ISC-3133 Introduction to Scientific Computing using C++

Peter Beerli

Professor

Department of Scientific Computing

Office: 150-T Dirac Science Library

Phone: (850) 559 9664

Email: beerli at fsu dot edu

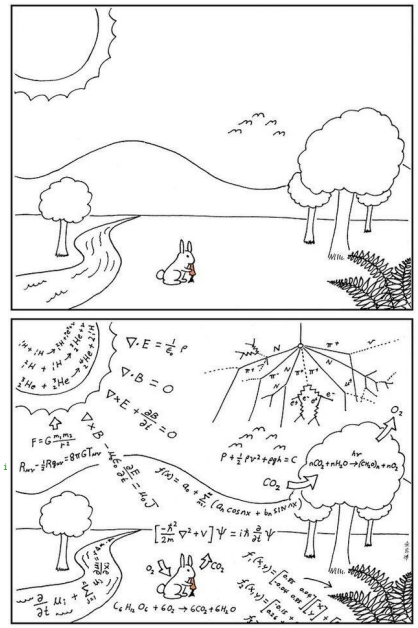
Web: http://people.sc.fsu.edu/~pbeerli


```
/* push dates to agree with a most recent sample date at endTime and oldest sample date i
/* will fail if used on contempory samples */
void CoalescentTree::pushTimesBack(double startTime, double endTime) {

    tree<Node>::iterator it, jt;
    double presentTime = getPresentTime();

    if (startTime < endTime) {

    // STRETCH OR SHRINK //////////////
```



This is how scientists see the world.

Course Description:

This course introduces you to the science of computations. Algorithms for standard problems in computational science are presented. The basics of the object-oriented programming language C++ are taught to facilitate the implementation of algorithms.

Course Objectives:

- Identify the components of scientific computing;
- Identify standard problems in scientific computing;
- Implement basic algorithms for standard problems in computational science using the programming language Java.
- Write, debug, and verify computer codes;
- Output results of computer simulations on a meaningful manner.

Grading Policy:

The student's grade for the course will be based upon classwork, homework, and a final capstone project. This work is weighted as follows:

- Classwork/Quizzes (weekly) - 10%
- Assignments (weekly to biweekly: description, code) - 50%
- Capstone Project (project description, code, presentation) - 40%

Contents

We have a total of 17 weeks of instruction, we will spend about two weeks per topic.

Computational science [or Scientific Computing]

From Wikipedia, the free encyclopedia

Not to be confused with computer science.

Computational science (or **scientific computing**) is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyse and solve scientific problems. In practical use, it is typically the application of computer simulation and other forms of computation to problems in various scientific disciplines.

The field is distinct from computer science (the study of computation, computers and information processing). It is also different from theory and experiment which are the traditional forms of science and engineering. The scientific computing approach is to gain understanding, mainly through the analysis of mathematical models implemented on computers.

Scientists and engineers develop computer programs, application software, that model systems being studied and run these programs with various sets of input parameters. Typically, these models require massive amounts of calculations (usually floating-point) and are often executed on supercomputers or distributed computing platforms.

Numerical analysis is an important underpinning for techniques used in computational science.

Computer Science

Computer science or **computing science** (sometimes abbreviated **CS**) is the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.[1][2][3][4] It is frequently described as the systematic study of algorithmic processes that create, describe, and transform information.

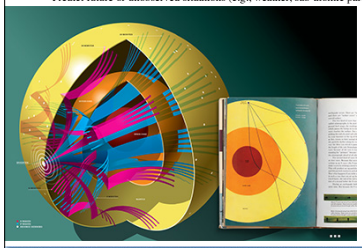
Applications of computational science

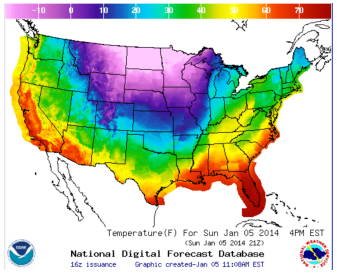
Problem domains for computational science/scientific computing include:

