Les Cahiers de droit

Droit d'auteur et logiciel

Denis Buron

Volume 28, Number 2, 1987

URI: https://id.erudit.org/iderudit/042817ar
DOI: https://doi.org/10.7202/042817ar

See table of contents

Publisher(s)

Faculté de droit de l'Université Laval

ISSN

0007-974X (print)
1918-8218 (digital)

Explore this journal

Cite this note

Buron, D. (1987). Droit d'auteur et logiciel. *Les Cahiers de droit*, *28*(2), 421–440. https://doi.org/10.7202/042817ar

Article abstract

Under the present Canadian Copyright Act, it appears that protection is granted to computer programs, or software, as well as to flowcharts and firmware, or chips, whithout any relevant distinction.

But the real problem is one of evidence. As a matter of fact, no layman can be expected to appreciate correctly a reproduction of a substantial part of a program where even experts are facing a troublesome task. New rules of evidence should be contemplated, based on behavior, since it is easier for the defendant to show that he did exercise creativity than for the plaintif to sustain that his work has been copied. That is especially important, considering that copyright seems to be the only answer for such badly needed protection.

Note

Droit d'auteur et logiciel

Denis BURON *

Under the present Canadian Copyright Act, it appears that protection is granted to computer programs, or software, as well as to flowcharts and firmware, or chips, whithout any relevant distinction.

But the real problem is one of evidence. As a matter of fact, no layman can be expected to appreciate correctly a reproduction of a substantial part of a program where even experts are facing a troublesome task. New rules of evidence should be contemplated, based on behavior, since it is easier for the defendant to show that he did exercise creativity than for the plaintif to sustain that his work has been copied. That is especially important, considering that copyright seems to be the only answer for such badly needed protection.

---

* Avocat, Québec.

Les Cahiers de Droit, vol. 28, n° 2, juin 1987, p. 421-440
(1987) 28 *Les Cahiers de Droit* 421

Introduction

L'impact de l'ordinateur sur le droit d'auteur est presqu'inimaginable. On a écrit :

> Ce bond technologique revêt une importance comparable à celle de la révolution que provoqua la presse de Gutenberg ; en effet, les possibilités ainsi offertes, dont nous ne connaissons et n'exploitons encore qu'un certain nombre, sont pratiquement illimitées. [1]

Différents aspects de cet impact sont susceptibles de faire l'objet d'une étude. Mentionnons les banques de données, la création assistée par ordinateur et les logiciels.

Dans le premier cas, l'ordinateur se transforme en support pour œuvre de toute nature. Les droits de reproduction, de représentation ou de traduction peuvent poser des problèmes.

L'ordinateur est également un outil de création. Nous croyons qu'il doit alors être traité de la même manière qu'un instrument de musique, un pinceau ou même un dictionnaire entre les mains d'un auteur et, par conséquent, aucun droit d'auteur n'y est alors applicable.

Le logiciel est enfin le trait commun entre toutes les utilisations possibles de l'ordinateur. Nous nous concentrerons ici sur cette question de l'étendue

1. *Une charte des droits des créateurs et créatrices*, Approvisionnements et services Canada, 1985, p. 43.

de la protection accordée, ou souhaitée, au logiciel par le droit d'auteur. Nous verrons ensuite quels droits sont protégés en l'espèce, de même que l'applicabilité de ces droits compte tenu du contexte particulier auquel nous faisons face. Enfin, nous tenterons, très brièvement, de brosser un tableau des autres moyens de protection disponibles, dans le but de situer le droit d'auteur par rapport à ceux-ci.

Au départ, il est essentiel toutefois de se retrouver dans les aspects techniques et le jargon de l'informatique.

1. Aspects techniques

L'ordinateur comprend deux types de composantes : matériel et logiciel (*hardware* et *software*). Voyons ce qu'il en est.

1.1. Matériel

Plusieurs sont désormais familiers avec l'aspect extérieur d'un micro-ordinateur. Celui-ci se compose généralement d'un clavier, d'un écran et d'un lecteur. S'y ajoute fréquemment une imprimante. L'écran et l'imprimante ont des fonctions similaires, soit de prendre connaissance du contenu de la mémoire interne de l'ordinateur : données, résultats ou encore programme.

Le lecteur de disquettes, bandes magnétiques ou autre, de même que le clavier ont pour fonction d'inscrire les données ou programmes dans la mémoire interne de l'ordinateur. Le lecteur peut également servir de mémoire externe, c'est-à-dire servir à l'enregistrement de programmes, données ou résultats pour usage ultérieur.

L'ordinateur comprend aussi des interfaces, servant à relier ses composantes à la mémoire interne. C'est dans cette mémoire interne que tout se passe. Avant de voir son fonctionnement, voyons-en la composition.

Celle-ci se divise en deux parties : ROM et RAM. Toutes deux sont des puces (*Chips*), c'est-à-dire des microcircuits électroniques.

Le ROM (*Read Only Memory*) est une mémoire permanente, non-effaçable, dans laquelle est inscrite la programmation de base pour cet ordinateur, tel que conçu par le constructeur. Notamment, y est inscrite la signification des différentes touches du clavier. Cette mémoire peut également comprendre un ou plusieurs langages de programmation.

Le RAM (*Random Access Memory*) est la mémoire vive de l'ordinateur. Celle-ci contiendra toutes les informations, données, programmes et résultats que l'utilisateur y introduira ou obtiendra, selon le cas.

Cette mémoire, dont la capacité est exprimée en K bits (*K bytes* en anglais)[2], est non seulement effaçable ou modifiable mais elle est aussi automatiquement effacée lorsque s'éteint l'ordinateur. C'est en interchangeant le contenu du RAM, avant, pendant et après traitement, que l'on obtient les résultats demandés, à l'aide du programme approprié. Les résultats sont ensuite dirigés vers les périphériques de sortie, écran, imprimante, disque, etc., selon le cas. Ces différentes opérations internes se font dans le CPU (*Central Processing Unit*), sur lequel il n'est pas pertinent d'insister.

