—— Perl —— — Cours 5/6 —

Outils Perl pour le système





Sylvain Lhullier contact@formation-perl.fr https://formation-perl.fr/

La reproduction et/ou diffusion de ce document, même partielle, quel que soit le support, numérique ou non, est strictement interdite sans autorisation écrite des ayants droit.

— Perl5/6: outils Perl pour le système —



3

Bonnes pratiques en Perl

Toujours coder comme si le gars qui va finir par maintenir votre code est un psychopathe violent qui sait où vous habitez. – Rick Osborne

Des idées, des pistes. Pas des règles immuables.

Référence : Damian Conway - O'Reilly

Perl best practices / De l'art de programmer en Perl

Quick reference guide (2 pages):

http://www.squirrel.nl/pub/PBP_refguide-1.02.00.pdf

— Perl 5/6 : outils Perl pour le système —



Plan du cours

Bonnes pratiques Courriels

Appels système * réception

* système de fichiers

Formats mémoire et binaires

Gestion des paramètres

* envoi

Gestion du temps Filtres courts avec Perl

Sockets Lancer une commande

Threads Fonction eval

© Sylvain Lhullier

Diffusion et reproduction interdites

)<u>ĭ</u>(

2

— Perl5/6: outils Perl pour le système —



- Ouvrez les accolades et les parenthèses en fin de ligne
- Mettez un point-virgule après chaque instruction
- Mettez une virgule après chaque entrée de liste
- Codez en 80 caractères de large
- Indentez avec 3 ou 4 espaces, pas de tabulation
- $\bullet\,$ Ne mettez pas 2 instructions sur la même ligne
- Alignez verticalement les éléments correspondants
- Coupez les longues instructions avant un opérateur

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites







Bonnes pratiques : Conventions de nommage

- Nommez les tableaux au pluriel, les hash au singulier
- Nommez les variables stockant des références avec le suffixe _ref
- Utilisez l'underscore _ pour séparer les mots
- Abrégez les noms lors que ceux-ci sont totalement non ambigus
- Préfixez avec l'underscore _ les fonctions et variables à usage interne

© Sylvain Lhullier

Diffusion et reproduction interdites

5

Bonnes pratiques: Valeurs et expressions

- Utilisez " uniquement pour interpoler
- Utilisez l'underscore pour les nombres longs 1_000_000
- Gardez la double-flèche => pour les couples clef-valeur
- N'utilisez pas la virgule pour séparer les instructions
- N'utilisez pas les références symboliques

© Sylvain Lhullier

Diffusion et reproduction interdites

_

— Perl 5/6 : outils Perl pour le système —



7

Bonnes pratiques: Variables

- Utilisez des noms lexicaux (mots lisibles)
- N'utilisez ni mots-clefs ni fonction du langage comme nom
- Ne modifiez pas \$_
- Utilisez les indices négatifs pour accéder à la fin d'un tableau
- Si vous devez modifier une variable ponctuation, localisez la { local \$/ = undef; \$content = <\$fd>; }
- Si vous devez modifier une variable d'un module, localisez la

— Perl5/6: outils Perl pour le système —



6

) (

Bonnes pratiques : Structures de contrôle

- N'utilisez ni unless ni until
- N'utilisez pas les if for while postfix
- Utilisez une variable déclarée my comme variable de boucle for
- Utilisez foreach pour les parcours et map pour les transformations
- Utilisez for et redo plutôt qu'un while irrégulièrement compté

© Sylvain Lhullier Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites





))(

Bonnes pratiques: Documentation

- Distinguez doc utilisateur et technique
- Utilisez POD dans le code pour la doc utilisateur
- Décrivez vos algorithmes
- Documentez tous ce sur quoi vous avez passé du temps
- Imaginez devoir ré-écrire si vous ne commentez pas

© Sylvain Lhullier

Diffusion et reproduction interdites

9

Bonnes pratiques: Fonctions du langage

- Utilisez reverse pour inverser une liste
- N'utilisez pas eval sur une chaîne
- Utilisez glob plutôt que <...>
- Utilisez un block d'instruction avec map grep sort
- Ne modifiez par les valeurs dans un sort
- Découpez avec unpack si taille fixe
- Découpez avec split si séparateur simple
- Découpez avec Text::CSV_XS si séparateur complexe

© Sylvain Lhullier

Diffusion et reproduction interdites

10

— Perl5/6: outils Perl pour le système —



Bonnes pratiques: Vos fonctions

- N'appelez pas les fonctions avec leur sigil &
- N'utilisez ni mots-clefs ni fonction du langage comme nom
- Recopiez @_ en première instruction
- N'utilisez ni @_ ni \$_[..] ensuite
- Utilisez une hash de paramètres si la fonction a beaucoup de paramètres
- Testez defined ou exists sur les paramètres
- N'utilisez pas les prototypes de Perl 5
- Renvoyez vos valeur avec un return explicite

— Perl5/6: outils Perl pour le système —



12

Bonnes pratiques : hash de paramètres

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites



Bonnes pratiques : Entrées/sorties

- N'utilisez ni mots-clefs ni fonction du langage comme nom de descripteur
- Utilisez open avec 3 arguments et non 2
- Fermez les descripteurs explicitement au plus tôt
- Testez vos open print close
- Utilisez while(<>) et non for(<>)
- Pour print, mettez entre accolades le descripteur print {\$fd} ''

© Sylvain Lhullier

Diffusion et reproduction interdites

13

Bonnes pratiques : Modules

• Imaginez l'interface du module avant de coder

• Une variable ne doit pas faire partie de l'interface du module

• Code dupliqué ⇒ fonction / fonction dupliquée ⇒ module

• Indiquez un numéro de version dans vos modules

• Exportez judicieusement et si possible sur demande

• Ne ré-inventez pas la roue, utilisez CPAN



— Perl5/6: outils Perl pour le système —



15

Bonnes pratiques : Objet

- Ne faite pas de l'objet par défaut, ce doit être un choix
- Encapsulez totalement vos objets
- Utilisez toujours le même nom pour le constructeur
- Appliquez aux méthodes les règles des fonctions
- Écrivez des accesseurs
- Utilisez la surcharge d'opérateur avec parcimonie
- N'utilisez pas AUTOLOAD

© Sylvain Lhullier

— Perl5/6: outils Perl pour le système —

Diffusion et reproduction interdites



14

Bonnes pratiques : Tests et débogage

- Utilisez toujours use strict;
- Utilisez toujours use warnings;
- Ne croyez pas qu'une compilation sans erreur donne un programme correct
- Écrivez les tests avant la fonction (TDD test driven development)
- Faites vos tests avec Test::More
- Faites vos suites de tests avec Test::Harness (harnachement attelage)

© Sylvain Lhullier Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites



Bonnes pratiques : Divers

- Utilisez un système de gestion de version
- Utilisez une syntaxe de configuration simple
- Ne tentez pas d'optimiser sans mesurer
- Pensez aux caches (pensez à en mesurer l'efficacité)

Critiques de votre code :

- module Perl::Critic
- commande perlcritic

Des bonnes idées, des pistes. Pas des règles immuables.

