PERL

Dott.ssa Denise Angilica
Corso di Sistemi Operativi
Aggiornato all'A.A. 2023-24

Primi passi

- Prima riga dello script: individuare la posizione dell'interprete
  - #!/usr/bin/perl
- Eseguire lo script:
  - perl script.pl
  - chmod u+x script.pl; ./script.pl
- Maggiori dettagli su eventuali errori:
  - perl -w script.pl
- Avviare lo script in modalità debug:
  - perl -d script.pl
- Fatti aiutare!
  - use strict; use warnings; #all'interno dello script

Variabili

- Non tipizzate
  - scalari, $variabile
    - stringhe
    - valori numerici
  - array, @variabile
    - contengono una lista di scalari
- Valore di default
  - undef
- Variabili speciali
  - $_
  - @ARGV

Array

- Per accedere un elemento si usa []
  - $array[0]: attenzione al $! Il singolo elemento di un array è uno scalare!
  - $array[-1] restituisce l'ultimo elemento dell'array, -2 il penultimo ecc.
  - è possibile accedere a "pezzi" di array: @array[1,3], @array[1..5], @array[@indici]
    - NOTA la @ invece di $!
  - push/pop aggiunge/rimuove elementi alla fine di un array
    - push @array, $valore; push @array1,@array2;
    - $ultimo_elemento = pop @array;
  - unshift/shift aggiunge/rimuove all'inizio di un array;
    - unshift @array,$valore; unshift @array1,@array2;
    - $primo_elemento = shift @array;
  - per stampare in modo decente un array si può usare la funzione "join"
    - print join("\n",@array);

Funzioni utili

- join
  - $str = join ‘c’,@array; restituisce una stringa ottenuta concatenando gli elementi di @array con il caratter ‘c’;
    - $str = join ‘.’,(127,0,0,1); $str contiene la stringa “127.0.0.1”
- split
  - @ arr = split ‘c’,$str; spezza la stringa li dove è presente il carattere ‘c’
    - @arr= split ‘.’ , ’127.0.0.1’; @arr conterrà (127,0,0,1)
- chomp
  - rimuove lo “\n” a fine stringa
    - $res = chomp “denise\n”; $res contiene “denise”
    - $res = chomp “denise”; $res contiene “denise”
    - $res = chomp; $res contiene il valore contenuto in $_ senza “\n”
    - ATTENZIONE: “denise\n” **ne** “denise”!!

Contesto (cont.)

- Operatore booleano ternario
  - $val = cond ? se_vero : se_falso;
    - equivale a if(cond) {$val=se_vero;} else {$val=se_falso}
- Operatore di ripetizione (x)
  - @array = (‘ciccio’) x 10; @array conterrà 10 elementi il cui valore è ‘ciccio’
  - $str = ‘ciccio’ x 10; $str conterrà ‘cicciociccioci
- Nei cicli:
  - next; passa all’iterazione successiva
  - last; termina il ciclo

Contesto

- Le variabili assumono automaticamente un comportamento diverso in base al contesto in cui si trovano
  - print @a; stampa il contenuto dell'array
  - print "La lunghezza dell'array è: " . @a ."\n"; stampa la lunghezza dell'array (viene interpretato come scalare)
  - $num =12332; $str= "ciao" . $num; $num viene interpretato come stringa
- I valori numerici si confrontano con i classici operatori numerici
  - ==, !=, <, >, <=,>=, <=>
- Le stringhe si confrontano con operatori dedicati
  - eq, ne, gt, lt, ge, le, cmp
- Operatori booleani
  - &&, ||
    - valutano il valore a destra solo se quello a sinistra è vero per && e falso per ||
  - and, or
    - valutano SEMPRE sia il valore a sinistra che quello a destra

Espressioni regolari

- Servono a
  - controllare pattern all'interno di stringhe
  - recuperare pezzi di stringhe
  - sostituire pezzi di stringhe
- Racchiuse tra //
- $str =~ /expr/ non è un assegnamento, confronta la stringa $str con l'espressione regolare /expr/
  - cerca un match
    - $str='ciao ciccio'. L'espressione regolare /ci/ fa match solo con 'ci' di 'ciao'
  - /expr/g cerca tutti i match
    - $str='ciao ciccio'. L'espressione regolare /ci/g fa match con tutte le occorrenze di 'ci' in $str
  - /expr/**i** rende l'espressione regolare expr case insensitive
  - **senza $str =~ l'espressione regolare viene applicata a $_**

Espressioni regolari: metacaratteri

\	Annulla gli effetti del metacarattere successivo
^	Identifica l'inizio di una riga; inoltre all'inizio di un gruppo nega il gruppo stesso
.	Qualsiasi carattere ad eccezione di quelli che identificano una riga nuova
$	Identifica la fine di una riga
|	Indica una condizione OR
()	Indicano un gruppo di caratteri
[]	Indicano intervalli e classi di caratteri
\t	tab (HT, TAB)
\n	newline (LF, NL)
\r	return (CR)

Espressioni regolari: metacaratteri (cont.)

\d	Ricerca un numero (d sta per digit).
\D	Opposto di \d, ricerca qualsiasi cosa che non sia un numero.
\w	Ricerca un carattere "parola" (w sta per word) ovvero lettere, numeri e "_" -> [a-zA-Z0-9_]
\W	Ricerca un carattere che non sia \w, ovvero tutto quello che non sia lettere, numeri o "_"
\s	Ricerca uno spazio, comprese tabulazioni e caratteri di fine riga.
\S	Opposto di \s. Ricerca qualsiasi cosa che non sia uno spazio, una tabulazione o dei caratteri di fine riga
\N	Ricerca un carattere che non sia newline

Espressioni regolari: quantificatori

*	Indica 0 o più occorrenze
+	Indica 1 o più occorrenze
?	Indica al massimo 1 occorrenza
{n}	Ricerca esattamente n occorrenze
{n,}	Ricerca minimo n occorrenze
{n,m}	Ricerca minimo n e massimo m occorrenze

Espressioni regolari: catturare i match

- Variabili speciali ci aiutano a catturare le porzioni di stringa che fanno match con la regex
  - la regex deve fare uso di ()
    - uso (?:) se non voglio catturare il match ma ho bisogno di racchiudere la porzione di stringa tra parentesi
  - $1-$9 cattura i primi 9 gruppi