Chapitre 1

Pourquoi choisir Perl ?

Les langages de programmation sont des outils. Certains s'apparentent à des bistouris de précision, qui exigent un contrôle total des moindres détails mais offrent des performances étonnantes ; d'autres ressemblent à des grues pilotées par satellite, puissantes, mais lentes et excessivement complexes ; d'autres encore sont spécialisés sur un petit créneau qu'ils font très bien et deviennent catastrophiques sur tous les autres.

Perl serait plutôt du genre couteau suisse, ou, plus exactement, *tronçonneuse suisse* : un outil tous usages à l'efficacité spectaculaire, qui s'adapte à votre main.

Sa spécialité : optimiser votre temps.

Comme tous les langages à succès, Perl est un langage radical. Son choix radical, c'est l'humilité en toutes circonstances : ce n'est pas au langage de vous imposer une approche, c'est toujours à vous de décider ce qui convient le mieux au problème que vous souhaitez résoudre et à votre tempérament. C'est un serviteur zélé et efficace, pas un maître exigeant.

Si vous êtes familier avec d'autres langages, vous avez peut-être remarqué qu'ils sont souvent construits sur une bonne idée dont ils ne permettent, hélas, pas de sortir[1]. L'approche de Perl est exactement inverse ; c'est un rassembleur qui chérit la diversité. Vous pouvez programmer Perl par objets si vous le souhaitez ; vous pouvez aussi ne pas le faire ;

[1] Songez par exemple aux objets en Java, aux parenthèses en Lisp, aux piles en Forth, aux fonctions en Caml ou au backtrack en Prolog...

vous pouvez utiliser des fonctions récursives comme vous pouvez vous en passer. Une seule règle est de mise : en Perl, « il y a plusieurs façons de le faire ! »

D'où vient Perl ?

Le langage Perl[2] est la création de Larry Wall, qui était déjà l'inventeur des programmes Patch[3] et RN[4]. L'objectif initial[5] était de créer un langage de script très complet qui pourrait remplacer plusieurs outils standard d'Unix[6], tout en fournissant des fonctions analogues à celles de C[7] et C++[8].

Larry Wall étant par ailleurs un linguiste averti, il a intégré à Perl des tournures que l'on ne trouve habituellement que dans les langages naturels[9]. C'est cette caractéristique unique qui donne aux programmeurs le sentiment que Perl leur permet d'écrire leur pensée telle quelle !

La syntaxe de Perl a été définie selon deux grandes règles : d'une part, ce qui est simple doit s'écrire simplement et ce qui est difficile doit rester possible ; d'autre part, la syntaxe est optimisée pour le cas le plus courant : au lieu d'imposer un carcan uniforme, Perl vous permet d'utiliser une syntaxe assouplie pour les opérations les plus fréquentes.

[2] Initiales de *Practical Extraction and Report Language*, « Langage pratique d'extraction [d'informations] et de rapport ».

[3] Patch permet de modifier très simplement le code source d'un programme, ce qui a donné l'expression devenue courante : « patcher un programme ».

[4] *Reader for the News*, logiciel de lecture des forums d'Usenet qui a été le premier à mettre en œuvre des threads pour minimiser le temps d'attente de téléchargement. La technique est aujourd'hui standard dans tous les logiciels.

[5] La première version de Perl date de 1986.

[6] Notamment sh, sed, awk, csh, grep et cut.

[7] Le langage C est la référence pour la programmation système. Le langage Perl lui-même est écrit en C.

[8] Perl a également repris des idées de Pascal, Ada, Fortran et Basic+.

[9] Notamment la *redondance* (la même idée peut être exprimée par plusieurs méthodes, disposant chacune de plusieurs syntaxes) et la *contextualité* (la même instruction ou le même symbole peuvent avoir des interprétations différentes selon le contexte).

Une autre caractéristique importante de Perl est son ouverture : contrairement à l'immense majorité des autres langages, Perl a dès l'origine été explicitement conçu pour évoluer selon les commentaires des utilisateurs, pour parvenir à maturité avec l'aide active de la communauté des programmeurs, ainsi que pour s'adapter à tous les types d'outils et même aux autres langages.

Ainsi, le langage que nous allons vous présenter dans ce livre n'est pas seulement l'œuvre d'un seul homme ; il a évolué au fil des années en absorbant des milliers d'idées provenant d'utilisateurs, qui ont été intégrées au langage avec l'aide de plusieurs dizaines de développeurs. Perl n'est donc pas une simple création de Larry Wall, mais plutôt son enfant, qui a progressé par l'expérience et s'est enrichi de la critique et des propositions de ses utilisateurs. C'est pourquoi Perl est à la fois simple et de très haut niveau, expressif et complet, riche et efficace.

Qui utilise Perl ?

Perl compte environ un million d'utilisateurs de par le monde et une requête sur Google renvoie treize millions de pages : Perl est présent dès qu'il y a des informations à filtrer et manipuler, c'est-à-dire... partout.