© Sylvain Lhullier

Diffusion et reproduction interdites

17

Appels système

Comme le C, Perl dispose de tous les appels système.
Les interfaces peuvent changer un peu.
Portabilité sous windows, etc.

Appel système : routine en interaction avec le noyau.
Ne pas confondre avec la commande system('')!

perldoc perlfunc

© Sylvain Lhullier

Diffusion et reproduction interdites

18



— Perl5/6: outils Perl pour le système —



Appels système : gestion du système de fichiers

- chmod *mode*, *fichiers*Changement des permissions des fichiers chmod 0644, @filenames;
- umask mode
 Changement de la valeur d'umask
 \$oldumask = umask 0022;
- chown \$uid, \$gid, @filenames; Changement de propriétaires et/ou de groupe.

— Perl5/6: outils Perl pour le système —



Appels système : gestion du système de fichiers

• rename \$oldname, \$newname;

Déplace un fichier/dossier (dans le même FS)

File::Copy move - cross-FS

File::Rename rename - renommage multifichier

• unlink @filenames;

Supprime les fichiers

File::Remove remove - suppression récursive

© Sylvain Lhullier

Diffusion et reproduction interdites

19

© Sylvain Lhullier

Diffusion et reproduction interdites

Appels système : gestion du système de fichiers

- chdir '/new/path'; Changement de répertoire courant
- mkdir \$newdir(, \$mask); Création d'un nouveau répertoire File::Path mkpath création en profondeur
- rmdir \$dir; Suppression d'un répertoire vide
- use POSIX qw(getcwd);
 \$pwd = getcwd; Renvoie le répertoire courant
 Le module Cwd gère les problématiques de liens symboliques

© Sylvain Lhullier

Diffusion et reproduction interdites

21

)•(

Appels système : gestion du système de fichiers

Manipulation fine du contenu des répertoires

- opendir ouverture
- readdir lecture
- closedir fermeture
- rewinddir rembobinage
- telldir position actuelle
- seekdir déplacement à une position

Rappel: @f = glob('/path/*.pl');

© Sylvain Lhullier

Diffusion et reproduction interdites

22



— Perl5/6: outils Perl pour le système —



23

Appels système : gestion du système de fichiers

- link \$existingfile, \$newlink Création d'un lien physique (même numéro d'inode)
- symlink \$existingfile, \$newsyllink Création d'un lien symbolique

— Perl 5/6 : outils Perl pour le système —



Appels système : gestion du système de fichiers

stat - récupération d'information sur un fichier

```
($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,
$atime,$mtime,$ctime,$blksize,$blocks) = stat($filename);
```

0 dev - numéro de device du filesystem

1 ino - numéro d'inode

2 mode - mode du fichier (type et permissions)

3 nlink - nombre de liens physiques

4 uid - numéro du propriétaire

5 gid - numéro du groupe

6 rdev - identifiant du device (fichier spécial)

7 size - taille en octets

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites

 24

Appels système : gestion du système de fichiers

```
8 atime - dernier accès (secondes depuis 1970)
9 mtime - dernière modification (secondes depuis 1970)
10 ctime - dernière modification de l'inode (secondes depuis 1970)
11 blksize - taille des blocks du FS
12 blocks - nombre de blocks utilisés

(undef,undef,undef,undef,$uid,$gid,undef,$size) = stat($filename);
$mode = (stat($filename))[2];
($mode,$uid,$gid) = (stat($filename))[2,4,5];
```

© Sylvain Lhullier

Diffusion et reproduction interdites

25

) • (



— Perl5/6: outils Perl pour le système —

1stat : information sur le lien et non le fichier pointé

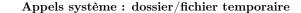


27

Appels système : gestion des utilisateurs/groupes

Consultation de la base de utilisateurs / groupes (lecture des fichiers /etc/passwd et /etc/group et/ou base PAM) :

- (\$login,\$passwd,\$uid,\$gid,\$quota,\$comment,\$gcos,\$dir, \$shell,\$expire) = getpwuid(\$uid); Informations sur un utilisateur à partir de son numéro
- (\$name,\$passwd,\$gid,\$members) = getgrgid(\$gid); Informations sur un groupe à partir de son numéro



use File::Temp qw/tempfile tempdir/; #Interface objet disponible

Création de fichier temporaire :

```
$fd = tempfile(); # Descripteur ouvert
($fd, $filename) = tempfile(); # Connaître le nom
$template = 'tempXXXXX'; # Modèle de nom de fichier
($fd, $filename) = tempfile( $template, SUFFIX => '.dat');
($fd, $filename) = tempfile( $template, DIR => $dir);
```

Création de dossier temporaire (et utilisation par tempfile) :

```
$dir = tempdir(); # Dossier créé
$dir = tempdir( CLEANUP => 1 ); # Suppression automatique
($fd, $filename) = tempfile( DIR => $dir );
```

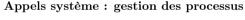
© Sylvain Lhullier

Diffusion et reproduction interdites

26

) (

— Perl5/6: outils Perl pour le système —

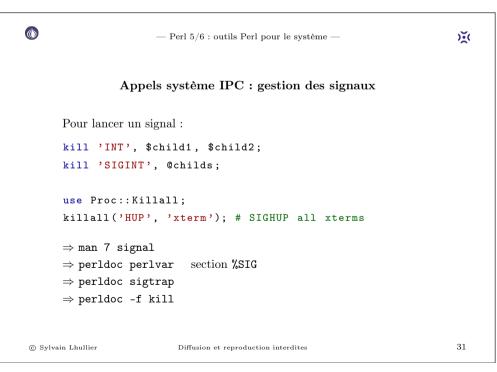


- \$\$ \$PID \$PROCESS_ID numéro de PID
- getppid numéro de PID du père
- fork création d'un processus fils
- \bullet $\,$ exec $\,$ $\,$ exécute un programme (écrasement de code)
- pipe création d'un tuyau
- \bullet wait $% \left(-\right) =\left(-\right) \left(-\right) =\left(-\right) \left(-\right) \left$
- $\bullet\,$ waitpid $\,$ attendre la fin d'un processus donné
- \bullet kill $\,$ envoyer un signal (lire la suite)
- %SIG gestion des signaux (lire la suite)

© Sylvain Lhullier

Diffusion et reproduction interdites

```
— Perl 5/6 : outils Perl pour le système —
                                                                               )•(
                  Appels système : gestion des processus
         pipe( my $readhandle, my $writehandle ) or die("pipe: $!");
         my $pid;
         if( ($pid = fork()) == -1 ) { die("fork: $!"); }
         if( $pid != 0 ) { #Père
           close($readhandle);
           $writehandle->autoflush(1); #use IO::Handle;
           while( .. ) {
              print $writehandle "....\n";
           }
         } else { #Fils
           close($writehandle);
           while( defined( my $ligne = <$readhandle> ) ) {
           }
         }
                                                                               29
© Sylvain Lhullier
                           Diffusion et reproduction interdites
```



```
— Perl 5/6 : outils Perl pour le système —
                                                                          ) • (
                Appels système IPC: gestion des signaux
      Table de hachage %SIG Clefs: noms des signaux
      Valeurs : références (de code) de handlers pour ces signaux
      my $get_signal = 0;
      $SIG{INT} = $SIG{QUIT} = sub { $get_signal++ };
      while ( not $get_signal ) {
          # traitement long
      $SIG{INT} = \&fonction_de_traitement;
      Le pragma sigtrap permet de gérer les signaux également
      use sigtrap 'handler' => \&my_handler, 'INT';
© Sylvain Lhullier
                                                                          30
                          Diffusion et reproduction interdites
```

