Unix Tools	**Lecture 5**
CS114 Fall 2003	**Wednesday, October 8, 2003**

Regular expressions

As you'll notice, text files turn out to be important under Unix. One consequence of this is that you'll often need to search for something in text files. For example, you are editing a file in Emacs and are looking for something, or you are in a directory and want to know in which files a given string occurs, etc.

The simplest searches are searches for a simple word. This is the kind of search you do in Emacs when you do a `C-s`. However, you often want to search for something more general. You don't want to search exactly for a string, but any string that matches some "shape" you care about. For example, you may want to search for all strings made up exclusively of 0's and 1's.

There is a convenient notation for writing such things: regular expressions. If you stay in Computer Science long enough, you'll encounter regular expressions again and again.

Formally (and we'll try *not* to be too formal here), a regular expression r (also called simply a regexp) represents a set of strings. A string is said to *match* a regexp r if the string is one of the strings represented by r.

By default, a regexp which is simply a string of characters matches a single string, namely itself. Hence, the regexp `abc` matches a single string, `abc`. However, some characters have a special meaning in a regexp:

- `.` matches any character. Hence, the regexp `a.c` matches strings such as `aac`, `abc`, `a3c`, `a;c`, etc.
- `[abcd]` matches *any* of the characters between the brackets. For example, `a[01b]c` matches any of `a0c`, `a1c`, or `abc`. The special notation x`-`y can be used anywhere in the brackets, to represent all the characters between x and y. For example, the regexp `[a-z]` matches any lowercase letter, while the regexp `[a-zA-Z]` matches any lowercase or uppercase letter.
- `*` matches 0 or more occurences of the preceding. For example, the regexp `ab*c` matches any string that starts with an `a`, followed by any number (including zero) of `b`'s, followed by a `c`. Similarly, `a[01b]*c` matches any string starting with an `a`, followed by any numbers of characters matching `[01b]`, followed by a `c`.

What if you want to match a "special" character exactly? For example, what if you want to match the `*` character? If you *escape* a special character, i.e., if you precede it by a backslash character `\`, then the character stands for itself, not for a regular expression operator. (To match a `\` exactly, you escape it as well, i.e., you write `\\`.). Thus, `a\**c` matches an `a` followed by any number of `*`'s, follwed by a `c`.

Some characters help delineate where a matching can occur:

- `^` matches only at the beginning of a line. For example, `^abc` will match the string `abc` if it is at the beginning of a line only.
- `$` matches only at the end of a line. For example, `abc$` will match the string `abc` if it is at the end of a line only.

The above operators define so-called *basic regular expressions*. These operators are supported in most programs using regular expressions of some sort. Some programs in fact implement *extended regular expressions*, which provide slightly more operators:

- `+` matches one or more of the preceding. Hence, `ab+c` will match a string starting with an `a` followed by one or more `b`'s, followed by a `c`.
- `?` matches zero or one of the preceding. Hence, `ab?c` will match `ac` and `abc`, and that's it.
- `( )` can be used to package together a regular expression, so that `*`, `+` or `?` can be applied to it. Hence, `a(bc)*d` matches `ad`, `abcd`, `abcbcd`, `abcbcbcd`, etc.
- `|` can be used to define choice. For example, `a(bc|de)f` matches `abcf` and `adef`.

More operators in fact exist, some allowing you to match the same substring in multiple places in a given string. I'll refer you to the links on the web page for more information.

Regexps in Emacs and grep

We saw that `C-s` and `C-r` can be used to perform searches for a string in Emacs. To search for a regular expression instead, we can use `M-C-s` and `M-c-r`. The minibuffer will ask you for a regexp to search for.

It is also possible to perform replacements of the strings matched by a regular expression into some other string, possibly including parts of the matched string. I'll refer you to the documentation for such. One such function is `replace-regexp`.

It is possible to search for a regular expression in a file directly from the command line. There are in fact many commands to do so, differing in the kind of regular expressions they can handle:

- `fgrep` *'string' file1 file2 ...* searches the given files for *string*.
- `grep` *'regexp' file1 file2 ...* searches the given files for the given basic regular expression *regexp*
- `egrep` *'eregexp' file1 file2 ...* searches the given files for the extended regular expression *eregexp*.

By default, `grep` and company returns every line of the files for which there is a match of the regular expression. Some interesting options to `grep` are *-i* to ignore the case of letters during matches, and *-v* to output the lines for which there is *no* match.