UNIX Tutorials

Typographical Conventions

Introduction to The UNIX operating system

Tutorial One

Tutorial Two

Assignment 1

Tutorial Three

Tutorial Four

Assignment 2

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

UNIX Tutorial

Typographical conventions

In what follows, we shall use the following typographical conventions:

- Characters written in `bold typewriter font` are commands to be typed into the computer as they stand.
- Characters written in *italic typewriter font* indicate non-specific file or directory names.
- Words inserted within square brackets `[Ctrl]` indicate keys to be pressed.
- Words inserted within parentheses `[additional information]` provide additional information on the command.

So, for example,

$ ls *anydirectory* [Enter]

means "at the UNIX prompt $, type ls followed by the name of some directory, then press the key marked Enter"

Don't forget to press the [Enter] key: commands are not sent to the computer until this is done.

Note: UNIX is case-sensitve, so LS is not the same as ls.

The same applies to filenames, so myfile.txt, MyFile.txt and MYFILE.TXT are three seperate files.

M.Stonebank@surrey.ac.uk, © 1 November, 2002

UNIX Introduction

This session concerns UNIX, which is a common operating system. By operating system, we mean the suite of programs which make the computer work.

There is a graphical interface between the user and UNIX. However, knowledge of UNIX and its command-line interface (shell) is required for operations which aren't covered by a graphical program, or for when there is no graphical interface, for example, in a telnet session.

Also, an interesting aspect of UNIX for a Computational Linguist is that it has many text processing utilities. It was originally designed by people familiar with syntax.

The UNIX operating system

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the filestore and communications in response to system calls.

As an illustration of the way that the shell and the kernel work together, suppose a user types `rm myfile` (which has the effect of removing the file **myfile**). The shell searches the filestore for the file containing the program `rm`, and then requests the kernel, through system calls, to execute the program `rm` on myfile. When the process `rm myfile` has finished running, the shell then returns the UNIX prompt `$` to the user, indicating that it is waiting for further commands.

The shell

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (`$` on our systems).

The shell has certain features to help the user inputting commands.

Filename Completion - By typing part of the name of a command, filename or directory and pressing the `[Tab]` key, the shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the arrow up and down keys to scroll up and down the list or type `history` for a list of previous commands.

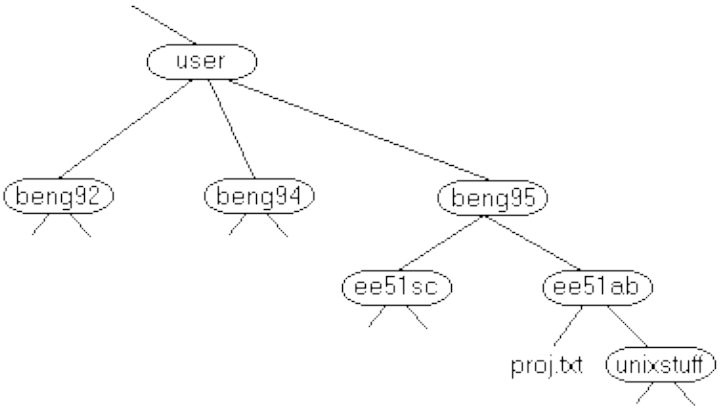
Files and processes (programs)

Everything in UNIX is either a file or a process. A process is an executing program identified by a unique PID (process identifier). A file is a collection of data. They are created by users using text editors, running compilers etc. Examples of files:

- a document (report, essay etc.);
- the text of a program written in some high-level programming language;
- instructions comprehensible directly to the machine and incomprehensible to a casual user;
- a directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

