Tekstmanipulatie, week 12

1.What is a 'regular expression'? `grep`

A more detailed description of what a regular expression is.

A *regular expression* is a set of strings, defined by using "concatenation" (joining substrings), Kleene-star and Kleene-plus (concatenation of taking finite times elements from a given set of strings; in the case of Kleene-star this can be zero time, too), as well as union, intersection or complement of previously defined regular expressions (concatenatie, repetitie, vereniging, intersectie, complement).

What does this definition mean in a human-readable language? A *regular expression* is a way to define a set of strings. It is similar to wild cards: the wild card expression `a*b`, for instance, refers to all strings starting with an a, and ending with a b, thus this expression defines the following infinite set: {ab, aab, abb, acb, adb,...,azb, aAb,..., aZb, a*b, a+b,..., aaab, aabb, aaBb, a%bcb,...}. What shell does is to checkwhich elements of this set are existing file names. If at least one file name is found, then the wild card expression is replaced by the relevant file names. Otherwise, the wild card expression is left unchanged, and this string of characters is used as the argument of the given command.

The idea of a *regular expression* is the same, however, the symbols used and their meaning are a little bit different. You have basically the following tools at your hand:

1. Defining a *set* of characters:
   - A set composed of one character, such as: a
   - A set composed of some characters, such as: `[abcABC]`
   - An abbreviation of the previous, such as: `[a-z]`
   - Any character: . (which corresponds to ? in a wild card expression)
2. The *Kleene-star* of a set, is a list of elements of a set, including the empty string. For instance:
   - The strings composed of zero or more instances of the character a: `a*`
   -

   - The strings composed of zero or more instances of the character a or b (such as the empty string, a, b, aa, ab, ba, bb, aab, babb, etc.): `[ab]*`
   - The strings composed of zero or more instances of a digit: `[0-9]*`
   - The *Kleene-plus* of a set is the same, but requiring at least one element of the set (i.e., without the empty string).
3. *Concatenation* is writing a string after the other. Examples are:
   - The concatenation of two characters form a string of two characters: ab
   - The concatenation of two sets: `[abc][0-9]` refers to all strings composed of two characters that include an a, a b or a c, followed by a digit.
   - The concatenation of a set and the Kleene-star of a set: `ba*` is the set {b, ba, baa, baaa,...}.
4. In addition to that, in Unix you can refer to the position of a string within the line (beginning or end of the line)

grep

'grep' is a very useful command, we will use it a lot. Its simplest syntax is:

```
    grep <reg_ex> [ file_names]
```

What does it do? It outputs the lines of the given file(s - if more than one given) (or, if not specified, from the input) that match the given regular expression. It can be seen as a filter to collect only the useful information (e.g. if too much output from a program).

If you want the lines that match both of two conditions (conjunction), then use a pipe-line. If you want lines that match (at least) one of two conditions (disjunction), then use the -F option or the 'fgrep' command. A few others of its most important options:

-c	returns you only the number of lines matching the given regular expression
-i	ignore case distinction: does not differentiate between capital and lowercase letters
-v	inverse: returns those lines that don't match the condition

But, the syntax of a regular expression here is slightly different from the one used for file names (remember the wildcards).
The metacharacters are the following: . (period), *, [ ], \, ^and $, as well as '-' within the [ ] brackets. Their meanings are:

1. Definition of characters, sets of characters:
   . any character (as ? for file names)
   Character classes: see below.
   [ ] any character within the brackets (a set of given characters). Special rules for this:
   - an interval of characters can be abbreviated by '-': [a-z], [0-9], [m-p]
   - a ^written in the first position means the inverse of the listed characters (anything except those)
   - if you want to list the character "]" within this list, you should put it into the first position, thus '[][]' matches a left bracket and a right bracket.

2. Repetition of some sets:
   * Kleene-star (Kleene closure): the repetition of the expression before it, any times (even 0 times)
   [In egrep, + stands for Kleene-plus (once, or more times)]
   [In egrep, ? stands for optionality (zero times or once)]
   [See the examples below on how to refer to n times, or more than n times, etc. in egrep.]

3. Position within the line:
   ^ beginning of the line (only at the beginning, otherwise it matches itself)
   $ end of the line (at the end of the outermost expression, otherwise it matches itself)

Furthermore, you have so-called "character classes", like:
[:upper:] uppercase letters (A-Z, and including some further, non English characters depending on your system)
[:lower:] lowercase letters (similarly)

[:alpha:] all letters (A-Z, a-z, and maybe more)
[:digit:] the digits 0 through 9, precisely
[:xdigit:] the hexadecimal digits (0-9, A-F, a-f)
[:punct:] the punctuation characters, such as `!"#$%'()*=,-./;:<=>?@[\]^_`{}|~`
[:graph:] all "graphic" characters, including the mutually excluding 'alpha', 'punct' and 'digit' classes, except <space>
[:print:] all "printable" characters, like graphic characters and <space>
[:blank] <space> and <tab>

E.g. `grep [[:digit:]]` will return all lines containing a number. Notice the double brackets, that you will need in some systems!