Par exemple, il recense la population américaine, déchiffre le génome humain, établit les prévisions macroéconomiques à la réserve fédérale américaine, parcourt le web sous forme de robots (*crawlers*), gère les retraites des Suédois et analyse les statistiques de vols dans l'aéronautique. Il est utilisé par Amazon.com, Yahoo, Oracle, Siemens, Palm Inc., la Deutsche Bank et bien d'autres encore...

Ces succès à grande échelle ne doivent pas faire oublier que Perl est aussi largement utilisé dans une multitude de petites configurations (particuliers, associations et PME). Ce n'est pas un langage des extrêmes, qui ne serait adapté qu'aux micro-projets ou aux défis pharaoniques, c'est simplement un outil versatile qui est à l'aise dans toutes les situations.

Pourquoi un tel succès ?

Le succès phénoménal de Perl tient à trois ingrédients.

Le premier ingrédient n'est qu'indirectement lié à Perl : le monde informatique subit un mouvement de fond qui déplace l'affectation des ressources du *hardware*[10] vers le *brainware*[11]. Ce qui coûte cher, ce n'est plus le matériel informatique mais le développement de logiciels. Or, écrire un programme en Perl prend cinq à dix fois moins de temps qu'écrire le même programme en C : le confort du programmeur rejoint ici l'intérêt de l'entrepreneur. En outre, Perl n'est pas la propriété d'une entreprise, c'est un projet *open source* gratuit, comme Linux et les protocoles internet, ce qui minimise les investissements et garantit une haute qualité du code. C'est aussi un langage portable, qui fonctionne sans modification sous Windows (XP, NT, 2003, etc.), Unix (notamment Linux et BSD), Mac OS et bien d'autres architectures.

La deuxième raison du succès de Perl est son universalité, qui résulte de l'ouverture du langage voulue par Larry Wall dès l'origine : il sait manipuler toutes les sortes de fichiers, gérer des processus, piloter des bases de données, il maîtrise tous les protocoles réseau, les architectures client/serveur, etc.[12] En raison de toutes ces capacités, Perl est idéal pour interfacer plusieurs applications ; par exemple, il est très utilisé pour faire communiquer un serveur web[13] et une base de données[14]. Songez à internet : ce n'est, après tout, que la mise en rela-

[10] Terme d'origine anglaise qui désigne le matériel informatique à toutes les échelles, du composant au serveur. Il est souvent opposé au terme *software*, synonyme de « logiciel ».

[11] Jeu de mot sur *hardware* et *software* qui indique que la ressource précieuse est la pensée, le cerveau (*brain* en anglais).

[12] Perl est même utilisé dans des environnements tout à fait inattendus, comme les calculs scientifiques intensifs (avec la bibliothèque PDL) et les jeux (comme *Frozen Bubble*, bibliothèque SDL).

[13] Perl est intégré aussi bien à Apache (via mod_perl) qu'à l'ASP du serveur web (IIS) de Microsoft.

[14] Perl possède des « pilotes » pour toutes les bases usuelles, notamment Oracle, MySQL, PostgreSQL et DB2.

tion de plusieurs ordinateurs, mais ce lien décuple l'intérêt de chacun des postes. Perl joue un rôle analogue vis-à-vis des logiciels en leur permettant de travailler ensemble.

Enfin, la troisième raison du succès de Perl tient au langage lui-même. Du point de vue du programmeur, il est facile à apprendre, très agréable à utiliser, amusant, créatif et adaptable ; c'est un choix qui va de soi. Du point de vue de la machine, c'est un langage robuste, efficace et sécurisé qui a fait ses preuves. Ce n'est pas un hasard si Perl a reçu de nombreux prix, notamment celui de la *Free Software Foundation*. Quand on a goûté à Perl, il est difficile de revenir à un autre langage !

Les performances de Perl

On a coutume de diviser les langages informatiques en deux catégories : d'un côté les langages compilés, de l'autre les langages interprétés. Les premiers imposent une étape de traduction du code source (lisible par l'homme) en binaire (suite de 0 et de 1), l'avantage étant que l'ordinateur exécute très rapidement ce dernier puisqu'il est écrit dans sa propre langue. Les seconds sont lus ligne à ligne par l'ordinateur au moment de l'exécution, ce qui est bien plus lent, mais de la sorte le code source est toujours disponible. Où se situe Perl ?

Comme d'habitude, Perl n'est partisan ni de la première méthode, ni de la deuxième, il cherche à tirer le meilleur parti des deux. Un programme Perl est toujours distribué sous forme de fichier source, comme s'il était interprété, mais il est compilé au vol, avec de nombreuses optimisations (notamment sur les boucles), et le binaire ainsi produit est stocké dans la mémoire vive au lieu d'être écrit sur le disque. Ceci implique un léger surcoût au lancement (imperceptible pour l'utilisateur), puisqu'une compilation doit avoir lieu[15], mais en pratique c'est confortable et efficace.