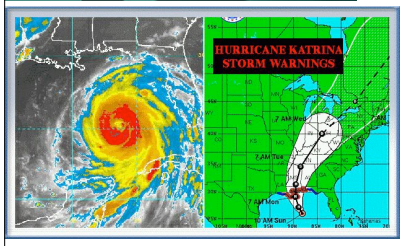
Numerical simulations

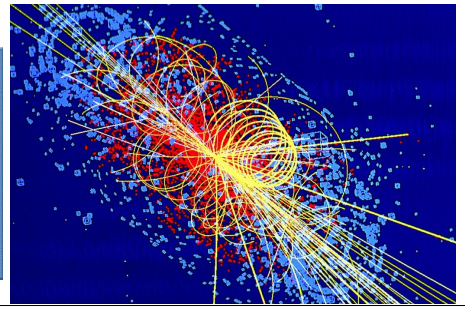
Numerical simulations have different objectives depending on the nature of the task being simulated:

- Reconstruct and understand known events (e.g., earthquake, tsunamis and other natural disasters).
- Predict future or unobserved situations (e.g., weather, sub-atomic particle behavior).









Model fitting and data analysis

- Appropriately tune models or solve equations to reflect observations, subject to model constraints (e.g. oil exploration geophysics, computational linguistics)
- Use graph theory to model networks, especially those connecting individuals, organizations, and websites.

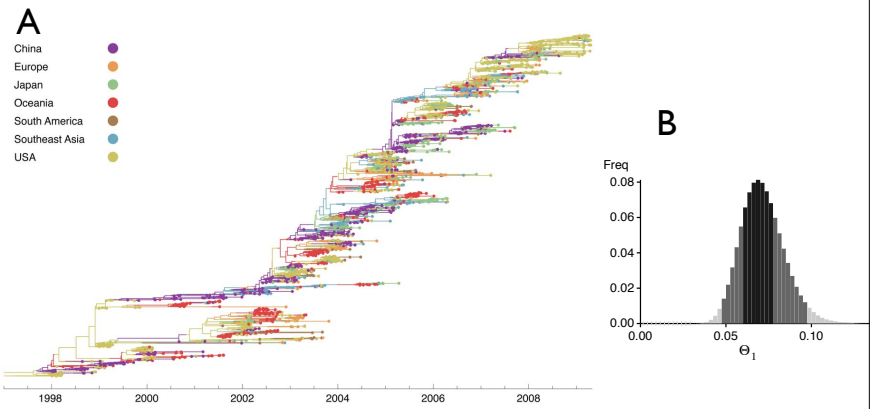
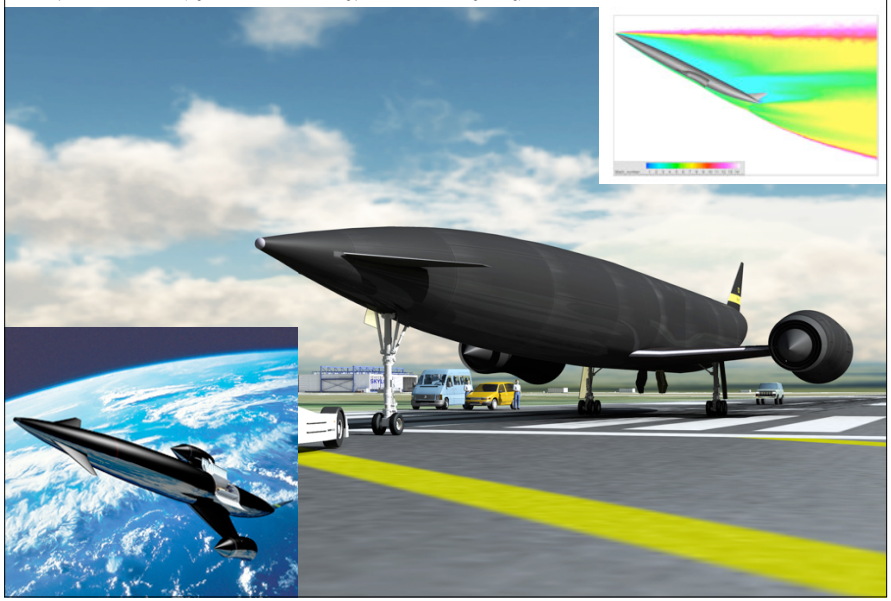


