Christopher Kelty

Sharing Source Code

Trad. Camille Paloque-Berges

Complément à
Histoires et cultures du Libre. Des logiciels partagés aux licences échangées.

Framabook
le pari du livre libre

Source

Christopher KELTY, *Two Bits, The Cultural Significance of Free Software*, chapitre « *Sharing Source Code* », Duke University Press, 2008, pp. 118-143.
URL : http://twobits.net.

Traduction par Camille Paloque-Berges, post-doctorante du LabEx Hastec, laboratoire DICEN (CNAM)

NdE (Framasoft) – Nous tenons ici à remercier Duke University Press d'avoir bien voulu autoriser la collection Framabook à faire figurer la présente traduction dans l'ouvrage *Histoires et cultures du Libre*. Cependant, moyennant une redevance auprès de Duke University Press, la clause restrictive d'usage « non commercial » de ce chapitre ne peut être levée que temporairement, pour chaque édition « papier » et vendue du livre, y compris pour un éditeur associatif et sans but lucratif, et même pour une traduction bénévole ayant vocation à être diffusée librement. Toute forme de diffusion de ce chapitre implique que les utilisateurs ne peuvent publier eux-mêmes l'ouvrage sous licence libre en version papier et le vendre pour couvrir les frais de l'impression… à moins d'en sortir ce texte ou de payer et informer Duke University Press. La collection Framabook a donc décidé de ne pas l'inclure dans la version papier de cet ouvrage. Néanmoins, compte tenu des apports conceptuels de cet extrait, nous avons décidé de le proposer en téléchargement gratuit, en guise de complément pour les lecteurs.

PRÉSENTATION

Par Camille PALOQUE-BERGES

Si l'histoire des technologies logicielles est bien documentée, celle du logiciel libre l'est beaucoup moins, explique dans ce chapitre Christopher Kelty en s'attaquant à la saga d'Unix.

Une des raisons en est le fait que la « philosophie » qui lui est attachée est tissée de narrations et de représentations, un imaginaire hybride à la croisée de la technique et de l'idéologie – et, prenant cette voie, le propos de Kelty s'inscrit dans la trajectoire de la sociologie culturelle appliquée à l'imaginaire des technologies informatiques.

Le texte suivant est au cœur de l'ouvrage *Two-Bits, The Cultural Significance of Free Software* publié en 2008 (sous licence Creative Commons-By-NC-Sa) ; il décrit le deuxième moment d'une partie centrale détaillant les cinq principes qui, selon Kelty, sont les piliers explicatifs de cette philosophie : le « mouvement », émergeant à partir d'un certain nombre de valeurs et de pratiques liées au monde des technologies informatiques de la fin des années 1990 (de l'*open source* aux navigateurs dits « libres ») ; le « partage du code source », un retour en arrière sur les premières et multiples formes de la circulation des codes informatiques des années 1960 aux années 1980, prenant comme paradigme le système d'exploitation UNIX (objet du texte traduit ci-dessous) ; la « conception des systèmes ouverts » revenant sur la manière dont la notion d'ouverture a été inventée et conceptualisée ; l'apparition et la systématisation d'une « écriture propre aux licences du libre » ; et enfin, les modes de « coordination des formes de la collaboration ».

Kelty procède par une analyse culturelle à la croisée des discours ayant accompagné et conditionné dans une très large mesure les exploits informatiques des adeptes du logiciel libre et des réalisations techniques à proprement parler. C'est pourquoi les « geeks » qui peuplent l'ouvrage sont moins une figure folklorique que les moteurs d'un développement à la fois social et technique d'une série d'alternatives aux trajectoires d'innovation et de développement des technologies informatiques académiques et commerciales ; sans territoire ni milieu propres, mais cependant marqués par des effets de structure et de pouvoir, ils sont partout comme le code source qu'ils partagent, et sont à la base de la notion de « public récursif », concept central de l'ouvrage.

Selon Kelty, un public récursif se constitue autour d'un besoin commun de maintenir les moyens d'association qui lui permettent de se penser comme public. Chez ces publics, la technologie est utilisée comme un argument, un discours, aussi bien qu'elle l'est pour des raisons instrumentales, comme le précise Kelty : le public parle de technologie, mais exprime aussi des idées grâce à elle. Ce public conçoit également les infrastructures au travers desquelles les idées peuvent circuler et être exprimées. Internet et le logiciel libre sont ainsi deux environnements privilégiés de ces publics, à la fois développés, regardés, commentés, et habités par eux.

Remontée généalogique aux origines de ces publics, ce chapitre a aussi une valeur archéologique dans la mesure où il va fouiller dans les couches profondes du code pour mieux comprendre les pratiques de la programmation, ainsi que le savoir-faire technique et les croyances technologiques qui les sous-tendent.

Sharing Source Code

1. « Le partage du code source »

Le logiciel libre n'existerait pas sans le partage du code source. Ce principe est au cœur même de la formule « Open Source » ; il est un prérequis de toutes les licences de logiciel libre : le code source doit être ouvert, c'est-à-dire visible, et non pas enfermé dans une boîte. Parmi les cinq composantes du logiciel libre, le code source est à la fois un médium expressif, comme l'écriture ou la parole, et un outil permettant d'exécuter des actions concrètes. Il sert de support mnémonique ainsi que d'interface de traduction entre la machine, qui traite à la vitesse de l'électron des calculs pour nous illisibles, et les hommes, qui persistent à essayer de comprendre et maîtriser ces phénomènes, même partiellement. L'adage « l'information veut être libre » est suivi par de nombreux informaticiens défenseurs du logiciel libre qui arguent du fait que le partage est quelque chose de naturel pour l'humanité. Cependant, j'aimerais défendre l'idée contraire : le partage produit sa propre morale et son propre ordre technique, et l'on devrait plutôt formuler les choses ainsi : « l'information pousse les gens à désirer la liberté ». La façon dont ce désir s'exprime est liée à la manière dont on produit de l'information et dont on la fait circuler. Ce chapitre raconte les revirements et contingences qui ont amené le code source et le partage à prendre les formes techniques, légales, et pédagogiques que l'on connaît aujourd'hui, des normes devenues une seconde nature chez les geeks.

Si le code source est essentiel au logiciel libre, c'est en raison des spécificités historiques qui ont déterminé son partage, sa « portabilité » et sa « forkabilité » [1]. Rien chez lui n'implique le partage *a priori* : pour les entreprises, le secret et la protection jalouse du code est une norme, tandis que pour les universitaires et les geeks, le code source est en général l'expression, voire l'implémentation unique d'une plus grande idée se devant d'être partagée. Mais au cours des trente dernières années le partage du code s'est accompagné de l'apparition de normes qui en apparence sont devenues naturelles. Elles sont le fruit de diverses initiatives pour faire du logiciel un produit

1. (NdT) Les deux termes sont des anglicismes que nous conservons dans la mesure où ils sont d'usage courant dans le jargon informatique. Des traductions françaises ont été proposées pour le *fork*, à savoir « embranchement » ou « fourche », que nous ne retenons pas ici afin de garder un équilibre linguistique et esthétique avec le terme *port*. L'acte accompli dans un *port* est devenu en français « portage » et on utilisera, pour garder le parallèle phonétique, le terme « forkage » ; le jargon verbalise également ces termes en « porter » et « forker » (*cf.* les articles Wikipédia version langue française correspondants).

(comme en 1968 quand IBM décide de vendre le logiciel et le matériel de ses ordinateurs séparément), pour le définir et le contrôler juridiquement grâce au secret de fabrique, aux droits d'auteur et aux licences, et enfin pour enseigner aux ingénieurs ce qu'est un logiciel et comment en créer toujours davantage.

Ce n'est pas une coïncidence si l'histoire des normes qui régissent le code source est aussi celle des systèmes d'exploitation UNIX [2]. Ils forment un hybride protéiforme de l'esprit académique et de l'esprit d'entreprise, une expérimentation sur la portabilité et le partage qui a fait ses preuves chez les geeks (qui le révèrent) mais a été sous-estimé ailleurs. L'histoire d'UNIX démontre en détail comment le code source en est venu à être partagé, techniquement, légalement et pédagogiquement. UNIX et le langage de programmation C dans lequel il a été écrit illustrent en théorie et en pratique plusieurs notions clefs liées au genre du système d'exploitation. Par ailleurs, ils ont été un médium de portage d'UNIX sur de nombreux appareils informatiques disponibles dans les années 1970, et dans le monde entier. En termes juridiques, le propriétaire d'UNIX, AT&T [3], a créé une licence autorisant l'utilisation large et libérale du code aussi bien sous sa forme binaire que comme code source. En termes techniques, sa définition est différente : UNIX a été le fruit d'une évolution expérimentale sur les systèmes d'exploitation portables, une instabilité constitutive de son existence. En termes pédagogiques, UNIX est devenu le paradigme même du système d'exploitation : sa portabilité n'était ainsi pas seulement d'ordre technique (d'une machine vers une autre) mais aussi d'ordre cognitif (des machines vers l'esprit), puisque les étudiants en science informatique apprenaient grâce à lui ce qu'était un « système d'exploitation » et étudiaient les détails d'un code source partagé quasi légalement [4].

2. Le partage du code source n'est pas le seul type de partage parmi les geeks (par exemple, on partage de manière informelle pour communiquer des idées), et UNIX n'est pas le seul logiciel partagé. Aujourd'hui, on trouve d'autres exemples de prolifération logicielle (le langage de programmation LISP, le système de formatage de texte TEX…) qui sont omniprésents comme UNIX. Si on inverse l'argument, on obtient l'idée que la commercialisation produit un ordre de nature différente : de nombreux produits plus utilisés qu'UNIX au départ ont disparu parce qu'ils n'ont pas été « portés » ou « forkés » ; ils font aujourd'hui partie des collections et des musées de l'informatique obsolète, si on les a jamais conservés.

3. (NdT) AT&T (American Telephone and Telegraph Company) est une des entreprises de télécommunication américaines les plus importantes dont l'existence remonte à la fin du XIX^e siècle. Comme il y sera fait allusion plusieurs fois dans ce chapitre, les innovations sur lesquelles travaille son laboratoire de recherche et développement (Bell Labs) ne peuvent être commercialisées (frais de licence mis à part) si elles ne relèvent pas strictement de l'équipement téléphonique (selon un décret remontant à 1956 et qui sera abrogé au milieu des années 1980), ce qui explique les relations relativement privilégiées entretenues avec les universités. Cf. note 32 page 16.

4. L'histoire d'UNIX, si elle a été racontée des centaines de fois, reste encore à écrire. Chaque

UNIX s'est aussi propagé à la faveur d'un accord hybride entre le monde de l'entreprise et celui de l'université : ce n'était ni un objet du « domaine public » que les universitaires se partageaient ni un produit commercial habituel. Les nouvelles formes du partage universitaire et des licences relatives au statut spécial d'AT&T (bénéficiaire d'un monopole, mais régulé et interdit d'entrée dans l'industrie des ordinateurs et des logiciels jusqu'à 1984) ont favorisé cette propagation. Il ne s'agit pas d'un processus de réplication, mais bien de prolifération, reposant non pas sur la simple vente d'UNIX, mais sur un réseau complexe d'extraits de code partagés encore et encore, ainsi que sur l'implémentation nouvelle d'un modèle conceptuel simple et élégant. En même temps qu'UNIX s'est propagé, le système s'est stabilisé de multiples façons : les universitaires cherchaient à conserver un système complet et compatible malgré les diverses contributions au code source ; les avocats d'AT&T, à poser des limites via des lois, des licences, des versions et des régulations ; et les enseignants, à en faire un exemple archétypal des concepts clefs de la théorie des systèmes d'exploitation. UNIX a généré un prototype de public récursif en créant un réseau d'acteurs prenant sens dans l'utilisation, la modification et la stabilisation d'UNIX.