[15] Ce surcoût peut toutefois être éliminé dans certains environnements exigeants ; ainsi, le module mod_perl du serveur web Apache conserve le binaire un certain temps en mémoire après la fin de l'exécution du

Lorsque des morceaux de code doivent impérativement être finement optimisés, Perl vous permet de les écrire dans un autre langage, par exemple C ou C++, et d'incorporer ensuite ces lignes à votre programme Perl. Vous pouvez ainsi bénéficier de la vitesse de développement caractéristique de Perl pour 90% du code, sans renoncer à des performances maximales pour les 10% restants. Une autre manière d'établir un compromis entre temps de développement et performance brute est d'écrire les prototypes en Perl, ce qui permet de tester de très nombreuses idées en un minimum de temps, et d'écrire la version finale en C.

Ne croyez pas cependant que Perl soit un escargot : un programme équivalent écrit en C sera au mieux trois fois plus rapide dans les contextes les plus exigeants[16], et en moyenne 20 à 50% plus rapide seulement. En pratique, il est rarement payant de se contraindre à utiliser un langage particulièrement exigeant comme le C : hormis pour les composants système lourdement sollicités, ce qui intéresse le programmeur c'est le temps total pour effectuer une tâche, c'est-à-dire le temps de programmation plus le temps d'exécution du programme. Bien rares aujourd'hui sont les programmes dont le temps d'exécution n'est pas négligeable devant le temps qu'il a fallu pour les écrire ! Le fort surcroît de temps nécessaire pour développer dans un langage de bas niveau est rarement compensé par un modeste gain d'efficacité. Par conséquent, si l'on ne tient pas pour nul le temps (et le plaisir) du programmeur, Perl est en général *nettement* plus rapide que C.

Au demeurant, le paramètre crucial dans le temps d'exécution d'un programme n'est pas le langage choisi pour l'écrire mais la méthode retenue pour résoudre le problème ainsi que la compétence du programmeur. Un programme Perl bien pensé sera toujours largement plus rapide qu'un programme naïf en C.

---

programme.

[16] Le langage C étant de très bas niveau, il est le plus rapide de tous les langages à l'exécution – après l'assembleur.

Que devez-vous apprendre ?

Sans hésiter, n'apprenez du langage Perl que le minimum utile à la réalisation de l'objectif que vous vous êtes fixé. La syntaxe étant simple et souple, cela vous demandera peu de temps et peu d'efforts. Plus tard, lorsque de nouveaux besoins apparaîtront, lorsque vous serez déjà un peu à l'aise avec ce que vous savez de Perl, vous apprendrez progressivement d'autres fonctionnalités, d'autres possibilités.

L'objectif premier de Perl n'est en aucun cas de présenter au monde une théorie harmonieuse, que vous devriez assimiler avant de faire vos premiers pas, mais de vous fournir des outils précis, efficaces et directement utilisables.

Ce n'est pas en lisant une grande quantité de documentation que vous progresserez en Perl, mais en le pratiquant, en faisant des essais, en voyant naturellement apparaître des besoins pour réaliser des tâches bien délimitées. Apprenez Perl comme on apprend une langue étrangère et non comme on apprend les mathématiques : des rudiments minimaux vous permettront de survivre au quotidien et vous serez sans peine amené, par l'usage, à accroître vos connaissances en consultant ponctuellement une documentation.

De nombreuses sources d'information sur Perl sont disponibles. Sur votre ordinateur, vous pouvez utiliser la commande `perldoc`, installée en même temps que Perl, pour rechercher la syntaxe exacte d'une fonction (`perldoc -f` *fonction*), consulter la FAQ (`perldoc -q` *mot-clef*) ou approfondir un thème (`perldoc perl`). De nombreux sites web offrent des documentations gratuites sur Perl, par exemple `perldoc.com` ; un moteur de recherche vous en montrera bien d'autres. Vous trouverez une aide bienveillante et personnalisée sur les forums de discussion comme `fr.comp.lang.perl` . Enfin, d'épais ouvrages de bonne qualité sont publiés aux éditions O'Reilly si vous ressentez le besoin de livres exhaustifs.

Les objectifs de cet ouvrage

Nous vous proposons dans cet ouvrage un panorama transversal du langage Perl, agrémenté de nombreux exemples et exercices, qui vous montrera les bases (installer Perl, écrire un premier programme) et les fondements du langage (structures de données, boucles, expressions régulières) avant d'aborder des sujets plus pointus (réaliser des sous-programmes, créer et utiliser des bibliothèques) et enfin quelques exemples d'usages avancés à vocation professionnelle. Ce livre ne vous apprendra pas *tout* ce que l'on peut savoir sur Perl, mais vous serez déjà opérationnel dans quelques pages.

Progressez à votre rythme dans la lecture, lisez en détail tous les exemples pour vérifier que vous les comprenez parfaitement et ne négligez pas les exercices (ils se font de tête) : c'est par la pratique que l'on apprend. Si vous disposez d'un ordinateur à portée de main, testez les morceaux de code, modifiez-les, améliorez-les et, surtout, jouez avec : Perl n'est pas qu'une tronçonneuse suisse, c'est aussi un langage amusant qui vous fera passer de bons moments.