Breu introducció a Perl

1. El primer *script* en Perl

```
#!/usr/bin/perl
#
# Programa que fa allò que sembla
#
print "Hello World!\n";   # Escriu un missatge
```

- **La primera linia**: Indica quin és l'intèrpret de comandes que volem fer servir per interpretar el *shellscript*. En aquest cas "/usr/bin/perl" és l'intèrpret de Perl.
- **Comentaris**: Els comentaris en Perl són tot allò que segueix el caràcter "#", sigui a començament de línia, o després d'una comanda.
- **Mostrar un missatge per pantalla**: La comanda "print" mostra una cadena per pantalla. En l'exemple, la cadena "Hello World!". Totes les comandes en Perl acaben en un ";".
- **Executar el *shellscript***: Guardeu el *shellscript* en un fitxer, i doneu-li permisos d'execució.

2. Tipus de dades

En Perl no cal declarar les variables prèviament a fer-les servir (excepte les variables locals a una subrutina, secció "Subrutines"). El tipus de la variable es determina segons l'ús que se'n fa. Per evitar confusions, no assigneu el mateix nom a dues variables de tipus diferents.

2.1. Escalars

- Engloba els tipus *string*, enter, i coma flotant.
- S'identifica amb un símbol $ davant del nom de la variable
- Operacions enteres/coma flotant: +, -, *, /, % (mòdul)
- Operacions lògiques: &&, ||, ! (not)
- Operacions sobre *strings*:
  - Concatenació: `$c = $a . $b;`
  - Eliminar el "\n" del final d'un *string*: `chomp $string;`

2.2. Llistes

- Arregles composats per escalars. Indexats a partir del zero.
- No es pot fer una llista de llistes.
- No tots els elements de la llista han de pertànyer al mateix tipus bàsic.
- Podem barrejar *strings*, enters, i coma flotant.
- Operacions:
  - Assignació:
    ```
    @list = (1,2,3,4);
    ```
  - Extracció:
    ```
    $string = $list[2];
    ```
  - Afegir elements:
    ```
    push @list, $elem;
    push @list, $elem1, $elem2, ...;
    push @list, @list2;
    ```

- Obtenir el nombre d'elements
  ```
  $num = @list;
  ```
- Obtenir i eliminar el primer element
  ```
  $string = shift @list;
  ```
- Obtenir i eliminar el darrer element
  ```
  $string = pop @list;
  ```

2.3. Operacions mixtes

- Divisió d'un *string* en una llista:
  ```
  @list = split <divisor>, $string;
  ```
  El caracter divisor no apareix en la llista.
- Unió d'una llista en un *string*
  ```
  $string = join <divisor>, @list;
  ```

3. Estructures condicionals

```
if (<condició>) {
} elsif (<condició>) {
} else {
}
```

- Les claus d'inici i final són obligades, fins i tot si hi ha una sola comanda (no com en C).

- **Comparacions**: Tingueu en compte que no és el mateix comparar dos nombres, que comparar dos cadenes de caràcters qualsevol (*strings*).
  - Comparació numèrica:
    ```
    $a == $b
    ```
  - Desigualtat numèrica
    ```
    $a != $b
    ```
  - Comparació de strings
    ```
    $a eq $b
    ```
  - Desigualtat de strings
    ```
    $a ne $b
    ```

- **Atributs sobre fitxers:** Podem especificar condicions que avaluen condicions sobre un fitxer donat:
  - Cert si el fitxer existeix
    ```
    -e <file>
    ```
  - Cert si el fitxer es pot llegir
    ```
    -r <file>
    ```
  - Cert si el fitxer es pot executar
    ```
    -x <file>
    ```
  - Cert si el fitxer és un fitxer de dades
    ```
    -f <file>
    ```
  - Cert si el fitxer és un directori
    ```
    -d <file>
    ```
  - Cert si el fitxer és un *soft link*

-l <file>

4. Estructures iteratives

- Similars a les usades en C.

```
while (<condició>) { }

for (<init>; <test>; <incr>) { }
```

- S'executa una vegada per cada element de la llista, on $string pren el valor corresponent.

```
foreach $string (@list) { }
```

5. Fitxers

- Obrir un fitxer per a lectura
  open FILE, "filename";
- Obrir un fitxer per a escriptura
  open FILE, ">filename";
- Obrir un fitxer per agregar
  open FILE, ">>filename";
- Detectar errors en obrir un fitxer
  open (FILE, "filename") or die “No es pot obrir el fitxer $!”;
- Llegir una línia del fitxer
  $line = <FILE>; (els angles formen part de la sintaxi)
- Llegir tot el fitxer: Cada element de la llista té una línia del fitxer.
  @lines = <FILE>;
- Escriure una string en un fitxer
  print FILE $string; (no hi ha coma entre FITXER i $string)
  printf FILE <format>, <arg>, ... (similar a la printf de C)
- Si no especifiquem FITXER, escriuen per pantalla.

Exemples:

- Escriure un fitxer per pantalla, línia a línia:

```
open TEXT, "text.txt";
while ($line = <TEXT>) {
        print $line;
}
close TEXT;
```

Una altra forma:

```
open TEXT, "text.txt";
@lines = <FILE>;
close TEXT;
foreach $line (@lines) {
        chomp $line;
        print $line . "\n";
}
```

6. Directoris

- Obrir un directori:
  opendir DIR, "dirname";
- Llegir un directori, fitxer a fitxer:
  $string = readdir DIR;
  Les entrades "." (directori actual) i ".." (directori pare) també es retornen en llegir un directori.
- Llegir tots els fitxers d'un directori:
  @list = readdir DIR;
  Cada element de la llista conté un fitxer.