L'envers de la propagation est la différenciation : le fork. UNIX suscite l'admiration pour son intégrité en tant qu'objet conceptuel, mais aussi le trouble face à son arbre généalogique où les portages et les forkages s'enchevêtrent, faisant naître de nouvelles versions, dérivées directement ou non du code source, des variations sur les licences ainsi que des branches complètement indépendantes.

Le fork, comme toute mutation aléatoire, a de bons comme de mauvais effets. D'un côté, la succession de forks a débouché sur des versions d'UNIX incompatibles entre elles (une sorte de réponse immunitaire), mais cela a aussi permis la fusion d'UNIX et de l'ARPANET [5], UNIX devenant un paradigme

hacker, programmeur, chercheur en informatique, chaque geek a sa propre version. Les sources étudiées pour écrire ce chapitre sont en partie porteuses de ces histoires, entendues et enregistrées pour les besoins de mon étude de terrain, mais nourrissent aussi la recherche académique qui travaille sur la question du logiciel libre et encourage avec enthousiasme cette histoire en conserve dans laquelle tout le monde vient faire son marché. Voir par exemple *The Success of Open Source* par Steven Weber, *The Internet Galaxy* par Castells, *The Hacker Ethic* par Himanen, *The Weath of Networks* par Benkler. Il existe aujourd'hui qu'une seule histoire détaillée d'UNIX : Peter SALUS, *A Quarter Century of Unix*, Reading : Addison-Wesley, 1994, sur laquelle je m'appuie largement. La thèse de Matt Ratto, « *The Pressure of Openness* », recèle également une excellente histoire analytique des événements narrés dans ce chapitre.

5. (NdT) L'ARPANET est le premier réseau d'ordinateurs à distance financé et développé, en partenariat avec quelques laboratoires de science informatique prestigieux, par l'ARPA (Advanced Research Projects Agency), une agence gouvernementale de recherche américaine née en 1958 en même temps que la NASA, en pleine course à l'innovation technologique avec l'URSS.

des systèmes d'exploitation aussi bien que des réseaux d'ordinateurs après sa rencontre avec le développement du TCP/IP, suite protocolaire au cœur d'Internet [6]. Au milieu des années 1980, UNIX était devenu un passage obligatoire pour quiconque s'intéressait à l'ingénierie de réseau, aux systèmes d'exploitation, à l'Internet et, plus particulièrement aux manières de créer, partager, et modifier le code source ; à tel point qu'UNIX est devenu célèbre parmi les geeks non plus en tant que système d'exploitation, mais en tant que philosophie, répondant à une vieille question renouvelée dans le contexte de l'informatique : comment vivre dans un monde de machines, de logiciels et de réseaux ?

Aux temps primitifs des machines informatiques, le code source n'existait pas. On dit qu'Alan Turing [7] aimait parler aux machines en code binaire. Grace Hopper, qui a inventé l'un des premiers compilateurs [8], posait le plus souvent possible les mains sur le Harvard Mark 1 : tournant les boutons, branchant et débranchant les relais constituant le « code » de ce que la machine allait exécuter. Un travail si mécanique et méticuleux n'est pas vraiment une lecture ni même une écriture ; il n'y avait pas d'expression GOTO [9], de numéros de ligne [10], seulement des calculs devant être traduits, des écrits pseudo-

Le réseau, fonctionnel à partir de 1968, sera raccordé à d'autres réseaux informatiques, nés entre temps et basés sur le modèle de l'ARPANET, par le biais du TCP/IP (dont il sera question à la fin du chapitre), ce qui a donné naissance à ce qu'on appelle aujourd'hui l'Internet (littéralement, « l'entre-réseaux », dont le Web est un des réseaux les plus célèbres mis en place entre 1989 et 1993). L'ARPA, soucieuse de réaffirmer ses liens avec la défense américaine, sera rebaptisée la DARPA (*Defense Advanced Research…*) dans les années 1970, d'où l'utilisation de ce dernier acronyme par l'auteur plus bas dans le texte.

6. UNIX et le TCP/IP se sont croisés vers 1980, ce qui a débouché sur le célèbre basculement du *Network Control Protocol* (NCP) au *Transmission Control Protocol / Internet Protocol* mis en place le 1er janvier 1983 (SALUS, *op. cit.*).

7. (NdT) Alan Turing est une figure clef des débuts de l'informatique moderne ; il a notamment inventé le principe de la « machine universelle », un modèle conceptuel de traitement numérique qui fait sortir l'informatique de sa préhistoire et la fait entrer dans son histoire.

8. (NdT) Grace Hopper a joué un rôle très important dans l'invention des langages de programmation de haut niveau, en proposant notamment l'un des premiers compilateurs (le « A Compiler », dans le cadre des travaux sur l'UNIVAC, le premier ordinateur commercialisé auprès des entreprises dans les années 1950), un logiciel permettant la traduction automatique des langages de haut niveau (pouvant être écrits et lus par les hommes grâce à un langage symbolique mixte d'anglais et de formules mathématiques) en langages de bas niveau (le langage machine et le code binaire). Avant cela, elle avait travaillé à l'université d'Harvard sur le Mark 1, l'un des premiers ordinateurs programmables automatiques à la fin entre 1944 et 1949.

9. (NdT) L'expression GOTO est une instruction célèbre en programmation (elle renvoie l'exécution vers une autre ligne de code), héritée des premiers langages de haut niveau et ayant fait l'objet de controverses chez les différentes écoles de programmeurs.

10. (NdT) Les logiciels permettant d'écrire du code (des éditeurs de texte) ont peu à peu intégré la numérotation des lignes du code pour des raisons pratiques.

mathématiques, des ingénieurs et des ordinateurs humains [11] en une configuration physique ou mécanique [12]. L'activité de lecture et d'écriture du code source et des langages de programmation s'est développée avec le temps, et ne devint relativement répandue qu'au milieu des années 1970. Les « langages de haut niveau » [13] ont commencé à apparaître à la fin des années 1950 : FORTRAN, COBOL, Algol, ainsi que les « compilateurs » permettant aux programmes écrits dans leur langage d'être transformés en représentations des mécanismes et valves lisibles seulement par la machine. C'est à cette époque que les expressions langage source et langage cible sont apparues afin de désigner l'activité de traduction d'un haut niveau à un bas niveau de langage [14].

L'ordinateur présente un aspect paradoxal rarement remarqué : c'est parce qu'on peut le programmer dans son ensemble qu'il est omnipotent, à en croire le principe d'universalité qu'on lui attribue ; en théorie, il peut effectuer n'importe quel calcul, et c'est prouvé mathématiquement par ce que l'on appelle la « machine universelle de Turing » [15]. Toutefois, si cette certitude repose sur la

11. (NdT) Le premier usage du terme *computer* s'appliquait à des employés de bureau au XIX^{e} siècle dont le métier était d'opérer un très grand nombre de calculs simples (en particulier dans les banques).

12. Jenifer LIGHT, "When Computers Were Women", dans : *Technology and Culture* 40.*3* (1999), p. 455–483 ;David A. GRIER, *When Computers Were Human*, Princeton : Princeton University Press, 2005.

13. (NdT) *cf.* note 8 page 8.

14. Les recherches académiques sur l'histoire du logiciel ont fait leurs preuves et continuent de se développer : les volumes 1 et 2 de *History of Programming Languages* édités respectivement par Wexelblat puis Bergin et Gibson sont des ouvrages collectifs d'historiens. Parmi les ouvrages clefs, citons Martin CAMPBELL-KELLY, *From Airline Reservations to Sonic the Hedgehog : A History of the Software Industry*, Cambridge, Mass. : MIT Press, 2003, Akera ATSUSHI et Frederik NEBEKER, *From 0 to 1 : An Authoritative History of Modern Computing*, New York : Oxford University Press, 2002, Ulf HASHAGEN, Reinhard KEIL-SLAWIK et Arthur L. NORBERG, éds., *History of Computing - Software Issues*, 1re éd., Berlin : Springer, 2002, Donald MACKENZIE, *Mechanizing Proof : Computing, Risk, and Trust*, Cambridge, Mass. : The MIT Press, 2001, Michael Mahoney est celui qui a le plus écrit sur l'histoire précoce du logiciel (voir par exemple « The roots of Software Engineering », « The Structures of Computation », « In Our Image », et « Finding a History for Software Engineering »). Sur UNIX plus particulièrement, il est étonnant de ne trouver que si peu de travaux. Seules deux pages y sont consacrées dans Martin CAMPBELL-KELLY et William ASPRAY, *Computer : A History of the Information Machine*, New York, NY : Basic books, 1996. Dès 1978, Ken Thompson et Dennis Ritchie ont réfléchi à l'« histoire » d'UNIX dans « The UNIX Time-Sharing System : A Retrospective ». Ritchie maintient un site web qui propose une collection de valeur sur les premiers documents relatifs au système ainsi que ses propres souvenirs (`http://www.cs.bell-labs.com/who/dmr/`). Mahoney a aussi conduit des entretiens avec les protagonistes du développement d'UNIX aux Bell Labs, qui n'ont pas été publiés et sont restés dans ses fichiers personnels, mais ont été prêtés à l'auteur et exploités pour ce chapitre.

15. Alan TURING, "On Computable Numbers, with an Application to the Entscheidungsproblem", dans : *Proceedings of the London Mathematical Society* 2.*1* (1937), p. 230. On trouve une explication simple de ce principe dans Davis MARTIN, *Engines of Logic : Mathematicians and*

puissance de l'abstraction, nous ne vivons pas dans le monde de l'Ordinateur, mais dans un monde d'ordinateurs. Les systèmes matériels que les constructeurs ont fabriqués dans les années 1950 et suivantes étaient si spécifiques et idiosyncrasiques qu'il était inconcevable que l'on puisse écrire un programme pour une machine et l'exécuter sur une autre. On programmait sur mesure, pour une machine spécifique, et si les programmeurs spécialistes d'une machine ont probablement échangé des programmes entre eux, partager avec des utilisateurs d'une autre machine ne présentait pas d'intérêt. Les informaticiens étaient certes aussi enthousiastes à l'idée d'échanger leurs descriptions mathématiques d'algorithmes et leurs avancées sur l'automation, que les entreprises étaient jalouses de protéger les leurs, mais ces formes de partage ou de secret ne concernaient pas directement le programme lui-même. Le besoin que certains ont eu de « réécrire » le programme pour chaque machine n'est pas un accident historique, mais la conséquence de certaines nécessités concernant les ingénieurs couplées aux difficultés posées par le marché de l'informatique, les machines étant fort coûteuses [16].