L'ensemble des puces composant la mémoire interne de l'ordinateur constitue ce que certains appellent le *firmware*. Ces puces, individuellement, sont de minuscules enchevêtrements de lignes dont des sections sont susceptibles d'être chargées positivement ou négativement. En fait, ce sont des circuits imprimés contenant, actuellement, jusqu'à 100 000 transistors, le tout de la taille d'une tête d'épingle[3].

1.2. Logiciel

Nous ferons abstraction de toute querelle de terminologie en utilisant, indifféremment, les termes « logiciel » et « programme ».

Qu'est-ce qu'un logiciel? On s'entend généralement pour dire que c'est une suite d'instruction s'adressant à l'ordinateur dans le but d'obtenir un résultat[4].

Comment se présente un logiciel? On peut le retrouver par écrit, sur disquette ou même sur une puce. Selon le support utilisé, le logiciel ne sera pas tout à fait le même. En effet, lorsqu'il est écrit, le programme sera généralement exprimé sous forme de texte à l'aide d'un langage évolué. S'il est inscrit sur disquette, il est composé d'une série d'impulsions électriques positives ou négatives, représentées symboliquement par les chiffres « 0 » et « 1 ». C'est ce que l'on appelle le langage machine.

Enfin, s'il est gravé sur une puce, par un procédé photo-lithographique, il est un assemblage de circuits de silicium dans lesquels des particules de phosphore ou de bore, en permettant au courant de circuler ou en l'empêchant selon le cas, sont compris comme étant, symboliquement toujours, « 0 » ou « 1 ».

---

2. 1 K bit est égal à 1024 bits, c'est-à-dire 1024 caractères binaires, chacun comprenant 8 chiffres (toujours 0 ou 1).
3. Pour plus de détails, voir : HART, R.J., « Questions Raised on Legally Protecting Semiconductor Chips in the UK », (1985) 1 *Computer Law & Practice*, 146.
4. Sur cette question et la problématique qui en découle, voir : Palmer et Resendes, *Le droit d'auteur et les ordinateurs*, Consommation et corporation Canada, 1982, p. 3 à 14.

Un type particulier de programme mérite d'être mentionné : l'assembleur. L'ordinateur, en effet, ne peut comprendre que deux choses, à ce jour, soit le positif et le négatif. En termes d'électronique, l'ordinateur comprend que le courant passe, ou qu'il ne passe pas. Ainsi, lorsqu'un programme est écrit en langage évolué, il doit être « traduit » pour être compris par l'ordinateur. L'assembleur est ce programme de traduction.

On appelle parfois[5] programme-source le programme écrit en langage évolué et programme-objet celui qui est « traduit » en binaire, ou langage machine. Pour des raisons exposées plus loin, cette distinction semble parfaitement inutile, quant au droit d'auteur.

Quant à la distinction entre langage évolué et langage machine, dont nous discutons déjà depuis un certain temps, qu'il suffise de dire que le premier est accessible directement à l'être humain, puisqu'il s'apparente à la langue anglaise, alors que le second est une suite de « 0 » et de « 1 » que même un programmeur chevronné aura peine à démêler sans l'aide... d'un ordinateur. Ce langage, diront certains, s'adresse à la machine.

On a donc inventé, par commodité, un certain nombre de langage dont certains sont d'application générale et d'autres plus spécialisés[6]. Le programmeur choisira celui qui convient le mieux aux impératifs qui le contraignent. Rappelons qu'au début de l'informatique, il devait composer ses instructions, en binaire, pour ensuite « allumer » ou « éteindre » de nombreux interrupteurs à la main. L'ordinateur accomplit désormais cette tâche pénible.

1.3. Création d'un logiciel

Le point de départ de la création d'un logiciel est, bien entendu, de déterminer ce que l'on veut accomplir à l'aide de l'ordinateur. Une fois cet objectif précisé, le programmeur décomposera les éléments nécessaires à la solution du problème en étapes distinctes et choisira, à cet effet, les algorithmes correspondants à la structure en question. Au besoin, il s'efforcera de trouver de nouveaux algorithmes.

L'étape suivante est de schématiser l'ensemble des algorithmes et, surtout, de les relier entre eux dans l'ordre dans lequel ils doivent être exécutés. Pour ce faire, le programmeur a avantage à dessiner un ordinogramme (*flowchart*), où les algorithmes sont symbolisés et où sont également représentés certains éléments distinctifs du problème.

---

5. Voir, en autre, *Dynabec Ltée* c. *Société d'informatique R.D.G. inc. et al.*, [1985] C.A. 236.
6. e.g. BASIC, FORTRAN, COBOL, PASCAL, APL, PL/1, LISP, PROLOG et encore bien d'autres...

Le plus gros de la tâche de la programmation est désormais complétée. En effet, il ne reste plus qu'à choisir le langage et à transposer l'ordinogramme dans ce langage, tout en y ajoutant différents commentaires ou remarques au besoin. Notons que certains programmeurs, plutôt que de choisir un langage évolué, préfèrent travailler directement avec un langage d'assemblage, celui-ci étant à mi-chemin entre le langage évolué et le langage machine. Finalement, après expérimentation, le programme sera révisé et amélioré jusqu'à l'obtention d'un produit final de qualité, c'est-à-dire le plus efficace possible.

Il est essentiel de remarquer que la possibilité que deux programmeurs, ou deux équipes, en arrivent au même résultat devant le même problème n'est que théorique [7]. Une grande part de créativité est en effet nécessaire dans le choix des algorithmes et dans la séquence utilisée par la suite. Enfin, toutes les variables [8] du programme final sont à la discrétion de l'auteur, de même, bien entendu, que les remarques.

