

Rechercher



Les personnages clés de l'informatique : la naissance d'UNIX

Licence CC BY-SA

Biographies de Ken Thompson, Dennis Ritchie, Brian Kernighan et Rob Pike

Auteurs :  Renault  jmiven

informatique c ordinateur unix bell labs

Catégories : [Programmation et algorithmique](#) , [Systèmes d'exploitation](#) et [Histoire](#)

Temps de lecture estimé : 44 minutes

Dernière mise à jour samedi 18 mai 2019 à 11h39

Récap' communautaire #13 — Mars 2019

Récap' communautaire #14 — Avril 2019

Cet article est le [premier d'une longue série censée rendre hommage à des personnes](#) qui ont façonné profondément l'histoire de l'informatique en tant que science et en tant que technique.

Notons que seuls les sujets concernant les Logiciels Libres, les débuts de la micro-informatique et UNIX sont prévus à ce stade. Toute initiative pour prendre en charge les autres périodes ou domaines est la bienvenue.

Aujourd'hui, nous allons aborder la naissance d'un système d'exploitation de 1969, dont les principes sont toujours d'actualité 50 ans plus tard. De plus, de part son influence, nous allons voir des personnalités qui se sont basées sur UNIX pour aller plus loin encore.

La naissance d'UNIX est particulièrement liée à l'histoire des Bell Labs, des laboratoires qui verront la naissance du transistor, du laser, d'UNIX, du langage C et C++ ou encore de la fibre optique.



Logo des Bell Labs de 1969 à 1983



Logo des Bell Labs de 1984 à 1995

Lucent Technologies
Bell Labs Innovations

Logo des Bells Labs de 1996 à 2015



- [Ken Thompson](#)
- [Dennis Ritchie](#)
- [Brian Kernighan](#)

- [Rob Pike](#)
- [Sources](#)

Ken Thompson

Naissance d'UNIX

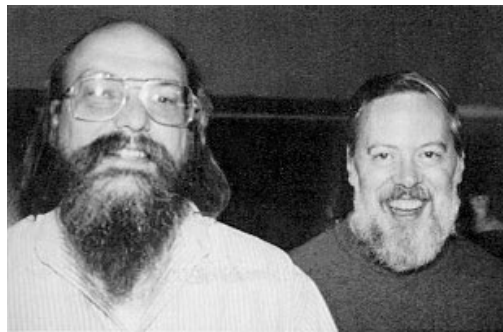
Kenneth Lane Thompson est né le 4 février 1943 aux États-Unis à la Nouvelle-Orléans.

En 1966, il ressort de la célèbre université de Berkeley, en Californie, avec une maîtrise en électronique et science informatique. Peu après, il va rejoindre l'équipe de développement au Bell Labs du système d'exploitation Multics où il va rencontrer Dennis Ritchie.

Le système Multics hérite d'un système plus ancien CTSS, dont la principale caractéristique était d'être à temps partagé et non en traitement par lot. Plusieurs utilisateurs pouvaient utiliser la machine en même temps pour des tâches différentes. Le système d'invite de commande et de traitement de texte vont être les précurseurs du shell et de Troff au sein d'UNIX.

Mais en plus de cet héritage, Multics va inclure le support du terminal distant permettant à chacun d'exécuter des tâches depuis son terminal dans son bureau sur l'ordinateur central. Pour mener cela à bien, un système d'anneaux de sécurité va se mettre en place permettant aux applications de se lancer avec des droits plus ou moins restreints. Technique que l'on va retrouver dans l'architecture des processeurs modernes comme le x86. Le système de fichier hiérarchisé va servir de référence à l'arborescence d'UNIX.

Cependant, en 1969 le travail de Dennis Ritchie et de Ken Thompson s'arrêtera sur Multics, les Bell Labs se retirent du projet. Cela va laisser le champ libre à la création d'UNIX. Ken profita du temps libéré pour créer son nouveau système, un prototype de ce qui deviendra UNIX sur un PDP-7 et le tout écrit dans un langage d'assemblage. Son collègue Brian Kernighan suggéra de nommer le système en Unics car contrairement à Multics, il n'y avait qu'une seule méthode pour faire les choses. Pour des raisons obscures, sans doute commerciales, Unics deviendra UNIX.



Ken Thompson (à gauche) et Dennis Ritchie (à droite)

Le problème de ce système est sa difficulté à être portée. Ken travaillait en parallèle sur un langage de programmation nommé le B qui descend du BCPL. Ce langage est le prédécesseur du C. Cependant UNIX ne sera pas porté en B, en effet ce langage n'était pas adapté à un portage sur une autre machine aisément en plus de performances jugées médiocres. UNIX a failli être porté à ce moment-là en TMG ou Fortran.

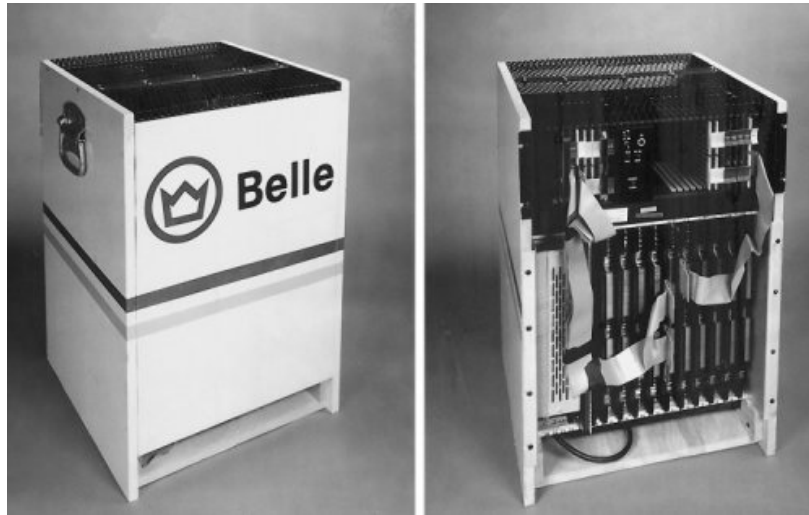
En 1969 et 1970, Ritchie travailla sur le langage C pour succéder au B. Après sa disponibilité en 1971, Ken s'attela à porter UNIX en C sur un PDP-11, cet ordinateur ouvrait la porte du traitement de texte avec le langage roff et ses descendants futurs. À partir de 1972, UNIX est enfin portable et sa diffusion pourra bientôt commencer. Ken sera très impliqué dans son développement jusqu'à la diffusion de la version 6 en 1975. Les versions 4, 5 et 6 avaient apporté notamment le pipe permettant de lier les programmes entre eux grâce à l'apport de Doug McIlroy. Cette année-là UNIX commença à quitter le Bell Labs. Il prendra un congé sabbatique pour apporter UNIX dans l'université de Berkeley. Université qui sera à l'origine du schisme d'UNIX avec la naissance de BSD.

Pendant son travail sur UNIX, il sera à l'origine des utilitaires grep et ed, l'éditeur de texte qui servira de base à la création de vi et Vim. En effet, à cette époque les informaticiens étaient souvent proches des mathématiciens et Ken Thompson va exploiter des

travaux récents dans le milieu. Dans les années 1940 et 1950, Michael Rabin, Dana Scott et Kleene vont travailler sur la formalisation des expressions régulières. Ken Thompson va être le premier à exploiter cette découverte mathématique. Il ira plus loin en publiant [un article en 1968 à ce sujet](#). Il en découlera un algorithme qui rend triviale la compilation d'une expression régulière en un automate fini non-déterministe (AFN) et la conception d'une *machine virtuelle* permettant d'exécuter un tel AFN de façon performante. Ce travail sera utilisé dans l'éditeur de texte *qed* sur CTSS avant de poursuivre ce travail sur UNIX avec *ed* et *grep*. Par la suite la plupart des logiciels de gestion du texte ou les langages de programmations vont reprendre ces concepts.

Ken Thompson n'a eu aucun regret sur la conception d'UNIX, quand on lui demandait ce qu'il aurait changé avec le recul dans UNIX, il répond avec humour « J'aurais orthographié *creat* avec un *e* ». La fonction système *creat()* n'est pas correctement orthographiée sans raison apparente, l'orthographe réelle rentrant dans les 6–8 caractères qui était une limite à cette époque.

