

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
```

```
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



# UNIX/Linux environment

## Regular expressions and find

Stefano Quer, Pietro Laface, and Stefano Scanzio

Dipartimento di Automatica e Informatica

Politecnico di Torino

[skenz.it/os](http://skenz.it/os)

[stefano.scanzio@polito.it](mailto:stefano.scanzio@polito.it)

## Regular expressions (RE)

- ❖ Introduced in 1956 by the mathematician Stephen Cole Kleene in the automaton and formal language domain
- ❖ Used from the seventies in the UNIX environment
  - Editors (vi, emacs, etc.)
  - Shell commandi di shell (find, grep, etc.)
  - Scripting languages (sed, awk, perl, python, etc.)

# Regular expressions (RE)

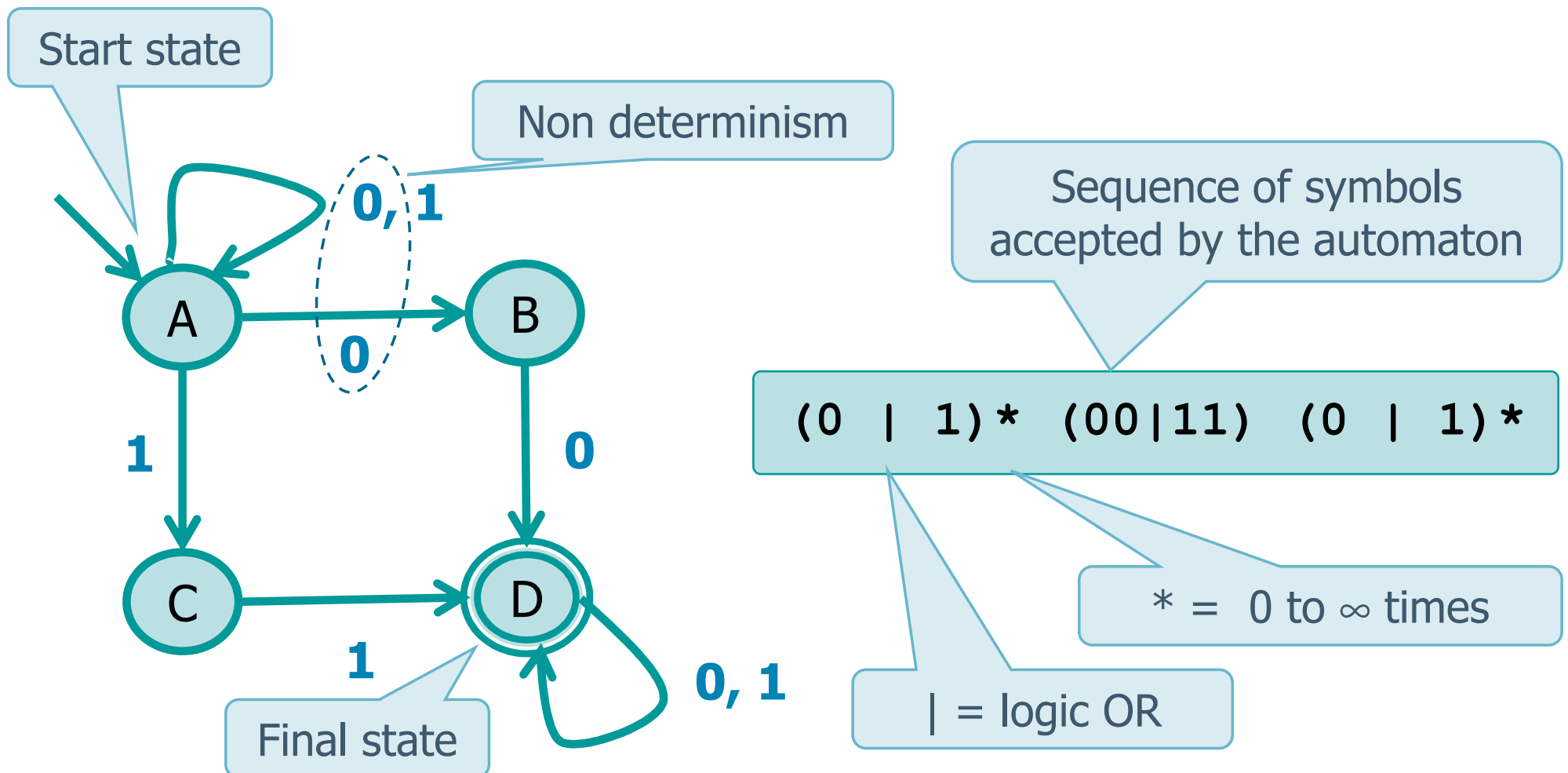
- ❖ A standard defined for POSIX in 1992
- ❖ Several versions exist, using similar but **different** formalisms
  - BRE, Basic Regular Expression
  - ERE, Extended Regular Expression
  - PCRE, Perl Compatible Regular Expression
    - C Library of Regular Expressions (Hazel, 1997)
    - More flexible than POSIX version
    - De-facto standard with Perl 5
- ❖ In this course
  - We will use simple regular expressions in find, grep, and shell scripting