Appels système : gestion des IPC

- File de messages
 msgctl msgget msgrcv msgsnd
- Mémoire partagée shmctl shmget shmread shmwrite
- Sémaphore semctl semget semop

© Sylvain Lhullier Diffusion et reproduction interdites



Appels système : gestion du système de fichiers

- fcntl

 Manipulation des descripteurs de fichier
- ioct1

 Modifie le comportement des périphériques sous-jacents des fichiers spéciaux.

© Sylvain Lhullier

Diffusion et reproduction interdites

— Perl 5/6 : outils Perl pour le système —

33

) (

35

Appels système: gestion du système de fichiers • utime \$atime, \$mtime, @filenames; Change les dates des fichiers (secondes depuis 1970) • binmode \$fd; Passe les écritures/lectures à venir en mode binaire. Pas de transformation dues au texte: \n windows, etc



Fonction utile: rand

La fonction rand revoie un nombre aléatoire entre 0 et son premier paramètre (1 par défaut)

rand() renvoie un nombre décimal entre 0 et 1 (1 exclu)
rand(256) renvoie un nombre décimal entre 0 et 255
int(rand(256)) renvoie un nombre entier entre 0 et 255

La fonction **srand** est appelée automatiquement par **rand** si cela n'a jamais été fait.

— Perl 5/6 : outils Perl pour le système —



Fonctions utiles: attendre

- sleep(\$seconds);
- use Time::HiRes qw(usleep); usleep(\$microseconds);
- use Time::HiRes qw(nanosleep); nanosleep(\$nanoseconds);

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites





))(

Formats mémoire et binaires

Conversion de données : format Perl <=> format binaire

- pack format Perl => format binaire
- unpack format binaire => format Perl

```
$binaryint = pack('i', 125);
$binarydouble = pack('D', 3.14159);
$valeur = unpack('s', $binaryshort);
```

© Sylvain Lhullier

Diffusion et reproduction interdites

37

Formats mémoire et binaires

Code de formats :

- 'c' char 'C' unsigned char
- 's' short 'S' unsigned short
- 'i' int 'I' unsigned int
- 'F' float
- 'D' double
- etc

perldoc -f pack

© Sylvain Lhullier

Diffusion et reproduction interdites

38

— Perl5/6: outils Perl pour le système —



Accès en parallèle à des flux avec select

L'appel système select permet une attente non-coûteuse sur plusieurs flux :

```
select( RBITS, WBITS, EBITS, TIMEOUT )
```

- RBITS : flux observés afin d'y lire
- WBITS : flux observés afin d'y écrire
- EBITS : flux observés pour des événements exceptionnels
- TIMEOUT : délai maximum d'attente (undef pour infini)

— Perl5/6: outils Perl pour le système —



40

Accès en parallèle à des flux avec select

Il faut construire les champs de bits :

```
my ($rin, $win, $ein);
$rin = $win = $ein = '';
# Descripteurs en lecture
vec($rin, fileno($TDIN), 1) = 1;
vec($rin, fileno($fd1), 1) = 1;
# Descripteurs en écriture
vec($win, fileno($TDOUT), 1) = 1;
vec($win, fileno($fd2), 1) = 1;
# Descripteurs pour événements
$ein = $rin | $win;
```

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites

Accès en parallèle à des flux avec select

```
# Appel classique :
    ($nfound,$timeleft) =
     select($rout=$rin, $wout=$win, $eout=$ein, $timeout);
```

- \$nfound : nombre de flux en attente (utilisables)
- \$timeleft: temps restant

Puis appel à select :

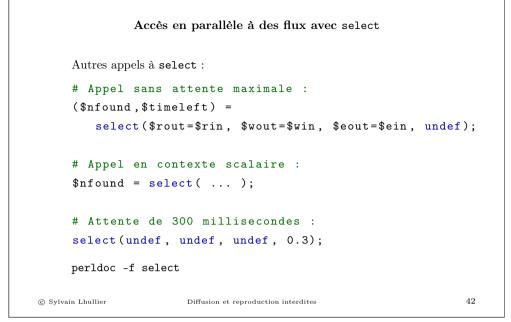
• \$rout, \$wout et \$eout : vecteurs indiquant les flux en attente

© Sylvain Lhullier

© Sylvain Lhullier

Diffusion et reproduction interdites

41





— Perl5/6: outils Perl pour le système —



43

Accès en parallèle à des flux avec IO::Select

Diffusion et reproduction interdites

```
use IO::Select;
$s = IO::Select->new();
$s->add(\*STDIN);
$s->add($some_handle);
@ready = $s->can_read($timeout);
# $s->can_write( ... );
# $s->has_exception( ... );
@ready = IO::Select->new(@handles)->can_read($timeout);
perldoc IO::Select
```

© Sylvain Lhullier

— Perl5/6: outils Perl pour le système —



44

Gestion d'événements multiples avec AnyEvent

```
Appel de callback (cb) selon l'événement
```

```
use AnyEvent;
# Attente de pouvoir lire dans un flux
AnyEvent->io( fh=>$fd, poll=>'r', cb=>sub{ ... } );
# Timer unique ou plusieurs fois
AnyEvent->timer( after=>$seconds, cb=>sub { ... } );
AnyEvent->timer( after=>$sec1, interval=>$sec2, cb=>... );
# Signal POSIX
AnyEvent->signal( signal=>'TERM', cb=>sub{ ... } );
# Fin de sous-processus
AnyEvent->child( pid=>$pid, cb=>sub{ ... } );
```

Diffusion et reproduction interdites





Gestion du temps

Système de référence :

- Unité : seconde
- Epoch: 1er janvier 1970 à 00h00 UTC (0 UNIX)

 UTC (Universal Time Coordinated). Ne pas confondre avec

 GMT (Greenwich Mean Time): heure observée à Greenwich

 (vitesse variable de la Terre sur son orbite elliptique)

Variantes:

- Décalage depuis *Epoch*
- Fuseau horaire local

© Sylvain Lhullier

Diffusion et reproduction interdites

45

Gestion du temps : secondes depuis Epoch

time()

Renvoie le nombre actuel de secondes depuis Epoch

 ${\tt localtime()}$ convertit un nombre de seconde depuis Epoch en une représentation locale

- En contexte de chaîne : chaîne de ctime \$now_string = localtime(); # 'Mon Oct 31 13:25:46 2042'
- En contexte de liste :
 (\$sec,\$min,\$hour,\$mday,\$mon,\$year,\$wday,\$yday,\$isdst) =
 localtime();