Les jeux d'échecs



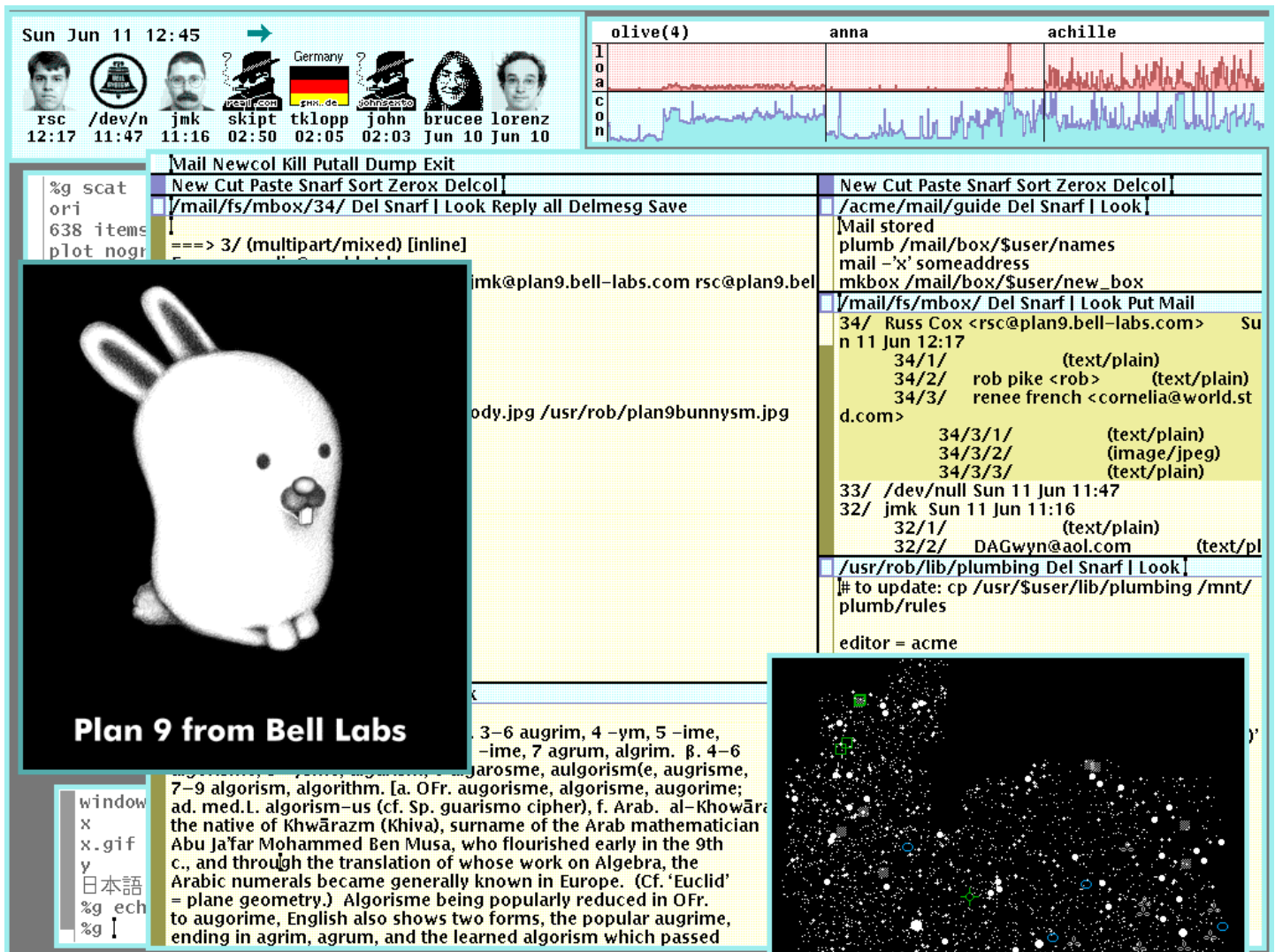
L'ordinateur Belle

En parallèle de son développement sur UNIX, il va s'atteler à une autre passion, les échecs avec l'ordinateur Belle. Avec Joe Condon, il va modifier un PDP-11/23 pour créer un ordinateur capable de jouer aux échecs, de manière similaire au célèbre projet Deep Blue de IBM deux décennies plus tard. Entre 1978 et 1986, cet ordinateur gagnera de nombreux championnats dont celui des ordinateurs d'échecs en Amérique du Nord. L'ordinateur a été confisqué temporairement par les États-Unis lors d'une volonté de participer à un match soviétique en URSS. En pleine guerre froide, cela a été annulé pour éviter un transfert illégal de technologies à une puissance ennemie.

Pendant ces années-là, il travaillera également sur les tables de finale aux échecs pour trouver les coups lorsqu'il reste 3 à 5 pièces qui permettent de gagner. Pour sa part, il se concentrera sur la manière d'obtenir la promotion des pions restants afin de renverser la situation du jeu en fin de partie (sans préciser si cela mènera effectivement à une victoire ou à une défaite). Cela est à mettre en relation avec son travail sur le moteur de base de données (non relationnelles) *dbm* en 1979 qui met l'accent sur la performance par le stockage de données via une clé primaire et des fonctions de hashages extensibles.

La création de Plan 9

Il ne s'arrête pas là. En 1984 il publiera également [un article célèbre](#) à propos de la confiance qu'on peut accorder à un compilateur. Ensuite, à la fin des années 80, il va s'occuper de son dernier projet au sein des Bell Labs, le système d'exploitation Plan 9. Selon lui, les défauts d'UNIX étaient trop profonds pour nécessiter de simples correctifs, d'où ce changement de projet et d'architecture. Tout d'abord, depuis la naissance d'UNIX l'informatique a eu des évolutions importantes : interface graphique, le réseau mais aussi l'internationalisation du secteur et ces éléments ne sont pas inclus dans l'architecture du système ce qui la rend moins puissante.



L'interface de Plan 9 : rio

Plan 9 va inclure dans le concept *tout est fichiers* de son prédécesseur le réseau et l'interface graphique. Le dialogue avec ces concepts se fera à travers le système de fichier virtuel `/proc`, pour les processus, `/net` pour la pile réseau, `/dev` gère en plus des périphériques virtuels pour l'interface graphique, etc. Ceci est plutôt novateur pour dialoguer avec l'ensemble du système sans modifier le code source et sans relancer la machine. Le protocole 9P sera conçu pour limiter la quantité d'appels systèmes pour exploiter le réseau ou le dialogue entre processus ou encore les systèmes de fichiers distribués. L'union des répertoires fera partie intégrante du système pour permettre de fusionner des sous-espaces du système de fichiers au-delà d'un simple lien symbolique avec la commande `bind`.

La dernière problématique de Plan 9 est l'internationalisation, à savoir que tout le système puisse interpréter les langues non-latines comme le chinois et qu'un système non chinois puisse interpréter les caractères chinois donc sans changer forcément d'encodage de caractères. Au cours d'un repas au restaurant avec Rob Pike, ils vont mettre en place ce qui sera l'encodage UTF-8 au sein du format Unicode et qui permet effectivement avec un seul système d'encoder n'importe quel caractère existant, d'être compatible avec les anciennes normes comme l'ASCII mais aussi d'économiser de la place, car les caractères latins ne seront encodés que sur un octet contre quatre pour un idéogramme chinois. Ce système de caractères encodés sur des longueurs variables rendra ce système plus complexe à réaliser. Aujourd'hui UTF-8 est presque partout, dans la plupart des systèmes d'exploitations, des logiciels ou des langages de programmations modernes comme sur ce site.

Le langage Go

Il prendra sa retraite en 2000 des Bell Labs, sera un conseiller pour Entrisphere avant de rejoindre Google en 2006. C'est là-bas qu'il concevra le langage de programmation Go avec encore et toujours Rob Pike qu'il a rejoint chez Google. Fort de son expérience passée, après avoir reconçu UNIX avec Plan 9 il va s'atteler à essayer de reconcevoir le langage C avec les dernières évolutions de l'informatique. Ce langage va notamment reprendre l'aspect de programmation concurrente nécessaire avec les ordinateurs à plusieurs cœurs, incorporer un ramasse-miette pour la gestion de la mémoire, et un typage plus fort pour être plus sûr.

Épilogue

Ken Thompson est comme nous avons pu le voir un programmeur d'exception qui a su participer à de nombreux projets d'envergure et dont la plupart ont été une grande source d'inspiration pour les logiciels actuels. C'est pour cela qu'il a été, comme avec ses collègues des Bell Labs, souvent récompensé. Il a reçu notamment le prix Turing, la National Medal of Technology et la médaille Richard Hamming en compagnie de Dennis Ritchie.

Dennis Ritchie

Début de carrière

Dennis MacAlistair Ritchie, est né le 9 septembre 1941 à Bronxville aux États-Unis.