Further possibilities:

\{m\}	matches exactly m times
\{m,n\}	matches between m and n times
\{0,n\}	matches maximum n times
\{m,\}	matches minimum n times

These are called BRE = Basic Regular Expressions.

Remarks: The concatenation (written one after the other) of two regular expressions is also a regular expression. The so-called Kleene-plus (any number of repetitions of the given regular expression, but at least one) can be realized as: <reg_ex> <reg_ex>*.

Further possibilities (ERE = Extended Regular Expressions), using `egrep` or `grep -E` :

- don't use the backslash (\) before the {, } symbols in {m, n}, etc.
- ? matches 0 or 1 time
- + matches at least one time (Kleene-plus)
- | means disjunction (OR), like in: `echo aaa | grep 'a|b'`
- you can form groups with (and), e.g. when having a disjunction

A few examples:

[oai]n	either 'on' or 'an' or 'in'
[0-9][0-9]	two consecutive digits
^[aeiou]	a vowel at the beginning of the line
^.[aeiou]	a vowel at the second position of a line

^[aeiou]$ a line consisting exactly of a vowel
[^0-9] anything but a digit (it will return you all lines containing (also) something different from a digit
[^0-9]$ a line ending with something different from a digit
^[d\-] a line beginning with a 'd' or a '-' (when is it useful?)
abb* an occurence of 'a', followed by any number of occurrences of 'b' (but at least one)
[0-9][0-9]* a sequence of any number (but at least one) of digits (an unsigned integer)

Don't forget using escape characters or quotes, when needed!

What is the difference between `grep apple` and `grep ^apple$` ? The first one will return all lines containing the string `apple`, whereas the second one will return only the lines that contain exactly `apple` and nothing else.

2. Permissions, `chmod`

What information does the "long list" contain?

```
total 49
drwxr-xr-x      2 birot     aistaff       172 Aug 26 15:37 info
-rw-r--r--      1 birot     aistaff       951 Aug 26 13:30
internet.gif
-rw-r--r--      1 birot     aistaff       653 Aug 26 15:29
intro.htm
```

- colors according to file types
- very first character: '-' for simple file, 'd' for directory, 'l' fpr symbolic link, etc.
- 3 times 3 character: permission for user, group and others:
  r = permission to **r**ead, w= permission to **w**rite, x = permission to e**x**ecute
- number of links belonging to this file (directories have additional links to their

subdirectories)
- user and the group owning the file (group is usually the group of the owner, but not necessarily)
- size of the file in characters ('total' on the top: total number of diskblocks occupied by the listed files)
- Date and time (or date and year) when the file was last modified (using the -u option: last access)
- file name

Users are grouped into groups. The owner can transfer the file to another user or to another group with the commands 'chown' and 'chgrp' respectively.

Changing the permissions of a file: 'chmod'

chmod <permissions> <file_name>
u = user, g = group, o = others, a = all (also: ug, uo, etc., ugo = a)
+ = give permission; - = remove permission
r = permission to **r**ead, w= permission to **w**rite, x = permission to e**x**ecute

E.g.: `chmod g+w intro.htm` or `chmod o-x info`

Setting the permissions:
4 = permission to **r**ead, 2 = permission to **w**rite, 1 = permission to e**x**ecute

E.g. `chmod 753 intro.htm` means: rwx to owner, r-x to group and -wx to others.
What permissions would you give to a file that contain your private mails? To a secret document that the group is writing together? To a program you want the others to use, but not change it, nor checkits content?

The meaning of read, write and execute depends on the file type:

	regular files	directories
read	read contents	list directory
write	change file	alter files in directory
execute	run program	files accessible (to read or execute them)

3. Link, `ln`

A person can have several names, like diminutive or aliases. Similarly, you can link several names to the same file.

First, let's understand a little bit the mechanism of the file structure. The information relating to a given file is to be found on three levels:

- Within the catalogue you can find the name of the file and a pointer to the relevant line within the so-called "i-node table".
- In the "i-node table" you will find all relevant information (e.g. permissions) and a pointer to the actual place where the content of the file is to be found.
- The actual place where the content of the file is to be found (e.g. the winchester, the floppy, and other machine,...)

There are two types of linking:

- The "hard link" creates a new file name with a pointer to the same line in the "i-node table".
- The "soft link" (symbolic link) creates a pointer to the original file name.

The command '`ln <existing_file> <new_name>`' creates a hard link. Adding the -s will create a soft link.