© Sylvain Lhullier

Diffusion et reproduction interdites

46



— Perl5/6: outils Perl pour le système —



47

Gestion du temps : valeurs renvoyées par localtime

```
0 sec - nombre de secondes (de 0 à 59, voire 60)
```

1 min - nombre de minutes

2 hour - nombre d'heures

3 mday - numéro de jour dans le mois

4 mon - mois (0=janvier, 11=décembre)

5 year - année-1900 (142 pour 2042)

6 wday - jour de la semaine (0=dimanche, 6=samedi)

7 yday - jour de l'année (de 0 à 364 ou 365 / 0=1er jany)

8 isdst - booléen indiquant que l'on est ou non en période de changement d'heure d'été / heure d'hiver

(\$sec,\$min,\$hour,\$mday,\$mon,\$year,\$wday) = localtime(); \$year += 1900; @d = qw(Dim Lun Mar Mer Jeu Ven Sam);

@m = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);

 $\label{lem:min:sec $d[$wday] $mday $m[$mon] $year\n";} \\$

© Sylvain Lhullier Diffusion et reproduction interdites

— Perl 5/6 : outils Perl pour le système —



Gestion du temps : paramètre de localtime

 ${\tt localtime()}\ \ {\tt peut}\ \ {\tt prendre}\ \ {\tt un}\ \ {\tt paramètre}:\ \ {\tt un}\ \ {\tt nombre}\ \ {\tt de}\ \ {\tt seconde}$

depuis *Epoch*, par défaut : time()

 $\label{localtime} \mbox{localtime()-60*60*24*2);} \quad \mbox{Il y a deux jours}$

localtime((stat(\$filename))[9]); Date de modif. du fichier

Agir sur le fuseau horaire : \$ENV{TZ} (lire plus loin)

gmtime() fait la même chose mais en temps UTC

Attention au nom : il s'agit bien de UTC

Attention : un seul t!

© Sylvain Lhullier

Diffusion et reproduction interdites

Gestion du temps : formatage avec strftime

```
strftime() permet de convertir les dates (string format time)
perldoc POSIX
                   man strftime
strftime(format, sec. min, hour, mday, mon, year)
sec - nombre de secondes de 0 à 59
min - nombre de minutes
hour - nombre d'heures
mday - numéro de jour dans le mois
mon - mois (0=janvier, 11=décembre)
year - année-1900 (142 pour 2042)
(mêmes valeurs que celle renvoyées par localtime)
```

© Sylvain Lhullier

Diffusion et reproduction interdites

49

) • (

Gestion du temps : codes de format de strftime

Le format de strftime() est une chaîne de caractères pouvant comporter:

%A - nom du jour de la semaine (suivant localisation)

%B - nom du mois (suivant localisation)

%c - représentation classique date&heure (suivant localisation)

%d - numéro du jour dans le mois (entre 01 et 31)

%H - heure (entre 00 et 23)

%m - numéro du mois (entre 01 et 12)

%M - minute (00 à 59)

%s - nombre de secondes écoulées depuis le Epoch

%S - seconde (00-59)

%Y - année (par exemple : 2042)

%Z - fuseau horaire

%% - caractère %

© Sylvain Lhullier

Diffusion et reproduction interdites

50

).(

— Perl 5/6 : outils Perl pour le système —



51

Gestion du temps : exemples de strftime

```
use POSIX qw(strftime);
      $str = strftime('%A, %d %B %Y', 0, 0, 0, 15, 2, 142);
      print "$str\n"; #samedi, 15 mars 2042
      $str = strftime('%A, %d %B %Y', localtime() );
      print "$str\n"; #vendredi, 7 novembre 2042
      $str = strftime(',%c', localtime());
      print "$str\n"; #ven 07 nov 2042 09:03:22 CET
      $str = strftime( '%d/%m/%Y %H:%M:%S', localtime($nbrSecEpoch) );
      $str=strftime('%d/%m/%Y %H:%M',gmtime(0)); #01/01/1970 00:00
      $str=strftime('%d/%m/%Y %H:%M',localtime(0));#01/01/1970 01:00
© Sylvain Lhullier
```

Diffusion et reproduction interdites





— Perl 5/6 : outils Perl pour le système —

La fonction localtime est sensible à \$ENV{TZ}

```
$ENV{TZ} = 'Europe/Paris';
$str1 = strftime('%c', localtime($dateseconds));
$ENV{TZ} = 'America/Montreal';
$str2 = strftime('%c', localtime($dateseconds));
Possibilité de localiser l'affectation à $ENV{TZ} dans un bloc :
```

local \$ENV{TZ} = 'Australia/Sydney'; \$str3 = strftime('%c', localtime(\$dateseconds));

© Sylvain Lhullier

}

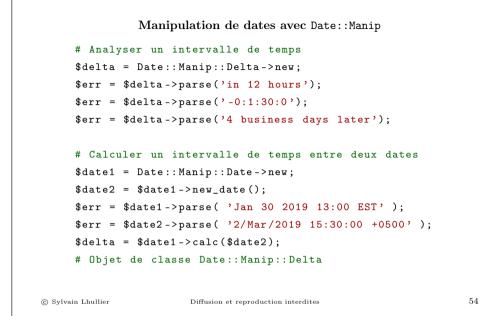
Diffusion et reproduction interdites

Manipulation de dates avec Date::Manip

```
use Date::Manip; # 2 syntaxes : objet (vue ici) / fonctionnelle
# Analyser une date
$date = Date::Manip::Date->new;
$err = $date->parse('today');
$err = $date->parse('1st Thursday in June 1992');
$err = $date->parse('05/10/93');
$err = $date->parse('12:30 Dec 12th 1880');
$err = $date->parse('8:00pm December tenth');

my $sec = $date->printf('%s'); # Nombre de secondes depuis Epoch
my $string = $date->printf('%Y/%m/%d %H:%M:%S %Z');
# Codes de formats similaires à ceux de strftime

Pour plus d'infos perldoc Date::Manip::Date
```





© Sylvain Lhullier

— Perl5/6: outils Perl pour le système —

Diffusion et reproduction interdites



55

53

Cervantes et Shakespeare sont morts à la même date mais pas le même jour ©Jean Forget

```
use DateTime::Calendar::Christian;
my $ws = DateTime::Calendar::Christian->new(
                     year => 1616, month => 4, day => 23,
                     reform_date => 'uk' );
my $mc = DateTime::Calendar::Christian->new(
                     year => 1616, month => 4, day => 23,
                     reform_date => 'italy' ); # même date que l'Espagne
my $dt_ws = DateTime->from_object(object => $ws); # pour avoir un overload
my $dt_mc = DateTime->from_object(object => $mc); # sur les comparaisons
my $comp = 'le même jour que';
if( $dt_mc < $dt_ws ) { $comp = 'avant' } else { $comp = 'après' }</pre>
print "Cervantes est mort $comp Shakespeare\n";
my $d = $dt_mc - $dt_ws;
print abs($d->delta_days), " jour(s) d'écart\n";
⇒ On apprend que Cervantes est mort 10 jours avant Shakespeare.
http://datetime.mongueurs.net/presentation/Dates_et_heures_en_Perl.html#modules_de_calendrier
```