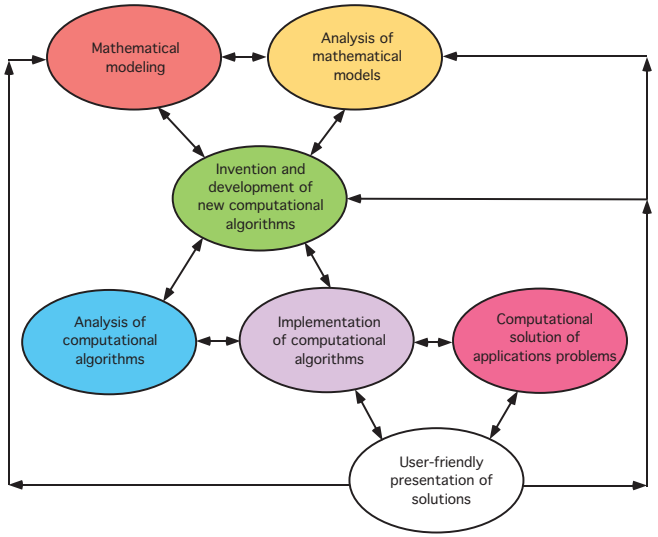
Figure 2. Genealogy of 2165 influenza A (H3N2) viruses sampled from 1998 to 2009. Each point represents a sampled virus sequence, and the color of the point shows the location where it was sampled. Samples are explicitly dated on the x-axis. Tracing a vertical line gives a contemporaneous cross-section of virus isolates. The genealogy is sorted so that lineages that leave more descendants are placed higher on the y-axis than other, less successful lineages. This sorting places the trunk along a rough diagonal, and it places lineages that are more genetically similar to the trunk higher on the y-axis than lineages that are farther away from the trunk. The tree shown is the highest posterior tree generated by the Markov chain Monte Carlo (MCMC) procedure implemented in the software program Migrate v3.0.8 [14,20].
doi:10.1371/journal.ppat.1000918.g002

Computational optimization

Main article: Mathematical optimization

- Optimize known scenarios (e.g., technical and manufacturing processes, front-end engineering).





The universe of scientific computing/computational science

in this course, ...

- we are mainly interested in implementing computational algorithms.
- we will use C++ to implement these algorithms
- we will learn the basics of C++ in the context of basic methods in scientific computing such as
  - ★ approximate integrals: $\int_a^b f(x)dx$
  - ★ solving a single nonlinear equation, e.g. find x such that $x = sin\ x$
  - ★ interpolating or fitting data, e.g. find a line $y = mx + b$
  - ★ vector and matrix operations $A\vec{x} = \vec{y}$
  - ★ solving simple differential equation, numerically $\frac{dy}{dt} = e^{-gt}, y(0) = y_0$
- We will visualize some of the results with GNUPLOT or other visualization methods.

HISTORY

Bjarne Stroustrup, creator of C++

Bjarne Stroustrup, a Danish and British trained computer scientist, began his work on "C with Classes" in 1979.[4] The idea of creating a new language originated from Stroustrup's experience in programming for his Ph.D. thesis. Stroustrup found that Simula had features that were very helpful for large software development, but the language was too slow for practical use, while BCPL was fast but too low-level to be suitable for large software development. When Stroustrup started working in AT&T Bell Labs, he had the problem of analyzing the UNIX kernel with respect to distributed computing. Remembering his Ph.D. experience, Stroustrup set out to enhance the C language with Simula-like features.[9] C was chosen because it was general-purpose, fast, portable and widely used. Besides C and Simula, some other languages that inspired him were ALGOL 68, Ada, CLU and ML. At first, the class, derived class, strong typing, inlining, and default argument features were added to C via Stroustrup's "C with Classes" to C compiler, Cpre.[10]

In 1983, the name of the language was changed from *C with Classes* to C++ (++ being the increment operator in C). New features were added including virtual functions, function name and operator overloading, references, constants, user-controlled free-store memory control, improved type checking, and BCPL style single-line comments with two forward slashes (//), as well as the development of a proper compiler for C++, Cfront. In 1985, the first edition of *The C++ Programming Language* was released, providing an important reference to the language, as there was not yet an official standard.[11] The first commercial implementation of C++ was released in October of the same year.[12]

Release 2.0 of C++ came in 1989 and the updated second edition of *The C++ Programming Language* was released in 1991.[13] New features included multiple inheritance, abstract classes, static member functions, const member functions, and protected members. In 1990, *The Annotated C++ Reference Manual* was published. This work became the basis for the future standard. Late feature additions included templates, exceptions, namespaces, new casts, and a Boolean type.