Comme de nombreux informaticiens de sa génération, il a suivi une formation très portée sur les mathématiques appliquées et la physique à Harvard. Peu après il rejoindra son père aux Bell Labs, où en 1968 il travailla sur ALTRAN qui est une extension du langage FORTRAN portant sur l'algèbre rationnelle. Il y rencontra Ken Thompson pour travailler sur Multics peu après.

Le langage C

Avant de concevoir le C, il faut se remettre dans le contexte de l'époque. À la fin des années 60, Thompson et Ritchie travaillaient beaucoup sur le PDP-7, une machine peu puissante qui ne pouvait manipuler que des programmes simples dans des langages qui le sont également. Comme les langages populaires à l'époque ne convenaient pas, tels que FORTRAN ou COBOL car trop verbeux et complexes, Thompson écrivit le langage B en reprenant les concepts d'un langage déjà simplifié : le BCPL. Le B était vraiment simple, typiquement le seul type existant était le mot mémoire ce qui le rendait portable mais peu flexible. Quand UNIX a eu plus de succès, le développement devait se produire sur un PDP-11 nouvellement acquis qui avait deux particularités par rapport au PDP-7. En plus d'avoir plus de mémoire, il pouvait manipuler des octets en plus des mots standards de 16 bits. Comme les caractères sont encodés principalement sur 8 bits, il fallait exploiter convenablement cette faculté de la machine. En plus, le PDP-11 était doté d'un jeu d'instruction permettant la manipulation des flottants qui étaient de fait absents du langage B.

C'est à partir de là que Ritchie commença à écrire un *New B*, la base était donc la création des nombreux types tels que char, float et int. Seulement il ira bien plus loin qu'une simple modification du B. Plutôt que d'avoir un type unique et dont l'interprétation du type se ferait automatiquement suivant le contexte (typiquement entre pointeur et entier), il ajoutera un jeu de types complets avec les pointeurs, les tableaux et les structures ce qui permettrait à terme de simplifier le portage d'UNIX mais aussi son évolution sur le PDP-11.

Nous sommes en 1973, le langage C est maintenant assez complet. Suffisamment pour que le portage d'UNIX puisse se faire cette année-là, tâche qui a été confiée à son plus fidèle collègue Thompson et lui-même. Comme nous pouvons le voir, le C est un langage qui a été conçu comme un assembleur évolué et portable. Le tout devait bien entendu rester très léger et peu verbeux, valeurs que Dennis Ritchie tiendra à conserver tout le long de sa vie.

Contrairement au B qui était peu répandu et peu décrit car il n'a pas existé très longtemps, le succès d'UNIX débutant il était nécessaire d'écrire un ouvrage présentant l'ensemble du langage C. Ce livre écrit en 1978 par Kernighan, qui s'occupait de l'essentiel du texte, et Ritchie, qui rédigeait les annexes et le code source, sera surnommé *K&R* et décrit l'ensemble des possibilités du langage. Il popularisera notamment la pratique du *Hello World* en guise de premier programme d'exemple. De 1978 à 1989, il servait avec le compilateur *pcc* de références sur les pratiques du langage C au point d'en être des normes *de facto*. De plus, pour des raisons d'économie de papier, les codes sources de ce livre utilisent un style d'indentation très compact tout en conservant un maximum de lisibilité. Ce style très codifié aujourd'hui est surnommé *K&R* en honneur à son origine et est l'un des styles les plus utilisés encore aujourd'hui, notamment dans le code source du noyau Linux.

Ce livre, tout comme UNIX, ont amorcé la popularité du C dans le milieu informatique. Il ajouta avant la normalisation du langage les énumérations, les unions, les types non signés et d'autres choses encore. Quand l'ANSI et l'ISO prennent part à la normalisation du langage à la fin des années 80, Dennis prôna de défendre la cohérence du C, sa légèreté et d'éviter les ambiguïtés.



Réception du National Medal of Technology en 1998 remis par le président des États-Unis Ken Thompson (à gauche), Dennis Ritchie (au centre) et Bill Clinton (à droite)

Après la normalisation du langage, il réédita avec Kernighan le *The C programming language* pour respecter la norme, après quoi il ne prit plus part aux évolutions du C standard. Notons que durant les années 80, il travailla beaucoup aux côtés d'un certain Bjarne Stroustrup qui sera en 1986 le père du langage C++. Bjarne garda un grand respect envers le travail de Dennis dont il dira :

Si Dennis avait décidé de consacrer cette décennie à des mathématiques ésotériques, Unix aurait été mort-né.

En parallèle du C, sur Unix

Ritchie travailla beaucoup sur Unix pendant qu'il poursuivait ses travaux sur le C. En plus de l'avoir porté sur de nombreuses architectures comme le Interdata 8/32, il faisait partie du groupe de travail sur System V 7e édition. Ses travaux aboutirent notamment sur les *ioctl* et *IPC* pour faciliter la discussion entre processus et entre les processus et les entrées/sorties du système. L'ensemble de ses travaux sur le sujet ont été inclus dans la norme POSIX.

Il travailla sur de nombreuses commandes Unix dont : *cc*, *ld*, *as*, *db*, *strip*, *bcd*, *cmp*, *date*, *du*, etc. Comme nous pouvons le voir, l'essentiel porte sur le développement d'un programme avec le compilateur, l'éditeur de lien, le débogueur ou encore l'assembleur...

Il s'occupa également d'un langage de traitement de macros M3. En effet à cette époque utiliser l'assembleur était courant et demandait des instructions très répétitives pour réaliser des tâches simples tels que l'affichage de caractères. Les macros permettaient de définir des raccourcis pour inclure des bouts de code redondant facilement dans le code source final, M3 servant à effectuer cette inclusion automatiquement. En 1977, il améliora les possibilités de M3 avec Kernighan pour obtenir M4.

Après le C

L'équipe originelle d'Unix prit part à son évolution et Ritchie n'y fit pas exception. En 1996, profitant de la restructuration des Bell Labs, il devient un responsable du pôle de recherche sur les logiciels systèmes. Il donna des contributions à l'élaboration des systèmes d'exploitation Plan 9 et Inferno mais aussi du langage de programmation Limbo dont le lien Limbo-Inferno rappelle fortement le lien C-Unix. Ressemblance également troublante, Kernighan et Ritchie écrivirent séparément des textes présentant le langage Limbo similaire à celui qu'ils avaient rédigé pour le C. Ritchie prit le nom de *The Limbo programming language*.

Fin de sa vie

Après une carrière brillante et avoir obtenu de nombreuses récompenses pour ses travaux, il va se retirer de l'informatique en 2007. Plutôt solitaire et très malade, il décèdera au début du mois d'octobre 2011, son corps ayant été retrouvé le 12.

Son décès fait suite à celui de Steve Jobs quelques jours plus tôt, le second ayant subi une telle médiatisation que le décès de Dennis Ritchie se déroula dans une grande indifférence. Mais nul doute qu'avec un héritage vivace de ses projets 40 ans après ses débuts, l'histoire saura retenir son nom.

Épilogue

Comme nous avons pu le voir, Ritchie a eu une carrière fortement liée à son langage de programmation : le C ; dont son histoire se mêla très fortement à celui d'Unix. Ses choix de conceptions du C qu'il a voulu maintenir, et que nous avons légèrement abordés, expliquent encore la main mise du C dans la programmation système et les systèmes embarqués aujourd'hui. C'est pour cela qu'il a été, comme avec ses collègues des Bell Labs, souvent récompensé. Il a reçu notamment le prix Turing, la National Medal of Technology et la médaille Richard Hamming en compagnie de Ken Thompson.

Brian Kernighan

Brian Wilson Kernighan est né le 1^{er} janvier 1942 à Toronto au Canada.

Il a suivi une formation en physique à l'université de Toronto pour y obtenir un Bachelor en 1964. Ensuite il suivra un PhD en ingénierie électrotechnique à l'université de Princeton. C'est par la suite qu'il intégra les équipes de Bell Labs autour d'Unix dans le département informatique.

Un homme aux multiples langages

Aux alentours de 1977–1985, il sera à l'origine ou le mainteneur de quatre langages d'envergure : AWK, M4, AMPL et troff.

L'aventure troff

Troff est un logiciel de formatage de texte qui a débuté sa vie au sein des projets CTSS et Multics dont nous avons évoqué l'existence dans la biographie de Ken Thompson. C'est tout logiquement qu'il se retrouve sur Unix grâce au portage de Joseph Ossanna avec de nombreuses améliorations. À l'heure où les interfaces graphiques n'existaient pas encore, troff était le moyen de formater du texte à l'aide de macros pour ensuite en avoir un affichage sur écran ou imprimante correspondant. C'est une version plus primitive et simple que LaTeX par exemple, qui remplit le même rôle aujourd'hui.

