

```
#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 )
```



L'ambiente UNIX/Linux

Espressioni regolari e comando find

Stefano Quer

Dipartimento di Automatica e Informatica

Politecnico di Torino

Espressioni regolari

- ❖ Nate nel 1956 a opera del matematico Stephen Cole Kleene nel dominio degli automi e dei linguaggi formali
- ❖ Utilizzate ampiamente a partire dagli anni '70 in ambienti UNIX per
 - Ambienti di editing (vi, emacs, etc.)
 - Comandi di shell (find, grep, etc.)
 - Linguaggi di scripting (SED, AWK, Perl, Python, etc.)

Espressioni regolari

- ❖ Standardizzate da POSIX nel 1992
- ❖ Ne esistono diverse versioni, con formalismi distinti, simili ma **non** identici
 - BRE, Basic Regular Expression
 - ERE, Extended Regular Expression
 - PCRE, Perl Compatible Regular Expression
 - Libreria C di Regular Expression (Hazel, 1997)
 - Molto più flessibili e potenti della versione POSIX
 - Diventato uno standard de-facto con Perl 5
- ❖ Nel corso di SO
 - Ne faremo un uso superficiale con find, grep, scripting di shell, etc.

Espressioni regolari

- ❖ Una espressione regolare (o **pattern**) è una espressione utilizzata per specificare un insieme di stringhe
 - Operatori compatti sono utilizzati per rappresentare sequenze complesse
 - Esempi
 - $a \mid b^*$ indica l'insieme delle stringhe $\{a, \phi, b, bb, bbb, bbbb, \dots\}$
 - $\backslash d \backslash . \backslash d$ i numeri decimali con una sola cifra intera ($\backslash d$), il punto decimale ($\backslash .$) e una sola cifra decimale ($\backslash d$), ovvero $\{0.0, 0.1, 0.2, \dots, 9.8, 9.9\}$