The number appearing after the permissions in the long list ('`ls -l`') shows the number of hard links to the given file. The very first character in the long list is 'l' in the case of symbolic links.

Changing the content of a linked file will affect the third level. Moving and removing a file will affect only the first level, that is only the file name. If you delete by chance a file that has been linked with a hard link to another one, you are on the safe side, because the content still exist, and the other file name points to it. One the other hand, if the given content is pointed to by only one file name (independently of whether this file is pointed to by a soft link), then deleting this file will result in the lost of the content, too.

Try out what happens if you have a soft link 'A' pointing to a file 'B', and then you delete 'B': what happens to 'A'? And what happens if you create a new file with the

name 'B'?

4. Protokols (`telnet`, `ssh`, `lynx`, `ftp`)

Suppose you are home and you want to log in to Hagen. Or you are anywhere else but in the Unix computer room. How can you log in to a remote computer? How can you make your local computer become a terminal of another computer?

Telnet is one solution. It exists under almost any systems, such as DOS, Windows, UNIX, etc. Just run it, and you will get a window that is a terminal of a remote computer in front of you. The only disadvantage is that it is not a graphic interface, therefore you won't be able to use the most comfortable tools, like clicking with the mouse, etc.

In fact "telnet" is a protocol. What is a protocol? It is a standard of communication between two different systems, that may be very far from each other, and may be operating in very different ways. Independently of their inner structure, they still can understand each other, due to the standardized protocols.

For what reason would two computers communicate with each other? They may want to exchange information, like emails or files. Therefore one basic protocol is `FTP` (for File Transfer Protocol, see later in this course). Another one is the well-known `http` (hypertext transfer protocol), which is a more advanced protocol to transfer more specialized files (allowing e.g. links). Telnet is thus the protocol using which you can log in into a remote computer, and opening a terminal of the remote machine using your local computer.

There is a newer version of it, which is called '`ssh`', that stands for 'secure shell protocol'. It is the same as telnet, i.e. it opens a terminal (a shell) belonging to the remote computer on your local computer. The only difference is that it encodes the information, thus no third party can have access to the information going between you and the remote computer, in both directions. Therefore it gets more and more popular, and many system administrators tend to forbid telnet, allowing only ssh. (There is a secure version of http, too, it is called https.)

How would I surf on the web if I don't have a graphical interface? How can I run my favourite Netscape? Well, this is a big disadvantage of not having a graphical interface, indeed. But, either you believe or not, life existed even before graphical interfaces! The prehistorical browser you still can use is called: '`lynx`'. Use the cursor-up and cursor-down keys to see the page and to walk between the links (these are highlighted), use the cursor-right to follow a link, and use the cursor-left to come back. To open a new URL, either you type it in the command line when starting lynx, or -- once it has been run -- press 'G'. Other commands appear in the bottom line (including H for help and O for options). To quit, press Q.

It is worth trying out once to make up your mind: press Q, then you are asked if you are sure that you want to leave. Press N for 'no', and look at lynx's reaction!

If you design a web-site, no matter how wonderful graphic it has, it should be usable also using lynx...

The abbreviation `ftp` stands for File Transfer Protocol. This is the simplest way to transfer files from one machine to another one, supposing you can log in to the remote machine, as well. The ftp program creates an interface which allows you only to do the basic steps that you need to transfer files. (The interfaces created by other protocols let you do more: in the case of 'telnet' you can do on the further machine whatever you could do from a non graphic terminal; while in the case of web browsers, like lynx and others :-)), they will automatically present you the files transferred.)

You will need it if for instance a Windows fun friend of you will send you some funny attachments that you are not able to open on Linux. Something happening pretty often...

In the early and middle 90s, before the web became so popular, there used to be lot of "anonymous ftp servers". If you knew their addresses, you could just log in as a "guest", without any password, and download public files, programs, images, etc. that were made available by others. Nowadays people prefer putting these files on their web page.

Unless you have a fancy ftp program that shows you your local directory and your remote directory, and let you transfer files by clicking on a button, you have the following commands:

> ftp machines.name.nl - connect to the given machine (you can give the name of the remote machine as an argument already when running ftp).

> disconnect - disconnect from the remote machine
>
> bye - exit the ftp program (not 'exit', not 'quit', but 'bye'!)
>
> bin - change the transfer protocol to binary (8 bits, instead of 7 bits; very important when transferring images, word files, etc.)
>
> put - put a file from the local system to the remote system
>
> get - get a file from the remote system to the local system
>
> mput, mget - the same but allow you using standard UNIX wildcards
>
> cd, pwd - change directory, print working directory on the remote system
>
> lcd, lpwd - same on the local system
>
> help - for more commands

Bíró Tamás:
e-mail
English web site
Magyar honlap

Last modified: Thu Jul 3 11:39:17 METDST 2003