C'est cette faculté qui l'a rendu populaire au sein des Bells Labs pour la rédaction des documents, dont des brevets, mais aussi pour la rédaction des pages de manuels d'Unix (fonctionnalité encore d'actualité sur Linux et la famille BSD).

En 1977, Joseph Ossanna meurt. Brian Kernighan reprit le flambeau du développement et de la maintenance de troff. Tout d'abord il réécrit l'ensemble du code pour en modifier l'architecture pour le renommer *Ditroff* comme *Device Independent troff* car l'objectif était de se rapprocher de l'architecture des compilateurs : un front-end pour transformer le langage source en langage universel intermédiaire pendant que le back-end traduit le langage intermédiaire en affichage destiné à un moniteur ou une imprimante particulière (qui sont souvent incompatibles entre eux).

Quand d'autres projets autour de troff apparaîtront après 1979, tous reprendront la description du langage intermédiaire tel que décrit par Kernighan. Mais avant cela, Kernighan ajouta le support des images et des notations mathématiques en collaboration avec Lorinda Cherry.

Son travail autour de troff s'explique en grande partie par la quantité d'ouvrages qu'il écrivit, comme *The C Programming Language* qu'il exploita pour sa mise en page.

AWK

AWK c'est l'acronyme de ses créateurs que sont : Alfred Aho, Peter Weinberger et Brian Kernighan. Leur travail qui débuta en 1977 avait pour objectif de concevoir un langage de traitement de texte simple et puissant pour aussi, contournant certaines limitations du programme sed dans le cadre de traitement des nombres avec des tableaux, de passage de document balisé. En effet, sed a plus été conçu comme un ed (l'éditeur de texte de Ken Thompson) non interactif ce qui le rend moins générique et puissant qu'un langage de programmation complet.

En 1985 il participa à l'amélioration de `awk` pour apporter les fonctions définies par les utilisateurs, les multiples sources d'entrées et les expressions régulières. Pour étendre l'adoption de `AWK`, il rédigea un livre *The AWK programming language* en collaboration des autres auteurs du langage. Ce livre servira, comme le K&R pour le C, de référence pour la standardisation du langage ici dans POSIX.

Ce langage sera notamment une des inspirations fortes pour l'élaboration du langage Perl.

Le langage des macros M4

En 1976, il s'attela à écrire le livre *Software Tools* avec P.J. Plauger où il décrit notamment le concept d'un langage de macro qui servit ensuite à Dennis Ritchie pour produire `M3` pour l'ordinateur `AP-3`. L'année suivante, Kernighan et Ritchie ont œuvré pour améliorer le langage `M3` de Ritchie qui se nomme aujourd'hui `M4`.

Ce n'est qu'après qu'il l'aide à [définir et publier autour de la nouvelle version du langage de macro M4](#), avec 21 macros internes.

Le `M3` a servi de moteur pour le préprocesseur du rational `FORTRAN` tandis que le `M4` remplira ce rôle également pour le C et `COBOL`. Aujourd'hui il sert encore comme outil essentiel à des programmes comme `SELinux`, `sendmail` ou `autotools` essentiellement à travers l'implémentation de GNU.

L'AMPL

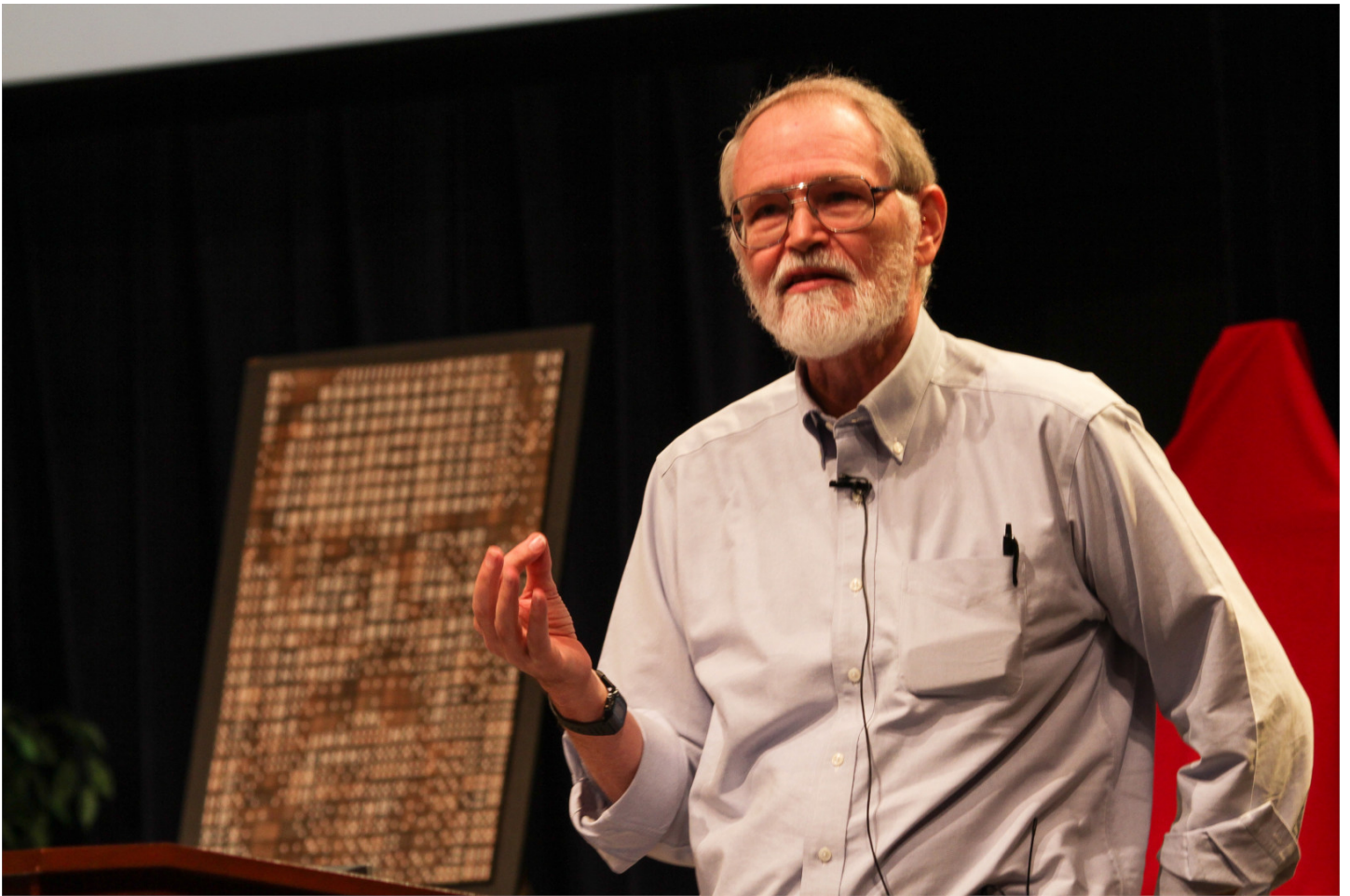
A *Mathematical Programming Language* a été conçu en 1985 par Robert Fourer, David Gay et bien entendu Brian Kernighan. Comme son nom l'indique, c'est un langage de programmation plus orientée sur la résolution de problèmes mathématiques. En ce sens, sa syntaxe est plus concise, haut niveau et proche des notations mathématiques pour qu'un mathématicien puisse le prendre en main.

Ce langage est accompagné d'un certain nombre de résolveurs pour résoudre un certain nombre de problèmes ou d'optimisations mathématiques. Par exemple les optimisations linéaires, quadratiques, non-linéaires, l'optimisation globale, etc.

Kernighan sera bien entendu auteur du livre fondateur du langage [AMPL: a Mathematical Programming Language](#).

Durant sa conception, il a été avec Rob Pike l'un des responsables de l'utilisation de `LEX` et de `YACC` pour la mise au point de l'interpréteur de syntaxe qui changeait beaucoup au début. Il a bien sûr aidé à la mise au point de certains modèles mathématiques comme `EGYPT2`.

Ses différents auteurs recevront le prix ORSA/CSTS en 1993 et le prix INFORMS Impact en 2012. En 2003 ils fondent l'entreprise AMPL Optimization LLC pour gérer l'évolution du langage en reprenant l'activité de Lucent Technologies (qui a racheté les Bell-Labs et était de fait le dépositaire du langage).



Cérémonie d'hommage à Dennis Ritchie avec Brian Kernighan en 2012

Un mathématicien également

Comme nous l'avons vu précédemment, Kernighan était impliqué avec ses collègues sur les mathématiques, suffisamment pour donner le coup de main à l'élaboration d'un langage pour cela.