As the C++ language evolved, the standard library evolved with it. The first addition to the C++ standard library was the stream I/O library which provided facilities to replace the traditional C functions such as printf and scanf. Later, among the most significant additions to the standard library, was a large amount of the Standard Template Library.

It is possible to write object oriented or procedural code in the same program in C++. This has caused some concern that some C++ programmers are still writing procedural code, but are under the impression that it is object oriented, simply because they are using C++. Often it is an amalgamation of the two. This usually causes most problems when the code is revisited or the task is taken over by another coder.[14]

C++ continues to be used and is one of the preferred programming languages to develop professional applications.[15]

C++

Java/C#

Ruby

C

Unix

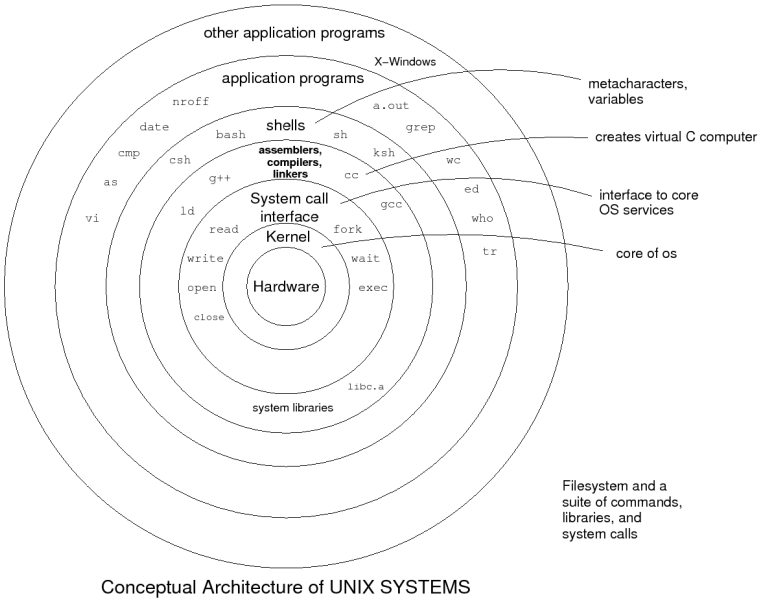
Operating system
Multitasking
Multi-user
1969
AT&T
Running on PDP-7

http://www.faqs.org/docs/artu/ch02s01.html

AIX IBM

ORACLE SOLARIS

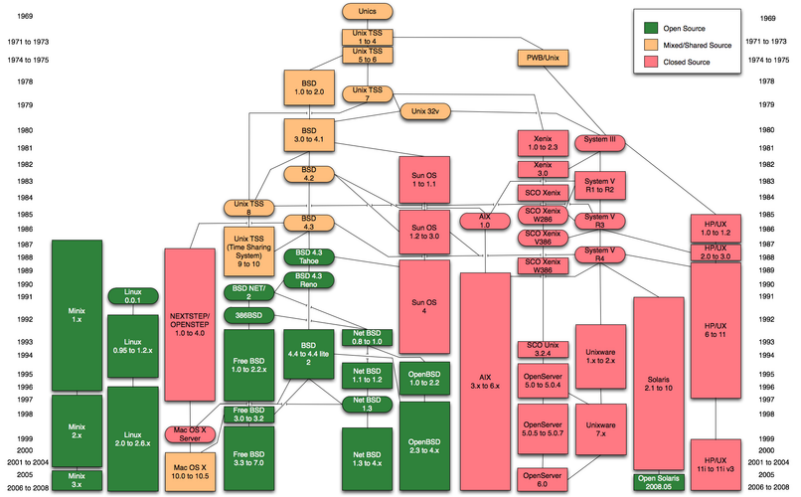
HP-UX 11i v3



Conceptual Architecture of UNIX SYSTEMS

http://academic.udayton.edu/SaverioPerugini/courses/cps346/lecture_notes/UNIXphilo.html

Evolution of Unix and Unix-like OS



From Wikipedia

Linux

[Unix-like free Operating system]
1991 ~
Linus Tovalds & many others
[Linux distribution]
- Unix-like operating system based on "*Linux kernel*"