À l'âge doré des ordinateurs-grands-comme-une-salle, les langages de programmation humains [(NdT) différents du code binaire, mais avant les langages de haut niveau] étaient mnémoniques ; ils n'existaient pas dans l'ordinateur, mais sur un morceau de papier ou une carte de code standard. La carte permettait aux hommes qui n'étaient pas Alan Turing de garder la trace, de partager avec d'autres, et de penser de manière systématique les calculs d'un appareil complexe, calculs invisibles et aussi rapides que la lumière. Une telle mnémonique devait être « codée » sur des cartes perforées ; si les ingénieurs devaient en discuter, ils le faisaient à l'aide de feuilles de papier où figuraient les câbles, relais et interrupteurs – ou, plus tard, des codes imprimés représentant un programme et des données.

Quand les langages de programmation firent leur apparition, la distinction entre un langage « source » et un langage « cible » fut introduite dans la pratique : les langages source étaient « traduits » dans le langage-machine, illisible pour l'homme. Les langages source, de haut niveau, appartiennent en quelque sorte à un genre mnémonique. Certes, ils étaient plus faciles à écrire et à lire pour les hommes (la plupart du temps sur des carnets jaunis ou sur des cartes de code standard), mais ils étaient aussi assez structurés pour

the Origin of the Computer, W. W. Norton, New York, NY, 2001.

16. Le partage des programmes n'était pertinent à cette époque que dans le cadre de groupes d'utilisateurs professionnels comme SHARE (IBM) ou USE (DEC). S'ils échangeaient bien du code source et des programmes (Akera ATSUSHI, "Volunteerism and the Fruits of Collaboration : The IBM User Group SHARE", dans : *Technology and Culture* 42.*4* (2001), p. 710–736), c'était seulement sur des machines similaires, la loyauté à la marque et l'idéal d'une machine unique étant de mise.

qu'un langage source puisse être saisi dans un ordinateur et traduit dans un langage cible spécifié par les concepteurs du matériel. L'acte de saisir des commandes, des cartes et du code source requérait une série d'actions spécifiques à chaque machine : un lecteur de carte en particulier, ou, plus tard, une perforatrice à cartes avec un « éditeur » particulier pour entrer les commandes. Un code source bien saisi et traduit produisait dans la machine un programme en binaire assemblé qui pourrait, à proprement parler, s'exécuter (calcul, opération, contrôle). Cela a introduit une séparation, une abstraction qui a permis une certaine division du travail entre des auteurs humains ingénus et des machines rapides de traduction mécanique.

Même après l'invention des langages de programmation, programmer « sur » un ordinateur – être assis devant un écran lumineux et hacker toute la nuit – est resté longtemps difficile. C'est seulement aux alentours de 1969 que l'on a pu s'asseoir devant un clavier, écrire du code source, donner l'ordre à l'ordinateur de le compiler, puis exécuter le programme – tout cela sans quitter le clavier – une série de gestes impensable aux premiers jours du « traitement par lot » [17]. Avant 1975, seuls quelques programmeurs avaient adopté cette nouvelle manière de travailler, puis les éditeurs de texte permettant de voir le texte sur l'écran plutôt que sur un morceau de papier commencèrent à proliférer [18].

Aujourd'hui, l'image d'un utilisateur assis devant un écran et interagissant avec la machine est si familière qu'il est presque impossible de s'imaginer comment la lente accumulation des outils et techniques de travail a créé cette nouvelle forme d'écriture, et surtout comment l'explosion babélienne des langages et des machines a en quelque sorte trahi la promesse d'une machine à calculer universelle.

La prolifération de machines et architectures différentes a créé le désir, chez les universitaires en particulier, d'une standardisation des langages de

17. Michell WALDROP, *The Dream Machine : J. C. R. Licklider and the Revolution that Made Computing Personal*, New York : Viking, 2002, p. 142-147. (NdT) Le traitement par lot (*batch processing*) est une des premières solutions en informatique pour mieux répartir les ressources de traitement d'un ordinateur : si les tâches dites interactives (où un opérateur ou un utilisateur intervient) sont prioritaires, les tâches par lot sont exécutées de manière automatique quand toutes les ressources ne sont pas utilisées par les premières.

18. Un grand nombre de ces éditeurs virent le jour à cette époque, les plus célèbres étant l'EMACS de Richard Stallman et le vi de Bill Joy. On n'en finit pas de glorifier l'héritage de Douglas Engelbart en matière d'interactivité informatique (Thierry BARDINI, *Bootstrapping : Douglas Engelbart, Coevolution, and the Origins of Personal Computing*, Stanford : Stanford University Press, 2001), mais il faut réévaluer le travail précoce de Butler Lampson et de Peter Deutsch à Berkeley, ainsi que celui de l'équipe Multics (Ken Thompson entre autres) sur les éditeurs, qui pose des bases probablement plus fondamentales en termes de manipulation de fichiers à l'écran. Cette histoire se doit d'être davantage documentée et analysée.

programmation : on ne croyait pas tant en un langage qui surpasserait tous les autres, mais il fallait ne pas se répéter, et pour cela on devait nécessairement partager un corpus émergent d'algorithmes, de solutions et de techniques. Algol, un langage simplifié adapté aux représentations algorithmiques et algébriques, s'est révélé au début des années 1960 un bon candidat pour la standardisation internationale. D'autres langages mettaient en avant des points forts différents : FORTRAN et COBOL [19] étaient utilisés pour des applications commerciales ; LISP, pour du traitement de symboles. En même temps, un langage de haut niveau standard impliquait un bouquet de programmes de traduction : des outils pour la compilation, la décomposition analytique, l'analyse de lexique et d'autres conçus pour transformer le langage de haut niveau (lisible par l'homme) en un langage de bas niveau (spécifique à la machine) c'est-à-dire le langage machine, le langage assembleur, jusqu'aux « zéro et un » mystiques qui se courent après dans nos machines. L'idée d'un langage standard et la nécessité d'inventer des outils spécifiques pour la traduction sont à l'origine des problèmes de portabilité : la capacité à faire passer du logiciel – non plus seulement de bonnes idées, mais de véritables programmes écrits dans un langage standard – d'une machine à une autre.

Le langage source standard était vu comme une manière de contrebalancer la prolifération de machines variées au moyen d'architectures aux différences subtiles. La portabilité du code permettait de faire naviguer un programme comme un bateau toujours en mouvement – aucune escale ne l'altère véritablement. La portabilité du code source est devenue une sorte d'Espéranto parlé par des tribus, chacune cependant réunie autour de sa propre machine.

Dans les années 1960, la portabilité du code source n'était d'aucun intérêt pour l'industrie informatique. On ne séparait pas le logiciel et le matériel informatique, cela coûtait cher et ce qui comptait était que la tâche requise par le client soit exécutée, et non pas le type de langage utilisé. Chaque nouvelle machine devait se démarquer de la précédente en termes de rapidité, fonctionnalité, taille (au départ, on les voulait toujours plus grandes, puis toujours plus petites). Ce besoin pressant de différencier les machines les unes des autres n'était pas motivé par une expérimentation académique ou la recherche de la pureté esthétique, mais par les intérêts du marché, les avantages compétitifs, et la transformation des machines en produits. Chaque machine devait exceller dans son domaine, et nécessitait d'être développée en secret afin de prendre une longueur d'avance sur les conceptions et innovations des rivaux.

19. (NdT) Ces deux langages sont parmi les tout premiers langages de haut niveau à avoir été largement adoptés par la communauté des informaticiens. FORTRAN a été développé au sein des équipes d'IBM et COBOL par des chercheurs universitaires (soutenus notamment par Grace Hopper).

Dans les décennies 1950 et 60, le logiciel était un composant central de ce produit commercialisable ; avant la décision d'IBM en 1965 de séparer logiciel et matériel, il n'avait pas ses propres circuits de distribution, n'existait pas en tant que produit à part entière.

Avant 1970, les employés d'une entreprise d'informatique écrivaient le logiciel en interne. La machine était le produit principal, et le logiciel était simplement un extra dans la facture. Mais les commerciaux d'IBM n'étaient pas les premiers à réfléchir à un marché indépendant pour le logiciel. Informatics et Applied Data Research avaient déjà pris cette voie [20]. Informatics en particulier a développé le premier logiciel ayant eu un succès commercial : un système de gestion appelé Mark IV qui coûtait US$ 30,000 en 1967. Le président d'Informatics, Walter Bauer rappelle qu'à l'époque, « les acheteurs potentiels étaient stupéfaits par ce prix. Dans un monde habitué à du logiciel gratuit, US$ 30,000 était en effet une somme élevée » [21].

La décision d'IBM a marqué un tournant décisif : l'industrie a commencé à croire en la portabilité, à considérer sa réalité pratique [22]. Plutôt que proposer un lot complet de matériel et logiciel, IBM a décidé de différencier ses produits : vendre du logiciel et du matériel séparément aux consommateurs [23]. Mais la question de la portabilité n'était pas seulement technique ; elle était aussi d'ordre politico-économique. La décision a été motivée à la fois par le désir de créer du logiciel de marque IBM pouvant être exécuté par toutes les machines IBM (c'était le but central du célèbre projet 0S/360 supervisé et analysé par Frederick Brooks), et comme réponse à un procès pour monopole intenté par le département de la justice américain [24]. Ce procès suggérait dans son argument l'idée que lier matériel et logiciel était une forme de comportement monopolistique, ce qui a poussé IBM à changer de stratégie en séparant l'ensemble en deux produits distincts.

Cependant, pour les entreprises le terme « portabilité » avait un sens bien particulier. Même si le logiciel pouvait être rendu portable sur un plan tech-

20. CAMPBELL-KELLY, *op. cit.*

21. *Ibid.*, p. 107.

22. CAMPBELL-KELLY et ASPRAY, *op. cit.*, p. 203-205.

23. Dans le procès qui oppose le Ministère de la Justice américaine à IBM, le pack logiciel-matériel a été considéré comme une preuve de comportement monopolistique–en plus de l'accusation contre la création des « Plug Compatible Machines », des machines faisant l'objet d'un processus d'ingénierie-inversée méticuleux : on construisait à la fois l'interface mécanique et le logiciel qui pourraient communiquer avec les ordinateurs centraux d'IBM (Franklin M. FISCHER, *Folded, Spindled, and Mutilated*, Cambridge, Mass. : MIT Press, 1983 ; Gerald BROCK, *The Second Information Revolution*, Cambridge, Mass. : Harvard University Press, 2003). (NdT) Pour une définition d'« ordinateur central », *cf.* note 50 page 28.

24. Frederick P. BROOKS, *The Mythical Man-Month : Essays on Software Engineering*, Reading, MA : Addison-Wesley Professional, 1975.

nique – transférable entre deux machines IBM – il n'existait aucune garantie que cette portabilité soit possible entre deux clients d'IBM. Le programme de comptabilité d'une entreprise, par exemple, pourrait ne pas convenir aux pratiques d'une autre. La progression vers la portabilité a ainsi été freinée à la fois par la diversité des architectures informatiques et par la variété des pratiques organisationnelles des entreprises. Par conséquent, IBM et les autres constructeurs n'ont pas vu les avantages qu'il y avait à standardiser le code source – pour eux, c'était favoriser la compétition [25].

Par la suite, le sens que l'on a donné au mot produit n'était pas le même que celui donné à matériel ou logiciel, mais davantage une combinaison des deux. Les contours d'un débat sur la signification de la portabilité et du partage sont alors devenus visibles dans une série de défis à relever, à la fois en matière de création de langages de haut niveau, et de contraintes politico-économiques que les entreprises ont dû prendre en charge dans la mise en place de produits propriétaires bien distincts.