Il travaillera avec Shen Lin autour de deux problèmes connus dits NP-complets que sont : le voyageur du commerce et le partitionnement de graphe. Cela donnera lieu respectivement à l'heuristique de Lin-Kernighan en 1973 et à l'algorithme Kernighan-Lin en 1970.

Les deux reposent sur la même logique à savoir de traiter la solution de manière locale, en opérant sur les voisinages d'un nœud donné sans considérer le graphe dans son ensemble. Et d'effectuer itérativement des permutations au fur et à mesure au sein de celui-ci. Ces résultats ne sont pas les plus performants et ne fournissent pas forcément les solutions optimales mais ils ont l'avantage d'être simples à mettre en œuvre et de fournir un compromis intéressant.

Ils seront améliorés par d'autres chercheurs avec le temps, mais le concept de base reste. C'est par exemple encore utilisé pour disposer sur un circuit intégré les portes logiques afin d'améliorer la densité des composants et l'organisation du circuit.

Pour finir enseignant

En tant qu'auteur, il rédigea avec Rob Pike deux ouvrages intéressants que sont *L'environnement de programmation UNIX* et *La programmation en pratique*. Ensuite il quitte les Bell Labs en 2000 pour devenir enseignant à l'université de Princeton au département informatique.

Il est toujours actif à ce jour à ce poste. Il ne boude pas son plaisir pour l'écriture, co-rédigeant également l'ouvrage *The Go Programming Language*, sur le langage Go conçu par ses anciens collègues Rob Pike et Ken Thompson.

Épilogue

Nous l'avons vu que Brian était un auteur de référence dans les Bell Labs. Il a rédigé la plupart des ouvrages techniques de références des langages qu'il a conçus ou ceux de ses collègues. Cette implication dans l'écriture l'a mené à travailler significativement sur *troff*, en étant l'un de ses principaux utilisateurs.

Son caractère pluridisciplinaire l'a mené également à travailler sur des projets de recherches en électronique, ce qui lui permet de travailler sur des optimisations mathématiques et algorithmique dans ce but. Cela le rend sans doute plus atypique que ses collègues présentés ici qui ont travaillé surtout sur l'informatique.

Rob Pike

Robert C. Pike est né en 1956 au Canada. Il sera diplômé d'un bachelor de l'université de Toronto avant d'obtenir un master à l'institut de technologie de Californie en informatique. Juste après la fin de ses études en 1979, il rejoindra l'équipe de développement d'UNIX pour travailler sur la version 8 du système.



Rob Pike

Débuts aux Bell Labs sur UNIX

Ses débuts coïncideront avec l'arrivée progressive des interfaces graphiques dans le milieu. C'est ainsi qu'il a conçu avec Bart Locanthi Jr. en 1982 *Blit*, un terminal graphique en bitmap. Ce terminal monochrome verdâtre avait une conception plutôt originale. Contrairement à la plupart des terminaux qui sont assez peu puissants, leur but étant juste de service d'interface visuelle et de saisis entre la machine et l'utilisateur, ce terminal dispose d'une capacité de calcul plus élaboré.

En fait pour éviter que la liaison série entre l'hôte et le terminal soit saturé pour échanger en permanence les ordres graphiques, et limiter la latence, le terminal avait la capacité de récupérer le logiciel de l'hôte pour l'exécuter en local. Ainsi les tâches trop

gourmandes graphiquement sont exécutées par le terminal, délaissant le reste à l'hôte ce qui en fait un petit système distribué.

À cette occasion, il a travaillé sur les premiers environnements graphiques d'UNIX que sont *mpx* et *mux* pour les versions 8 et 9 du système. Ce travail sur le sujet servira à la mise au point du protocole du serveur d'affichage X. [Il a publié une vidéo de présentation de son travail de l'époque.](#)

C'est durant cette période qu'il co-rédige le livre *L'environnement de programmation d'UNIX* en 1984 avec le fameux Brian Kernighan. Il a également conçu, avec l'aide de Ken Thompson, l'éditeur de texte *sam* qui est une évolution de *ed* et qui profite également des capacités graphiques nouvelles d'UNIX.



Le terminal Blit

L'aventure Plan 9...

S'il n'a pas assisté aux débuts d'UNIX, ayant rejoint le projet que plus tardivement, il est l'un des membres essentiels du projet Plan 9 avec Ken Thompson dès la fin des années 80. Grâce à son expérience d'UNIX, mais aussi sur des interfaces graphiques et des systèmes distribués, il est le plus à même à régler les défauts d'UNIX dans ce nouveau système.

Il participe à rendre le système dès le départ distribué, à travers le concept des espaces de noms mais aussi par un protocole orienté message pour le système de fichier : 9P. Il fait en sorte également que l'interface graphique soit aussi intégré dans le système que ne l'est la *shell* par exemple. Il sera ainsi le développeur principal des gestionnaires de fenêtres de base de Plan 9 que sont *8 1/2* et son successeur *rio*.

En 1992 avec Ken ils mettent au point l'encodage de caractères UTF-8 qui sera pleinement employé dans Plan 9 pour des raisons d'internationalisation.

Il a porté *sam* sur Plan 9 avant d'écrire son successeur *acme* et son écosystème qui est un environnement de développement plus complet, étant capable également de lire le courrier électronique et d'autres choses encore !

À titre purement anecdotique, sa femme Renée French a dessiné la mascotte du système, un lapin nommé Glenda.

... puis Inferno

En 1995, fort de son expérience sur Plan 9, il participe avec Phil Winterbottom et Sean Dorward à la conception d'un nouveau système d'exploitation nommé Inferno.

L'objectif est de succéder à Plan 9, en reprenant tous ses concepts d'origine. La nouveauté est de faire en sorte qu'il soit le plus portable possible et sur une variété de machines bien plus large. Allant du système embarqué au centre de données.

Pour cela, ils créent le langage Limbo. Cette évolution très poussée du C doit être aussi liée au système Inferno que ne l'est le C avec UNIX. Par rapport au C, il est adapté à la programmation concurrente, il dispose d'un ramasse-miette automatique et d'un typage fort. Pour faciliter le portage, il reposera sur une machine virtuelle nommée Dis. Le compilateur génère donc des objets dans un langage intermédiaire que Dis transforme en instructions machines.

Mais un autre langage fait son apparition dans cette période qui vise des buts similaires : Java. Lucent, qui vient de racheter les Bell Labs en 1996, passe un accord avec Sun pour obtenir une licence Java et sonne le glas du développement de Limbo et donc de fait d'Inferno qui seront revendus quelques années après.

Il reprend du coup le développement de Plan 9 jusqu'à sa dernière version en 2002.



Glenda, la mascotte de Plan 9

À la fin des années 90, il co-écrit le livre *La programmation en pratique* toujours avec Brian.

Départ chez Google

Comme Ken, il quitte les Bell Labs pour Google.

Il travailla d'abord sur le langage de script Sawzall, dont l'objectif est de traiter facilement les logs de Google. Ces logs très nombreux et distribués à travers beaucoup de serveurs nécessitent des traitements spécifiques. Jusqu'alors ils appliquaient la technique MapReduce en C++ ou Java ce qui rendait l'écriture longue et trop verbeuse. Ce que Pike résolu avec son équipe avec ce nouveau langage.

Puis il rejoint Ken et Robert Griesemer pour travailler ensemble sur le projet d'un nouveau langage qui deviendra Go. Rob reprit de nombreux concepts de Limbo qu'il a conçu pour permettre la programmation concurrente et performante. Étant donné que Go ne vise pas à devenir le langage de référence d'un système d'exploitation, en étant plutôt orienté programmation système ou pour écrire des serveurs capables tenir une forte montée en charge, la machine virtuelle n'est pas retenue dans la conception.

Il travaille toujours chez Google à ce jour.

Épilogue

Des quatre développeurs principaux de ce qui deviendra la famille de systèmes d'exploitation UNIX et sa lignée directe, il est le plus jeune. De part cette caractéristique il a travaillé essentiellement sur l'apport des innovations informatiques dans UNIX

comme le concept d'interface graphique. Il est également le seul du groupe à avoir établi la conception des deux successeurs d'UNIX que sont Plan 9 et Inferno qui resteront cependant dans l'ombre de leur ancêtre.