# Regular expressions (RE)

- ❖ A regular expression (or **pattern**) is an expression that specifies a set of strings
  - Compact operators are used to represent complex sequences of characters
    - Example
      - $a \mid b^*$  represents the set of strings  $\{a, \phi, b, bb, bbb, bbbb, \dots\}$
- ❖ Expressions are useful to find if a **match** exists between objects
  - Directories or file names, lines or fields of a file, strings or sub-strings, etc.

# Regular expressions and automata

- ❖ A regular expression corresponds to a Non Deterministic Automaton (NFA)





# Definitions

## ❖ Literal

- Any character (or character sequence) used in the search of a match
  - ind matches **w**indows, **i**ndifferent, etc.

## ❖ Metacharacter

The power of RE is hidden in the use of metacharacters

- One or more characters having special meaning
  - \* indicates 0 to  $\infty$  preceding symbols, e.g.,  $b^* = \{\phi, b, bb, bbb, \dots\}$

## ❖ Escape sequence

- Allows using literally a metacharacter
  - Character '.' must be given as '\.'

# Metacharacters

Operator	Meaning
[...]	Specifies a list or range of symbols
(...)	Manages operator precedence Groups sub-expressions Allows reference to previous expressions (backward reference)
	Logical OR between REs

Basic RE: `\[...\]` and `\(...\)`

Anchors	Meaning
\<	Beginning of word
\>	End of word
^	Beginning of line
\$	End of line

Special characters	Meaning
\+ \? \.	Characters '+', '?', '.'
\n	New line
\t	TAB

# Meta-characters

Quantifiers And ranges	Meaning
*	$[0, \infty]$ times
+	$[1, \infty]$ times
?	$[0, 1]$ times
$[c_1c_2c_3]$	Any character in parenthesis
$[c_1-c_5]$	Any character in range
$[^c_1-c_5]$	Any character not in range
$\{n\}$	Exactly $n$ times
$\{n_1, n_2\}$	$n_1$ to $n_2$ times

Superset

grep command:  
Allows also  $\{n_1,\}$  or  $\{,n_2\}$



# Meta-characters

Characters	Meaning
<b>c</b>	Any symbol c (excluding special ones)
<b>.</b>	Any character (excluding '\n')
<b>\c</b>	Any control character
<b>\s</b>	A space or TAB
<b>\d</b>	A digit [0-9]
<b>\D</b>	Not a digit [^0-9]
<b>\w</b>	Letters, numbers and _ [0-9A-Za-z_]
<b>\W</b>	Not letters, numbers and _ [^0-9A-Za-z_]

Some are not recognized  
by some commands

# Examples

Regular expression	Meaning
ABCDEF	String "ABCDEF"
a*b	Any number of 'a' followed by a single 'b'
ab?	a or ab
a{5,15}	5 to 15 repetitions of 'a'
(fred){3,9}	3 to 9 repetitions of string "fred"
.+	Any, non empty, sequence (entire line)
myfunc.*(.*)	A function with name beginning by "myfunc"
^ABC.*	A line beginning by "ABC"
.*h\$	A line ending by "h"
hello\>	Word ending by "hello"
a+b+	One or more 'a' followed by one or more 'b'

# Examples

Regular expression	Meaning
<code>.*b.*3</code>	Matches string <code>"/fbar3"</code>
<code>[a-zA-Z0-9]</code>	A letter or a digit
<code>A b</code>	A or b
<code>\w{8}</code>	A 8 character word
<code>((4\.[0-2])  (2\.[0-2]))</code>	Numbers 4.0, 4.1, 4.2 or 2.0, 2.1, 2.2
<code>(.)\1</code>	Two times the same character (ex. "aa")
<code>(.)(.)\2\1</code>	Any 5 character palindrome string (e.g., radar, civic, 12321, etc.)