Le programme terminé sera ensuite fixé sur un support quelconque. Évidemment, les différentes étapes ont pu être fixées. La fixation dont nous parlons ici est celle qui sera disponible pour l'utilisateur éventuel. Selon la nature du programme, le support choisi pourra être une disquette, une puce ou, possiblement, un imprimé. Dans ce dernier cas, l'utilisateur devra lui-même enregistrer le programme. Cette technique est grandement utilisée par les magazines spécialisés.

2. Étendue de la protection

Pour plusieurs raisons, incluant l'effort créateur, on a cherché à protéger le logiciel contre toute appropriation non-autorisée. Une des raisons principalement considérée fut la facilité extrême avec laquelle quiconque peut copier un logiciel, généralement sans même devoir débourser [9]. L'impact économique que peut avoir cette propension est catastrophique, compte tenu des coûts de développement énormes d'un logiciel performant.

---

7. Voir, à cet effet, BORKING, J. « Copyright protection of Software in Continental Europe, a status Quo », (1985) 1 *Computer Law and Practice* 42.
8. Variables: Tout ce que l'auteur décide d'utiliser pour nommer les données, sous-programmes, etc. Cela peut être une lettre, un mot, un symbole, bref n'importe quoi, sujet à certaine restriction tel l'usage des mots qui sont « réservés » à un langage, ou que l'ordinateur comprend une fois traduit en langage machine, ce contrairement aux variables, qui ne sont que des étiquettes.
9. Si l'on ne tient pas compte de l'achat d'un support, tel une disquette.

Voyons donc les motifs qui nous permettent de croire à la protection du logiciel par le droit d'auteur. Voyons, plus particulièrement, si les différentes étapes de l'élaboration d'un logiciel peuvent être protégées séparément.

2.1. L'algorithme

La première étape qui peut être protégeable est constituée des algorithmes choisis pour un programme. Nous devons noter, immédiatement, qu'il existe une certaine confusion entre les notions d'algorithme et d'ordinogramme, pour l'excellente raison que ces deux étapes sont rarement explicitées séparément par écrit. Dans le but de déterminer ce qu'il en est, reportons-nous à cette définition d'algorithme :

> Ensemble des règles opératoires qui définissent la suite des calculs à effectuer pour obtenir la solution d'un problème. [10]

De façon plus concise, l'algorithme s'apparente donc à une formule mathématique. Cette qualification entraîne des conséquences considérables en droit d'auteur. Effectivement, si l'algorithme est une formule mathématique, il n'est pas protégeable.

Une des règles fondamentales du droit d'auteur veut, justement, que l'on ne protège pas les idées mais uniquement l'expression des idées [11]. Or, une formule mathématique est une idée et, qui plus est, une idée qui ne peut être exprimée que d'une seule façon. Puisque l'on se refuse à accorder le monopole d'une idée, fusse-t-elle mathématique, celle-ci fait partie du domaine public. Par conséquent, nous sommes d'opinion que l'algorithme n'est pas protégé par le droit d'auteur et qu'il fait parti du domaine public.

2.2. L'ordinogramme

Qu'en est-il de l'ordinogramme? Celui-ci est, rappelons-le, la représentation schématique de l'ordre dans lequel les opérations doivent être réalisées pour obtenir un résultat. L'ordinogramme comprend un certain nombre d'algorithmes.

À première vue, l'ordinogramme est une œuvre littéraire. Rappelons la définition que donne la loi de ce type d'œuvre :

> « œuvre littéraire » comprend les cartes géographiques et marines, les plans, tableaux et compilations ; [12]

10. Le *Robert méthodique*, édition 1982.
11. On consultera à cet effet : Vincke, Côté et Nabhan, *Problèmes de droit d'auteur en éducation*, Éditeur officiel du Québec, 1977, p. 16 à 20.
12. *Loi sur le droit d'auteur*, S.R.C. 1970, c. C-30, article 2.

Il semble donc que l'on puisse considérer l'ordinogramme comme un compromis entre le tableau et la compilation, soit une compilation d'algorithmes représentée sous forme de tableau. Il ne reste donc plus qu'à déterminer si l'ordinogramme est l'expression d'une idée et non seulement une idée.

La réponse est simple. Il existe probablement des dizaines de façons d'exprimer un problème ; un ordinogramme n'est qu'une manière parmi tant d'autres de le faire. L'expérience apprend que l'ordinogramme est aussi distinctif que le logiciel lui-même. La protection à y accorder sera donc la même, en toute logique.

Nous verrons plus loin l'utilité que peut présenter la protection de l'ordinogramme séparément du logiciel. Ajoutons toutefois qu'il serait également possible de considérer qu'un logiciel est une copie ou une traduction de l'ordinogramme dans un langage quelconque.

2.3. Le logiciel

Bien que le droit d'auteur apparaisse comme un moyen de protection approprié du logiciel, il est nécessaire, néanmoins, que celui-ci puisse entrer dans une des catégories d'œuvres prévues par la loi, à défaut de quoi aucune protection ne saurait être accordée [13]. Puisqu'il était hors de question de considérer le logiciel comme une œuvre musicale ou artistique et qu'à première vue un logiciel est ou peut être écrit, c'est dans la catégorie des œuvres littéraires qu'il fut intégré [14]. Un logiciel, en effet, se présente comme une série d'instruction indiquant à l'ordinateur quoi faire, comment le faire et, parfois, indiquant au lecteur le pourquoi de telle ou telle instruction, au moyen de remarques et commentaires.