The Directory Structure

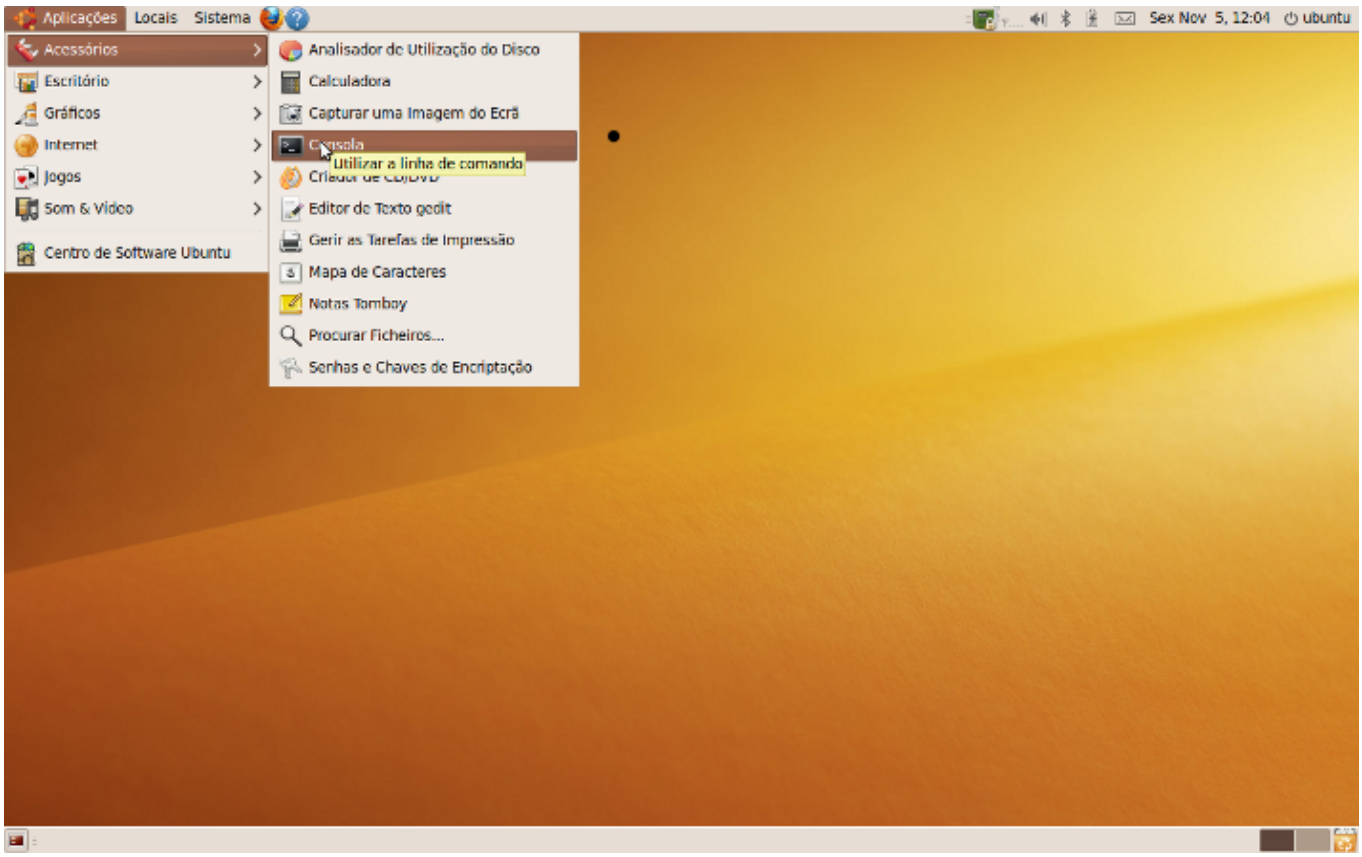
All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root**.



In the diagram above, we see that the directory `ee51ab` contains the subdirectory `unixstuff` and a file `proj.txt`

Starting a terminal session

Insert the Ubuntu disk. (Re)-start your computer and make sure it can boot from the disk.
Select language = Português
Select '`Experimentar Ubuntu sem fazer nenhuma alteração ne seu computador`'.
To start a terminal session, use the drop-down menus



A terminal window will appear with a Unix prompt, waiting for you to start entering commands.

To see your home directory, click on '`Locais/Pasta Pessoal`'.

Important Note: because we are using a live version of UNIX, all the files created during a session will not be stored in the system when you end the session. You will need to save your work on a different media, e.g. a USB pen. The UNIX folder for storage media is '`/media`'. You can activate the wireless connection through the top-right corner icon of the desktop view. The Mozilla browser icon, located on the top bar, will allow you to browse the internet.

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

UNIX Tutorial One

1.1 Listing files and directories

ls (list)

When you first login, your current working directory is your home directory. Normally, your home directory has the same name as your user-name. Here, because we use a live CD, it is called *ubuntu*, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type

$ ls (short for list)

The `ls` command lists the contents of your current working directory.