Basic RE: `\(.\)\1` and `\(.\)\(.\)\2\1`  
Extended RE: `(.)\1` and `(.)(.)\2\1`

`\1, \2`  
Backward Reference

## Exercise

## ❖ Write a RE to match

- All the lines containing an integer number between 1 and 50 (extremes included)

```
^[1-9]$ | ^[1-4][0-9]$ | ^50$
```

- Any date with format yyyy/mm/dd

- A year can be expressed on 2, 3 or 4 digits. Assume that the number of days of each month is 31.

```
\d{2,4} \\/ (0[1-9] | 1[0-2]) \\/ (0[1-9] | [1-2][0-9] | 3[01])
```

\d{2,4} equal to [0-9]{2,4}

## ❖ Allows

- Searching and listing the file, directories or links that match a given criterion
- Possibly executes a shell command on every listed file

## ❖ Notice that

- Find outputs the relative path of the matching files, not their basenames
- This is important to write the REs to match a give path and the actions to be performed



## ❖ Format

- find directory options actions

## ❖ In details, the find command

- Visits the **directory** subtree
- Outputs the list of pathnames satisfying the **options**
- Possibly performs the **actions** on every file of the list

## ❖ We need to analyze how to specify directory, options and actions

## find: specify directory

❖ Specifies the search directory tree in which execute the command

- .
- /usr/bin
- ./subDirA/subDirB

```
find directory options actions
```

# find: specify options

Option	Meaning
-name pattern	Match with the file name. The initial path (basename) is removed. In some versions it is possible to place the pattern between double quotes to specify regular expression. <b>-iname</b> is equal but case insensitive
-path pattern	Like the previous one, but you need to specify path+filename <b>-ipath</b> is equal but case insensitive

```
find directory options actions
```

# find: specify options

Option	Meaning
-regex expr	Specifies a regular expression that matches the found <u>relative path</u> (full path) <b>-iregex</b> is equal but case insensitive
-regextype type	Indicates the type of regular expressions used: posix-basic, posix-egrep, posix-extended, etc. You must specify the type before the regular expression ( <b>regextype</b> must precede <b>regex</b> )
-atime [+,-]n -ctime [+,-]n -mtime [+,-]n	Last access, status or modification time n=1 specifies from 0 to 24 hours back n value with sign : + means $\leq$ , - means $\geq$

find directory **options** actions

# find: specify options

Option	Meaning
<b>-size [+,-]n[bckwMG]</b>	<p>File dimension</p> <p>Sign + means <math>\geq</math>, - means <math>\leq</math></p> <p>Next character indicates the size :</p> <ul style="list-style-type: none"><li>• b blocks (of 512 bytes)</li><li>• c bytes</li><li>• k kbytes</li><li>• w word (2 bytes)</li><li>• M Mbytes</li><li>• G Gbytes</li></ul>
<b>-type type</b>	<p>File type</p> <p><b>f</b> per regular file (i.e., text files, executable, etc.), <b>d</b> for directories, <b>p</b> for pipes, <b>l</b> for symbolic links, <b>s</b> for sockets</p>

```
find directory options actions
```



# find: specify options

Option	Meaning
-user name -group name	File owner identifier ( <b>user name</b> ) File group identifier ( <b>group name</b> )
-readable -writable -executable	File access permissions
-mindepth n -maxdepth n	Search limited to a subtree section: <b>mindepth</b> and <b>maxdepth</b> indicate the minimum and maximum depth from <b>directory</b> (-maxdepth 1 means search only on <b>directory</b> )
-quit	Quit the search after the first match

```
find directory options actions
```