Exemples:

- Llistar el contingut d'un directori:

```
opendir CURRENT, ".";
while ($file = readdir <CURRENT>) {
        print $file;
}
closedir CURRENT;
```

Llistar sols els fitxers de dades (no directoris):

```
opendir CURRENT, ".";
@files = readdir CURRENT;
closedir CURRENT;
foreach $file (@files) {
        if (-f $file) { print $file . "\n"; }
}
```

7. Expressions regulars

Cerca d'expressions regulars

$string =~ /<expr>/ # Cert si es troba l'expressió en la cadena

Possibles valors per <expr>:

abdc : La cadena "abcd"
[abcd] : Correspon als caracters "a", "b", "c", o "d"
[^jkl] : Qualsevol caracter excepte "j", "k", o "l"
[a-f] : Correspon als caracters de la "a" a la "f"

[0-9] : Correspon als caracters del "0" al "9"

- Metacaracters
  . : Correspon a qualsevol caracter excepte "\n"
  + : Qualsevol cadena d'un o més caracters
  * : Qualsevol cadena de caracters (inclosa la cadena buida)
  ^<expr>: L'expressió ha de figurar al començament de la cadena
  <expr>$: L'expressió ha de figurar al final de la cadena
  ^[ ]*$: Correspon a línies buides, o que sols tenen espais en blanc
  /^[0-9]+/: cadenes numèriques de longitud 1 o mes al principi de la línia
  /[0-9]*$/: cadenes numèriques de longitud 0 o mes al final de la línia
  /^something$/: una línia que comença i termina amb el text “something”
- Clases de caracters
  \d : Correspon a un digit simple
  \D : Correspon a un caracter que no es un digit simple
  \w : Correspon a un caracter de paraula simple (digits, lletres y “_”)
  \W : Correspon a un caracter que no es de paraula simple
  \s : Correspon a un caracter de espai (espai i tabulador)
  \S : Correspon a un caracter que no es un espai
  /\d+\s+/: Correspon a un o mes dígits seguits de u o mes espais buits
- Repeticions
  {N} : Correspon a N repeticions del caracter precedent
  {N,M} : Correspon a mínim N i màxim M repeticions del caracter precedent
  {N, } : Correspon a mínim N del caracter precedent
  /[aeiou]{2}/: una cadena amb dues vocals.
- Subexpressions
  /(\w+)(\d+)/: correspon a una cadena amb u o mes caracteres de paraula seguits de u o mes dígits. Els caracters de paraula es emmagatzemaran a la varaible $1 i els digits a la variable $2. Per exemple: “dma123”, o “d123”, o “dma1”
  /^(\w+)\s*/ : una cadena que comença amb u o mes caracters de paraula seguits de cero mes espais, per exemple “hola”, o “hola “, o “hola “
  /\((\d+):(\d+)\)$/: un cadena amb u o mes dígits separats per dues punts seguits de u o mes dígits, al final de la línia, tot inclòs en parèntesi. Per exemple (123:456)
- Substitució d'expressions regulars

  $string =~ s/<cerca>/<subs>/g

  - Substitueix totes les aparicions de <cerca> per <subs>.
  - Sense la "g" final, sols substitueix la primera aparició.
- Exemple complet: processament de la sortida de la comanda **last.** La variable $1 conté el nom d'usuari, la variable $2 les hores de connexió i la variable $3 els minuts.

```
#!/usr/bin/perl
@lastlog = `last`;
foreach $line (@lastlog ) {
  if ( $line =~ /^(\w+)\s*.*\((\d+):(\d+)\)\s*$/ ) {
    $hours{$1}  += $2;
    $minutes{$1} += $3;
    $logins{$1}++;
  }
}
foreach $user (sort (keys %hours)) {
  $minutes{$user} += int ( $hours{$user}* 60) ;
  print "User $user, total login time ";
  printf "%02d, ",  $minutes{$user};
  print "total logins $logins{$user}.\n";
}
```

d'usuari, la variable $2 les hores de connexió i la variable $3 els minuts.

8. Paràmetres

- Tot *script* en Perl rep el seus paràmetres en la llista @ARGV.
- El nombre d'arguments podem trobar-lo fent `$ARGC = @ARGV;`

9. Subrutines

- Definició de subrutines

  `sub <routine name> { }`
  - No hi ha cap declaració de paràmetres.
  - Les rutines reben tots els paràmetres en la llista @_, per tant són accessibles com $_[0], $_[1], $_[2], ...
- Crida a subrutines

  `&<routine name> (params);`
- Com no hi ha declaració de paràmetres, no hi ha comprovació de paràmetres.

10. Variables locals

```
my $var;
my $var = $value;
my @list;
```

Si una variable no es declara local, serà global per defecte.

11. Execució de comandes

- Execució de comandes de *shell*

  system <command>;

- Executar comandes en el shell, i capturar el resultat de pantalla en la variable $string;

  $string = `<command>`;

- Accés a variables d'entorn

  $ENV{'<variable>'}

Exemple:

```
print "Adding: ".$ENV{'HOME'}."/bin al PATH";
```