— Perl5/6: outils Perl pour le système —



Réseau : sockets

Deux interfaces principales :

- Appels sytème socket bind listen connect accept send recv
- Interface objet perldoc IO::Socket perldoc IO::Socket::INET

© Sylvain Lhullier

Diffusion et reproduction interdites

56

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Réseau : sockets serveur

— Perl5/6: outils Perl pour le système —



Threads

Perl dispose de threads intégré depuis Perl 5.6.0 appelés threads interpréteur ou ithreads.

perldoc threads perldoc perlthrtut

Implémentation : généralement pthreads sous Linux

use threads;

Création: threads->new(fonction, paramètres)

- \$thr = threads->new(\&fonction); sub fonction { print "Dans le thread\n"; }
- \$thr = threads->new(\&fonction, 'Param1', 'Param2'); sub fonction { print "Dans le thread (0_)\n"; }
- \$thr = threads->new(sub{print "Dans le thread\n";});

© Sylvain Lhullier Diffusion et reproduction interdites

59

Réseau : sockets client

— Perl 5/6 : outils Perl pour le système —



58

Threads: gestion de la fin de vie

Diffusion et reproduction interdites

Gestion par le créateur du thread :

- \$thr->join();
 Attendre qu'un thread termine et nettoyer la mémoire
 sub fonction {; return \$calcul; }
 \$donnees_renvoyees = \$thr->join();
- \$thr->detach();
 Ignorer la fin d'un thread (mémoire nettoyée automatiquement)

© Sylvain Lhullier Diffusion et reproduction interdites



))(

Threads

Pour qu'un thread rende explicitement le contrôle du processeur à un autre thread :

threads->yield(); (méthode de classe)

- cas de calculs très lourd et interface graphique
- gestion des threads ne pas supportant le multitâche préemptif

© Sylvain Lhullier

© Sylvain Lhullier

Diffusion et reproduction interdites

61

Threads: partage de données

Par défaut toutes les données sont propres au thread (même les variables globales)

Le package threads::shared permet d'indiquer lors de la déclaration de variables si elles sont partagées ou non.

Un scalaire, un tableau entier, une table de hachage entière peuvent être déclarés partagés :

```
my $var : shared = 2;
my @tab : shared = (4,5,6);
my %hash : shared = (Paul=>8);
```

L'atomicité de l'accès et modification d'une donnée est garantie.

© Sylvain Lhullier

© Sylvain Lhullier

Diffusion et reproduction interdites

62



— Perl5/6: outils Perl pour le système —



63

Threads: partage de données

```
use threads;
use threads::shared; # Ordre important !

my $partage : shared = 1;
my $personnel = 1;
threads->new( sub{ $partage++; $personnel++ } )->join();

print "$partage\n";
# affiche 2 car $partage est partagée
print "$personnel\n";
# affiche 1 car $personnel n'est pas partagée
```

Diffusion et reproduction interdites



— Perl5/6: outils Perl pour le système —



64

Threads: exclusion mutuelle

La fonction lock() pose un verrou sur une variable partagée.

```
my $verrou : shared = 0;
sub calcul {
    while(1) {
        tache1();
        {
            lock($verrou); # Bloque jusqu'â obtenir le verrou
                tache2(); # Code avec exclusion mutuelle
        } # Verrou automatiquement rendu en fin de bloc
        tache3();
    }
}
my $t1 = threads -> new(\&calcul);
my $t2 = threads -> new(\&calcul);
$t1->join();
$t2->join();
```

Diffusion et reproduction interdites



Courriel

Un courriel (mail, email, etc) est constitué:

- d'une enveloppe (perdue à l'arrivée),
- d'en-têtes.
- d'un contenu.

Attachements : le contenu peut être composé de plusieurs entités.

SMTP : protocole d'envoi de courriels

POP3, IMAP : protocoles de réception de courriels

MIME : format des courriels

© Sylvain Lhullier

Diffusion et reproduction interdites

65



— Perl5/6: outils Perl pour le système —



67

Courriel: envoi

Gestion complète du protocole SMTP : module Net::SMTP

- my \$smtp = Net::SMTP->new('mailhost');
 Connexion à un serveur
 Net::SMTP->new('mailhost', Port=>25, Timeout=>60);
- \$smtp->mail('moi@ici.org'); Adresse de l'expéditeur (MAIL FROM)
- \$smtp->to('lui@la-bas.org');
 Adresse du destinataire (RCPT TO)



Diffusion et reproduction interdites



— Perl 5/6 : outils Perl pour le système —



Exemple d'usage du module Net::POP3

```
#!/usr/bin/perl
use strict;
use warnings;
use Net::POP3;
my $pop = Net::POP3->new('host') or die("pop->new: $!");
$pop->login($username, $password) or die("pop->login: $!");
my $messages = $pop->list();  # hashref of mesgId => size
foreach my $mesgId (sort {$a<=>$b} keys %$messages) {
    my $lineRef = $pop->get($mesgId);  # tabref of lines
    print @$lineRef;
    $pop->delete($mesgId);
}
$pop->quit();
```

© Sylvain Lhullier

Diffusion et reproduction interdites

66

— Perl5/6: outils Perl pour le système —



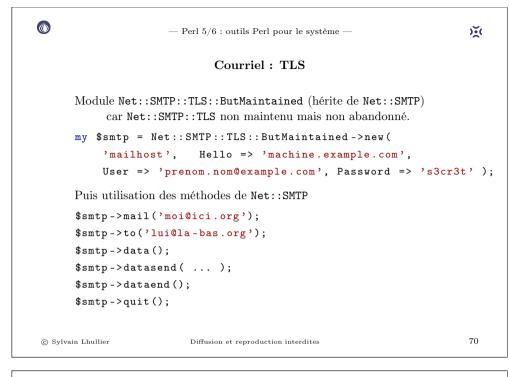
Courriel: envoi

- \$smtp->data(); Début des données (DATA)
- \$smtp->datasend('Blabla');
 Transmission des en-têtes et du contenu
- \$smtp->dataend(); Fin des données (ligne avec point seul)
- \$smtp->quit(); Ferme la connexion avec le serveur (QUIT).