2. Le système de temps partagé UNIX

Dans ce contexte, l'invention, le succès et la prolifération du système d'exploitation UNIX semblent atypiques, une aberration dans les pratiques académiques aussi bien que commerciales qui aurait dû échouer au lieu de devenir le système d'exploitation portable le plus utilisé dans l'histoire, ainsi qu'un paradigme pour tous les systèmes d'exploitation en général. L'histoire d'UNIX démontre comment la portabilité est devenue une réalité et comment l'habitude de partager le code source d'UNIX est devenue de fait et dès ses débuts un standard.

UNIX a été écrit en 1969 par Ken Thompson et Dennis Ritchie aux Bell Telephone Labs à Murray Hill, dans le New Jersey. UNIX était l'aboutissement du projet Multics mené au MIT, financé en partie par les Bell Labs et dont Ken Thompson avait la responsabilité. Multics était l'un des premiers systèmes d'exploitation sur des systèmes à temps partagé (c.-à-d. plusieurs utilisateurs peuvent utiliser en même temps une même machine), une plateforme de démonstration pour un certain nombre d'innovations pionnières dans le domaine [26]. En 1968, les Bell Labs avaient cessé de le financer et

25. L'industrie informatique a toujours beaucoup reposé sur le secret de fabrique plutôt que sur les brevets et les droits d'auteur, ce qui crée un certain ordre, des formes d'accès et de circulation spécifiques ; cela a également déterminé les premières années de l'industrie logicielle. Pour un compte-rendu qui fait consensus des pratiques compétitives du secret dans l'industrie informatique, voir Tracy KIDDER, *The Soul of A New Machine*, Boston : Little, Brown, 1981.

26. Sur la question du temps partagé, voir Lee et *al.* : 1992. Multics est mentionné dans qua-

rappelé Ken Thompson à Murray Hill aux côtés de Denis Ritchie ; ils se retrouvaient sans machine, sans argent, et sans projet.

Thompson et Ritchie étaient tous deux des spécialistes des systèmes d'exploitation, des langages et de l'architecture informatiques dans un groupe de recherche sans financement, sans même une incitation à continuer leur travail dans ces domaines. Grâce à l'utilisation astucieuse d'équipement recyclé, et relativement isolés par rapport au reste du laboratoire, Thompson et Ritchie ont créé en deux ans un système d'exploitation complet, un langage de programmation appelé C, ainsi qu'un panel d'outils toujours très utilisés aujourd'hui. Le nom UNIX (brièvement UNICS) était notamment un jeu de mots puéril – un Multics castré [27].

Aucune mission, donc, pour susciter ou cadrer la créativité et le travail de Thompson et Ritchie ; mais cela était courant aux Bell Labs ; les chercheurs y étaient libres de travailler sur à peu près n'importe quoi, à la seule condition que cela ait un vague rapport avec les intérêts d'AT&T. Cependant, ils manquaient d'argent pour investir dans une machine plus puissante, propice au type de travail qu'ils effectuaient. Cela a eu pour effet d'influencer la conception du système : une unité de contrôle très mince (un kernel) a été favorisée pour diriger les opérations de base de la machine ainsi qu'une série extensible d'outils petits et indépendants qui avaient chacun une tâche à accomplir correctement mais qui, mis bout à bout, pouvaient exécuter des instructions plus complexes et plus performantes [28]. Avec l'aide de Joseph Ossana, Douglas McIlroy et d'autres, ils ont finalement réussi à faire campagne pour un nouveau PDP-11/20 en insistant non pas sur les avantages techniques du système d'exploitation UNIX lui-même, mais sur ses applications potentielles, en particulier celles appartenant à la catégorie du traitement de texte, qui se concentre sur le développement d'outils pour le formatage, la typographie, et l'impression, avec pour mission prioritaire de créer des applications brevetées, ce qui pour les Bell Labs et plus généralement pour AT&T était tout à fait recevable [29].

UNIX était unique pour de nombreuses raisons techniques, mais aussi

---

siment toutes les histoires de l'informatique ; le site de Tom Van Vieck reste l'une des meilleures sources sur le sujet(`http://multicians.org`).

27. (NdT) Le jeu de mots s'appuie sur l'idée que Multics avait plusieurs manières d'accomplir une même tâche, UNIX n'en avait qu'une seule, mais plus efficace.

28. Parmi les innovations techniques les plus admirées (héritées pour la plupart de Multics) figurent : le système de fichiers hiérarchique ; la console de commande pour interagir avec le système ; le choix de tout traiter comme une entité à part entière (un fichier), y compris les appareils externes ; l'opérateur « tube » [*pipe* ou *pipeline*, NdT] permettant aux résultats d'un outils d'être chaînés [*piped*, NdT], c'est-à-dire d'alimenter l'entrée de l'outil suivant, facilitant la création de tâches complexes à partir d'outils simples.

29. SALUS, *op. cit.*

pour une raison économique : ce système n'était ni vraiment académique, ni commercial. Martin Campbell-Kelly remarque qu'UNIX était un « système d'exploitation non-propriétaire d'une signification majeure » [30].

L'utilisation que fait Kelly du terme « non-propriétaire » n'est pas surprenante, mais elle est incorrecte. Bien que le langage de l'entreprise oppose régulièrement les mots ouvert et propriétaire tout au long des années 1980 et au début des années 1990 (et UNIX appartenait assurément au champ sémantique du premier), cette utilisation abusive illustre bien la confusion existant entre les termes propriété et distribution appliqués au logiciel, et à l'œuvre à la fois dans les usages populaires et académiques du terme. UNIX était en effet un logiciel propriétaire, puisque les Bell Labs détenaient tous les droits (ainsi que Western Electric puis AT&T), mais ces derniers n'en faisaient ni le commerce ni la promotion.

AT&T permettait par contre aux individus et aux entreprises d'installer UNIX et de créer des variantes de « Type Unix » [UNIX-like, NdT] pour des frais de licence très bas. Jusqu'à environ 1982, la licence d'UNIX était proposée aux universitaires pour une très petite somme : en général sans frais de droits d'auteur, mais avec une facture de service minimale (entre US$ 150 et US$ 800) [31]. La licence permettait aux chercheurs de faire ce qu'ils voulaient avec le logiciel à condition qu'ils le gardent secret : ils ne pouvaient ni le distribuer ou l'utiliser hors de leur laboratoire d'université (et encore moins l'utiliser à but commercial), ni en publier aucun extrait. En conséquence, le développement d'UNIX tout au long des années 1970 s'est fait selon des méthodes relativement informelles. Les Bell Labs ont gardé ce cap libéral non seulement parce qu'ils possédaient l'un des rares centres de recherche et développement à la fois académique et industriel, mais aussi parce qu'AT&T détenait un monopole gouvernemental qui fournissait des services téléphoniques au pays et était donc officiellement interdit d'entrée dans le marché des logiciels informatiques [32].

30. CAMPBELL-KELLY, *op. cit.*, p. 143.

31. Le site web de Ritchie contient une copie de la licence de 1974 (http://cm.bell-labs.com/cm/wo/dmr/unixad.html). Selon Don Libes et Sandy Ressler, « les licences originales étaient des licences source… Les institutions commerciales payaient des frais de l'ordre de US$ 20,000. Si on possédait plus d'une machine, on devait acheter des licences binaires pour chacune d'entre elles si on voulait installer UNIX [c'est-à-dire on ne pouvait copier la source et l'installer]. Le prix était honnête (US$ 8,000), dans la mesure où on pouvait les revendre. Par contre, les institutions éducatives pouvaient acheter des licences sources pour quelques centaines de dollars – juste assez pour que les Bell Labs couvrent leurs frais administratifs et le coût des bandes magnétiques » (Don LIBES et Sandy RESSLER, *Life with UNIX : A Guide for Everyone*, Englewood Cliffs, NJ : Prentice Hall, 1989, p. 20-21).

32. Selon Salus, cette pratique de licence résultait directement du décret antitrust du Juge Thomas Meaney en 1956 : AT&T devait révéler ses brevets et les licencier pour des frais minimaux

UNIX était à la frontière des milieux des affaires et de l'université, et cela impliquait deux choses. D'abord, le système était protégé de la pression du management et des marchés, ce qui lui a permis d'achever l'intégrité conceptuelle qui a attiré les universitaires. Ensuite, AT&T l'envisageait comme un produit potentiel au sein du marché émergeant du logiciel, et donc posant de nouvelles questions juridiques dans le contexte d'un régime de droits d'auteur en transformation permanente, de nouvelles formes de marché, ainsi que de nouvelles méthodes de développement, de support et de distribution du logiciel.

Malgré son positionnement marginal, UNIX a été un succès phénoménal. Les raisons en sont multiples : son esthétique, sa taille, l'intelligence de sa conception et de ses outils. Mais son adoption si large et rapide témoigne aussi de l'existence d'une communauté enthousiaste de scientifiques et d'ingénieurs à laquelle il était attaché [bootstrapped [33]], des utilisateurs pour lesquels un système d'exploitation puissant, flexible, bon marché, modifiable et rapide était une sorte de révélation. Il était une alternative évidente aux systèmes compliqués, mal documentés et mal ficelés qui accompagnaient par défaut les machines que les universités et les instituts de recherche achetaient. En d'autres termes, « ça marchait ».

L'inclusion du code source était une propriété clef d'UNIX. Quand les Bell Labs vendaient une licence d'UNIX, ils fournissaient une bande magnétique [34] contenant la documentation (c.-à-d. une documentation faisant partie du système, et non une fiche de type manuel technique qui lui était extérieure), une version binaire du logiciel, et le code source. L'habitude de distribuer le code source a encouragé sa maintenance, son expansion, sa documentation – ces modifications étant envoyées à Thompson et Ritchie. Ce faisant, les utilisateurs ont développé un intérêt à maintenir et soutenir le projet précisément parce que cela leur donnait la possibilité et les moyens d'utiliser leur ordinateur de manière créative et flexible. Une telle communauté mondiale et organisée autour d'utilisateurs ayant un intérêt commun (maintenir

(SALUS, *op. cit.*, p. 56). Voir aussi Brock : 2003, p. 116-120.

33. (NdT) Le terme *bootstrap*, qui désigne littéralement ce qui permet d'attacher les lacets sur les bottines, est probablement utilisé ici en référence au vocabulaire informatique qui l'a adopté (où il désigne principalement une procédure de démarrage d'un ordinateur ou une technique d'écriture d'un compilateur). *cf.*, dans la bibliographie, BARDINI, *op. cit.*, qui l'applique métaphoriquement aux premières conceptions de programmes facilitant l'interaction homme-machine.

34. (NdT) La bande magnétique est un support d'enregistrement standardisé dès les débuts de l'informatique numérique, et utilisé pendant longtemps pour la sauvegarde (et l'échange) de données. Malgré les progrès et la diversification des supports numériques, elle reste utilisée aujourd'hui pour la sauvegarde des données de grands sites web (en particulier ceux qui s'appuient sur des « fermes de serveurs »).

un système d'exploitation) préfigurait le public récursif, mais restait encore confinée au monde des scientifiques et chercheurs en informatique (les seuls ayant accès à des machines toujours hors de prix). UNIX n'était pas seulement un matériau logiciel quasi-commercial (une alternative au marché de détail où prime la valeur prix), mais aussi le premier à inclure systématiquement le code source comme partie intégrante de cette distribution, suscitant ainsi la motivation des universitaires et ingénieurs [35].