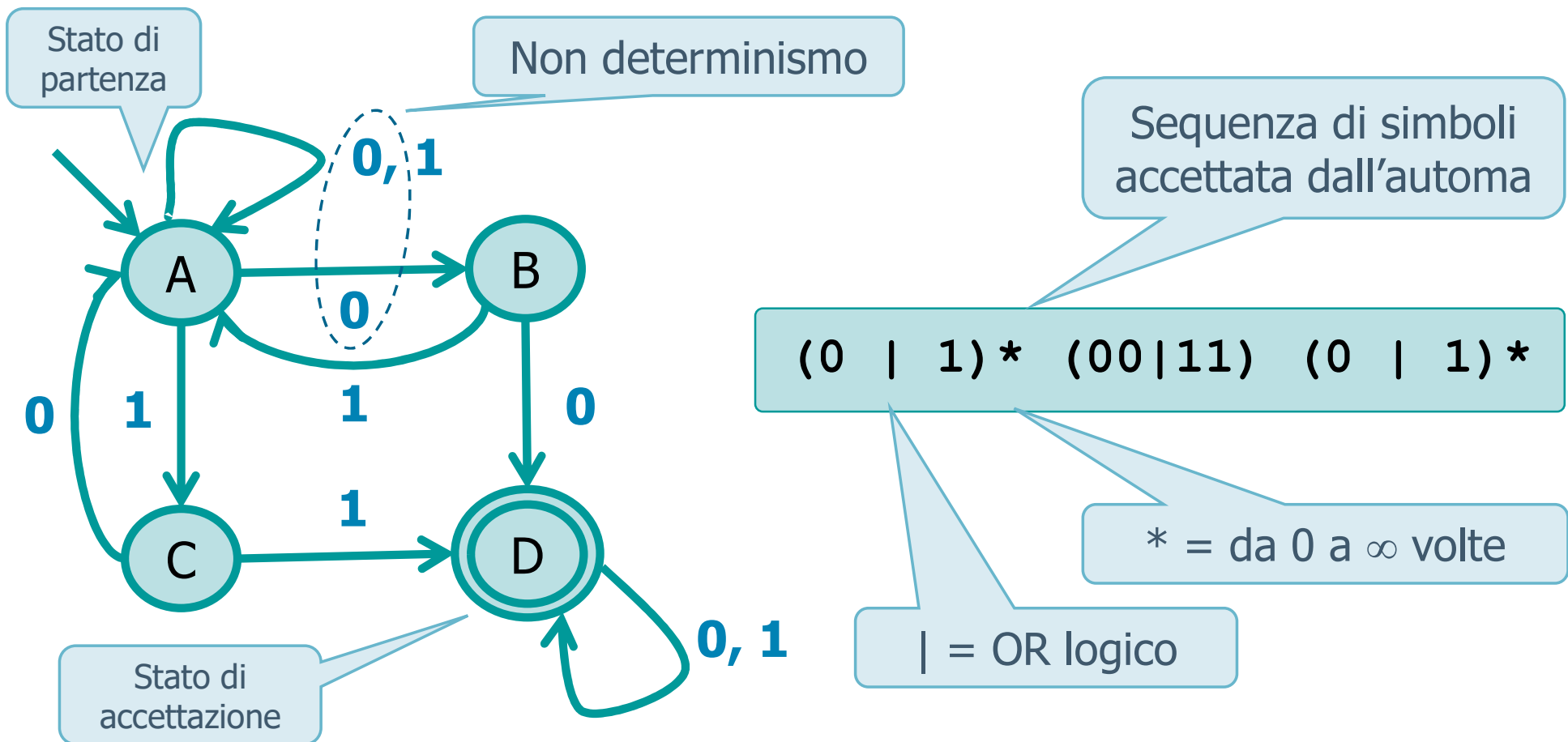
Espressioni regolari

- ❖ Le espressioni sono utilizzate per effettuare l'accoppiamento (**match**) tra oggetti
 - Nomi di direttori, nomi di file, righe o campi di file, stringhe o sotto-stringhe, etc.
- ❖ In generale quindi
 - Occorre capire quali caratteristiche identificano univocamente gli oggetti che si desidera manipolare
 - Esperimentare tali caratteristiche con una espressione regolare

Espressioni regolari e automi

❖ Una espressione regolare corrisponde a un Automa Non Deterministico

➤ NFA (Nondeterministic Finite Automata)



Definizioni base

❖ Letterale

- Qualsiasi carattere (o sequenza di caratteri) utilizzato nella ricerca del match
 - ind in **w**indows, **ind**ifferent, etc.

La potenza della RE è nascosta nell'utilizzo dei metacaratteri

❖ Metacarattere

- Uno o più caratteri con significato speciale
 - * indica da 0 a ∞ simboli precedenti, e.g., $b^* = \{\phi, b, bb, bbb, \dots\}$

❖ Sequenza di escape

- Metodo per indicare che un metacarattere deve essere utilizzato come letterale
 - Il carattere '.' si indica con '\.'

'\n' = new-line
(n non è un metacarattere)

Metacaratteri

Operatore	Significato
[...]	Specifica un elenco o un intervallo di simboli
(...)	Gestisce la precedenza tra operatori Raggruppa insieme di simboli in sottoespressioni Permette riferimenti a espressioni precedenti (backward reference)
	Effettua l'OR tra espressioni regolari

Basic RE: \[...\] e \(...\)

Ancore	Significato	Caratteri speciali	Significato
\<	Inizio parola	\+ \? \. *	Char '+', '?', '.', '*'
\>	Fine parola	\n	New line
^	Inizio riga	\t	Tabulazione
\$	Fine riga		

Metacaratteri

Quantificatori e Intervalli	Significato
*	Elemento presente $[0, \infty]$ volte
+	Elemento presente $[1, \infty]$
?	Elemento presente $[0, 1]$ volte
$[c_1c_2c_3]$	Uno qualsiasi dei caratteri in parentesi
$[c_1-c_2]$	Uno qualsiasi dei caratteri nel range
$[^c_1-c_2]$	Uno qualsiasi dei caratteri non nel range
$\{n\}$	Elemento presente esattamente n volte
$\{n_1, n_2\}$	Elemento presente da n_1 a n_2 volte

Sovra-insieme

Comando grep:
ammette anche le versioni $\{n_1,\}$ ovvero $\{,n_2\}$

Metacaratteri

Caratteri	Significato
c	Un qualsiasi simbolo c (Tranne quelli utilizzati a scopi speciali)
.	Un carattere qualsiasi (non '\n')
\s	Uno spazio o una tabulazione
\d	Una cifra 0-9
\D	Non una cifra
\w	Qualsiasi carattere tra 0-9, A-Z, a-z
\W	Qualsiasi carattere non in 0-9, A-Z, a-z