There may be no files visible in your home directory, in which case, the UNIX prompt will be returned.

`ls` does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as hidden files and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with UNIX!!!

To list all files in your home directory including those whose names begin with a dot, type

$ ls -a

`ls` is an example of a command which can take options: `-a` is an example of an option. The options change the behaviour of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behaviour of the command (see later in this tutorial).

1.2 Making Directories

mkdir (make directory)

We will now make a subdirectory in your home directory to hold the files you will be creating and using in the course of this tutorial. To make a subdirectory called `unixstuff` in your current working directory type

$ mkdir unixstuff

To see the directory you have just created, type

$ ls

1.3 Changing to a different directory

cd (change directory)

The command `cd` *directory* means change the current working directory to 'directory'. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type

$ cd unixstuff

Type `ls` to see the contents (which should be empty)

Exercise 1a

Make another directory inside the **unixstuff** directory called **backups**.

1.4 The directories . and ..

Still in the **unixstuff** directory, type

$ ls -a

As you can see, in the **unixstuff** directory (and in all other directories), there are two special directories called (**.**) and (**..**)

In UNIX, (**.**) means the current directory, so typing

$ cd .

NOTE: there is a space between cd and the dot

means stay where you are (the **unixstuff** directory).

This may not seem very useful at first, but using (**.**) as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.

(**..**) means the parent of the current directory, so typing

$ cd ..

will take you one directory up the hierarchy (back to your home directory). Try it now.

Note: typing `cd` with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

1.5 Pathnames

pwd (print working directory)

Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type `cd` to get back to your home-directory and then type

$ pwd

The full pathname will look something like this -

/home/ubuntu

which means that *ubuntu* (your home directory) is in the directory *home*,which is located at the root of the file

system '/'.

Exercise 1b

Use the commands `ls`, `pwd` and `cd` to explore the file system. What's under the root directory ?

(Remember, if you get lost, type `cd` by itself to return to your home-directory)

1.6 More about home directories and pathnames

Understanding pathnames

First type `cd` to get back to your home-directory, then type

$ ls unixstuff

to list the contents of your unixstuff directory.

Now type

$ ls backups

You will get a message like this -

não consigo aceder a backups: Ficheiro ou directoria inexistente

The reason is, **backups** is not in your current working directory. To use a command on a file (or directory) not in the current working directory (the directory you are currently in), you must either `cd` to the correct directory, or specify its full pathname. To list the contents of your backups directory, you must type

$ ls unixstuff/backups

~ (your home directory)

Home directories can also be referred to by the tilde **~** character. It can be used to specify paths starting at your home directory. So typing

$ ls ~/unixstuff

will list the contents of your unixstuff directory, no matter where you currently are in the file system.

What do you think

$ ls ~

would list?

What do you think

$ ls ~/..

would list?

Display a directory and all its content recursively

$ ls -R

or possibly:

$ tree

1.7 Accessing the USB pen to save your work

Looking up the folder where the content of your USB is stored:

$ ls /media

1.8 Wildcards

The characters * and ?

The character * is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your **unixstuff** directory, type

$ ls list*

This will list all files in the current directory starting with **list....**

Try typing

$ ls *list

This will list all files in the current directory ending with **....list**

The character ? will match exactly one character.
So `ls ?ouse` will match files like **house** and **mouse**, but not **grouse**.
Try typing

$ ls ?list

1.9 Filename conventions

We should note here that a directory is merely a special type of file. So the rules and conventions for naming files apply also to directories.

In naming files, characters with special meanings such as **/ * & %** , should be avoided. Also, avoid using spaces within names. The safest way to name a file is to use only alphanumeric characters, that is, letters and numbers, together with _ (underscore) and . (dot).

File names conventionally start with a lower-case letter, and may end with a dot followed by a group of letters indicating the contents of the file. For example, all files consisting of C code may be named with the ending .c, for example, prog1.c . Then in order to list all files containing C code in your home directory, you need only type `ls *.c` in that directory.

Beware: some applications give the same name to all the output files they generate.

For example, some compilers, unless given the appropriate option, produce compiled files named **a.out**. Should you forget to use that option, you are advised to rename the compiled file immediately, otherwise the next such file will overwrite it and it will be lost.