Au cours des années 1970, le bas prix de la licence, l'inclusion du code source, et l'intégrité de sa conception encouragèrent le portage d'UNIX sur un nombre remarquable d'autres machines. La motivation suscitée par l'engagement dans la création et l'amélioration d'un système de pointe était bien plus importante pour les universitaires, qui prenaient en charge eux-mêmes le processus de licence et de portabilité, que si on leur avait vendu le logiciel sans code source. Peter Salus avance ainsi que l'on percevait l'absence de support de la part des Bell Labs comme une incitation à développer et partager leurs propres trucs et astuces de maintenance. Les moyens, normes et pratiques de partage, de portage, de forkage, ainsi que la modification du code source se sont développés au sein même de la programmation d'UNIX – la conception technique du système facilite et dans certains cas reflète les normes et les pratiques du partage, et donc les systèmes d'exploitation reflètent les systèmes sociaux [36].

3. Partager UNIX

Entre 1974 et 1977 l'utilisation et le portage d'UNIX se répandent à une très grande échelle pour un système d'exploitation sans circuit de distribution formel et sans support officiel d'une entreprise ; cette évolution se fit petit à petit, à travers les contributions d'utilisateurs tout autour du monde. Un groupe avait fini de développer en 1975 USENIX [37]. UNIX avait atteint

35. Dans la science informatique, le code source était rarement partagé, du moins pas officiellement ; il circulait plutôt sous la forme de théorèmes, de preuves, ou de langages de haut niveau divers et théoriques, comme le MIX de Donald Knuth dont le but était d'illustrer des algorithmes (Donald E. KNUTH, *Art of Computer Programming*, Addison-Wesley Professional, 1997). C'est sous forme imprimée (des manuels, tutoriels et autres publications professionnelles destinées à l'apprentissage de la programmation) que l'on trouvait de véritables morceaux de code source.

36. On désigne du terme « philosophie UNIX » le développement simultané du système d'exploitation et les normes de création, de partage, de documentation et d'expansion qui le gouvernent. L'idée clef est que chacun doit travailler à partir des idées (du logiciel) des autres (Mike GANCARZ, *Linux and the Unix Philosophy*, Cambridge, Mass. : MIT Press, 1983 ; Eric S. RAYMOND, *The Art of UNIX Programming*, Boston : Addison-Wesley, 2004).

37. Les Bell Labs ont menacé la toute jeune liste de diffusion UNIX NEWS, arguant d'une utilisation abusive de marque déposée ; le terme « USENIX » a été trouvé comme un compromis,

le Canada, l'Europe, l'Australie et le Japon, et un certain nombre d'outils et applications circulaient de manière indépendante mais étaient aussi inclus par les Bell Labs eux-mêmes dans leurs fréquentes distributions. Pendant ce temps, le département des licences d'AT&T cherchait à trouver un équilibre entre la permissivité (favorisant la circulation et l'innovation) et la rigueur (pour garder au logiciel le statut de secret de fabrique). En 1980, UNIX était sans aucun doute devenu le secret le plus partagé de l'histoire de l'informatique.

Le modèle contributif qui sous-tend la circulation et le développement d'UNIX est mal documenté ; il présente des aspects à la fois techniques et pédagogiques. Sur le plan technique, les formes prises par la distribution sont nées de l'opposition aux tentatives d'AT&T pour le contrôler mais ont aussi été facilitées par sa politique de licence logicielle inhabituellement libérale. Sur le plan pédagogique, UNIX devient rapidement un objet paradigmatique pour les étudiants en science informatique précisément parce qu'il incluait le code source et était assez simple pour être parcouru en un ou deux semestres. Dans A Quarter Century of UNIX, les modalités du partage technique d'UNIX (en particulier les détails relatifs à un contexte où ce partage n'était pas strictement illégal, sans pour autant être autorisé légalement) sont relatées dans des récits fondateurs.

Le premier de ces récits trouve son origine chez Ken Thompson :

> Il faut avant tout se rendre compte que le monde extérieur tournait sur des versions successives d'UNIX (V4, V5, V6, V7) au contraire de nous, qui avions une vision continue. Nous avions utilisé V5 à un moment donné mais il se périmait vite, ne serait-ce qu'à cause du temps passé à le mettre en forme afin de l'exporter. Après V6, je me préparais à aller enseigner à Berkeley pour un an et avais bricolé un système que je pourrais prendre avec moi. Comme il était quasiment prêt pour la distribution, j'en ai fais un *diff* sous la forme de V6 [une bande magnétique contenant seulement les différences entre la version précédente et celle que Ken prenait avec lui]. En route pour Berkeley je me suis arrêté à Urbana-Champaign [l'Université, dans l'état de l'Illinois, près de Chicago, NdT] pour aller voir ce que faisait Greg Chesson… J'y ai laissé la bande en lui disant que je n'avais pas d'objection à ce qu'elle passe de mains en mains. [38]

évitant l'utilisation du nom UNIX et faisant référence à USE, le groupe original des machines DEC (LIBES et RESSLER, *op. cit.*, pp. 9).

38. SALUS, *op. cit.*, p. 138.

Le fait que la bande magnétique doive nécessairement « passer de mains en mains » témoigne de la différence entre les années 1970 et aujourd'hui : la distribution de logiciel impliquait à la fois le transport matériel de média et la copie numérique de l'information. La volonté de distribuer des solutions de maintenance pour réparer les bugs (via la bande *diff*) préfigure l'émergence imminente du logiciel libre : le fait que d'autres aient réparé des problèmes et partagé les solutions avec Thompson et Ritchie avait créé l'obligation de veiller à la diffusion la plus large de ces solutions, afin qu'elles soient portées sur de nouvelles machines. On peut avancer l'idée que les Bell Labs, au contraire, ont perçu ce problème à travers le filtre du développement logiciel : il fallait penser à une nouvelle distribution, renégocier le contrat ainsi que de nouveaux frais de licence pour la nouvelle version. La notion de « continuum » mise en avant par Thompson par opposition à celle de succession des distributions marque aussi la différence entre l'idée d'un ensemble d'objets régi par divers acteurs éloignés les uns des autres et celle d'un package logiciel produit et « emballé », dont la valeur de commodité économique était en train de s'imposer. Quand Thompson parle du « monde extérieur », il réfère non pas seulement aux gens hors des Bell Labs mais à la manière dont le monde serait perçu depuis les Bells Labs par les avocats et les vendeurs s'ils avaient dû créer une nouvelle version. Pour les avocats, la circulation du code source était un problème non pas pour des raisons commerciales ou légales (chaque nouvel extrait de logiciel requérant une licence), mais parce qu'il avait besoin d'être stabilisé. Distribuer des mises à jour, des solutions de maintenance, et surtout de nouveaux outils écrits par des utilisateurs qui n'étaient pas employés par les Bell Labs brouillait la clarté juridique même si cela en renforçait la qualité technique. Lou Katz l'explique clairement :

> On a récupéré un grand nombre de solutions pour réparer les bugs, et plutôt que de les distribuer un par un, Ken a créé une bande magnétique qui les compilait (« the 50 fixes ») [il s'agit probablement de la « diff tape » évoquée plus haut]. Certaines de ces réparations étaient relativement importantes, bien que je ne me rappelle d'aucune en particulier et **j'ai dans l'idée qu'une grande partie a en fait été effectuée par des gens extérieurs aux Bell Labs**. Ken a essayé de la faire circuler, mais les avocats n'arrêtaient pas de tergiverser. Finalement, alors que Ken se décourageait, quelqu'un « a trouvé la bande sur Mountain Avenue » [où étaient situés les Bells Labs]. Quand les avocats ont appris la nouvelle, ils ont fait appel à toutes les licences imaginables et ont menacé de graves conséquences si elle n'était pas détruite – après avoir cherché d'où elle provenait, ce que personne ne leur a ja-

mais dit, j'imagine (pas moi en tout cas) [39].

La distribution des réparations impliquait un rapport de force entre les ingénieurs et le management, mais était surtout et clairement motivée, comme Katz l'explique, par le fait qu'une « grande partie a en fait été effectuée par des gens extérieurs aux Bell Labs ». Cela impliquait deux choses : la première, qu'il existait un avantage indéniable à faire circuler les mises à jour sur le système ; la seconde, que la mainmise ou les réclamations d'AT&T sur les droits quant à ces réparations étaient contestables – ils auraient eu à se couvrir au niveau juridique, ce qui expliquait en partie les tergiversations et les menaces des avocats, qui essayaient probablement de gagner du temps afin de créer une version « légale » accompagnée des permissions adéquates.

Pour comprendre ce rapport de force, inutile d'imaginer un combat entre les forces rebelles du développement d'UNIX et celles, obscures, de l'empire des avocats et des managers. Il s'agit plutôt de l'opposition de deux méthodes pour stabiliser l'objet UNIX. Pour les avocats, la stabilité impliquait de faire d'UNIX un produit, entrant dans un cadre légal existant, répondant aux demandes spécifiques d'un monopole régulé, interdit de compétition libre avec d'autres constructeurs ; on devait pouvoir strictement rendre des comptes à propos de la propriété du matériel informatique, des idées et des contributions. Pour les programmeurs, elle émergeait de la redistribution d'un système constamment mis à jour et du partage exhaustif des innovations afin que ces dernières puissent être portées sur différentes machines. Pour les avocats, il était urgent de stabiliser UNIX sur le plan légal ; pour les ingénieurs, il était urgent de le stabiliser sur un plan technique et de favoriser la compatibilité entre les versions, en empêchant le forkage de UNIX qui sonnerait le glas de la portabilité. Cette tension a accompagné l'évolution d'UNIX et celle de ses dérivés – et au-delà, avec des conséquences dans d'autres domaines dans lesquels le système a des ramifications.

L'identité et les frontières d'UNIX se sont formées de manière complexe grâce à son partage et à sa distribution. Le partage créait sa propre morale et son propre ordre technique. Des questions troublantes ont immédiatement émergé : les versions réparées, accrues et développées étaient-elles toujours UNIX, et donc encore sous le contrôle d'AT&T ? Ou alors y avait-il assez de différences pour parler d'un nouvel objet ? On peut imaginer que la fameuse bande magnétique contenant des réparations (proposées par divers contributeurs hors des Bell Labs) a circulé parmi ceux qui ont participé à la création de nouvelles licences pour UNIX ; ces réparations modifiaient le système : était-ce toujours UNIX ? Ces questions se posaient aux plans légal et tech-

39. *Ibid.*. La mise en gras est de l'auteur du chapitre.

nique, mais aussi à leur croisement. Elles peuvent sembler rhétoriques, mais l'histoire du développement d'UNIX illustre quelque chose qui les subsume : alors que toutes les modifications possibles, au niveau technique et légal, ont pu être effectuées, le concept UNIX est resté remarquablement stable.

