

Programming in shell 1

Text processing.
Regular expressions.
Filter grep.

Jan Trdlička



Czech Technical University in Prague, Faculty of Information Technology
Department of Computer Systems

1 Text processing

2 Pattern specification

- Fixed-character strings: `grep -F / fgrep`
- Extended regular expressions: `grep -E / egrep`
- Basic regular expressions: `grep`

Text processing

- The discipline of mechanising the creation or manipulation of electronic text.
- **Electronic text**
 - Character encoding
 - It defines how to interpret sequence of bits into real characters.
 - **ASCII** (using the bottom 7 bits of byte).
 - **ANSI** (using 8 bits with several different code pages for the symbols 128 to 255).
 - **Unicode-based encodings**: UTF8 (1 byte), UTF16 (2 bytes), UTF32 (4 bytes).
 - In different character codings, the characters can be in different order (abc...zABC...Z vs. aAbBcC...zZ).
 - Text
 - The sequence of abstract characters.
- **The most common text operations**
 - **Find** text containing pattern.
 - **Find** text containing pattern and **replace** it by a new text.

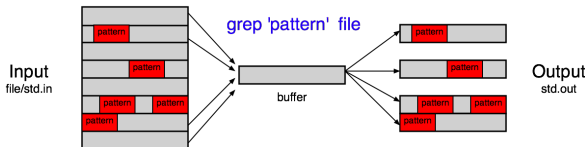
Pattern specification

- The pattern can be specified in several different ways.
- **Fixed-character string**
 - All characters are interpreted literally (no character has special meaning).
- **Regular expression**
 - The pattern may contain meta-characters that have special meaning to the application that interprets the regular expression.
 - According to the set of supported meta-characters we define
 - Basic regular expressions (BRE),
 - Extended regular expressions (ERE),
 - Perl Compatible Regular Expressions (PCRE).
 - Regular expressions can be interpreted by
 - library functions (`match`, `split`, `gsub`, ...) of different programming languages (Java, C#, Python, ...),
 - applications (`less`, `vi`, `grep`, `sed`, `awk`, ...).

Fixed-character pattern: `grep -F` / `fgrep`

`grep -F [options] patter [files]` (GNU implementation)

`fgrep [options] patter [files]`



- The `pattern` is interpreted literally (fixed-character string).
- The filter `fgrep` (fast `grep`) searches standard input/text files for a character string and prints all lines that contain that string.
- Useful options
 - `-i` ... ignores upper/lower case.
 - `-v` ... prints all lines except those that contain the pattern.
 - `-c` ... prints only a count of the lines that contain the pattern.
 - `-l` ... prints only the names of files with matching lines.
 - `-n` ... precedes each line by its line number in the file (first line is 1).

Fixed-character pattern: `grep -F` / `fgrep`

• Examples

- Print all lines from file `/etc/passwd`, that contain string "root".

```
grep -F 'root' /etc/passwd
```

- Print all lines from file `/etc/passwd`, that don't contain string "root".

```
grep -F -v 'root' /etc/passwd
```

- How many lines from the file `/etc/passwd` contain string "root".

```
grep -F -c 'root' /etc/passwd
```

- How many lines from the file `/etc/passwd` don't contain string "root".

```
grep -F -vc 'root' /etc/passwd
```

Fixed-character pattern: `grep -F` / `fgrep`

- **Examples**

- Print only names of files from the directory `/etc` that contain the string `"root"`.

```
grep -F -l 'root' /etc/* 2>/dev/null
```

- Print only name of file from the directory `/etc` that contains the string `"root"` and the string is located on the line with the highest number.

```
grep -F -n 'root' /etc/* 2>/dev/null | \
    sort -t':' -k2,2n | tail -1 | cut -d':' -f1
```

- How many students have an account on the server `fray3.fit.cvut.cz`?

```
ssh $USER@fray3.fit.cvut.cz 'getent passwd' | \
    grep -Fc 'student:'
```

- Is it correct solution?

Extended regular expressions: `grep -E` / `egrep`

`grep -E [options] patter [files]` (GNU implementation)
`egrep [options] patter [files]`

- The filter searches standard input/text files for a pattern and prints all lines that contain that pattern.
- Pattern is defined as basic regular expression (ERE) in which the following meta characters are interpreted.
 - **System V supports** (in Solaris `/usr/bin/egrep`)
 - Disabling/enabling meta-character: `\.`
 - Anchoring : `^`, `$`.
 - Single character: `.`, `[]`, `[^]`.
 - Repetition: `?`, `+`, `*`.
 - Subexpressions: `()`.
 - Alternation: `|`.
 - **In addition, GNU supports** (in Solaris `/usr/bin/ggrep -E`)
 - The backslash character: `\<`, `\>`.
 - Repetition: `{ }`.
 - Back references: `()`, `\n`.

Extended regular expressions: `grep -E` / `egrep`

- Anchoring

<code>^</code>	The caret matches the the begin of a line.
<code>\$</code>	The dollar sign matches the end of a line.