1.10 Getting Help

On-line Manuals

There are on-line manuals which gives information about most commands. The manual pages tell you which options a particular command can take, and how each option modifies the behaviour of the command. Type man command to read the manual page for a particular command.

For example, to find out more about the `wc` (word count) command, type

$ man wc

Alternatively

$ whatis wc

gives a one-line description of the command, but omits any information about options etc.

Apropos

When you are not sure of the exact name of a command,

$ apropos keyword

will give you the commands with keyword in their manual page header. For example, try typing

$ apropos copy

history

The C shell keeps an ordered list of all the commands that you have entered. Each command is given a number according to the order it was entered.

$ history (show command history list)

Summary

`ls`	list files and directories
`ls -a`	list all files and directories
`mkdir`	make a directory
`cd` *`directory`*	change to named directory
`cd`	change to home-directory
`cd ~`	change to home-directory
`cd ..`	change to parent directory
`pwd`	display the path of the current directory
`tree`	list the contents of directories in a tree-like format
`*`	match any number of characters
`?`	match one character
`man` *command*	read the online manual page for a command
`whatis` *command*	brief description of a command
`apropos` *keyword*	match commands with keyword in their man pages
`history`	show command history list

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

UNIX Tutorial Two

2.1 Dowloading files

wget

Download a file from the internet.

$ wget URL

2.2 Copying Files

cp (copy)

`cp` *file1 file2* is the command which makes a copy of **file1** in the current working directory and calls it **file2**

What we are going to do now, is to take a file stored in an open access area of the course website, and use the `cp` command to copy it to your unixstuff directory. Grab a copy of the file science.txt:

$ wget http://alfclul.clul.ul.pt/crpc/curso/CL2012/files/science.txt

Note: you could also use `Ficheiro/Guardar Página como...` from the menu bar to save it into your **Ambiente de Trabalho** directory.

First, `cd` to your unixstuff directory.

$ cd ~/unixstuff

Then at the UNIX prompt, type,

$ cp ~/Desktop/science.txt .

(Note: Don't forget the dot (`.`) at the end. Remember, in UNIX, the dot means the current directory.)

The above command means copy the file **science.txt** to the current directory, keeping the name the same.

Exercise 2a

Create a backup of your **science.txt** file by copying it to a file called **science.bak**

2.3 Moving files

mv (move)

`mv` *file1 file2* moves (or renames) **file1** to **file2**

To move a file from one place to another, use the `mv` command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two.

It can also be used to rename a file, by moving the file to the same directory, but giving it a different name.

We are now going to move the file science.bak to your backup directory.

First, change directories to your unixstuff directory (can you remember how?). Then, inside the **unixstuff** directory, type

$ mv science.bak backups/

Type `ls` and `ls backups` to see if it has worked.

2.4 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the `rm` command. As an example, we are going to create a copy of the **science.txt** file then delete it.

Inside your **unixstuff** directory, type

$ cp science.txt tempfile.txt
$ ls (to check if it has created the file)
$ rm tempfile.txt
$ ls (to check if it has deleted the file)

You can use the `rmdir` command to remove a directory (make sure it is empty first). Try to remove the **backups** directory. You will not be able to since UNIX will not let you remove a non-empty directory.

Exercise 2b

Create a directory called **tempstuff** using `mkdir` , then remove it using the `rmdir` command.

2.5 Displaying the contents of a file on the screen

clear (clear screen)

Before you start the next section, you may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood.

At the prompt, type

$ clear

This will clear all text and leave you with the $ prompt at the top of the window.

cat (concatenate)

The command `cat` can be used to display the contents of a file on the screen. Type:

$ cat science.txt

As you can see, the file is longer than than the size of the window, so it scrolls past making it unreadable.

less

The command `less` writes the contents of a file onto the screen a page at a time. Type

$ less science.txt

Press the `[space-bar]` if you want to see another page, type `[q]` if you want to quit reading. As you can see, `less` is used in preference to `cat` for long files.

head

The `head` command writes the first ten lines of a file to the screen.

First clear the screen then type

$ head science.txt

Then type

$ head -n 5 science.txt

What difference did the 5 do to the *head* command?

tail

The `tail` command writes the last ten lines of a file to the screen.

