Come si cerca un pattern con libpcre

libpcre è una libreria che implementa la ricerca di pattern in un testo. I pattern da cercare sono descritti usando **espressioni regolari**, e che vengono trasformate da **libpcre** in un automa a stati finiti che scandisce il testo da riconoscere per trovare il punto in cui il pattern corrisponde al testo. Il pattern può contenere dei sub-patterns racchiusi tra parentesi tonde che, una volta individuati, possono essere estratti dal testo separatamente (quindi le espressioni regolari possono essere molto utili ad esempio per implementare un semplice parser).
I passi da seguire per cercare un pattern nel testo sono:

1. il pattern viene "compilato" ovvero trasformato nel corrispondente automa a stati finiti, (usate **pcre_compile**)
2. il pattern compilato viene usato per cercare la sua prima occorrenza nel testo (usate **pcre_exec**)
3. se il pattern è stato trovato avrete nel vettore ***ovector*** i dati che indicano dove è il match (indici 0 e 1) e dove sono i match dei sub-patterns (indici successivi)
4. a questo punto è possibile estrarre dal testo le parti corrispondenti ai subpatterns con una delle funzioni **pcre_get_substring** oppure **pcre_copy_substring**

I prototipi delle funzioni di **libpcre** di cui sopra sono:

pcre *pcre_compile(const char *_pattern_, int _options_, const char **_errptr_, int *_erroffset_, const unsigned char *_tableptr_);

Trasforma un pattern nel corrispondente automa a stati finiti, codificato nella struttura di tipo pcre che viene allocata e tornata da questa funzione.

- ***pattern*** A zero-terminated string containing the regular expression to be compiled
- ***options*** Zero or more option bits
- ***errptr*** Where to put an error message
- ***erroffset*** Offset in pattern where error was found
- ***tableptr*** Pointer to character tables, or NULL to use the built-in default

NOTA: Alla fine dovete ricordarvi di deallocare la struttura dati **pcre** che viene tornata da questa funzione.

int pcre_exec(const pcre *_code_, const pcre_extra *_extra_, const char *_subject_, int _length_, int _startoffset_, int _options_, int *_ovector_, int _ovecsize_);

This function matches a compiled regular expression against a given subject string, and returns offsets to capturing subexpressions. Its arguments are:

- ***code*** Points to the compiled pattern (ottenuto da pcre_compile)
- ***extra*** Points to an associated pcre_extra structure, or is NULL
- ***subject*** Points to the subject string (il testo in cui cercare)
- ***length*** Length of the subject string, in bytes
- ***startoffset*** Offset in bytes in the subject at which to start matching
- ***options*** Option bits
- ***ovector*** Points to a vector of ints for result offsets
- ***ovecsize*** Number of elements in the vector (a multiple of 3)

NOTA: Dovete allocare voi il vettore ***ovector***, in cui la funzione inserisce gli offset nel testo dei subpatterns trovati. Deve avere lunghezza **<u>multiplo di 3</u>**, pari almeno 3 volte il numero di subpatterns (più un pò per gli usi interni della libreria).
NOTA: il vettore ***ovector*** contiene coppie di valori che indicano l'offset di inizio e di fine+1 (ovvero il carattere <u>DOPO</u>) di ciascuna stringa trovata che corrisponde al pattern ed ai subpatterns.
Esempio:

- alle posizioni 0 ed 1 ci sono gli offset di inizio e fine+1 per il pattern intero
- alle posizioni 2 e 3 ci sono gli offset di inizio e fine+1 per il primo sub-pattern
- alle posizioni 4 e 5 ci sono gli offset di inizio e fine+1 per il secondo sub-pattern

  ...

int pcre_get_substring(const char ****subject*, int ****ovector***, int** *stringcount***, int** *stringnumber***, const char *****stringptr***);**

This is a convenience function for extracting a captured substring. The arguments are:

- ***subject*** Subject that has been successfully matched
- ***ovector*** Offset vector that pcre_exec() used
- ***stringcount*** Value returned by pcre_exec()
- ***stringnumber*** Number of the required substring
- ***stringptr*** Where to put the string pointer

Per estrarre la sottostringa trovata che corrispondono ad un certo subpattern potete usare questa funzione.
NOTA: la funzione alloca la stringa, quindi dovete ricordarvi di deallocarla

Se preferite allocare voi il buffer usate la funzione

int pcre_copy_substring(const char ****subject*, int ****ovector***, int** *stringcount***, int** *stringnumber***, char ****buffer***, int** *buffersize***);**

This is a convenience function for extracting a captured substring into a given buffer. The arguments are:

- ***subject*** Subject that has been successfully matched
- ***ovector*** Offset vector that pcre_exec() used
- ***stringcount*** Value returned by pcre_exec()
- ***stringnumber*** Number of the required substring
- ***buffer*** Buffer to receive the string
- ***buffersize*** Size of buffer

NOTA: per sapere di che dimensioni è la stringa corrispondente al pattern N potete esaminare il vettore ovector, dove troverete (la stringa che corrisponde all'intero pattern corrisponde ad N=0):

- in ovector[2N]: l'offeset del primo carattere che corrisponde al sub-pattern
- in ovector[2N+1]: l'offeset del carattere **che segue** il sub-pattern

Per altre informazioni ed esempi di come le funzioni vanno chiamate eseguite (in linux o cygwin):

- **man pcreapi**

Il manuale PDF allegato alla versione per windows della libreria fa abbastanza schifo (mancano pagine) e lo sconsiglio.