© Sylvain Lhullier

Diffusion et reproduction interdites

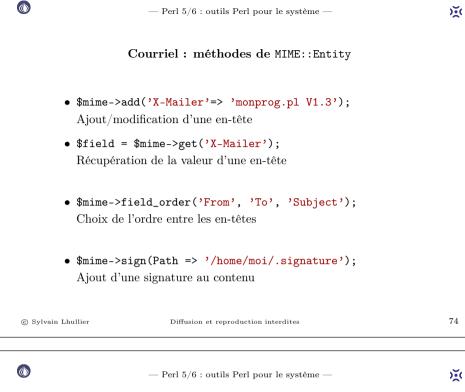
```
— Perl 5/6 : outils Perl pour le système —
                                                                    )•(
                           Courriel: envoi
      my $smtp = Net::SMTP->new('serveur', Timeout => 30);
      $smtp->mail('moi@ici.org'); # Enveloppe
      $smtp->to('lui@la-bas.org'); # Enveloppe
      $smtp->data();
      $smtp->datasend( "From: Moi <moi\@ici.org>\n" );
      $smtp->datasend( "To: Lui <lui\@la-bas.org>\n" );
      $smtp->datasend( "Subject: RVD demain\n" );
      $smtp->datasend( "\n" );
      $smtp->datasend( "C'est ok pour toi ?\n" );
      $smtp->datasend( "\n" );
      $smtp->datasend( "Moi.\n" );
      $smtp ->dataend();
      $smtp->quit();
                        Diffusion et reproduction interdites
                                                                    69
© Sylvain Lhullier
```

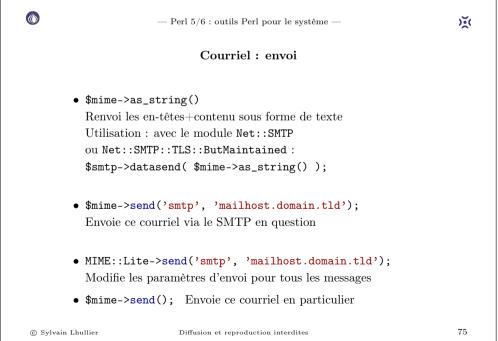


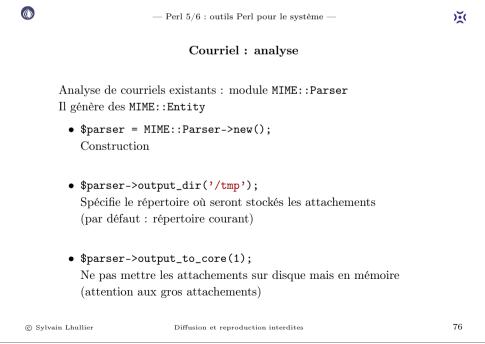
```
— Perl 5/6 : outils Perl pour le système —
                                                                            )•(
                              Courriel: envoi
      Gestion du MIME : modules MIME: :
      dont MIME::Lite qui génère des MIME::Entity.
      Version simplissime:
      use MIME::Lite;
      $msg = MIME::Lite->new(
                  From
                             => 'moi@ici.org',
                  Tο
                             => 'lui@la-bas.org',
                  Subject => 'RVD demain',
                  Data
                             => "C'est ok pour toi ?\n\nMoi.",
              );
      $msg ->send();
      Cette syntaxe nécessite un serveur de mail installé en local.
© Sylvain Lhullier
                          Diffusion et reproduction interdites
                                                                           71
```

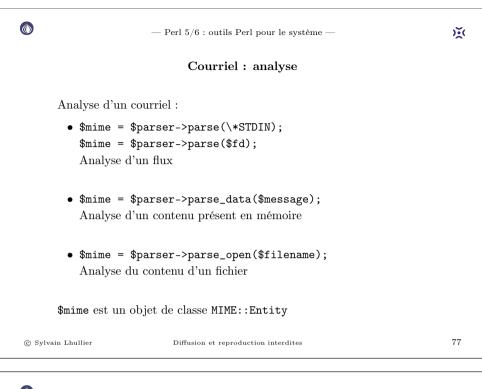
```
— Perl 5/6 : outils Perl pour le système —
                                                                         ).(
                             Courriel: envoi
       • my $mime = MIME::Lite->new(); - Construction
          my $mime = MIME::Lite->new(
             From
                          => 'Moi <moi@ici.org>',
             Τo
                          => 'Lui <lui@la-bas.org>',
                          => 'RVD demain'.
             Subject
             Type
                          => 'TEXT',
             Encoding
                          => 'quoted-printable',
                          => "C'est ok pour toi ?\n\nMoi.\n",
             Data
          );
© Sylvain Lhullier
                                                                         72
                          Diffusion et reproduction interdites
```

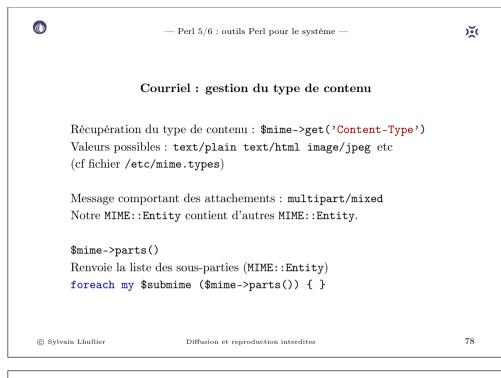
```
— Perl 5/6 : outils Perl pour le système —
                                                                             ) • (
                       Courriel: envoi d'attachement
        • $mime->attach()
          Ajoute un attachement
          $mime ->attach(
               Type
                          => 'image/pjpeg',
               Encoding => 'base64',
               Path
                          => '/path/to/file/badname.jpg',
               Filename => 'file.jpg'
          );
                                                                             73
© Sylvain Lhullier
                           Diffusion et reproduction interdites
```

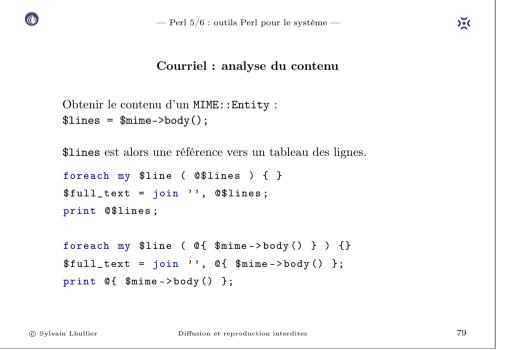


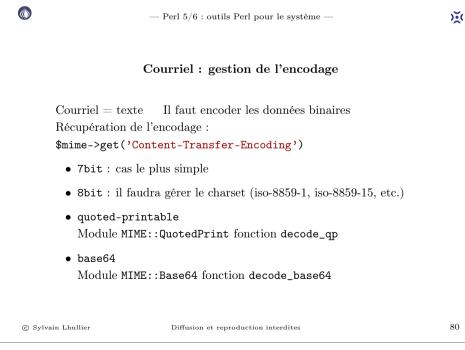














) (

Courriel: analyse

© Sylvain Lhullier

Diffusion et reproduction interdites

81

Courriel: Mail::Box::Manager

Une boîte courriel est constituée de plusieurs courriels successifs. Le module Mail::Box::Manager permet d'en gérer les messages.

```
use Mail::Box::Manager;
my $mgr = Mail::Box::Manager->new;
my $folder = $mgr->open(folder => 'boite-mail.mbox');
foreach my $message ($folder->messages) {
    # Extraction de chaque courriel
    # (à traiter avec MIME::Parser ?)
    print $message->string;
}
```



— Perl5/6: outils Perl pour le système —



83

Gestion des paramètres avec Getopt::Long

Gestion des paramètres de la ligne de commande :