[Used by all of world top 10 Supercomputers](June, 2011)

ubuntu

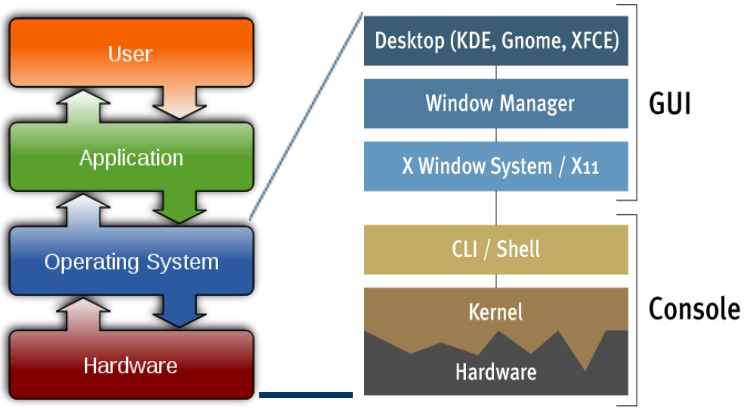
debian

redhat.

CentOS

openSUSE

Operating System



Remote Access

	Windows	Linux/Mac
Login	putty	ssh
Copy files	winscp	scp
Check availability	ping	ping

Free download from

putty
http://www.chiark.greenend.org.uk/~sgtatham/putty/
*Configuration http://sc.tamu.edu/help/access/windows.php

winscp
http://winscp.net/

A UNIX command line consists of the name of a UNIX command (actually the "command" is the name of a built-in shell command, a system utility or an application program) followed by its "arguments" (options and the target filenames and/or expressions). The general syntax for a UNIX command is

```
$ command -options targets
```

Here `command` can be though of as a verb, `options` as an adverb and `targets` as the direct objects of the verb. In the case that the user wishes to specify several options, these need not always be listed separately (the options can sometimes be listed altogether after a single dash).


```
nagal:gugus>echo "Basic UNIX commands"
Basic UNIX commands
nagal:gugus>ls
nagal:gugus>ls -la
total 0
drwxr-xr-x+   2 beerli  staff    68 Jan  7 09:51 .
drwxr-xr-x+ 128 beerli  staff  4352 Jan  7 09:51 ..
nagal:gugus>ls .
nagal:gugus>ls ..
#nescent-report#     dada
%backup%~         dreamhost_conversation
Adobe Illustrator CS6.dmg drunken_sailor.sce
Applications         echo
Data           galaxy-python
Desktop           game.sce
anim.py           montyhall.sce
anim2.py       nescent-report
bear281.log       nescent-report~
…
nagal:gugus>nano hello.txt
nagal:gugus>ls
hello.txt
nagal:gugus>cat hello.txt
The quick fox jumps over the lay dog.
The difference between stupidity and genius is that genius has its limits.
Dogs have owners, cats have staff.

nagal:gugus>cp hello.txt byebye.txt
nagal:gugus>ls
byebye.txt  hello.txt
nagal:gugus>diff byebye.txt hello.txt
nagal:gugus>rm byebye.txt
nagal:gugus>cp hello.txt byebye.txt
nagal:gugus>emacs byebye.txt
nagal:gugus>diff byebye.txt hello.txt
1c1
< A quick fox jumps over the lay dog.
---
> The quick fox jumps over the lay dog.
nagal:gugus>
```

