system: `finger > users.txt`
   Use `grep` to isolate the line in `users.txt` that contains your login details.
4. Use `grep` and `sort` to display a sorted list of all files in the `/usr/common/classes/beerli` subdirectory tree that contain the word `"Romanshorn"` somewhere inside them, you may need to recurse through all directories, how?
5. Use `locate` to find all filenames that contain the word `emacs`. Can you combine this with `grep` to avoid displaying all filenames containing the word `share`?
6. On Linux systems, the file `/dev/urandom` is a constantly generated random stream of characters. Can you use this file with `od` to printout a random decimal number?

UNIX files searching, and other interrogation techniques

To use the example do this: `cp /usr/common/classes/beerli/animals.txt .`
The pipe ('|') operator is used to create concurrently executing processes that pass data directly to one another. It is useful for combining system utilities to perform more complex functions. For example:

```
$ cat animals.txt | sort| uniq
```

creates three processes (corresponding to `cat`, `sort` and `uniq`) which execute concurrently. As they execute, the output of the who process is passed on to the `sort` process which is in turn passed on to the `uniq` process. `uniq` displays its output on the screen (a sorted list of users with duplicate lines removed). Similarly:

```
$ cat animals.txt | grep "dog" | grep -v "fish"
```

finds all lines in `animals.txt` that contain the string "`dog`" but do not contain the string "`fish`".

Piping into files

The output from programs is usually written to the screen, while their input usually comes from the keyboard (if no file arguments are given). In technical terms, we say that processes usually write to **standard output** (the screen) and take their input from **standard input** (the keyboard). There is in fact another output channel called **standard error**, where processes write their error messages; by default error messages are also sent to the screen.
To redirect standard output to a file instead of the screen, we use the > operator:

```
$ echo hello
hello
$ echo hello > output
$ cat output
hello
```

In this case, the contents of the file `output` will be destroyed if the file already exists. If instead we want to append the output of the echo command to the file, we can use the >> operator:

```
$ echo bye >> output
$ cat output
hello
bye
```

stdin, stdout, stderr

To capture standard error, prefix the > operator with a 2 (in UNIX the file numbers 0, 1 and 2 are assigned to standard input, standard output and standard error respectively), e.g.:

```
$ cat nonexistent 2>errors
$ cat errors
cat: nonexistent: No such file or directory
$
```

You can redirect standard error and standard output to two different files:

```
$ find . -print 1>errors 2>files
```

or to the same file:

```
$ find . -print 1>output 2>output
```

or

```
$ find . -print >& output
```

Standard input can also be redirected using the < operator, so that input is read from a file instead of the keyboard:

```
$ cat < output
hello
bye
```

stdin, stdout, stderr

You can combine input redirection with output redirection, but be careful not to use the same filename in both places. For example:

```
$ cat < output > output
```

will destroy the contents of the file `output`. This is because the first thing the shell does when it sees the > operator is to create an empty file ready for the output.

use always something like this if you want to replace, because with an error in to above you destroy your data

```
$ cat < output > output2
```

```
$ mv output2 output
```

more fun with extracting

2. Use `grep` and `sort` to display a sorted list of all files in the `/usr/common/classes/beerli` subdirectory tree that contain the word `"Romanshorn"` somewhere inside them, you may need to recurse through all directories, how?
3. How many "species" are in the animals.txt file?
4. How many fish? [aproximately]
5. Looking at the file animals.txt (use more or less) you see that the second word contains names for how babies of that species are called, give a list of all names that are used to call babies
    1. the command awk can extract words among other things. We use it to extract words with this awk '{ print $2 }'
    2. now combine this with grep and sort and uniq and you can create the list
6. You want to read the list into excel or other program and need to use the tab character.
    1. Another useful command is tr , it changes characters,
    2. a common usage: tr -s ' ' '\t' < in > out