- Examples

- Print entries of the directory `/etc` that represent symbolic links.

```
ls -la /etc | grep -E '^l'
```

- How many users who have an account on this system are named Jiri?

```
getent passwd | cut -d':' -f5 | grep -E '^Jiri_'
```

- Print login names of users that have bash set up as their login shell on this system?

```
getent passwd | grep -E '/bash$' | cut -d':' -f1
```

Extended regular expressions: `grep -E` / `egrep`

- The backslash character

<code>\<</code>	The symbol matches the begin of a word (only GNU).
<code>\></code>	The symbol matches the end of a word (only GNU).

- These meta-characters are not supported by `egrep` in Solaris.
- Examples
 - Print all lines of command `man ls`, that contain the **string** "the".

```
man ls | grep -E 'the'
```

- Print all lines of command `man ls`, that contain the **word** "the".

```
man ls | grep -E '\<the\>'
```

- Copy the file `/usr/dict/words` from the server `fray1.fit.cvut.cz` to your working directory.
- Print lines from the file `words`, that contain words starting with the character 'b'.

```
grep -E '\<b' words
```

```
grep -E '^b' words      # file has one word per line
```

Extended regular expressions: `grep -E` / `egrep`

- Single character

<code>.</code>	The period matches any single character.
<code>[]</code>	It matches any single character in the list/range expression.
<code>[^]</code>	It matches any single character not in the list/range expression.

- Examples

- Print all the words of length 4 from the file `words`.

```
grep -E '^....$' words
```

- Print all the words of length 3 from the file `words`, that have in the middle the vowel (a, e, i, o, u, y).

```
grep -E '^.[aeiouy].$' words
```

- Print all the words of length 3 from the file `words`, that have not in the middle the vowel (a, e, i, o, u, y).

```
grep -E '^.[^aeiouy].$' words
```

- Print all words from the file `words`, that begin uppercase character.

```
export LC_ALL=C          # dependent on character encoding
grep -E '^[A-Z]' words
```

Extended regular expressions: `grep -E` / `egrep`

• Character classes

- The set of characters, that are defined by a range of characters, depends on the locales (character encoding).
- In order to avoid dependence on locales, character classes have been defined (not supported by `egrep` in Solaris).

<code>[[:digit:]]</code>	Digits: '0 1 2 3 4 5 6 7 8 9'.
<code>[[:lower:]]</code>	Lower-case letters: 'a b c ... x y z'.
<code>[[:upper:]]</code>	Upper-case letters: 'A B C ... X Y Z'.
<code>[[:alpha:]]</code>	Alphabetic characters.
<code>[[:alnum:]]</code>	Alphanumeric characters.
<code>[[:blank:]]</code>	Blank characters: space and tab.

• Examples

- Print all the words from the file `words`, that contain digit.

```
grep -E '[[:digit:]]' words
```

- Print all the words from the file `words`, that have the digit only as the first character.

```
grep -E '^[[[:digit:]]][^[:digit:]]*$' words
```


Extended regular expressions: `grep -E` / `egrep`

- Alternation, subexpressions and back references

	Two regular expressions may be joined by this infix operator. The resulting regular expression matches any string matching either alternate expression.
()	Defines a marked subexpression. The string matched within the parentheses can be recalled later.
\n	Matches what the <i>n</i> -th marked subexpression matched.

- Meta-character `\n` is not supported by `egrep` in Solaris.

- Examples

- Print all the words from the file `words`, that begin with string "work" or end with string "work".

```
grep -E '^work.*|.*work$' words
```

- Print all the words from the file `words`, that contain the following strings: "boy", "girl", "woman" or "man".

```
grep -E 'boy|girl|woman|man' words
```

- Print all the words from the file `words`, that end with one of the following strings: "boy", "girl", "woman" or "man".

```
grep -E '(boy|girl|woman|man)$' words
```

Extended regular expressions: `grep -E` / `egrep`

- Subexpressions and back references

- Examples

- Print all the words from the file `words`, that begin and end with the same character.

```
grep -E '^(\.).*\1$' words
```

- Print all the words from the file `words`, that begin and end with the different character.

```
grep -Ev '^(\.).*\1$' words
```

- Print all **palindromes** of the length 3 from the file `words`.

```
grep -E '^(\.).\1$' words
```

- Print all **palindromes** of the length 4 from the file `words`.

```
grep -E '^(\.())\2\1$' words
```

Basic regular expressions: grep

`grep [options] patter [files]`

- The filter `grep` searches standard input/text files for a pattern and prints all lines that contain that pattern.
- Pattern is defined as basic regular expression (BRE) in which the following meta characters are interpreted.
 - **System V supports** (in Solaris `/usr/bin/grep`)
 - Anchoring : `^`, `$`.
 - The backslash character: `\<`, `\>`.
 - Single character: `.`, `[]`, `[^]`.
 - Repetition: `*`, `\{ \}`.
 - Subexpressions and back references: `\(\)`, `\n`.
 - **In addition, GNU supports** (in Solaris `/usr/xpg4/bin/grep`)
 - Repetition: `\?`, `\+`.
 - Alternation: `\|`.
- **Note**
 - Originally, the characters `?`, `+`, `|`, `<`, `>`, `{ }` and `()` were not meta-characters. Special meaning was added later.
 - Therefore, the character `\` allows the special meaning of these characters in BRE (backward compatibility).