4. Le portage d'UNIX

La portabilité technique explique seulement en partie le succès d'UNIX. Ressource pédagogique, UNIX est rapidement devenu un outil indispensable pour les universitaires à travers le monde entier. On l'enseignait et on l'apprenait en même temps qu'on l'installait et qu'on l'améliorait. Le fait qu'UNIX a proliféré d'abord dans les départements universitaires de science informatique, et non pas dans les entreprises, au sein du gouvernement ou d'autres organisations, implique qu'il a joué un rôle pédagogique central dans cette génération de programmeurs et de scientifiques. Tout au long des années 1970 et 1980, il a exemplifié le concept même d'un système d'exploitation de type temps partagé et multi-utilisateurs. Deux nouveaux récits relatent la manière dont UNIX a été porté depuis les machines jusque dans les esprits, montrant les étapes de son développement et les résistances légales et techniques : le premier est rapporté par John Lions dans Commentary on Unix 6th Edition ; le deuxième est l'histoire de Minix racontée par Andrew Tanenbaum.

Le développement d'une pédagogie unixienne a stabilisé le concept au niveau du code source et au niveau légal. Le portage d'UNIX a eu tant de succès que même dans les cas où une version portée d'UNIX ne partage aucun code source avec l'original, elle est toujours considérée comme relevant d'UNIX. La nature protéiforme et excessive d'UNIX se révèle le plus clairement dans les narrations qui l'entourent, en particulier quand on la compare avec l'intégrité légale et technique d'un objet comme Microsoft Windows, dont il n'existe qu'une poignée de versions (NT, ME, 95, 98, etc.) et dont le code propriétaire, soigneusement contrôlé et emmuré dans des protections légales est seulement distribué à travers les ventes et les packs de service aux clients ou aux constructeurs d'ordinateurs personnels. Alors que Windows est beaucoup plus largement utilisé qu'UNIX, il est loin d'avoir sa portée paradigmatique : son intégrité est d'abord et avant tout légale. Pédagogiquement parlant, Windows est à un poisson ce qu'UNIX est aux leçons de pêche.

Le Commentary de Lions est célèbre : il est « le document le plus photocopié en science informatique. » Lions était enseignant-chercheur à l'Université de New South Wales au début des années 1970 ; après avoir lu le premier article de Ritchie et Thompson sur UNIX, il a convaincu ses collègues d'ache-

ter une licence auprès d'AT&T[40]. Comme de nombreux chercheurs, Lions admirait la qualité du système et était, comme tous les utilisateurs d'UNIX à ce moment-là, familier avec son code source – un prérequis pour l'installer, l'exécuter, ou le réparer. Lions a commencé à l'utiliser dans le cadre de ses cours sur les systèmes d'exploitation, et ce faisant il publia une sorte de manuel, composé de l'ensemble du code source de la version 6 d'UNIX (V6) et de commentaires et explications détaillés ligne après ligne. La valeur de cet ouvrage ne doit pas être sous-estimée, en particulier à cette époque où l'accès aux machines et aux logiciels pour s'entraîner était difficile : « les vrais ordinateurs avec de vrais systèmes d'exploitation étaient verrouillés dans les salles de machine et réservés au traitement de données 24h sur 24. UNIX a changé tout cela.[41] » Berny Goodheart, louant le Commentary de Lions, rappelle l'utilité pratique qu'avait le code source accompagné d'un commentaire : « Il faut bien saisir la signification du travail de John à cette époque : pour les étudiants en science informatique dans les années 1970, des problèmes complexes comme l'ordonnancement, la sécurité, la synchronisation, les systèmes de fichiers et d'autres concepts étaient au-delà de la compréhension normale et extrêmement difficiles à enseigner – on n'avait tout simplement pas assez de temps d'accès aux machines pour faire des études de cas. À la place, on apprenait la discipline en faisant des trous dans des cartes, en collectionnant des feuillets imprimés en continu, et ainsi de suite[42]. En gros, un système d'exploitation informatique à cette époque était considéré comme une énorme pièce de code propriétaire inaccessible.[43] »

C'était un document unique dans le monde de la science informatique, recelant une sorte de clef d'accès à la compréhension d'un composant central de l'ordinateur (ce qui était rare à cette époque) : le portage ne se faisait pas seulement entre les machines, mais aussi et surtout entre les esprits des jeunes chercheurs et étudiants en programmation dont le nombre compensait la faible quantité des machines. Il a suffi d'une bonne dose de toner et d'un circuit de distribution disposant de photocopies pour que plusieurs générations soient formées à l'aide du Commentary – la plupart finissant par

40. Ken THOMPSON et Dennis RITCHIE, "The UNIX Time-Sharing System", dans : *Communications of the ACM* 17.7 (1974), p. 365–375.

41. Greg Rose cité *in* John LIONS, *Lions' Commentary on UNIX 6th Edition with Source Code*, Réed. 1977, San jose : Peer to Peer Communications, 1996 (sans pagination).

42. (NdT) Les cartes à perforations étaient des moyens primitifs de programmation informatique : on y saisissait sous forme de trous et de pleins les données pour traitement dans la machine. Les résultats étaient donnés après le temps de traitement, par exemple sous la forme d'impression papier en continu (on parle aussi de « papier accordéon » afin d'illustrer la manière dont les différents feuillets se pliaient – une technique d'ailleurs toujours d'actualité).

43. LIONS, *op. cit.* (*idem*).

travailler chez des constructeurs informatiques. Par là même on peut dire que la distribution du système d'exploitation s'est effectuée sous une forme textuelle.

Malheureusement, la distribution de l'ouvrage était elle aussi restreinte au plan légal. AT&T et Western Electric, espérant pouvoir conserver à UNIX le statut de secret de fabrique, freinaient la circulation du livre. Au départ, Lions avait la permission de distribuer des copies à ceux qui possédaient déjà une licence pour UNIX V6 ; plus tard les Bell Labs allaient eux-mêmes le distribuer pendant une courte période, mais seulement à des utilisateurs ayant la licence, en interdisant la vente, la distribution ou la copie. Néanmoins, presque tout le monde semble avoir eu sous les yeux les pages cornées d'une copie issue de photocopies successives. Peter Reintjes écrit : « Nous avons vite été en possession de ce qui ressemblait à une photocopie de cinquième génération et quelqu'un, dont je tairai le nom, a passé toute une nuit dans la salle des photocopies afin de donner naissance à la sixième génération, une action expressément interdite par un avertissement explicite sur la première page. Il se passait plusieurs choses remarquables en même temps. Tout d'abord, nous pouvions enfin nous amuser avec du logiciel, alors qu'auparavant cela nous ennuyait. Ensuite, nous avions sous les yeux une sorte de critique littéraire appliquée à un logiciel. Nous étions de plus en train d'opérer l'avancée la plus significative de notre éducation en science informatique en déchiffrant intégralement un système d'exploitation. Enfin, nous faisions un pied de nez à la légalité ». [44]

Ces générations d'étudiants et de chercheurs en informatique partageaient ainsi un secret de fabrique qui était devenu secret de polichinelle. Tous les étudiants apprenant les bases du système UNIX à partir d'une photocopie du commentaire de Lions étaient également mis au courant des tentatives d'AT&T pour contrôler sa distribution légale sur la couverture du manuel. Le développement parallèle de la technique de photocopie a son importance dans ce contexte ; aux côtés des radio-cassettes et des magnétoscopes, la photocopieuse a précipité l'évolution de la loi sur les droits d'auteur adoptée en 1976 [45].

Trente ans plus tard, et bien après la réécriture complète du code source qu'il décrivait, le Commentary de Lions est toujours largement admiré par les geeks. Même si le logiciel libre a bouclé la boucle, en fournissant aux étudiants un véritable système d'exploitation qui peut être légalement étudié, enseigné, copié et implémenté, le genre de « critique littéraire » que le travail

44. *Ibid.*

45. (NdT) Le *Copyright Act* de 1976 prolonge les droits d'auteur à 50 ans après la mort de d'un auteur, et à 75 ans pour les œuvres collectives aux états-Unis.

de Lions représente reste extrêmement rare. Lire du code, même obsolète, avec un commentaire qui l'éclaire est toujours l'un des rares moyens pour véritablement comprendre les éléments de conceptions et les implémentations intelligentes qui ont fait d'UNIX un système si différent de ses prédécesseurs ; et même parmi ses nombreux successeurs, presque aucun n'a été porté avec tant de succès dans les esprits de tant d'étudiants.

Le Commentary a contribué à la création d'une communauté mondiale de gens liés par le corps d'un code source, à la fois dans sa forme implémentée et textuelle (photocopiée). Ce public récursif naissant n'était pas seulement en train de se définir comme appartenant à une élite technique constituée par sa création, sa compréhension, et la promotion d'un outil technique particulier, mais se reconnaissait aussi comme « hors la loi », une communauté constituée en opposition aux formes de pouvoir gouvernant la circulation, la distribution, la modification et la création des outils mêmes dont il faisait l'apprentissage par vocation. La connexion matérielle partagée autour du monde par les geeks amateurs d'UNIX avec leur code source n'est pas une expérience simplement technique, mais aussi sociale et légale.

Lions n'était pas le seul chercheur à reconnaître que l'enseignement du code source était la méthode la plus rapide pour une compréhension optimale. Le deuxième récit détaillant la circulation du code source est celui d'Andrew Tanenbaum, un chercheur en informatique reconnu et auteur d'un manuel standard sur l'architecture des ordinateurs, les systèmes d'exploitation, et les technologies de réseau [46]. Dans les années 1970, Tanenbaum avait aussi utilisé UNIX comme outil pédagogique pour ses cours à l'Université Vrije, à Amsterdam. Grâce au fait que le code source était distribué avec le code binaire, les étudiants étaient incités à explorer l'implémentation du système, et Tanenbaum utilisait fréquemment le code source ainsi que le livre de Lions dans ses cours. Mais selon son Operating Systems : Design and Implementation publié en 1987, « Quand AT&T a sorti la Version 7 [aux alentours de 1979], l'entreprise s'est rendu compte qu'UNIX était un produit commercial valable et l'a donc fait accompagner d'une licence interdisant que le code source soit étudié dans les universités afin d'éviter de mettre en danger son statut de secret de fabrique. De nombreuses universités s'y plièrent, en abandonnant l'étude d'UNIX et en se consacrant seulement à la théorie ». Pour Tanenbaum, cette alternative était inacceptable – de même, apparemment, que continuer à enseigner UNIX, ce qui le plaçait hors la loi. Il a donc décidé de créer un système d'exploitation nouveau, complet, et similaire à UNIX qui n'utilisait pas une seule ligne de code d'AT&T. Il baptisa cette création

46. Les deux manuels les plus célèbres de Tanenbaum sont *Operating Systems* et *Computer Networks*, qui ont été publiés en trois et quatre éditions respectivement.

Minix. C'était une version condensée, faite pour être exécutée sur des ordinateurs personnels (les PC d'IBM) et distribuée avec le manuel Operating Systems publié par les éditions Prentice Hall [47].