Sources

Général :

- [Wikipédia](#)
- [Présentation résumée des membres des Bell Labs \(avec interviews, photos et publications\)](#)
- [Vidéo d'archive d'At&T à propos d'UNIX](#)
- [Histoire de M4](#)
- [Histoire de l'UTF-8](#)
- [Histoire du langage C par Dennis Ritchie](#)

Ken Thompson :

- [Historique de Ken Thompson](#)

Dennis Ritchie :

- [Page personnelle de Dennis Ritchie](#)
- [Page IEEE portant sur Dennis Ritchie](#)
- [Article sur l'histoire du C par Dennis Ritchie lui-même](#)
- [Histoire du langage C résumée sur Développez.com](#)

Brian Kernighan :

- [Page personnelle de Brian Kernighan](#)
- [L'histoire simplifiée de troff sur linuxfr.org](#)

Rob Pike :

- [Page personnelle de Rob Pike](#)

Nous venons de voir comment une poignée d'hommes a pu jeter les bases d'une informatique encore très utilisée aujourd'hui. Que ce soit le langage C ou UNIX, ces éléments ont grandement influencé les systèmes d'exploitation modernes (Linux et macOS sont très liés à UNIX, Windows en a tiré une partie de son architecture actuelle aussi) ou même les langages, le C++ qui en est le direct descendant ou d'autres qui ont repris partiellement sa syntaxe comme Java ou Python.

Sans leurs travaux, de larges pans de notre environnement informatique seraient différents. Et nous allons voir dans un prochain article ceux qui ont conçu des systèmes d'exploitations libres à partir d'UNIX via les projets GNU, BSD et Linux.

Récap' communautaire #13 — Mars 2019

Récap' communautaire #14 — Avril 2019

Ces contenus pourraient vous intéresser



Hommage à Ray Tomlinson, l...
Retour sur l'histoire du mail !

13 dans Autres (informatique) et Histoire
Vendredi 11 mars 2016 à 11h03 par ...

histoire
tcp-ip

27 commentaires

1	2	Suivante
---	---	----------

Renault, mercredi 24 avril 2019 à 20h2924/04/19 à 20h29

Et oui, enfin ! Merci à qwerty pour la relecture. 🍊

C'est le premier d'une série de 3 articles que nous souhaitons écrire. Les suivants seront à propos de :

OS basés sur UNIX :

- Andrew Tanenbaum (Minix)
- Linus Torvalds (Linux)
- Richard Stallman (GNU)

Les débuts de la micro informatique personnelle :

- Steve Wozniak (Apple)
- Steve Jobs (Apple et NeXT)
- Bill Gates (Microsoft)
- Paul Allen (Microsoft)

Qui ont l'avantage d'être en général mieux documentés (j'ai même la biographie de 4 d'entre eux) donc plus simple pour l'écriture. 🍊

Nous attendons quelques retours avant de s'y mettre. Donc n'hésitez pas à commenter cet article pour qu'on fasse au mieux les prochaines. Merci !

Amateur de Logiciel Libre et de la distribution GNU/Linux Fedora. #JeSuisArius

+5

nohar, mercredi 24 avril 2019 à 21h2224/04/19 à 21h22 • Modifié

Super article. Merci à vous !

À propos de Ken Thompson, son travail sur les expressions régulières ne s'est pas borné à utiliser les théories existantes dans grep et consorts. On lui doit [un papier légendaire](#), dans les années 1960, d'où découlent :

- Un algorithme qui rend presque triviale la compilation d'une expression régulière en un automate fini non-déterministe (AFN),
- La conception d'une "machine virtuelle" permettant d'exécuter un tel AFN de façon performante.

Si je qualifie ce papier de "légendaire", c'est parce que c'est grâce à lui que des utilitaires comme grep (qui compilent les automates de recherche à la volée, donc qui ont besoin que ce soit rapide) sont possibles, et pour cause, après quelques errances dûes à un enthousiasme un peu trop débordant autour des regexps (le moteur PCRE inspiré de Perl rajoute de nombreux opérateurs supplémentaires aux AFN de Ken Thompson... en sacrifiant son modèle d'exécution en faveur du backtracking, et donc la stabilité des performances, car certains patterns pathologiques peuvent prendre *la vie* à s'évaluer), [les ingénieurs reviennent à l'algorithme de Thompson](#) depuis une ou deux décennies, en *droppant* les excroissances un peu trop dégueulasses des PCRE dans leurs bibliothèques.

Ce papier, qui date des années 1960, (à l'instar de celui d'Aho et Corasick sur les AFD qui date des années 70-80), est encore de toute première fraîcheur aujourd'hui : si on veut comprendre les problématiques de l'exécution des expressions régulières, c'est par lui qu'il faut commencer, et on peut parfaitement s'en contenter, encore en 2019.

I was a llama before it was cool

+3

Renault, mercredi 24 avril 2019 à 21h4124/04/19 à 21h41

Merci pour cette précision, en effet nous sommes passés à côté.

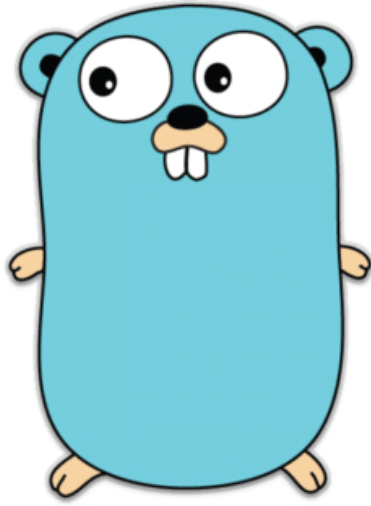
Nous avons également oublié de mentionner son article également célèbre [à propos de la confiance envers le compilateur](#). Nous allons corriger cela bientôt. 🍊

Amateur de Logiciel Libre et de la distribution GNU/Linux Fedora. #JeSuisArius

+1

nohar, mercredi 24 avril 2019 à 22h0524/04/19 à 22h05 • Modifié

Au passage, sur des considérations plus légères, je viens seulement de remarquer la ressemblance troublante entre la mascotte de Plan 9 et celle de Go.



Go

Cela dit, ce n'est certainement pas un hasard. 🍊

PS: Si j'étais mauvaise langue, je me hasarderais à supposer que ce sont deux autoportraits caricaturaux de Rob Pike.

I was a llama before it was cool

ache, mercredi 24 avril 2019 à 22h4324/04/19 à 22h43 • Modifié

@nohar: Ce n'est pas du tout un hasard.

Renée French qui est la femme de Rob Pike à dessiner Glenda également! 🍊 J'allais posté un commentaire pour dire que justement, ce n'était pas si anecdotique! x)

J'ajouterais que ce n'est pas claire dans la partie de Denis Ritchie que la photo illustre 3 personnages illustres que je ne saurais du coup même pas reconnaître x"D

Bref, article passionnant la prochaine fois, je le relirais avec un carnet pour noter mes remarques.

[ache.one](#)



jniven, mercredi 24 avril 2019 à 22h5124/04/19 à 22h51

René French qui est la femme de Rob Pike à dessiner Glenda également! 🍊

— [ache](#)

Glenda 🍊

J'ajouterais que ce n'est pas claire dans la partie de Denis Ritchie que la photo illustre 3 personnages illustre (oui, je sais, mais l'article vise tout le monde non ?) Bjarne, Ritchie et Nixon de gauche à droite.

Effectivement, nous aurions dû légender. S'il s'agit bien de Dennis Ritchie au milieu, c'est Ken Thompson qui l'accompagne et Bill Clinton qui lui serre la main!

PS: Si j'étais mauvaise langue, je me hasarderais à supposer que ce sont deux autoportraits caricaturaux de Rob Pike.

— [nohar](#)

Haha, c'est une théorie... intéressante 🍊

ache, mercredi 24 avril 2019 à 22h5324/04/19 à 22h53 • Modifié

J'ajouterais que ce n'est pas claire dans la partie de Denis Ritchie que la photo illustre 3 personnages illustre (oui, je sais, mais l'article vise tout le monde non ?) Bjarne, Ritchie et Nixon de gauche à droite.

Effectivement, nous aurions dû légender. S'il s'agit bien de Dennis Ritchie au milieu, c'est Ken Thompson qui l'accompagne et Bill Clinton qui lui serre la main !

— [jmiven](#)

Comme quoi ! C'est super important 🍊
Je vais arrêter de dire des bêtises moi x)

J'ai également fait une faute à Renée French.

[ache.one](#)



Javier, vendredi 26 avril 2019 à 18h1426/04/19 à 18h14

Un grand merci et un grand bravo pour cet article et la série qui va suivre.

Continuez ! C'est top.

Happiness is a warm puppy

+2

Bam92, dimanche 28 avril 2019 à 02h0528/04/19 à 02h05

C'est plutôt intéressant ce billet. J'avais à coeur de faire quelque chose de semblable il y a bien longtemps. Bravo donc !

J'ai cependant quelques questions||suggestions:

- y a-t-il un ordre de présentation des personnages ou c'est arbitraire ? Pourquoi ne pas commencer par Dennis au lieu de Ken par exemple ? Il en est de même pour rms et Linux.
- n'est-il pas possible d'ajouter des légendes bien plus claires sur les images ?