La ressemblance, sous certains aspects, avec un scénario, une mise-en-scène ou un arrangement scénique est relativement aisée à établir. Mais la définition d'«œuvre dramatique» de la *Loi sur le droit d'auteur* réfréna l'enthousiasme des plus entreprenants [15]. Puisqu'un logiciel n'est pas destiné à être récité, qu'il n'est certainement pas une chorégraphie ou une pantomime et qu'il est encore moins une production cinématographique, il semble exclu de la catégorie. Même s'il faut éviter de sauter trop vite aux conclusions, il semble malgré tout difficile de rapprocher suffisamment le logiciel des types d'œuvres énumérées pour l'inclure dans cette catégorie.

13. *Id.*, a. 45.
14. *Dynabec Ltée* c. *Société d'informatique R.D.G. inc.*, *supra*, note 5 ; voir aussi : *IBM Corp.* c. *Ordinateurs Spirales inc.*, 80 C.P.R. (2d) 187 ; [1985] 1 C.F. 190, de même que *Apple Computer Inc.* c. *Mackintosh Ltd.*, 10 C.P.R. (3d) 1.
15. *Supra*, note 12, a. 2.

Reste donc la possibilité, plus probable, que le logiciel soit une œuvre littéraire au sens large. L'arrêt *IBM* c. *Spirales* [16] a judicieusement analysé le pour et le contre de cette hypothèse. Revoyons-en les grandes lignes.

Quant à la forme, on avait tout d'abord objecté que le programme n'est pas écrit ou imprimé lorsqu'il est utilisé, de telle sorte que ne peut s'appliquer le test suggéré par l'arrêt *University of London Press Ltd v. University Tutorial Press Ltd* [18]. Le juge Reed répond à cela que le programme, sous au moins une forme, soit le programme-source, réalise les conditions de ce test [19]. Elle ajoute que la nécessité d'un écrit ne signifie pas que l'on doit être en présence de littérature, au sens artistique du terme.

Le juge écarte également l'argument voulant que le programme ne soit qu'une partie de la machine, l'ordinateur ; elle rejetait par le fait même toute analogie avec les rouleaux perforés pour piano mécanique [20]. Elle rejette aussi l'argument selon lequel le logiciel, ne s'adressant pas à l'être humain mais à une machine, ne peut être protégé par le droit d'auteur. Elle écarte ce dernier argument [21] en référant à une importante décision australienne, dont nous reproduirons le passage suivant :

> Copyright is essentially concerned with the expression of ideas in composition or language rather than with the function or purpose of those ideas. [22]

Le juge constate enfin que le critère de la fixation, issu de la décision *Canadian Admiral Corp. Ltd. v. Rediffusion* [23], se trouve respecté dans le cas du logiciel. Elle ne se prononce toutefois pas sur la suffisance d'une fixation dans une puce ou sur disquette [24]. L'argument tiré du fait qu'une telle fixation est invisible demeure donc en suspend [25].

Bien que convaincante, cette jurisprudence, de même que celle qui est survenue depuis [26], souffre d'un défaut considérable : toutes et chacune de ces décisions sont de la nature de l'injonction interlocutoire, ce qui revient à dire

16. *Supra*, note 14 ; les numéros de pages ci-après réfèrent au *Canadian Patent Reporter*.
17. Voir aussi, pour plus de détails : Schanfield Freedman et Winters, « Copyright goes Computer : A Case Law Survey of Recent Developments in Canada », (1985) 45 *R. du B.* 316.
18. [1916] 2 Ch. 601.
19. *Supra*, note 14, p. 193.
20. *Id.*, p. 195, l'argument étant basé sur *Boosey v. Whight*, [1900] 1 Ch. 122.
21. *Id.*, p. 195.
22. *Apple Computer Inc. v. Computer Edge Pty. Ltd.*, Cour fédérale d'Australie, 29 mai 1984, p. 32 des motifs du juge Lockhart.
23. (1954) 20 C.P.R. 75 ; [1954] Ex. C.R. 382.
24. *Supra*, note 14, p. 197-198.
25. Cet argument s'inspire également de *Boosey v. Whight*, *supra*, note 20.
26. *Dynabec* c. *R.D.G.*, *supra*, note 5.

que le tribunal n'avait qu'à décider s'il y avait un droit *prima facie*. Il est donc impossible d'affirmer que le tribunal serait arrivé à la même conclusion en rendant jugement sur le fond. Nous devrons donc nous contenter d'affirmer que la tendance est de considérer le logiciel en tant qu'œuvre littéraire, au sens de la *Loi sur le droit d'auteur*. Cette tendance est mondiale et nos tribunaux hésiteraient peut-être avant de la renverser. Néanmoins, une disposition claire de la Loi serait grandement appréciée. Nous prendrons donc pour acquis que le logiciel, de même que l'ordinogramme, sont protégés par le droit d'auteur au Canada.

2.4. La puce [27]

Rappelons que la mémoire interne d'un ordinateur (ROM et RAM) est composée de puces. Celles-ci sont donc des supports, au sens usuel du terme. Elles ont la particularité d'être de l'une ou l'autre catégorie, selon leur fonction, soit effaçable (RAM) soit permanente (ROM). Nous reviendrons plus loin sur la puce effaçable.

La puce à mémoire permanente est donc le support par excellence de la programmation centrale de l'ordinateur. C'est, économiquement, la partie de l'ordinateur la plus coûteuse à développer. Or, comme le logiciel considéré séparément, elle peut être copiée à peu de frais, toute proportion gardée.

Mais la protection de la puce vise un objectif beaucoup plus élevé. Il est établi qu'il est impossible de développer un ordinateur « compatible » à 100% sans copier les puces pertinentes [28]. Or la compatibilité permet de s'accaparer le marché développé par son concurrent sans payer aucun coût de recherche et de développement.

Par ailleurs, il serait possible de développer à ses frais des puces qui seraient compatibles à environ 98% [29]. Cette compatibilité est cependant insuffisante puisqu'à tout moment un logiciel risque dans ces conditions de ne pas fonctionner normalement et ainsi provoquer des erreurs inacceptables pour l'utilisateur.