Clear the screen and type

$ tail science.txt

How can you view the last 15 lines of the file?
Lines 10-15 ? (you need tutorial three for this one)

2.6 Searching the contents of a file

Simple search using less

Using `less`, you can search though a text file for a keyword (pattern). For example, to search through **science.txt** for the word 'science', type

$ less science.txt

then, still in `less` (i.e. don't press [q] to quit), type a forward slash `[/]` followed by the word to search

/science

As you can see, `less` finds and highlights the keyword. Type `[n]` to search for the next occurrence of the word.

grep

`grep` is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

$ grep science science.txt

As you can see, `grep` has printed out each line containg the word science.

Or has it????

Try typing

$ grep Science science.txt

The `grep` command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the -i option, i.e. type

$ grep -i science science.txt

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

$ grep -i 'spinning top' science.txt

Some of the other options of grep are:

-v display those lines that do NOT match
-n precede each maching line with the line number
-c print only the total count of matched lines
-E extended regex

Try some of them and see the different results. Don't forget, you can use more than one option at a time, for example, the number of lines without the words science or Science is

$ grep -ivc science science.txt

2.7 Counting words

wc (word count)

A handy little utility is the `wc` command, short for word count. To do a word count on **science.txt**, type

$ wc -w science.txt

To find out how many lines the file has, type

$ wc -l science.txt

2.8 Encoding of a file

file

`file` gives the file encoding. To report on all files in your home directory, type

$ file -i *

iconv

Translate from one encoding to another.

$ iconv -l (list all known coded character sets)

$ iconv -f iso-8859-1 -t utf-8 file (change encodings of *file* from latin to utf-8)

Summary

`cp` *file1 file2*	copy file1 and call it file2
`mv` *file1 file2*	move or rename file1 to file2
`rm` *file*	remove a file
`rmdir` *directory*	remove a directory
`cat` *file*	display a file
`more` *file*	display a file a page at a time
`head` *file*	display the first few lines of a file
`tail` *file*	display the last few lines of a file
`grep` *'keyword' file*	search a file for keywords
`wc` *file*	count number of lines/words/characters in file
`file -i` *file*	give the encoding of the file
`wget` *URL*	download a file from the internet
`iconv`	translate the encoding

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

Assignment 1

UNIX Tutorial Three

3.1 Redirection

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

In UNIX, we can redirect both the input and the output of commands.

3.2 Redirecting the Output

To print some text on the screen, type

$ echo pear

We use the **>** symbol to redirect the output of a command. For example, to create a file called **list1** containing the word *pear*, type

$ echo pear > list1

What happens is the result of the `echo`, which normally goes to the screen, is now redirected into a file called **list1**.

To read the contents of the file, type

$ cat list1

The symbol **>>** appends standard output to a file. So to add more items to the file **list1**, type

$ echo banana >> list1
$ echo apple >> list1

Exercise 3b

Using the above method, create another file called **list2** containing the following fruit: orange, plum and banana. Read the contents of **list2**.

You should now have two files. One contains three fruits, the other contains also three fruits. We will now use

the `cat` command to join (concatenate) **list1** and **list2** into a new file called **biglist**. Type

$ cat list1 list2 > biglist

What this is doing is reading the contents of **list1** and **list2** in turn, then outputing the text to the file **biglist**.

To read the contents of the new file, type

$ cat biglist

3.3 Pasting files

paste

Writes lines consisting of sequentially corresponding lines of each input file on the standard output.

$ paste list1 list2 > list3 (merge two files into a single one, side by side)

To view list3, type:

$ cat list3

3.4 Sorting files

The command `sort` alphabetically or numerically sorts a list. For example, to sort a list of fruit, type:

$ sort list1

To output the sorted list to a file, type

$ sort biglist > slist

Use `cat` to read the contents of the file **slist**.

To sort a file numerically, type

$ sort biglist -n

To sort a file in inverse order, type

$ sort biglist -r

To sort a file on the second field, type

$ sort list3 -k2

uniq

Given sorted input, writes to the standard output the unique lines, that is, one line in place of what may be

multiple identical lines in the input. If desired, uniq will print a count (-c) of the repeated lines.