Renault, dimanche 28 avril 2019 à 14h0328/04/19 à 14h03

y a-t-il un ordre de présentation des personnages ou c'est arbitraire ? Pourquoi ne pas commencer par Dennis au lieu de Ken par exemple ? Il en est de même pour rms et Linux.

L'ordre n'a pas été totalement choisi au hasard bien que entre Ken et Dennis c'est probablement là où le choix a été le plus arbitraire. Mais cela n'a pas une grande importance après.

L'objectif étant de parler d'UNIX, parler des deux auteurs originaux du système en premier est donc évident. Brian suit car il était là à ce moment là tandis que Pike a rejoint le projet en dernier.

L'ordre pour le projet article n'est pas arrêté et en tout cas mon premier message ne traduit pas un ordre particulier mais juste une liste.

n'est-il pas possible d'ajouter des légendes bien plus claires sur les images ?

Oui, c'était ce qui était prévu à l'origine. En fait je n'étais pas satisfait de la mise en page de la légende sous l'image (pas de saut de ligne possible à priori). Et mettre tout sur une ligne était un peu lourde. Et selon nous mettre la source et la licence d'une image est plus importante que l'explication de celle-ci.

On n'a pas trouvé le moyen qu'une partie de l'information soit éventuellement dans l'attribut alternative de l'image pour qu'au moins au survol cela soit accessible.

artragis, dimanche 28 avril 2019 à 18h5428/04/19 à 18h54 • Modifié

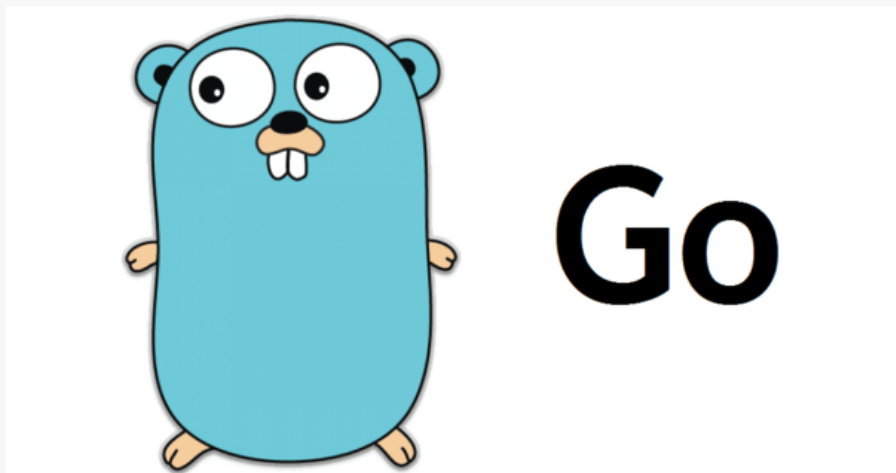
Genre comme ça?

```
1 ! [source: un site] (https://cdn0.tnwn.com/wp-content/blogs.dir/1/files/2018/07/go.png)
2 Figure: une légende
3 avec plusieurs
4 lignes
```

On notera les deux espaces après "légende" et "plusieurs"

qui donne:

Prévisualisation de votre message



une légende
avec plusieurs
lignes

lgende multiligne.PNG

et en html:

```
1 <figure>
3 lignes</figcaption></figure>
```

Torado, lundi 29 avril 2019 à 12h2129/04/19 à 12h21

Super article! Vivement le prochain l'un de mes idoles sera dedans: Linus Torvalds 🍊 En tout cas merci pour l'article

"Tout obstacle renforce la détermination. Celui qui s'est fixé un but n'en change pas." Léonard De Vinci

Oliv, lundi 29 avril 2019 à 18h0129/04/19 à 18h01

Superbe Article, lu avec un grand intérêt. Comme l'annonce sur le sujet des prochains articles.

Cela a bien entendu soulevé chez moi quelques questions existentielles, dont je n'ai pas les réponses 🍊

- Si UNIX et le Langage C et / ou C++ sont toujours usitées malgré de nouvelles avancées technologiques selon d'autres opinions; rust etc etc... (je suis incapable de juger...):
 - Est-ce du à la difficulté (coût financier trop important) de porter cet OS UNIX et les applications vers ces nouveaux langages:
 - Quid de l'évolution des architecture matérielles. CISC / RISC → IA64?
 - Quid des choix stratégiques des Acteurs Majeurs UNIX?
 - SOLARIS / HP UX / AIX...

Un petit regret, que l'on parle uniquement de Linux ou Richard... et les BSD alors ??? (malheureusement ici, je n'ai pas le temps de proposer quelque chose 🍌)

Voilà toutes les questions qu'ont soulevées chez moi la lecture de cet article, je ne pourrai donner de réponses, car à mon sens, malgré le temps passé sur le retour d'expérience, cela suscite chez les gens des discussions bien trop passionnées à la limite du troll 🍌

Merci encore pour l'article, je lirai les suivants avec plaisir. Peut-être apporteront-ils des réponses notamment via Steve Jobs et Steve Wozniak... Mac OS sur CISC → Mac OS X (BSD) sur PowerPC et x64 (depuis 2009 only)

Nous constatons bien ici l'abandon des systèmes CISC & RISC vers les architectures x86_64 (Merge des approches depuis le Pentium 2) avec l'apport de l'open source vers des dérivées UNIX.

À l'inverse, d'autres systèmes tel que le Z OS annoncé comme disparu, font plus que perdurer sur leur créneau respectif. Où les amateurs de COBOL peuvent encore s'exprimer...

Merci et encore Bravo !!!

+1

Renault, lundi 29 avril 2019 à 19h0529/04/19 à 19h05 • Modifié

Est-ce du à la difficulté (coût financier trop important) de porter cet OS UNIX et les applications vers ces nouveaux langages:

UNIX en tant que tel n'existe plus, c'est plus une famille et une API standard (POSIX) qui portent cet héritage.

Mais oui, porter les *BSD, Linux et autres vers un autre langage a un coût humain et financier non négligeable. Et il faut des compétences et du recul sur ces nouvelles technologies, ce qu'on a pas encore de manière aussi large.

D'autant plus que le noyau d'un OS est sans doute le programme qui tire le moins des bénéfices des langages comme Rust. Car un noyau par définition ne peut pas exploiter une bibliothèque standard du langage énorme (faut réimplémenter pas mal de choses) et certains concepts dans un langage ont peu d'intérêt ou sont difficiles à exploiter à ce niveau. Par exemple un noyau en Rust a probablement beaucoup plus de code *unsafe* qu'un logiciel normal écrit avec ce langage.

Cela peut arriver un jour mais le C et le C++ ont encore de beaux jours devant eux grâce à cette inertie et par une bonne prise en charge de ce genre de développements.

Quid de l'évolution des architectures matérielles. CISC / RISC → IA64?

Le rapport?

Quid des choix stratégiques des Acteurs Majeurs UNIX?

Globalement Linux et *BSD dominent le marché et allègrement. Les acteurs commerciaux ont encore une niche pour vendre leur solution mais globalement cela va plus dans le sens d'une maintenance que d'une innovation forte dans le secteur. D'ailleurs l'abandon de Solaris par Oracle est assez symptomatique de la situation.

Un petit regret, que l'on parle uniquement de Linux ou Richard... et les BSD alors ??? (malheureusement ici, je n'ai pas le temps de proposer quelque chose 🍌)

En effet les *BSD ne seront pas abordés pour trois raisons:

- *BSD est un écosystème où il y a beaucoup moins d'influence d'une personne (ou d'une poignée), donc c'est plus délicat de retracer l'histoire de BSD ainsi;*

- Ces contributeurs très éparés n'ont souvent pas de contributions si significatives, ce qui rend compliqué l'élaboration d'une biographie intéressante;
- Et en plus leurs vies et contributions sont moins documentées ce qui rend le travail plus difficile.

Après si tu arrives à trouver des noms et des histoires sympas pour narrer l'histoire de *BSD, lance-toi ! Nous ne sommes pas dépositaires du format, nous traitons les sujets qui nous intéressent, si tu veux aller plus loin ou traiter d'autres personnalités, n'hésite pas !

Amateur de Logiciel Libre et de la distribution GNU/Linux Fedora. #JeSuisArius

Oliv, lundi 29 avril 2019 à 22h10 29/04/19 à 22h10

Quid de l'évolution des architecture matérielles. CISC / RISC —> IA64 ?

Le rapport ?

Je parlais du postulat que les architectures matérielles évoluant au fil du temps, permettait des évolutions logiques sur les Systèmes d'exploitation.