Minix a été largement utilisé dans les années 1980 en tant qu'outil pédagogique de la même façon que le code source de Lions l'avait été la décennie précédente. Selon Tanenbaum, le groupe Usenet comp.os.minix [48] comprenait 40 000 membres à la fin vers 1990, et ses utilisateurs envoyaient constamment des suggestions pour faire évoluer et améliorer le système. Son propre engagement dans l'enseignement impliquait qu'il n'incorporait que peu de ces suggestions, afin de garder sa simplicité au système pour qu'il soit publié dans un manuel et compris par les étudiants de premier cycle. Minix était fait d'un code source librement accessible ; c'était aussi un système d'exploitation totalement fonctionnel, voire une alternative potentielle à UNIX qui tournerait sur un ordinateur personnel. On avait là un exemple parfait de l'intégrité conceptuelle d'UNIX qui se communiquait à une nouvelle génération d'étudiants en informatique : le titre du manuel de Tanenbaum n'est d'ailleurs pas « les systèmes d'exploitation de type UNIX » mais bien Operating Systems [« Systèmes d'Exploitation »]. UNIX est ainsi sans aucun doute l'une des représentations les plus claires des principes qui allaient guider la création de tout système d'exploitation : même vingt ans après son invention, il est resté le cas d'école le plus largement étudié, quelles que soient les intentions ou les applications qu'on lui assigne.

Minix n'était pas un logiciel commercial, mais il n'était pas non plus un logiciel libre à proprement parler. Il était sous droits d'auteur et contrôlé par l'éditeur de Tanenbaum, Prentice Hall. Parce qu'il n'utilisait aucun code d'AT&T, Minix était également indépendant sur le plan légal, un objet ayant sa propre intégrité juridique. Cela, ajouté au fait qu'il restait fidèle à UNIX sur le plan conceptuel, éclaire bien les types de tensions gouvernant la création et le partage de code source. Par un renversement ironique, Minix est lui-même devenu un standard pédagogique de choix pour l'étude d'UNIX

47. Tanenbaum n'a pas été le seul à suivre cette voie. Douglas Comer, autre grand auteur reconnu dans le genre du manuel, a créé Xinu et Xinu PC (UNIX à l'envers) décrits dans son *Operating Systems Designs* (1984).

48. (NdT) Usenet est un réseau créé en 1979 à partir du protocole *Unix to Unix Copy* (*cf.* note 57 page 30), rattaché à Internet via le TCP/IP en 1983, et qui a la particularité d'être décentralisé : les utilisateurs gèrent des groupes de nouvelles (similaires à des listes de discussions thématiques) à partir de leurs propres serveurs. Il a la réputation d'être l'ancêtre des forums, notamment en raison de la structure en arbre de ses groupes – ici, par exemple, la branche *computers* (comp.) suivie de la sous-branche *operating systems* 'os.) puis de la sous-sous-branche *minix* (qui est un système d'exploitation parmi tant d'autres). C'est un espace de discussion extrêmement populaire dans les années 1980, et presque tous les utilisateurs d'Internet s'y retrouvent pour parler de leurs sujets favoris.

en 1991-1992, quand le jeune Linus Torvalds a créé un fork de Minix, encore une fois entièrement réécrit, et qui allait devenir le paradigme du logiciel libre : Linux. Tanenbaum avait pour but que Minix reste un système d'exploitation pédagogique et utile – concis, léger, et illustratif – alors que Torvalds voulait accroître et développer Minix afin de profiter de tous les avantages des différents matériels informatiques construits dans cette décennie. Chacun, cependant, croyait que pour comprendre parfaitement et le plus rapidement possible les principes des systèmes d'exploitation, il fallait garder au code source sa visibilité et continuer à le partager.

5. Forker Unix

La création de Minix était motivée par le désir de partager le code source avec les étudiants, ce qui dérangeait manifestement AT&T et menaçait le statut de secret de fabrique de leur propriété. Le fait que Minix puisse être appelé un fork d'UNIX est un aspect clef de l'économie politique des systèmes d'exploitation envisagés comme systèmes sociaux. Le fork réfère en général à la création d'un code nouveau, modifié depuis une source originale, résultant en deux programmes distincts mais cousins. Alors que si l'on modifie un moteur, on obtient seulement un moteur modifié, modifier un code source implique à la fois une différenciation et une reproduction, grâce à la facilité avec laquelle il peut être copié.

Comment Minix – une réécriture complète – peut-il toujours être considéré comme le même objet ? Du point de vue des lois sur le secret de fabrique, les deux objets étaient distincts, mais du point de vue des droits d'auteur il a peut-être existé un cas de violation des droits d'auteur, bien que AT&T soit moins concerné par ce dernier que par le secret de fabrique. Du point de vue technique, les fonctions et processus accomplis par le logiciel sont les mêmes, mais les moyens de les coder sont différents. Enfin, du point de vue pédagogique, les deux sont identiques en tant qu'ils exemplifient certaines des propriétés centrales d'un système d'exploitation (la structure en fichier-système, la mémoire virtuelle paginée, la gestion de processus) – tout le reste relevant de l'optimisation et d'autres détails superflus. Comprendre la nature du fork requiert également qu'UNIX soit perçu depuis un point de vue social, c'est-à-dire depuis la perspective d'un système d'exploitation créé et modifié par des utilisateurs-développeurs à travers le monde selon des demandes partielles et spécifiques. Cela forme la base de l'émergence d'un public récursif robuste.

L'histoire de la Berkeley Software Distribution (BSD), qui raconte notamment la façon dont les protocoles TCP/IP ont été incorporés à UNIX, illustre

la déambulation de l'un des forks du code source d'UNIX, impliquant une communauté de co-développeurs, les plus prestigieux. En 1975 Ken Thompson avait décidé de prendre une année sabbatique dans sa ville d'origine, Berkeley, se rapprochant du département de science informatique de l'université et participant à l'implémentation d'UNIX à l'aide de sa « diff tape » et de la V6. Il avait commencé à programmer un compilateur pour le langage Pascal qui pourrait tourner sur UNIX ; deux étudiants-chercheurs ont alors pris le relais : Bill Joy et Chuck Hartley. (Joy allait devenir le co-fondateur de Sun Microsystems, une entreprise spécialisée dans les stations de travail basées sur UNIX ayant rencontré un très grand succès).

Joy était l'un de ceux qui distribuaient (de manière informelle) le code source avec le plus d'enthousiasme. Muni du système Pascal, largement utilisé et robuste, ainsi que d'un nouvel éditeur de texte appelé ex (plus tard, vi), il a inventé le Berkeley Software Distribution (BSD), un ensemble d'outils pouvant être utilisés avec le système UNIX. Ces outils étaient des extensions de l'UNIX original, mais non une version complète et entièrement réécrite pouvant le remplacer. Aux dires de tous, Joy serait devenu une usine à distributions à lui tout seul, créant les bandes magnétiques, les mettant à la poste, prenant les commandes et encaissant les chèques – tout en continuant à programmer le logiciel [49]. Les utilisateurs d'UNIX tout autour du monde ont vite compris la valeur de ces extensions, et il ne fallut pas beaucoup de temps pour que l'UNIX BSD trouve sa propre place, aux côtés de l'UNIX d'AT&T.

Selon Don Libes, les Bell Labs permettaient à Berkeley de distribuer ses extensions d'UNIX si l'université possédait la licence pour l'UNIX original (un arrangement similaire à celui passé avec Lions pour son Commentary) [50]. Entre 1976 et 1981, BSD est peu à peu devenue une distribution indépendante – ainsi qu'une version complète de UNIX – réputée pour son éditeur vi et son compilateur Pascal, mais aussi parce qu'elle ajoutait de la mémoire virtuelle et pouvait être implémentée sur les machines VAX de chez DEC [51]. Cette si-

49. Marshall K. McKusick, "Twenty Years of Berkeley Unix. From AT&T-Owned to Freely Redistributable", dans : *Open Sources : Voices from the Open Source Revolution*, sous la dir. de Chris DiBona, Sam Ockman et Marc Stone, Sebastopol, CA : O'Reilly, 1999, p. 32.

50. Libes et Ressler, *op. cit.*, pp. 16-17.

51. Un procès récent entre SCO (entreprise basée dans l'Utah), le propriétaire actuel des droits sur le code source original d'UNIX, et IBM a de nouveau posé la question de la quantité de code de l'UNIX original existant dans l'UNIX BSD. SCO argue qu'IBM (et Linus Torvalds) a inséré du code original dans le *kernel* Linux. Cependant, il suffirait de rappeler les détours vertigineux pris par ce code pour affaiblir ces arguments : il a été développé aux Bell Labs, de nombreuses universités ont bénéficié de sa licence, il a fourni la base de BSD, été vendu à l'entreprise devenue SCO (auparavant Santa Cruz Operation) qui a elle-même créé une version nommée Xenix en coopération avec Microsoft. Cf. le diagramme composé par Eric Lévénez sur son site (http://levenez.com/unix), et pour plus de détails sur cette affaire, le site

tuation unique a été rendue possible par le statut quasi-commercial de l'UNIX d'AT&T. De nombreux utilisateurs (par exemple, des étudiants) n'avaient aucun moyen de savoir s'ils travaillaient sur l'UNIX d'AT&T ou de BSD : les fonctionnalités étaient les mêmes et aucun système n'affichait clairement son identité, à l'exception des notices de propriété qui apparaissaient parfois à l'écran (ce qui allait changer au cours de la décennie suivante). Aucune entreprise n'aurait accepté cette situation sauf si BSD était incorporé et distribué en tant que produit identifiable (notamment par un nom retenant l'attention) ; mais AT&T a fermé les yeux, et la prolifération de l'UNIX original a alors donné lieu à des projets de fork, des instances de code source qui portaient chacune le nom d'UNIX.

Alors que BSD se développait, il a fini par acquérir de nouvelles fonctionnalités absentes dans l'UNIX d'AT&T. Une des plus significatives a été l'inclusion d'un code permettant de connecter des ordinateurs à l'ARPANET en utilisant la suite protocolaire TCP/IP conçue par Vinton Cerf et Robert Khan. Ces protocoles étaient au centre du projet ARPANET, mais n'ont pas attiré l'attention de l'IPTO (Information Processing and Techniques Office) à la DARPA [52] avant 1977. Le but de ces protocoles était de permettre à différents réseaux, chacun à partir de sa propre machine et avec ses propres délimitations administratives, d'être connectés les uns aux autres [53]. Il y a bien un héritage commun à UNIX et l'ARPANET, via la figure de J.C.R. Licklider [54], impliquant l'imaginaire des systèmes d'exploitation à temps partagé couplé au rêve du « réseau intergalactique » ; mais les développements d'ARPANET et d'UNIX ont d'abord été complètement indépendants [55]. UNIX permettait

groklaw.com.

52. L'IPTO était en charge en particulier du développement de l'ARPANET. *Cf.* note 5 7 pour plus de détails sur la DARPA.

53. Vinton G. CERF et Robert KAHN, "A Protocol for Packet Network Interconnection", dans : *IEE Transactions on Communications* 22.*5* (1974), p. 637–648, URL : http://www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf. Pour l'histoire d'ARPANET, voir Janet ABBATE, *Inventing the Internet*, Cambridge, MA : MIT Press, 1999, Arthur L. NORBERG et Judy E. O'NEILL, *A History of the Information Techniques Processing Office of the Defense Advanced Research Projects Agency*, Minneapolis : Charles Babbage Institute, 1992, ainsi que les chapitres 1 et 5 de cet ouvrage qui s'attardent sur le rôle des protocoles et des RFC (« Request For Comments ») [(NdT) les RFC étaient des notes de travail papier puis électroniques pour la mise en place des protocoles et standards de l'Internet].