Il semble donc essentiel de protéger la puce indépendamment du logiciel. Il est approprié de se demander, par conséquent, si la puce peut être protégée par la loi actuelle. Il semble bien que oui.

Si on observe une puce au microscope électronique, on aperçoit un enchevêtrement de lignes, légèrement surélevées. Si on se rapporte au procédé

27. Cette section est grandement inspirée de R.J. HART, *supra*, note 3.
28. Voir la preuve soumise dans l'affaire *Apple Computer*, *supra*, note 22.
29. *Idem*.

de fabrication, on constate que la puce est une superposition de photogravures, réalisée par procédé lithographique. La puce est à la fois un dessin, une photo, une gravure et une lithographie. Chacun de ces éléments est contenu dans la définition d'œuvre artistique de la *Loi sur le droit d'auteur*[30], explicitement ou non. La puce, indépendamment de son contenu, serait donc protégée au Canada.

Revenant à la puce effaçable, on peut formuler une objection pertinente : il est inapproprié d'accorder des droits à l'auteur d'un support qui est utilisé de la même manière qu'un ruban magnétique. Cela devrait normalement relever des brevets, s'il y a lieu, et non pas des droits d'auteur. Il n'y a pas de réponse certaine à cette affirmation. Ajoutons également qu'il n'est pas assuré que les créateurs de puces voient un intérêt à protéger ce type de puce.

On peut donc considérer la puce comme étant doublement protégée au Canada. D'une part le logiciel qui y est contenu, en tant qu'œuvre littéraire, et d'autre part le dessin particulier de cette puce en tant qu'œuvre artistique.

Dans le but de suppléer à leur législation sur le droit d'auteur, les États-Unis ont décidé de protéger spécifiquement les puces[31]. Dans les faits, ils ont protégé les négatifs servant à confectionner celles-ci, soit le *mask work*.

Cette Loi, qui fait partie des dispositions sur les droits d'auteur aux États-Unis[32], établit un régime légèrement différent pour les *mask works* que le droit d'auteur usuel. La période de protection, en particulier, est plus courte[33]. Caractéristique importante, l'enregistrement est nécessaire[34]. Enfin, mentionnons que la lettre « M » encerclée servira à identifier les appareils contenant des puces dont le *mask work* est protégé[35].

3. Droits protégés

Quelle est l'utilité et l'applicabilité de cette protection dont bénéficient logiciel et puce ?

3.1. Droit de reproduction

L'intérêt de la protection des logiciels par le droit d'auteur est, sans contredit, de prévenir la copie non-autorisée. Nous verrons, en deux sections,

30. *Supra*, note 12, a. 2.
31. *The Semiconductor Chip Protection Act 1984*, 17 U.S.C. 9.
32. Le titre 17 du *United States Code*.
33. 10 ans : article 904.
34. Dans les 2 ans de la première exploitation commerciale : article 908.
35. Article 909.

les particularités rencontrées quant à la reproduction des logiciels. Puis des puces.

3.1.1 Logiciel

Plusieurs problèmes d'application se posent lors de la mise en application du droit de reproduction quant au logiciel. Ces problèmes sont de deux ordres, selon la situation envisagée, soit la copie intégrale ou la copie partielle.

3.1.1.1. Copie intégrale

Il s'agit ici du cas classique, maintes fois soumis, du consommateur qui emprunte un programme pour s'en faire une copie, plutôt que de l'acheter. Il est ici question d'une véritable industrie de la copie et plusieurs variantes existent dans le but de parvenir au même résultat, soit une copie identique à un prix moindre.

Le problème ne se pose pas ici en terme de protection du logiciel : il est certain que cette copie n'est pas permise par le droit d'auteur. La seule difficulté qui subsiste est celle du contrôle de cette forme de copie privée. À cet égard, la question est comparable à celle qui se pose lorsque des cassettes audio ou vidéo sont utilisées pour enregistrer à domicile des copies d'œuvres musicales ou cinématographiques.

Le cas où une compagnie copie un concurrent pour mettre son produit en marché n'est cependant plus de la copie privée. Il est peu probable que celle-ci copie intégralement le logiciel, mot pour mot, car ici il serait relativement facile de découvrir le fautif.

3.1.1.2. Copie partielle

La copie partielle du logiciel est beaucoup plus plausible. Un logiciel est généralement composé d'une succession de sous-programmes. Il peut donc être avantageux d'emprunter un ou plusieurs des sous-programmes d'un logiciel pour en créer un nouveau.

En terme de droit d'auteur, la question est de savoir si un ou plusieurs de ces sous-programmes peut être considéré comme une partie importante du logiciel [36]. La notion de « partie importante » n'est pas définie par la Loi. On enseigne généralement qu'il s'agit tantôt d'une proportion de l'œuvre, tantôt

36. *Loi sur le droit d'auteur*, *supra*, note 12, a. 3(1).

d'une partie, même infime, considérée comme étant l'essence même de l'œuvre. Bref, l'on oscille entre des critères de quantité et de qualité, ce dernier l'emportant cependant le plus souvent [37].

Que devient ce test en informatique? Le problème en est surtout un d'appréciation. En effet, alors que le test normal est de savoir si la personne non-experte trouve importante la partie empruntée [38], il est nécessaire de se fier à l'opinion d'experts lorsqu'il s'agit de logiciel. Il n'existe donc pas de réponse certaine.

Mentionnons tout de même qu'il est peu probable qu'une ligne d'instruction, prise individuellement, soit une partie importante d'un logiciel. De même, les remarques et commentaires qui peuvent faire partie d'un programme n'en sont probablement pas des parties importantes, celles-ci étant inutiles à l'exécution du programme. Il y aurait donc lieu de croire qu'une partie importante serait constituée d'un certain nombre d'instructions, dans une séquence donnée. Un sous-programme, dans cette hypothèse, serait donc nécessairement une partie importante.