- *BSD est un écosystème où il y a beaucoup moins d'influence d'une personne (ou d'une poignée), donc c'est plus délicat de retracer l'histoire de BSD* ainsi ;
- Ces contributeurs très éparés n'ont souvent pas de contributions si significatives, ce qui rend compliqué l'élaboration d'une biographie intéressante ;
- Et en plus leurs vies et contributions sont moins documentées ce qui rend le travail plus difficile.

Je me permets quelques petites corrections :

- SUN SOLARIS était initialement un système BSD, à partir de la seconde version le noyau est issu de la branche SYSTEM V.
- OpenBSD sous l'impulsion de Theo de Raadt a fait fléchir nombre de fondeurs pour obtenir les spécifications techniques nécessaires au développement des drivers. cf le travail de Damien Bergamini notamment sur les drivers WIFI.

Un prix a été décerné, à Theo de Raadt en 2004, pour la promotion des drivers libres par la FSF (à l'inverse des distribution GNU Linux utilisant des drivers propriétaires)

- OpenBSD propose nombre de produit, qui sont repris très rapidement sur les distributions linux :
 - OpenSSH
 - LibreSSL (fork de OpenSSL car trop de failles)
 - Packet Filter
 - OpenBGPD
 - OpenNTPD
 - OpenSMTPD
 - OpenHTTPD
 - OpenCVS
 - ManDoc (et en général, les pages de man OpenBSD sont les plus riches du monde Open Source)

Tous ces projets, suivent la même philosophie "KISS", et avec la même rigueur et audit de code, pour un eco système sécurisé, simple et fiable.

Après si tu arrives à trouver des noms et des histoires sympas pour narrer l'histoire de *BSD, lance-toi ! Nous ne sommes pas dépositaires du format, nous traitons les sujets qui nous intéressent, si tu veux aller plus loin ou traiter d'autres personnalités, n'hésite pas !

— [Renault](#)

Sincèrement cela m'aurait plu.

Notamment :

- Sur les prises de positions de Theo de Raadt vs GNU / LINUX. "It's terrible," De Raadt says. "Everyone is using it, and they don't realize how bad it is. And the Linux people will just stick with it and add to it rather than stepping back and saying, 'This is garbage and we should fix it.'" https://www.forbes.com/2005/06/16/linux-bsd-unix-cz_dl_0616theo.html#7d4644e8171d

- L'anecdote du nom de domaine THEOS.COM. Ici une société a attaqué en justice Theo de raadt, pour obtenir l'extension commerciale du nom de domaine que Theo avait souscrit auprès d'un registrar.
- L'annonce, il y a plus de 10 ans des failles "Meltdown" des processeurs Intel.

Malheureusement, entre mes projets persos qui stagnent, la maison qui va démarrer, je n'ai pas de bande passante pour le moment. Je garde le lien... qui sait plus tard !

Je le réitère j'ai apprécié votre travail, l'article m'a plu. Il était pertinent et je me posais, à l'issue, certaines questions existentielles...

+1

Gabbro, lundi 29 avril 2019 à 22h3529/04/19 à 22h35

Quid des choix stratégiques des Acteurs Majeurs UNIX? SOLARIS / HP UX / AIX...

Concernant AIX, ils vendent une très grande stabilité, au sens qu'un programme compilé il y a très longtemps tourne encore sur un AIX récent. Mais ils ont du mal à suivre les nouveautés (en partie par manque de moyen alloués, en partie pour préserver la compatibilité).

L'inertie de tels systèmes et entreprises fait qu'ils dureront encore un bon moment, mais je pense qu'ils sont voués à disparaître à terme. Et l'achat par IBM (qui édite AIX) de RedHat (qui édite l'une des plus importantes distributions commerciales de Linux) me conforte.

Il y a bien des façons de passer à l'acte. Se taire en est une. *Attribué à Jean-Bertrand Pontalis*

ache, mardi 30 avril 2019 à 07h3730/04/19 à 07h37 • Modifié

@**Oliv**: OpenHTTPD n'est pas de OpenBSD, en tout cas, c'est ce que la page openhttpd.net affirme.

ache.one



Oliv, mardi 30 avril 2019 à 08h1830/04/19 à 08h18 • Modifié

@**Gabbro**, les stratégies commerciales d'IBM me sont incompréhensibles, le rachat "Red Hat" est à suivre par contre. A voir ce que Red Hat deviendra... 🍊

En parallèle, pour les AIX tournant sur les offres iSeries. La grande distribution se sépare depuis quelques années de ces technologies. Néanmoins, tu fais tourner des machines AIX sur du z/OS qui, malgré le coût, observe toujours un avenir florissant. Surtout sur les échanges de flux inter compagnies (achats / commandes...)

@**Ache**, My BAD. Effectivement absent des projets portés par la fondation OpenBSD: <http://www.openbsdfoundation.org/>

Reyk floter le créateur de OpenHTTPD est un des participants actifs, du projet OpenBSD.

Je ne souhaite pas ici troller sur Linux vs BSD. OpenBSD me convient, comme Linux convient à d'autres.

J'aurai pu évoquer aussi, le travail de Constantine A. MURENIN sur le "Hardware Sensors Framework" repris par d'autres: <https://www.openbsd.org/papers/asiabsdcon2009-sensors-paper.pdf>

Constantin A Murenin est, aujourd'hui, contributeur sur d'autres projets BSD.

Mes questions existentielles tournaient effectivement sur l'empreinte du commerce sur les évolutions techniques.

Aliciaz, samedi 04 mai 2019 à 17h3204/05/19 à 17h32

Je pensais que je connaissais bien l'histoire de l'informatique mais il y a plein de noms que je ne connaissais pas en fait 🍊 Content d'en savoir plus, merci !

jmiven, mercredi 08 mai 2019 à 09h4608/05/19 à 09h46

Si vous comprenez l'anglais et avez apprécié cet article, vous aimerez probablement [cette interview](#) récente de Ken Thompson par Brian Kernighan.

L'interview commence à 8:20.

Renault, dimanche 19 mai 2019 à 21h5519/05/19 à 21h55

Pour votre information, l'article a été mis en jour en tenant compte de vos remarques. En espérant que cela convienne. 🍊

Amateur de Logiciel Libre et de la distribution GNU/Linux Fedora. #JeSuisArius

+4

Abou, mercredi 27 novembre 2019 à 01h2327/11/19 à 01h23




Bonsoir, le sujet est de présenter des personnes qui ont beaucoup contribué à l'informatique, mais je ne vois qu'un seul groupe d'amis là, et les autres est-ce que vous allez en parler? Par exemple ADA.

En se partageant le savoir ne se divise pas, il se multiplie []____) (____ (.)() \ / | . | () |||

ache, mercredi 27 novembre 2019 à 05h4627/11/19 à 05h46

Normalement, c'est une série d'articles. M'enfin, cet article a commencé à être écrit il y a très longtemps.

Si tu veux, tu peux écrire sur l'un ou plusieurs des personnes cités dans le sujet de la première phrase de l'article. (cf [Personnages clés de l'informatique](#))

[ache.one](#)   

Abou, jeudi 28 novembre 2019 à 10h5228/11/19 à 10h52 • Modifié

Pourquoi ces seules personnes? Il y a plein d'autres personnes qui ont contribué au monde informatique.

En se partageant le savoir ne se divise pas, il se multiplie []____) (____ (.)() \ / | . | () |||

Renault, jeudi 28 novembre 2019 à 11h3728/11/19 à 11h37

Le but de cette liste n'est pas d'être exhaustive et chacun est libre de faire ce qu'il veut. 🍊

Personnellement je n'ai les connaissances et l'envie de ne traiter que ceux que j'ai listé ici (et non dans le topic) en gros UNIX, les LL et les débuts de l'ère PC. Donc si d'autres pans t'intéresse, tu peux les couvrir, je ne le ferais de toute façon pas.

Date estimée de publication: d'ici 10 ans (basé sur le temps requis pour écrire le premier :D).

Amateur de Logiciel Libre et de la distribution GNU/Linux Fedora. #JeSuisArius

nnmaire

1. [Ken Thompson](#)
2. [Dennis Ritchie](#)
3. [Brian Kernighan](#)
4. [Rob Pike](#)
5. [Sources](#)

tager

- [Twitter](#)
- [Facebook](#)
- [Mastodon](#)
- [Diaspora*](#)
- [Envoyer par mail](#)

écharger

- [HTML \(41,3 Kio\)](#)
- [PDF \(1,1 Mio\)](#)
- [LaTeX \(37,7 Kio\)](#)
- [EPUB \(2,7 Mio\)](#)
- [Archive \(37,3 Kio\)](#)