54. (NdT) Joseph C. R. Licklider est une des figures clefs de l'histoire d'Internet ; un psycho-acousticien ayant découvert le potentiel des ordinateurs et travaillé sur les systèmes à temps partagé au MIT dans les années 1950, il est considéré comme un visionnaire qui a lancé le grand projet de l'ARPANET dans la décennie suivante à partir de l'idée que les ordinateurs allaient augmenter les facultés cognitives de l'humain en supportant notamment la mise en réseau globale des cerveaux dans un « réseau intergalactique » – sa pensée entretient de nombreux liens avec la Cybernétique, très en vogue à cette époque. Cf. ABBATE, *op. cit.* ; WALDROP, *op. cit.*

55. *ibid.*, chap. 5 et 6.

le partage des ressources sur un seul ordinateur (selon le principe du temps partagé), qu'il soit ordinateur central [56] ou un mini-ordinateur, mais n'était pas destiné à être connecté à un réseau – c'est toujours le cas aujourd'hui [57]. À l'opposé, le but de l'ARPANET était explicite : permettre le partage des ressources à partir de machines à distance sur différents réseaux.

Afin de profiter des avantages de TCP/IP, il fallait implémenter les ressources sur chaque système d'exploitation connecté à l'ARPANET. Mais en 1977 les machines du réseau étaient devenues dépassées et très difficiles à maintenir ; ce qui grevait le budget, selon Kirk McKusik, était qu'il fallait porter les vieux logiciels protocolaires sur de nouvelles machines. IPTO a alors décidé d'une nouvelle stratégie en parallèle, à savoir travailler la coordination au niveau des systèmes d'exploitation : parce qu'il était à la pointe de la portabilité, le système UNIX a été choisi comme l'une des plate-formes clefs pour effectuer la standardisation. Aux alentours de 1978, l'IPTO a proposé un contrat à la BBN (Bolt, Baranek et Newman : l'un des premiers sous-traitants des technologies d'ARPANET) afin d'intégrer les protocoles TCP/IP au système d'exploitation UNIX. Mais selon Salus, les événements ont pris une drôle de tournure : « BBN a créé un premier prototype et l'a envoyé à Berkeley. Bill [Joy] a tout de suite commencé à le hacker afin de le faire tourner sur Ethernet à 56kbps/seconde en utilisant 100 % du CPU sur une machine 750... Il a lobotomisé le code et a poussé son processeur pour que sa puissance atteigne 700kbps/seconde. BBN, arrivant alors avec son prototype final, a protesté ; mais Bill ne voulait rien entendre. Ils se sont battus pendant des années pour savoir quelle version serait installée dans le système ; celle de Berkeley l'a finalement emporté. [58] »

Personne n'est tout à fait sûr de ce qu'il s'est vraiment passé. BBN aurait apparemment voulu donner le code à Joy pour inclusion dans son UNIX BSD et qu'il soit ainsi distribué ; Joy et ses collaborateurs auraient voulu coopérer avec Rob Gurwitz à BBN sur une implémentation finale, mais l'équipe de Berkeley aurait insisté pour « améliorer » le code afin que ses performances

56. (NdT) : les ordinateurs centraux, *mainframes* en anglais, sont parmi les premières générations de machines informatiques, de très grande taille, à l'architecture logiciel centralisée (et non répartie comme sur les autres ordinateurs) et dédiés au traitement de grands ensembles de données. On les appelle aussi les « superordinateurs » et ils sont toujours d'usage aujourd'hui dans les organisations ayant besoin d'une très forte puissance de calcul (banques, compagnies aériennes, administrations, laboratoires de physique...).

57. Ceci à l'exception d'un outil qui a son importance, le *Unix to Unix Copy Protocol*, ou uucp, qui était largement utilisé pour transmettre des données par téléphone et a fourni la base de la création de Usenet (Michael HAUBEN, Ronda HAUBEN et Thomas TRUSCOTT, *Netizens : On the History and Impact of Usenet and the Internet*, 1re éd., Los Alamitos : Wiley-IEEE Computer Society Pr, 1997).

58. SALUS, *op. cit.*, p. 161.

servent davantage ses besoins propres, ce que BBN aurait refusé [59]. Un des résultats de ces prises de bec a été la naissance d'un véritable fork : deux corps de code qui faisaient la même chose et étaient en compétition pour remporter l'implémentation d'un UNIX standard pour le TCP/IP. On a ici un cas de partage de code source menant à la création de différentes versions du logiciel – un partage sans collaboration. Certains sites sur le réseau ont décidé d'utiliser le code de BBN, d'autres celui de Berkeley.

Le fork n'implique pas toujours la divergence ; l'amélioration continue dans le port et le partage peut avoir des conséquences inattendues au moment où un fork se présente. D'un côté il s'agit de morceaux spécifiques de code source : ils doivent être précis, identifiables, et présentés avec une notice de droits d'auteur, comme c'était le cas pour le code de Berkeley. L'université de Californie avait d'ailleurs la réputation de contrôler cela de très près : on était permissif sur la distribution, du moment que la notice de propriété était respectée à la lettre. De l'autre, il existait des collections de code d'un genre particulier et fondées sur la collaboration, par exemple entre UNIX™, l'UNIX approuvé par DARPA ou plus tard, Certified Open Source [sm] [60], et identifiées par un symbole de propriété commerciale afin de différencier explicitement et légalement les produits – et non pas tant d'inscrire la propriété dans les instances spécifiques d'un produit.

Quelles sont les conséquences de tout cela ? Le code TCP/IP écrit par Joy a été incorporé non seulement dans l'UNIX BSD, mais aussi dans d'autres versions, notamment celle distribuée par AT&T (dont la licence, à l'origine, avait été attribuée à Berkeley) mais en retirant la notice de propriété de Berkeley. Cette étrange intrication de licences et de code a abouti à cette série de conflits entre AT&T et Berkeley, au cours desquels on a essayé de démêler les choses [61]. Quelqu'un qui ne ferait que passer par là et penserait innocemment qu'UNIX est un objet unique serait très surpris de voir que si les raisons de l'émergence de ces formes multiples sont obscures, les causes, elles, sont tout à fait claires : il existe une rivalité entre différentes versions du partage ; différentes manières, d'un point de vue moral et technique, d'envisager l'ordre s'enchevêtrant dans un complexe de valeurs et de codes.

Le fork BSD d'UNIX (et le sous-fork du TCP/IP) n'en était qu'un parmi tant d'autres à venir. Au début des années 1980, les forks d'UNIX prolifféraient et allaient être suivis par une commercialisation vigoureuse. Au même

59. Le *TCP/IP Digest* 1.6 (du 11 novembre 1981) recèle l'explication donnée par Joy au sujet des intentions de Berkeley (Message-ID : anews.aucbvax.5236).

60. (NdT) Abréviation de l'expression *standard model*.

61. Voir l'article de Andrew Leonard, « BSD Unix : Power to the People, from the Code », publié sur le magazine web *Salon* le 16 mai 2000 (http://archive.salon.com/tech/fsp/2000/05/16/chapter-2-partone/).

moment, la circulation du code source a commencé à ralentir tandis que les entreprises trouvaient des manières de rivaliser en ajoutant des caractéristiques et en créant du matériel spécifique à UNIX (par exemple, les stations de travail Sun Sparc et les systèmes d'exploitation Solaris, résultant de la commercialisation de BSD par Joy). Savoir comment faire fonctionner toutes ces versions entre elles est devenu la question centrale dans les discussions sur les systèmes ouverts, discussions qui allaient dominer les secteurs du marché informatique dédiés aux stations de travail et aux réseaux, mais se résorber en 1993, alors que l'arrivée d'Internet et le succès de Windows NT auprès du grand public étaient sur le point de bouleverser le sort de l'industrie UNIX.

Les conflits entre BBN et BSD ont eu une conséquence encore plus importante : l'adoption généralisée du TCP/IP. Il est estimé que 98 % des départements de science informatique aux États-Unis (et de très nombreux à travers le monde) ont incorporé cette suite protocolaire à leur système UNIX et ont ainsi pu accéder immédiatement à l'ARPANET [62]. Le moment où cela s'est fait n'est pas anodin : quelques années de plus (alors qu'UNIX commençait à être commercialisé) et ces protocoles n'auraient pu être implémentés aussi largement par les constructeurs (ou plus exactement, ils auraient été implémentés sous des formes non standard, et non compatibles), alors qu'avant 1993 les chercheurs en informatique voyaient clairement les bénéfices à faire participer leur université au plus large réseau informatique de la planète. L'implémentation générale, fonctionnelle et relativement standard de TCP/IP sur UNIX (et la possibilité d'avoir accès au code source) a donné à ces protocoles un énorme avantage : leur succès et leur survie sont les fondements d'un réseau global et singulier.

6. Conclusion

Le système d'exploitation UNIX n'est pas simplement une prouesse technique ; il est le fruit de la création d'un ensemble de normes pour partager le code source dans un environnement inhabituel : quasi commercial, quasi académique, en réseau, et à l'échelle de la planète. Le partage du code source d'UNIX a pris trois formes fondamentales : porter le code source (le transférer d'une machine à une autre), enseigner le code source (ou le « porter » vers les étudiants dans un contexte pédagogique dans lequel l'utilisation d'un système d'exploitation fonctionnel facilite grandement l'enseignement de la théorie et des concepts), et forker le code source (modifier le code existant afin

62. NORBERG et O'NEILL, *op. cit.*, pp. 184-185. Ils citent le schéma de Douglas E. COMER, *Internetworking with TCP/IP*, Upper Saddle River, NJ : Prentice Hall, 2000, p. 6.

de faire quelque chose de nouveau ou de différent). Ce jeu de prolifération et de différenciation est essentiel à l'identité remarquablement stable d'UNIX, mais cette identité a plusieurs formes : technique (un système d'exploitation qui fonctionne et qui est auto-compatible), légal (une version circonscrite par une licence et sujette aux droits d'auteur et la loi commerciale), et pédagogique (un modèle conceptuel, le paradigme d'un système d'exploitation). Le code source ainsi partagé est par essence différent de tout autre code dans le monde des ordinateurs, qu'il soit académique ou commercial. Cela pose des questions qui dérangent sur la standardisation, le contrôle et l'audit, ainsi que sur la légitimité qui hante non seulement UNIX mais aussi Internet et ses divers protocoles « ouverts ».

Si la pratique du partage du code source dans l'univers du logiciel libre est telle qu'elle est aujourd'hui, c'est grâce à UNIX ; mais UNIX doit son existence non seulement au génie inventif de Thompson et Ritchie, et à la brillante stratégie marketing d'AT&T, mais aussi au fait que le partage produit son ordre propre en termes de systèmes d'exploitation comme de systèmes sociaux. Si les geeks se réclament d'une « philosophie UNIX », c'est qu'UNIX n'est pas seulement un système d'exploitation mais une manière d'organiser les relations complexes de la vie et du travail à travers des moyens techniques ; de rendre visibles puis remettre en question les frontières entre l'académique, l'esthétique et le commercial ; d'implémenter des idées relevant de la technique et de la morale. Par ailleurs, alors que le code source inclut de plus en plus les activités de la communication et de la création quotidiennes – et en arrive à remplacer l'écriture et à augmenter les capacités cognitives –, la généalogie et les récits relatant sa portabilité et sa forkabilité permettent d'éclairer les types d'ordres qui émergent dans les pratiques liées à la technologie, même éloignées des systèmes d'exploitation – mais intimement liées à la philosophie UNIX.