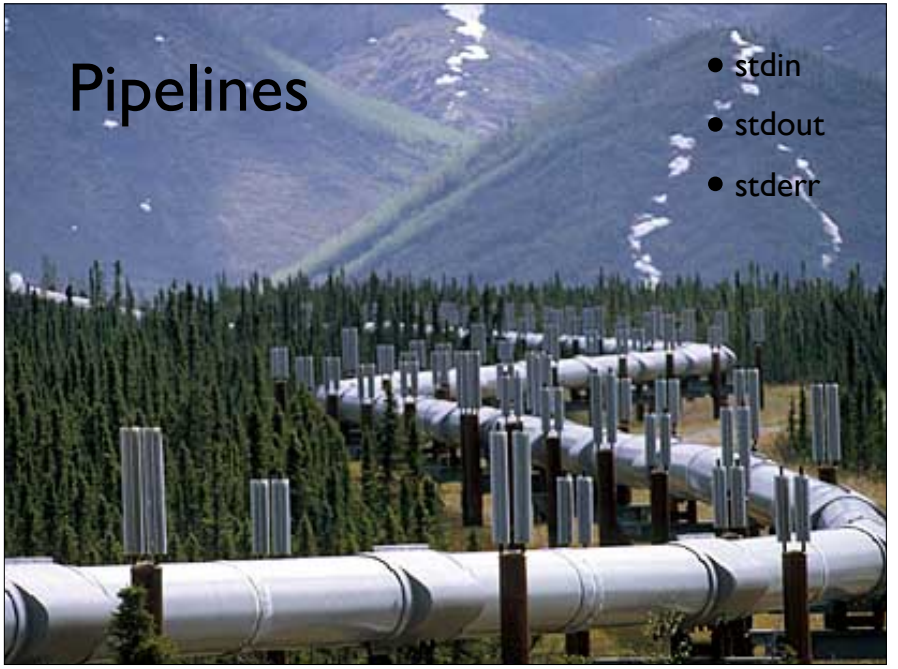
1. Log on a Linux machine or connect to one from a Windows machine (e.g. click on the Exceed icon and then use putty to connect to the server `kiwi`. Enter your login (user name) and password at relevant prompts.
2. Enter these commands at the UNIX prompt, and try to interpret the output. Ask questions and don't be afraid to experiment (as a normal user you cannot do much harm):

- `echo hello world`
- `passwd`
- `date`
- `hostname`
- `arch`
- `uname -a`
- `dmesg | more`
  (you may need to press `q` to quit)
- `uptime`
- `who am i`
- `who`
- `id`
- `last`
- `finger`
- `w`
- `top` (you may need to press `q` to quit)
- `echo $SHELL`
- `echo {con,pre}{sent,fer}{s,ed}`
- `man "automatic door"`
- `man ls` (you may need to press `q` to quit)
- `man who` (you may need to press `q` to quit)
- `who can tell me why i got divorced`
- `lost`
- `clear`
- `cal 2000`
- `cal 9 1752` (notice anything unusual?)
- `bc -l` (type `quit` or press Ctrl-d to quit)
- `echo 5+4 | bc -l`
- `yes please`
  (you may need to press Ctrl-c to quit)
- `time sleep 5`
- `history`

UNIX editors

http://en.wikipedia.org/wiki/Comparison_of_text_editors

Most common basic UNIX editors




```
cat hello.txt | sort | uniq
```

```
cat hello.txt | grep "dog" | grep -v "cat"
```

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
```

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
```

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.

UNIX shell cheat sheet

The shell allows maintenance tasks, such creating, copying, moving, renaming,... of files and directories/ Among many other things, it also allows to search for files and contents of files.

Focus Directory	`cd`	change directory to the users home directory
	`cd $HOME`	change directory to the users home directory
	`cd ..`	change directory to the directory that is outside of the current one
	`mkdir directory1`	Create directory `directory1`
Manipulating	`mv file1 file2`	Rename file1 to file2, works also with directories
	`cp file1 file2`	Copy file1 to file2
	`cat file1 > file2`	Copy file1 to file2 using pipelining
	`cat file1`	Show file1
	`less file1`	Show file1 with paging, leave this mode using `q`, Top is `g`, Bottom is `G`, paging is <spacebar>
	`nano file1`	Open text file editor (if all key-presses fail try Cntrl-G, or Cntrl-C)
finding	`find . -name file1`	find a file name starting in the current directory and all subdirectories
	`find dir1 -name file1`	find a file name starting in the directory dir1
	`find . -name '*fi*'`	find a file name containing the letters `fi` starting in the current directory
	`find . -name 'fi*'`	find a file name beginning with the letters `fi` starting in the current directory
	`grep "is this" file1`	find all lines in file1 that contain the text "is this"
	`grep "^is this" file1`	find all lines in file1 that begin with "is this"
	`grep "[tT]his" file1`	find all lines in file1 that contain the text "this" or "This"
Changing text	`tr -s '\r\n' '\n' < file1 > file2`: Change all windows end-of-line characters to UNIX end-of-line characters but piping file1 to file2	