Sovra-insieme
(non tutti sono disponibili
per tutti i comandi)

Esempi

RE	Significato
ABCDEF	La stringa "ABCDEF"
a*b	Un qualunque numero di a seguite da una b
ab?	Solo a oppure ab
a{5,15}	Da 5 a 15 ripetizioni della lettera "a"
(fred){3,9}	Da 3 a 9 ripetizioni della stringa "fred"
.+	Qualsiasi sequenza non vuota (una riga)
myfunc.*(.*)	Una funzione il cui nome inizia per "myfunc"
^ABC.*	Una riga che inizia con "ABC"
.*h\$	Una riga che finisce con "h"
hello\>	Una parola che termina con "hello"
a+b+	Una o più a seguite da una o più b

Esempi

RE	Significato
[a-zA-Z0-9]	Una lettera o una cifra
A b	A oppure b
\w{8}	Sequenza di 8 caratteri alfabetici o numerici (minuscoli o maiuscoli)
((4\.[0-2]) (2\.[1-3]))	Numeri 4.0, 4.1, 4.2 oppure 2.1, 2.2, 2.3
(.)\1	Due caratteri identici
(.)(.).\2\1	Qualsiasi stringa palindroma di 5 caratteri (e.g., radar, civic, 12321, etc.)

\1, \2
Backward Reference

Basic RE: \(.\)\1 e \(.\)\(.\).\2\1
Extended RE: (.)\1 e (.)\1 e (.)\1

Esercizio

❖ Scrivere una espressione regolare per rintracciare

➤ Una qualsiasi data con formato gg/mm/aaaa

- Giorno e mese possono essere espressi su 1 o 2 cifre

```
\d{1,2}\/\d{1,2}\/\d{4}
```

➤ Tutte le righe che contengono un solo numero intero incluso tra 1 e 50 (inclusi)

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

Il comando find

❖ Permette di

- Ricercare file, direttori o link che soddisfano (match) un particolare criterio, creandone un elenco
- Se necessario, eseguire sugli oggetti dell'elenco dei comandi di shell

❖ Osservazione

- Il comando ritorna il path (relativo) degli oggetti rintracciati non (solamente) il loro nome
- Questo è importante per la scrittura delle espressioni regolari di ricerca e delle azioni da effettuare

Il comando find

```
find direttorio [opzioni] [azioni]
```

- ❖ Il formato del comando risulta relativamente complesso
- ❖ Sostanzialmente
 - Visita tutto l'albero a partire dal direttorio **directory**
 - Crea l'elenco che soddisfa le **opzioni**
 - Eventualmente effettua per ogni file le **azioni** specificate
- ❖ Occorre analizzare come specificare directory, options e actions

Specifica del direttorio

❖ Direttorio

- Specifica l'albero di direttori in cui eseguire il comando
- Esempi
 - .
 - /usr/bin
 - ./subDirA/subDirB

```
find directorio [opzioni] [azioni]
```

Specifica delle opzioni

❖ Opzioni

Opzione	Significato
-name pattern	Match con il nome del file. Il path iniziale è rimosso. In alcune versioni è possibile racchiudere il pattern tra doppi apici per specificare espressioni regolari. -iname è identica ma case insensitive.
-path pattern	Come il precedente ma specifica path + nome -ipath è identico ma case insensitive.

```
find directorio [opzioni] [azioni]
```

Specifica delle opzioni

Opzione	Significato
-regex expr	Specifica una espressione regolare che deve avere un match con il path (completo) rintracciato. -iregex è identica ma case sensitive.
-regextype type	Indica il tipo di RE utilizzate, ovvero: posix-basic, posix-egrep, posix-extended, etc. Occorre specificare il tipo prima della RE (regextype va inserito prima di regex).
-atime [+,-]n -ctime [+,-]n -mtime [+,-]n	Ultimo access, status o modification time. n=1 specifica da 0 a 24 ore fa. Il valore di n può essere inserito con segno: + indica \leq , - indica \geq

```
find directorio [opzioni] [azioni]
```

Specifica delle opzioni

Opzione	Significato
-size [+,-]n[bckwMG]	<p>Dimensione del file. Il segno + indica \geq, quello - indica \leq. Il carattere successivo indica l'unità di misura:</p> <ul style="list-style-type: none">• b blocchi (di 512 byte)• c byte• k kByte• w word (2 byte)• M Mbyte• G GByte
-type tipo	<p>Tipo di file. Il tipo può essere: f per file regolari (i.e., file di testo, eseguibili, etc.), p per pipe, l per symbolic link, s per socket, d per direttori</p>

```
find directorio [opzioni] [azioni]
```