Afin de reconnaître les parties importantes, il pourra être utile de référer à l'ordinogramme de chacun des logiciels. Les parties y sont beaucoup plus apparentes et ce surtout aux yeux d'un non-expert.

3.1.1.3. Copie déguisée

Existe aussi la possibilité de cacher les éléments empruntés d'un logiciel. Ce genre de situation était jusqu'alors inconnue en droit d'auteur. Peuvent ainsi être dissimulées une partie de programme, importante ou non, ou même un programme en entier.

Comment peut-on ainsi « déguiser » ou « maquiller » un programme de façon telle qu'il sera invisible, c'est-à-dire que la copie ne pourra être décelée? Nous avons précédemment mentionné qu'un programme est composé de sous-programmes relativement indépendants. Or, il est possible de modifier l'ordre dans lequel se présenteront ces sous-programmes et même l'ordre dans lequel se présenteront les différentes instructions à l'intérieur d'un sous-programme. Bref, il est possible de tout mélanger. En conséquence, le *listing* — c'est-à-dire le programme sous forme écrite — apparaîtra considérablement différent, particulièrement aux yeux du non-expert. L'expert, pour sa part, saura y retracer les instructions indiquant à l'ordinateur dans quel ordre les instructions doivent être exécutées et pourra s'y retrouver, après un exercice long et fastidieux.

37. Pour plus de détails, se référer à VINCKE, CÔTÉ et NABHAN, *supra*, note 11, p. 34 à 39.
38. Autrement dit, si le juge en arrive à cette conclusion, *Deeks v. Wells*, (1931) 4 D.L.R. 533.

Mais ce n'est là que le maquillage le plus élémentaire et le moins efficace ! Tous les mots qui ne font pas parti du langage de programmation, toutes les variables, peuvent aussi être changés [39]. Enfin, rien n'empêche de changer le langage de programmation. Le maquillage est alors complet ; un expert n'a désormais que très peu de chance de s'y retrouver, encore moins d'être convainquant lors d'un éventuel témoignage.

On peut prétendre que la copie maquillée n'est plus une copie mais une création indépendante. À cet argument, nous rétorquons que fort peu de création est ici en cause. En effet, le point de départ est toujours une copie intégrale d'un programme, ou d'une partie de programme. Par la suite, chacune des opérations de maquillage peut se faire à l'aide d'un ordinateur et de logiciels appropriés. Ceux-ci permettront d'abord de copier le programme initial. Un traitement de texte permettra de changer mots et ordre apparent du programme, souvent en quelques instants à peine. Enfin, d'autres logiciels font la traduction d'un langage à un autre sans que l'on ait à intervenir.

Comment, alors, prouver l'atteinte aux droits du titulaire des droits d'auteurs ? Même les experts ne s'entendraient pas et, compte tenu de notre système contradictoire, n'éclaireraient pas beaucoup la cour. Deux moyens détournés de prouver cette atteinte sont possibles, afin d'éviter que toute protection ne soit illusoire. Le premier, le plus simple, consiste à recourir à l'ordinogramme. La seconde sera ce que nous appellerons la preuve comportementale.

Nous avons mentionné qu'il existe autant d'ordinogrammes, implicites ou explicites, que de programmes. Or, pour un logiciel donné, il n'existe qu'un seul ordinogramme. De plus, rappelons que lors de l'élaboration d'un programme l'ordinogramme survient avant le choix du langage.

La situation serait donc celle-ci : à défaut de pouvoir prouver la copie d'un programme, il est possible de prouver que l'ordinogramme est identique. Deux conclusions peuvent en être tirées: soit que l'ordinogramme a été copié, soit que, puisque l'ordinogramme est identique, une forte présomption pèse sur le défendeur à l'effet qu'il n'aurait pu obtenir ce résultat sans copier le programme du demandeur.

Dans le premier cas, puisque nous avons mentionné qu'à notre avis l'ordinogramme est une œuvre littéraire, alors l'atteinte aux droits du titulaire des droits de reproduction est prouvée. Dans le deuxième cas, une analyse du comportement du défendeur sera nécessaire.

Le principe applicable est ici relativement simple. Ce que la *Loi sur le droit d'auteur* vise à accomplir, c'est empêcher d'utiliser l'œuvre d'un autre

39. Voir *supra*, note 8.

comme étant sienne, ou partie de la sienne. La loi a, de fait, choisi d'accorder des droits exclusifs à l'auteur d'une œuvre. Ainsi, si l'on prouve qu'un tiers n'a pas développé lui-même son « œuvre » il ne peut en être l'auteur et a nécessairement enfreint les droits de cet auteur.

Ce type de preuve est déjà utilisée dans certains cas[40]. La condition préalable consiste à prouver la similarité des œuvres ainsi que l'accès à la source, sa disponibilité, pour ensuite renverser le fardeau de preuve et exiger du défendeur qu'il démontre que son œuvre n'est issue que de lui-même, de son propre travail.

Il est possible que la preuve de l'ordinogramme identique puisse servir de preuve de similarité. Par contre, la question se complique si cette preuve n'est pas disponible, soit parce que détruite, ou encore parce que le programmeur n'a pas fait d'ordinogramme. Il est d'autre part certain que le copieur n'aura pas d'ordinogramme. À moins de faire l'opération inverse, c'est-à-dire transformer le programme en ordinogramme, nous nous retrouverions donc avec une absence de preuve.

Pourrait-on, donc, aller un pas plus loin et exiger immédiatement une preuve comportementale, ou, à tout le moins, remplacer la preuve de similarité apparente par une présomption de fait ?