- flag, valeur numérique ou chaîne script.pl --verbose --nbr 42 --date now fichier
- valeurs multiples script.pl --rgbcolor 255 0 255 --coord 26.4 85.1
- paramètres condensés -l -a -c ou -lac
- syntaxe souple -d 32 -d32 --data 32
- prise en compte de -- (fin des paramètres) script.pl --verbose --debug -- --nom-de-fichier.txt

Suppression de @ARGV des paramètres traités (lecture dans <>)

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

— Perl5/6: outils Perl pour le système —

Diffusion et reproduction interdites



84

82

Getopt::Long

```
use Getopt::Long;
     my $file = 'data.txt';
     mv $size = 24;
     my $debug;
     my $r = GetOptions('size=i' => \$size,
                                                    # nombre
                          'file=s' => \$file,
                                                    # chaîne
                          'debug' => \$debug)
                                                    # flag
        or die(usage());
     # Aliases :
     GetOptions ('length|height=f' => \$length);
     # --length ou --height flottant
© Sylvain Lhullier
                        Diffusion et reproduction interdites
```

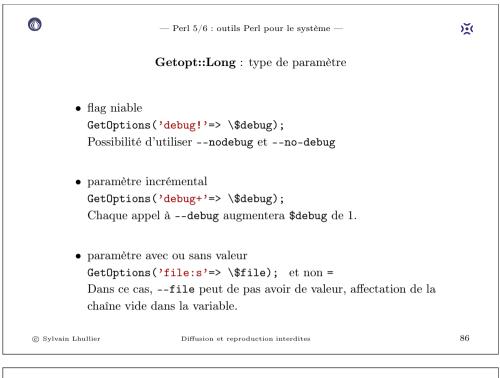
```
Getopt::Long: type de paramètre

• flag
GetOptions('debug'=> \$debug);

• entier
GetOptions('size=i'=> \$size);

• flottant
GetOptions('length=f'=> \$length);

• chaîne libre
GetOptions('file=s'=> \$file);
```



```
— Perl 5/6 : outils Perl pour le système —
                                                                            ).(
                    Getopt::Long : type de paramètre
        • valeurs multiples libres
          GetOptions('path=s'=> \@path);
          --path chemin1 --path chemin2 --path chemin3
        • valeurs multiples contraintes
          GetOptions('coord=f{2}'=> \@coord, 'rgbcolor=i{3}'=> \@color);
          --coord 26.4 85.1 --rgbcolor 255 0 255
        • valeurs multiples associatives
          GetOptions('value=s'=> \%values);
          --value os=linux --value vendor=debian
© Sylvain Lhullier
                                                                            87
                           Diffusion et reproduction interdites
```

```
— Perl 5/6 : outils Perl pour le système —
                                                                      ) (
                        Getopt::Long: callback
      GetOptions('data' => sub{ .... } );
      GetOptions('data' => \&maFonction );
      my $verbose = 0;
      GetOptions ('verbose' => \$verbose,
                   'quiet' => sub { $verbose = 0 });
      sub f { print "f : 0_\n"; }
      my $r = GetOptions( 'file=s' => \&f );
      # ./prog.pl --file=data.txt
      # Arguments de la fonction f : file data.txt
© Sylvain Lhullier
                                                                      88
                         Diffusion et reproduction interdites
```



Filtres courts en Perl

Objectif : écrire des filtres en ligne de commande (one liner) aussi rapidement qu'en shell, mais plus puissants

```
perl ...options... fichier
who | perl ...options... | wc -1
```

Usage des options:

- -e : code en ligne de commande
- -n : lecture automatique
- -p : lecture et écriture automatiques
- \bullet -i : traitement «en ligne»
- -M : chargement d'un module

© Sylvain Lhullier

Diffusion et reproduction interdites

89

 \bigcirc — Perl5/6: outils Perl pour le système —

Filtres courts en Perl: option -n

Option -n : lecture dans \$_ de chaque ligne de <ARGV> (fichiers passés en paramètres et/ou entrée standard).

Correspond à l'ajout des instructions suivantes :

```
LINE:
while (<>) {
    ... # votre programme ici
}
```

© Sylvain Lhullier

Diffusion et reproduction interdites

— Perl5/6: outils Perl pour le système —



91

Filtres courts en Perl: option -n

```
Exemple:
```

```
perl -ne 'print if m/regexp/' fichier
exécute en fait :
    LINE:
    while (<>) {
        print if m/regexp/
}
```

- ⇒ Affichage des lignes qui correspondent à une regexp Perl
- ⇒ Commande grep avec les regexp de Perl



— Perl5/6: outils Perl pour le système —



90

) • (

Filtres courts en Perl: option -n

```
Affiche tous les logins du système
```

```
perl -ne 'm/^(.+?):/;print"$1\n"' /etc/passwd
```

Affiche la longueur de chaque ligne du fichier

```
perl -ne 'print length."\n"' fichier
```

Affiche uniquement les lignes numéros impairs (1ère, 3ème, etc)

```
perl -ne 'print if $.%2' fichier
```

Efface les fichiers non modifiés depuis 7 jours

```
find . -mtime +7 -print | perl -nle unlink
```

© Sylvain Lhullier

Diffusion et reproduction interdites

© Sylvain Lhullier

Diffusion et reproduction interdites



Filtres courts en Perl: option -n

Compte le nombre de chiffres contenus dans le fichier perl -ne $\frac{n+=y}{0-9}$; END $\frac{n'}{n'}$, fichier

Compte le nombre de «mots» contenus dans le fichier perl -ne '@w=split/\W+/;\$w+=@w;END{print"\$w\n"}' fichier

Supprime les lignes en doublon (mieux qu'uniq) perl -ne 'print unless \$cpt{\$_}++' fichier

Affiche les lignes du fichier par ordre décroissant d'occurrence perl -ne $\frac{\$h}{\$}++;END\{print sort \{\$h\{\$b\}<=>\$h\{\$a\}\} keys\%h\}$, f

© Sylvain Lhullier

Diffusion et reproduction interdites

93

).(

95

) • (





Numérote les lignes d'un fichier

perl -pe 's/^/\$. /' fichier

Supprime les 5 premiers caractères de chaque ligne

perl -pe '\$_=substr(\$_,5)' fichier > fichier2

Met le fichier en majuscule

perl -pe 'tr/a-z/A-Z/' fichier > fichier2

© Sylvain Lhullier



Filtres courts en Perl: option -p

Option -p : lecture dans \$_ de chaque ligne de <ARGV> puis **affichage** de \$_ (qu'il faut donc modifier)

Correspond à l'ajout des instructions suivantes :

```
LINE .
while (<>) {
   ... # votre programme ici
} continue {
   print or die "-p destination: $!\n";
```