Specifica delle opzioni

Opzione	Significato
-user nome -group nome	Definizione del proprietario del file, ovvero identificativo del proprietario (user) oppure del gruppo (group)
-readable -writable -executable	Modalità di accesso, ovvero l'oggetto deve essere leggibile, scrivibile, eseguibile
-mindepth n -maxdepth n -quit	Sezione dell'albero in cui effettuare la ricerca: mindepth indica la profondità minima per la ricerca (nell'albero di direttori) e maxdepth quella massima. Con quit esce dalla ricerca dopo il primo match.

```
find directorio [opzioni] [azioni]
```

Esempi: find & options

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

File c con qualsiasi nome
(uso wildchar dos)

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

Errato: *.c non è una RE

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

RE equivalente

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

Tutti i file con nome uguale
ad a oppure A e hanno una
estensione qualsiasi

Tutti i file di dimensione
>500 Byte

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

Tutti i file leggibili nel
direttorio corrente (./) con
nome che inizia per ab, aab,
aaab e qualsiasi estensione

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

Esempi: find & options

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

Tutti i file con nome costituito da due caratteri iniziali, sequenza palindroma di 5 caratteri e altri caratteri in numero indefinito

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

Idem ma con RE standard

Tutti i file di estensione exe nel direttorio /home/usr dal livello 2 al 4 (inclusi)

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


Specifica delle azioni

❖ Azioni

➤ L'azione di default del comando **find** è la stampa

- L'azione di default equivale al comando **print**

```
find directorio [opzioni] -print
```

➤ È però possibile eseguire

- Un qualsiasi comando di shell
- Su ciascuno degli oggetti appartenenti all'elenco restituito

```
find directorio [opzioni] [azioni]
```

Specifica delle azioni

Azione	Significato
-print	Azione di default. Stampa un nome per ciascuna riga
-fprint	Come il precedente ma effettua l'output su file
-print0	Come -print ma non va a capo
-execdir command	Esegue il comando
-exec command	Versione sicura POSIX dell'azione precedente. Espande il comando includendo il path e il nome.
-delete	Elimina l'oggetto rintracciato

Specifica delle azioni

- L'esecuzione di un comando si effettua con l'opzione **exec** (o **execdir**)

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

Formati
equivalenti

- Dove
 - Il **comando** viene eseguito nel direttorio
 - In cui la entry è stata rintracciata con **execdir**
 - In cui si esegue la find con **exec**
 - Find sostituisce la stringa '{}' (\{}) con il file corrente dell'elenco
 - La stringa ';' (\;) termina il comando eseguito dalla find

Esempi: find & actions

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

-print è l'azione di default

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

Visualizza tutti i file
regolari o i symbolic link

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

Visualizza la prima entry che
ha un match e poi esce
(**-print** deve essere inserito)

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

Cancellazione di file
(comandi equivalenti)

Esempi: find & actions

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

Visualizza tutti i file elencati concatenandoli

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

Come il precedente ma la concatenazione viene ridiretta nel file file.txt

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

Visualizza le prime due righe di tutti i file elencati

Esempi: find & actions

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

Modifica tutti i permessi dei file di estensione "exe" contenuti nel secondo livello gerarchico di direttori a partire da "/home/usr/" aggiungendo il permesso di esecuzione

Esercizio

- ❖ Si riportino i comandi UNIX per effettuare quanto indicato, utilizzando eventuali ridirezioni e pipe
 - Visualizzare quanti file di estensione ".txt" sono presenti nell'albero con radice la working directory
 - Visualizzare quante righe sono presenti in tutti i file di estensione ".txt"
 - Visualizzare il numero di righe totali presenti in tutti i file di estensione ".txt"

Soluzione

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

La pipe applica wc (word count) **all'elenco** di tutti i file con estensione .txt. Fornisce il numero di file .txt (e altro)

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

La exec applica wc (word count) al **contenuto** di tutti i file (\{}) rintracciati. Fornisce numero di righe (e altro) in ogni file .txt

```
find . -name "*.txt" -exec cat \{} >> file.txt \; | wc
```

Come il precedente ma prima concatena i file poi esegue un unico wc (word count) su tale file