Cela a commencé à se faire aux États-Unis[41]. Le raisonnement est le suivant : moyennant une preuve d'accès et une preuve suffisante quant à l'identité des fonctions et du mode d'opération des programmes soumis, la cour est justifiée d'exiger du défendeur la preuve qu'il n'a pas copié le logiciel. Bien qu'il soit très incertain de pouvoir transposer cette procédure au Canada, il ne serait pas sans intérêt d'en faire une étude approfondie dans le but, possiblement, de la codifier lors d'une révision de la Loi.

Il est nécessaire d'insister sur l'utilité d'un tel raisonnement. Comprenons qu'en matière de preuve de copie de logiciel, le test de la similarité apparente aux yeux du non-expert n'est d'aucun secours. D'autre part, l'analyse des logiciels par des experts est très incertaine, sans compter qu'elle risque de favoriser la partie financièrement plus confortable. Quant à la preuve par ordinogramme, nous craignons qu'elle soit trop rarement disponible.

---

40. Voir VINCKE, CÔTÉ et NABHAN, *supra*, note 11, p. 40 à 45.
41. *Whelan Associates Inc. v. Jaslow Dental Laboratory Inc.*, n° 83-4583 (E.D. Pa. Jun. 22, 1985) ; *SAS Institute Inc. v. S & H Computer Systems Inc.*, n^{os} 82-3669, 82-3670 (M.D. Tenn.Mar. 6, 1985), toutes deux en cour de district et citées par : CONLEY and BRYAN, « A unifying theory for the litigation of computer software copyright cases », (1985) 6 *Computer Law Journal* 55, p. 116. Cet article est à l'origine de l'argumentation de la présente section.

Enfin, soulignons que le juge risque de ne rien comprendre aux expertises, alors qu'il lui sera beaucoup plus facile de constater d'abord que les programmes font la même chose de la même manière et, finalement, d'apprécier le comportement du défendeur pour en établir la permissibilité en regard des dispositions de la *Loi sur le droit d'auteur*. Il est évident que le juge sera beaucoup plus confortable devant cette tâche beaucoup plus familière, beaucoup plus de sa compétence que les questions techniques.

Il est donc fortement recommandé aux entreprises de conserver tous les documents pertinents au développement de leurs logiciels. Cela pourra toujours leur servir en défense et pourra aider à la preuve en poursuite, en recourant à l'ordinogramme en particulier, et ce que la théorie de la preuve comportementale soit adoptée ou non tant par le législateur que par les tribunaux.

3.1.2. Puce

La question de la copie des puces présente un cas différent et moins complexe malgré les apparences. La puce peut être protégée tant de par le programme qui y est inscrit que par sa configuration.

Dans le premier cas, la preuve de copie devra être la même que pour toute copie de logiciel, avec les écueils que nous avons soulignés. Dans le deuxième cas, quiconque se munira des outils appropriés, soit des agrandissements à l'aide d'un microscope électronique, pourra en comparer les dessins assez facilement. Même si ceux-ci sont très complexes, il suffirait de superposer, par transparence, des agrandissements, pour avoir une preuve certaine de copie. La comparaison des masques[42], de la même manière, donnerait un résultat identique.

Il est donc probable que les puces soient mieux protégées que les logiciels eux-mêmes, puisque ce droit est d'application beaucoup plus aisée et, surtout, plus naturelle.

3.2. Droit de traduction

Deux situations peuvent faire l'objet d'un droit de traduction, soit, d'une part, la traduction d'un langage évolué à un langage inférieur (assembleur ou machine), ou, d'autre part, la traduction d'un langage évolué à un autre.

---

42. Terme correspondant à *Mask work*, tel qu'utilisé dans la loi américaine. Plus ou moins les négatifs, rappelons-le.

Soulignons que le terme « traduction », bien qu'utilisé par les praticiens de l'informatique, doit être interprété largement pour être compris en droit d'auteur. Il semble, malgré tout, que cela ne soit pas incompatible, bien que le terme translitération puisse être le plus approprié [43].

Dans les faits, la traduction en informatique s'apparente à l'écriture d'un texte en morse ou en braille, c'est-à-dire que la même chose est dite sous une forme différente. Dans aucun cas la langue n'est changée. En informatique, la langue anglaise sert autant aux langages évolués qu'aux symboles mnémotechniques des langages d'assemblage.

3.2.1. Traduction entre niveaux de langage

Il semble inapproprié de parler de traduction d'un langage évolué à un langage d'un niveau inférieur et inversément. En effet, compte tenu des termes de la *Loi sur le droit d'auteur*, il semble plus opportun de parler de « copie sous une forme matérielle quelconque » [44].

Rappelons que pour être mis en mémoire, tant à l'intérieur de l'ordinateur lui-même que sur support externe, tel ruban magnétique ou disquette, un programme doit être transformé en langage machine, c'est-à-dire en impulsion positive ou négative. Le symbolisme utilisé (les chiffres « 0 » et « 1 ») n'existe que pour se représenter ce qu'il en est.

L'analogie suivante peut être utilisée : est-ce que les charges magnétiques d'un ruban sur lequel est enregistré une œuvre musicale constituent une traduction de l'œuvre musicale ? Sûrement pas. La distinction programme-source/programme-objet devrait donc être abandonnée pour ne considérer le deuxième que comme la forme matérielle d'un programme lorsque fixé sur certains supports particuliers [45]. Soulignons que cette nouvelle caractérisation ne change rien aux droits exclusifs du titulaire des droits d'auteur, s'il détient toujours tous ses droits.

3.2.2. Traduction entre langages évolués

La traduction d'un langage évolué à un autre est mieux désignée. En effet, il s'agit bien ici d'une forme d'expression différente, bien que toujours apparentée à la langue anglaise. Ce droit appartient donc en exclusivité au titulaire des droits d'auteurs.

---

43. Voir les motifs du juge Lockhart dans l'affaire *Apple Computer*, *supra*, note 22, p. 37.
44. Article 3(1) de la Loi.
45. La Cour d'appel a penché dans ce sens dans *Dynabec* c. RDG, *supra*, note 5, p. 240.