© Sylvain Lhullier

Diffusion et reproduction interdites

94

) (

— Perl 5/6 : outils Perl pour le système —



Remplace le mot vert par bleu

perl -pe 's/\bvert\b/bleu/g' fichier > fichier2

Conversion dos2unix

perl -pe 's/\r\n/\n/g' fichier_dos.txt

Ne garde que le premier mot de chaque ligne

perl -pe 's/ $\W*(\w+).*/\$1/$ ' fichier > fichier2

Garde les 10 premières lignes d'un fichier

perl -pe 'exit if \$.>10' fichier > fichier2

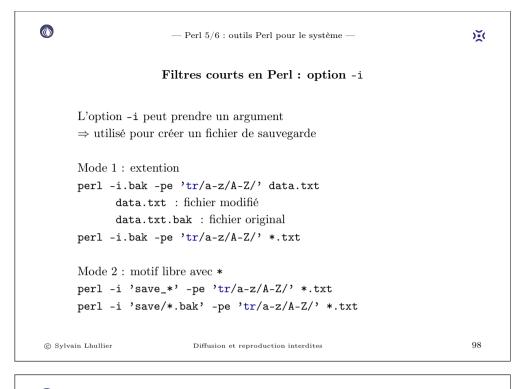
© Sylvain Lhullier

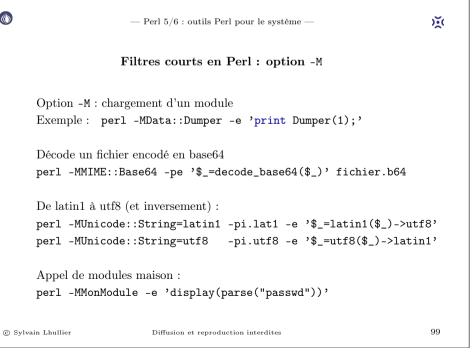
Diffusion et reproduction interdites

) • (

Diffusion et reproduction interdites

```
— Perl 5/6 : outils Perl pour le système —
                                                                                ) • (
                      Filtres courts en Perl: option -i
       Option -i : modification des fichiers «en ligne» (édition sur place)
       Avant:
       perl -pe 'tr/a-z/A-Z/' data.txt > data.new
      mv -f data.new data.txt
       Avec -i ·
       perl -i -pe 'tr/a-z/A-Z/' data.txt
       ⇒ Le fichier original est directement modifié
       Sur plusieurs fichiers à la fois :
       perl -i -pe 'tr/a-z/A-Z/' data.txt data2.txt
      perl -i -pe 'tr/a-z/A-Z/' *.txt
                                                                                97
© Sylvain Lhullier
                            Diffusion et reproduction interdites
— Perl 5/6 : outils Perl pour le système —
                                                                                ).(
```





— Perl 5/6 : outils Perl pour le système —) (Filtres courts en Perl: d'autres options -1 (comme «L»ine) : chomp sur chaque ligne lue et ajoute \n à la fin de chaque ligne affichée -I : ajoute des répertoires dans QINC -0 (zéro) : modifie la notion de ligne avec \$/ -a: ajout d'un split /\s+/ sur chaque ligne (valeurs dans @F) -F: changer le motif du split de -a etc Plus d'informations : perldoc perlrun http://articles.mongueurs.net/magazines/linuxmag50.html © Sylvain Lhullier 100 Diffusion et reproduction interdites



Lancer une commande: system

• system '/usr/bin/prog';
system 'ls -al *.txt';

La sortie standard est la même que le script Perl.

Les erreurs s'affichent également.

Attention: and pour tester

```
system 'ls -al *.txt' and die "erreur: $!";
```

Attente de la fin de la commande avant d'exécuter la suite du programme

© Sylvain Lhullier

Diffusion et reproduction interdites

101



— Perl 5/6 : outils Perl pour le système —



Lancer une commande: antiquotes

• \$sortie = '/usr/bin/prog';

En contexte scalaire : récupération en une seule chaîne de caractères de la sortie standard de la commande

v = 'df -h';

• @lignes = '/usr/bin/prog';
foreach('/usr/bin/prog') {...}

En contexte de liste : récupération de chaque le ligne de la sortie standard de la commande

```
foreach my $ligne ('df -h') {
  print $ligne;
}
```

• Attente de la fin de la commande avant d'exécuter la suite du programme

© Sylvain Lhullier

Diffusion et reproduction interdites

102



— Perl5/6: outils Perl pour le système —



103

Lancer une commande : open

• open(filedescr, 'commande|') exécution de la commande On lira sa sortie standard dans le descripteur de fichier.

```
open(my $fd,'ls|') or die "$!";
open(my $fd,"df -HT $device|") or die "$!";
while( defined( my $ligne=<$fd> ) ) {
   print $ligne;
}
close($fd);
```

© Sylvain Lhullier

Diffusion et reproduction interdites

— Perl5/6: outils Perl pour le système —



Lancer une commande : open

• open(filedescr,'|commande') exécution de la commande On écrira son entrée standard dans le descripteur de fichier. Attention au signal SIGPIPE reçu si la commande se termine avant que nous ayons fini de lui fournir à lire

```
open(my $fd,"|gzip > $x.gz") or die "$!";
open(my $fd,'|mail robert@bidochon.org') or die "$!";
print $fd "chaîne\n";
close($fd);
```

© Sylvain Lhullier

Diffusion et reproduction interdites

```
— Perl 5/6 : outils Perl pour le système —
```



Lancer une commande : contrôle

```
• system 'commande' and die "$!";
Attention: and
```

- open(my \$fd, 'commande|') or die "\$!";
- $\bullet\,$ La variable $\ref{eq:contient}$ contient des informations sur le dernier processus

```
$status = $? >> 8; valeur de sortie
```

signal = ?? & 127; signal qui a tué le processus

coredp = 228; vrai si suite à un core dump

NB: La variable \$PATH du shell est utilisée. NB en Perl : \$ENV{PATH}

© Sylvain Lhullier

Diffusion et reproduction interdites

105



— Perl 5/6 : outils Perl pour le système —



Lancer une commande : aller plus loin

En Perl, on a aussi accès aux appels système fork et exec.

Un gestionnaire de fork : Parallel::ForkManager

Lancer des processus à partir de Perl http://articles.mongueurs.net/magazines/linuxmag55.html

© Sylvain Lhullier

Diffusion et reproduction interdites

106

) (

— Perl5/6: outils Perl pour le système —



107

Fonction eval

Cette fonction évalue les chaînes de caractères comme du code Perl.

• \$0 : erreur du dernier eval eval '\$r = \$x / \$y'; warn(\$0) if(\$0);

À utiliser avec parcimonie.

© Sylvain Lhullier

Diffusion et reproduction interdites

Exemple : le rename de Larry Wall (allégé)

— Perl 5/6 : outils Perl pour le système —

```
Usage : rename 's/ancien(\..*)/nouveau$1/' *
       rename 'tr/A-Z/a-z/' *.txt
#!/usr/bin/perl
$op = shift @ARGV;
foreach(@ARGV) {
                           # $_ vaut chaque nom de fichier
  $was = $_;
                           # Sauvegarde du nom
                           # Modification de $
  eval $op;
  die($0) if($0);
                           # Gestion d'erreur
  rename($was,$_)
                           # Renommage
     if($was ne $_);
}
```