## Examples: find & options

```
find . -name "*.c"
```

File with ".c" extension

```
find . -regex "*.c"
```

Wrong: "\*.c" is not a RE

```
find . -regex ".*\\.c"
```

Correct equivalent RE

```
find /usr/bin -iname "a.*"
```

All files starting with 'a' or 'A' that do have any extension

All files with dimension  
>500 bytes

```
find . -size +500c
```

All readable files in the  
current directory (.) with  
name beginning by ab, aab,  
aaab,etc., and any extension

```
find . -readable \  
-regex "\./a+b.*\\.\\.\\."
```

## Examples: find & options

```
find /usr/bin \  
-regextype posix-extended \  
-regex ".*\./..(..)(.).\2\1.*"
```

```
find /usr/bin -regex \  
".*\./..\(..\)\(..\).\2\1.*"
```

All files in directory `/usr/bin` starting with two characters, followed by a palindrome word with 5 characters, and followed by other characters in indefinite number

The same but with standard RE

All files with extension `exe` in directory `/home/usr` from level 2 to 4 (included)

```
find /home/usr/ \  
-mindepth 2 -maxdepth 4 \  
-name "*.exe"
```

## find: specify actions

- ❖ The default action of **find** is to output the list of matching files
  - The default action is equivalent to the **print** command
    - find directory options **-print**
- ❖ It is possible to execute
  - any shell command
  - on every pathname of the matching list

```
find directory options actions
```

## find: specify actions

Option	Meaning
<b>-print</b>	Default action. Print a name for each line
<b>-fprint</b>	Like the previous one, but it performs the output on a file
<b>-print0</b>	Like -print, but without going in the next line
<b>-execdir command</b>	Executes command
<b>-exec command</b>	Secure POSIX version of the previous one. Expands the command by including the path and the name
<b>-delete</b>	Deletes the founded filename

find directory options **actions**



## find: specify actions

- ❖ The execution of a command is performed by means of the option **exec** (or **execdir**)

- **Format**

- `find directory options -exec command '{}' ';'`
- `find directory options -exec command \{} \;`

- **Where**

- The **command** is executed in the directory
  - in which the pathname has been found by using **execdir**
  - in the current directory if **exec** has been used
- **find** substitutes string `' {} '` (or `\{ }`) with the current pathname of the list
- String `' ; '` (or `\;`) terminates the command executed by **find**

```
find directory options actions
```

## Examples: find & actions

```
find . -name "*.c" -print
```

**-print** is the default action

```
find / -type f -print0  
find / -type l -print0
```

Searches all the regular files or the symbolic links

```
find . -name "*.c" -print -quit
```

Prints the first entry that has a match and then it exits (**-print** has to be inserted)

Cancels all the matching files (both commands are equivalent)

```
find . -name "*.old" -type f -exec rm -f \{} \;  
find . -name "*.old" -type f -exec rm -f '{}' ';' ;
```

## Examples: find & actions

```
find / -user root -exec cat \{} \;
```

Outputs the content of each file of the root filesystem belonging to user `root`, and concatenates their content

```
find / -user root -exec cat '{}' >> file.txt ';' 
```

Like the previous one, but the concatenation is redirected in the file `file.txt`

```
find . -name "*.txt" -exec head -n 2 \{} \;
```

Outputs the first two lines of each `.txt` file

## Examples: find & actions

```
find /home/usr/ \  
-mindepth 2 -maxdepth 2 \  
-name "*.exe" \  
-type f \  
-exec chmod +x {} \;
```

Adds the execution permissions of the ".exe" files in the second level directories of "/home/usr/"

## Exercise

- ❖ Report the UNIX commands to do what indicated, possibly using redirections and pipes
  - Search how many files with extension ".txt" are present in current working directory
  - Display how many lines are present in all founded files with extension ".txt"



## Exercise

- ❖ Report the UNIX commands to do what indicated, possibly using redirections and pipes
  - Search how many files with extension ".txt" are present in current working directory
  - Display how many lines are present in all files with extension ".txt"

```
find . -name "*.txt" | wc
```

```
find . -name "*.txt" -exec wc \{} \;
```

Pipe: applies wc (word count) **to the list** of files found by find (list of all files with extension ".txt")

-exec: applies wc (word count) **to the content of each file** (\{}) found by find