$ uniq slist -c

3.5 Pipes

When you want to connect the output of one command as input to another command, what you need is piping. The symbol for a pipe is the vertical bar | :

For example, typing

$ cat biglist | sort | uniq -c

will give the same result as

$ cat slist | uniq -c

or

$ uniq slist -c

Exercise 3b

Using pipes, print all lines of **list1** and **list2** containing the letter 'p', sort the result, and redirect to the file '*hockney*'.

Summary

command > *file*	redirect standard output to a file
command >> *file*	append standard output to a file
command1 | *command2*	pipe the output of command1 to the input of command2
`cat` *file1 file2* > *file0*	concatenate file1 and file2 to file0
`sort`	sort data
`uniq`	report or omit repeated lines
`paste`	paste two files side by side
`echo`	echo some string on the standard output

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

UNIX Tutorial Four

4.1 Regular Expressions

Regular Expressions (regex)

They describe a pattern to match, a sequence of characters within a line of text.

x match the letter 'x'

x+ one or more 'x'

x* zero or more 'x'

? at most one

. any single character

[a-z] lowercase characters

[A-Z] uppercase characters

[0-9] digits

[aeiou] vowels

[^] anything but

^ begin of line

$ end of line

x|y disjunction of patterns

$grep science science.txt (search files for lines with 'science')

$grep '^science' science.txt ('science' at the start of a line)

$grep 'science$' science.txt ('science' at the end of a line)

$grep '^science$' science.txt (lines containing only 'science')

$grep '\^s' science.txt (lines with '^s', "\" escapes the ^)

$grep '[Ss]cience' science.txt (search for 'Science' or 'science')

$grep '[tT][eE][lL]' science.txt (search for tel, teL, tEl, tEL, Tel, TeL, TEl, TEL)

$grep '^$' science.txt | wc -l (search for blank lines and count them)

$grep '[0-9][0-9]' science.txt (search for pairs of numeric digits)

Back Slash "\" is used to escape the next symbol, for example, turn off the special meaning that it has.

4.2 Text Analysis

tr

Translate one set of characters (SET1) into another (SET2). Can also replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character (-s).

$ cat science.tx | tr x X (replace x with X)

$ tr -s ' .,:;!?' '\n' < science.txt (tokenize a file)

awk

Copies its input to standard output, performing specified actions whenever the input matches a specified pattern. AWK automatically parses its input into records and the records into fields. By default, a record is a line, with fields separated by whitespace. With only a small amount of effort, however, it can be used to extract particular records and fields and to rearrange fields. Awk supports the following operators: > (greater than), >= (greater than or equal), < (less than), <= (less than or equal), == (equal), != (not equal). $0 is the entire line and $NF is the last column. $N corresponds to field N.

$ awk '{print $1}' science.txt (print the first column)

$ awk '$2 == "the"' science.txt (print line if the second column is "the")

rev

Reverse lines of a file or files.

$ rev science.txt

Text Analysis

Tokens (words).

$ cat science.txt | tr -s ' .,:;!?' '\n' > tokens

Counting and sorting words.

$ cat tokens | sort | uniq -c | sort -k1 -n -r

Rhyming order.

$ cat tokens | sort | uniq | rev | sort | rev

Ngrams.

tail -n +2 tokens > nextwords (create a next word list)

paste tokens nextwords | head -n -1 > bigrams (paste them together, except for the last)

cat bigrams | sort | uniq -c | sort -k1 -n -r > bigrams.sorted (sort the bigrams, most frequent first)

Find bigrams that appear exactly twice.

$ awk '$1 == "2"' bigrams.sorted

Find types.

$ cat tokens | sort | uniq > types

Find words that appear 30 times or more.

$ cat tokens | sort | uniq -c | awk '$1 > "30"'

Find palindromes.

$ rev types > types.rev

$ paste types types.rev | awk '$1 == $2'

1-syllable words (C*V+C*).

$ cat tokens | grep -iE '^[^aeiou]*[aeiou]+[^aeiou]*$'

Summary

`tr`	translate character set into another
`awk`	pattern scanning and language processing
`rev`	reverse a string
`paste`	paste two files side by side
`echo`	echo some string on the standard output
`who`	list users currently logged in

Acknowledgement: *info.ee.surrey.ac.uk/Teaching/Unix/*

Assignment 2