Nous avons déjà discuté des difficultés de preuve qui peuvent se présenter. Rappelons que l'ordinogramme peut être ici d'une utilité certaine pour comparer une traduction à l'original, s'il y a lieu.

3.3. Droit d'adaptation

Le droit d'adaptation en droit Canadien ne comprend que le droit de transformer une œuvre littéraire en œuvre dramatique et inversément[46]. Il n'y a donc pas lieu de discourir plus longuement sur le sujet, à moins de considérer le logiciel comme œuvre dramatique, sauf, peut-être, pour mentionner que la définition de droit d'adaptation est fort variable selon les juridictions et comprend, parfois, le droit de traduction[47]. Il est donc opportun de ne pas importer littéralement la jurisprudence étrangère à cet effet.

4. Autres moyens de protection

Plusieurs s'interrogent encore quant à savoir si le droit d'auteur est la meilleure technique de protection disponible pour les logiciels. Il est certain, d'autre part, que ce n'est pas la seule. Sans en faire une étude exhaustive, mentionnons quelques autres moyens susceptibles de rencontrer l'approbation de quelques-uns.

4.1. Moyens techniques

Les concepteurs de logiciels se sont évertués à protéger leur produit par eux-mêmes. Ils ont donc inventé différentes « barrières » techniques visant soit à empêcher la copie ou encore rendre celle-ci inutilisable. Celles-ci vont de la possibilité de trouer au laser la disquette supportant le programme, à l'option d'une *time bomb*.

Dans le premier cas, toute copie privée du trou minuscule à un endroit précis peut être inutilisable, moyennant des instructions appropriées dans le logiciel lui-même. Dans le second cas, il s'agit de doter le logiciel d'une horloge et de lui indiquer de cesser de fonctionner lorsqu'une date donnée surviendra. Il suffit, pour le concepteur, de reprogrammer la « bombe » lors de l'entretien du système pour qu'elle n'agisse jamais à l'encontre de l'utilisateur autorisé. Toute copie illégale, cependant, se verra affectée.

---

46. Article 3(1)b), c) et e) de la *Loi sur le droit d'auteur, supra*, note 12.
47. En Australie, par exemple.

Toutes ces techniques, dont nous n'avons donné que deux exemples parmi beaucoup d'autres, souffrent de trois défauts majeurs : elles coûtent presque aussi cher à développer que le logiciel lui-même, tout programmeur ayant l'expérience et la patience nécessaire pourra les contourner, en modifiant les instructions appropriées ou, plus simplement, en passant par-dessus. Finalement, le concepteur est responsable en dommages-intérêts des pertes susceptibles de se produire chez l'utilisateur autorisé, advenant un mauvais fonctionnement de la technique utilisée. Voilà un danger que tous ne sont pas prêts à affronter.

Les moyens techniques sont donc plus ou moins efficaces et, progressivement, sont sujet à l'abandon.

4.2. Moyens contractuels

Il faut ici distinguer entre deux situations complètement différentes, soit le logiciel sur mesure et le logiciel pour diffusion.

Dans le dernier cas, la protection contractuelle est quasi-impossible, compte tenu de l'effet relatif des contrats. Le logiciel ici visé est celui disponible chez le marchand du coin. Le nombre d'intermédiaires entre l'acheteur et le concepteur est trop grand pour que l'on puisse efficacement lier tous et chacun par contrat. S'y ajoute la difficulté de surveiller, chez-lui, l'acheteur.

Différentes tentatives naissent tout de même dans ce sens, tel un emballage mentionnant que l'acheteur, en brisant un sceau, s'engage à ne pas faire ni permettre de faire de copie. La valeur juridique d'un tel processus est très discutable.

Par contre, le logiciel sur mesure est facilement protégeable par contrat. En effet si je commande un programme à un concepteur, selon un certain nombre de spécifications, il est aisé pour celui-ci de prévoir au contrat ce que l'acheteur pourra ou ne pourra pas faire du programme. Ce moyen gagne alors à être utilisé, puisqu'il peut être d'une efficacité remarquable, plus grande même que les droits d'auteur, sans pour autant renoncer à ceux-ci.

4.3. Brevet

On s'interroge toujours sur la possibilité de qualifier d'invention le logiciel et, a fortiori, la puce.

Il semble bien que dans quelques rares cas cela soit possible. L'incertitude, de même qu'un régime de protection jugé moins approprié sous la *Loi sur les brevets*, entraîne progressivement l'abandon de cette

discussion. Cela n'empêche pas, semble-t-il, que certains produits de l'informatique sont potentiellement brevetables.

Soulignons également que la tendance mondiale est d'écarter la protection par brevet au profit des droits d'auteur. Le Canada a avantage, au plan international, à abonder dans le même sens.

Conclusion

L'ordinateur n'a pas fini de nous étonner, pas plus que de nous apporter différents problèmes légaux auxquels il faudra bien trouver des solutions.

En matière de droit d'auteur, nous avons déjà l'habitude d'interpréter une Loi dépassée pour l'adapter à des situations désormais courantes. Nous avons vu que le logiciel ne fait pas exception à cet effet.

Cependant, comme bien d'autres, nous nous devons d'espérer plus de clarté dans une nouvelle Loi qui n'en fini plus de se faire attendre. Le législateur pourra alors en profiter pour aller plus avant dans sa réforme et prévoir des moyens de preuve plus appropriés au cas du logiciel. En cas contraire, il est probable que toute nouvelle loi ne soit qu'un coup d'épée dans l'eau, en ce qui a trait au logiciel. Il ne faudrait pas, encore une fois, devoir compter sur les tribunaux pour faire le travail et laisser dans l'incertitude une industrie que l'on veut pourtant encourager.