Normalisation de systèmes d'exploitation des ordinateurs

Autor(en): **Gagnebin, T.**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **72 (1981)**

Heft 23

PDF erstellt am: **22.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-905177

Normalisation de systèmes d'exploitation des ordinateurs

Par Th. Gagnebin

006.44:681.3;

1. Introduction

Alors que le débat sur la normalisation des langages de programmation les plus courants est loin d'être clos, il s'étend aujourd'hui aux systèmes d'exploitation (S.E.). En effet, une standardisation dans ce domaine est aussi importante que dans celui des langages pour plusieurs raisons:

- Langages et systèmes sont liés: les S.E. imposent certaines restrictions ou offrent leurs propres extensions aux langages de programmation qu'ils supportent.
- Les S.E. créent un environnement logiciel complexe.
- Ils suscitent des méthodes de travail et des habitudes, notamment par le langage de commande et les possibilités qu'ils offrent.
- Chaque S.E. a sa propre représentation des informations sur les principaux supports d'informations magnétiques.

Passer d'un système à un autre coûte cher, or l'évolution technique dans le domaine des microordinateurs impose un rythme d'évolution du logiciel sans précédent. Se concentrer sur le développement et diminuer l'effort de maintenance du logiciel sont d'une nécessité vitale. Atteindre ces objectifs ne se fait pas au hasard, et la normalisation des S.E. n'est qu'un des aspects de ce problème. L'exposé qui va suivre a pour but d'évaluer ce qu'une normalisation pourrait apporter, ainsi que de dégager des tendances à la lumière des expériences passées et de l'état actuel du problème.

Pour illustrer l'importance du problème, il faut rappeler que le projet ADA, un des plus significatifs du point de vue des normes, est essentiellement motivé par la constatation que malgré une augmentation considérable des frais de logiciel du DOD[1]) ($ 3,5 · 10^9 en 1974), un pourcentage toujours plus important de ces frais est englouti dans la maintenance (80 %). Dans l'ensemble des services du DOD, on utilisait en 1974 environ 400 langages différents et 200 systèmes différents!

Ces chiffres justifient largement la décision de définir une norme interne avec effet contraignant y compris pour les fournisseurs de matériel. Dans l'optique du DOD, ADA ne sera pas seulement un langage de programmation normé mais aussi le langage de base de parties importantes de ce qui est actuellement du domaine des S.E. La notion de «ADA Programming Support Environment» (APSE) définit un interface standard sur un système hôte. En ce sens, ADA concerne cet exposé et est, de plus, une des meilleures synthèses actuelles d'idées apparues dès les années 1970. Deux de ces idées ont particulièrement marqué l'informatique actuelle, à savoir la programmation structurée et la notion de portabilité.

Avant d'entrer dans le vif du sujet, il est important de donner quelques définitions.

2. Définitions

Un système d'exploitation d'ordinateur (S.E.) est:

- un interface de base entre le logiciel et le matériel, caractérisé par un ensemble de fonctions fondamentales,
- une technique de réponse aux événements de l'environnement matériel (par exemple en temps réel ou en temps partagé),
- une collection de programmes et un langage de commande permettant de développer et maintenir le logiciel et de gérer les informations (filers, éditeurs, compilateurs, etc.),
- tout S.E. largement utilisé est de plus un réservoir d'applications diverses et un moyen d'échanger efficacement du logiciel.

Principales caractéristiques d'un S.E.:

- supporte ou non les processus concurrents (multi-tâche),
- supporte le déroulement de plus d'un programme à la fois (multi-programmable)
- supporte plus d'un utilisateur à la fois (multi-utilisateur),
- nature du moniteur: séquentiel, en temps réel, en temps partagé,
- organisation de l'espace de travail, en particulier sur disque: structure d'arbre, structure de liste simplement chaînée, structure à accès aléatoire, structure en blocs contigus, etc.,
- liste des fonctions du système et moyens d'y accéder,
- nature du langage de commande du système: par mots-clés, par touches de fonction, par menu à structure arborescente, par langage conventionnel,
- étendue et nature du logiciel implémenté ou disponible,
- faculté d'adaptation à un autre hardware.

Notion de portabilité

Il est souhaitable qu'un même programme puisse tourner sur différents ordinateurs pour des raisons d'économie: beaucoup d'utilisateurs ont plus d'un type de machine, ils remplacent d'anciens modèles, parfois ils échangent entre eux des programmes. On ne peut pratiquement envisager la portabilité de programmes que s'ils sont écrits en un langage abstrait qui cache au mieux les différences entre ordinateurs. Des programmes écrits dans un même langage sont rendus portables de différentes manières:

- il existe un compilateur unique mais modifiable au niveau du générateur de codes pour chaque machine. Le compilateur en question doit être conçu spécialement pour séparer les parties d'analyse du langage et la génération de codes.
- il existe un compilateur pour chaque machine. Ceci n'est envisageable que pour quelques langages normés très répandus.
- il existe une machine virtuelle unique, supportant le compilateur, qui peut efficacement être simulée sur différentes machines.

Pour qu'un S.E. soit portable, il faut non seulement qu'il soit écrit comme un programme portable mais que ses interfaces au hardware soient bien délimités et paramétrables.

3. Fondements et objectifs d'une normalisation

La politique en matière de logiciel des constructeurs de matériel est souvent fermée pour des raisons protectionnistes. Le logiciel est parfois insuffisamment adapté aux besoins du développement, surtout s'il s'agit d'un développement destiné à du matériel concurrent. En imposant aux utilisateurs des particularités plus ou moins intéressantes, les constructeurs escomptent s'assurer la fidélité de leur clientèle.

Le résultat de cette politique est l'émergence de familles de systèmes dont les ancêtres (encore vivants) sont parfois des systèmes provenant de constructeurs, parfois des systèmes développés par des utilisateurs industriels ou universitaires. Leurs points communs qui intéressent en matière de normalisation est qu'ils tentent d'ignorer le plus possible les caractéristiques propres à tel ou tel constructeur. Il s'ensuit que toute norme en matière de S.E. devrait avoir (entre autres) pour objectif:

[1]) DOD = Department of Defence (USA).

– de promouvoir une conception logique des périphériques les plus courants et de l'environnement matériel en général (par exemple associer une notion de fichier ou de volume à une imprimante ou une console, voire à la mémoire),

– promouvoir des définitions de certains outils généraux comme des éditeurs, des «linker», des «filer», etc. Ces définitions devraient comporter des listes exhaustives de fonctions attendues ainsi que des détails de réalisation,

– définir les fonctions élémentaires d'accès aux périphériques les plus courants (horloge, écrans, imprimantes, disques, lignes de communication, etc.),

– définir les fonctions et les entrées des moniteurs ainsi que les procédures d'interfaçage des «drivers»,

– définir un ensemble cohérent de structures logiques sur les supports d'informations magnétiques,

– proposer des niveaux de sophistication à l'intérieur d'une famille compatible de S.E. (du mono-utilisateur jusqu'au gestionnaire de base de données),

– proposer une méthodologie de l'échange d'informations et de logiciel.

Un programme d'une telle ambition aurait des retombées intéressantes s'il aboutissait:

– il favoriserait la mobilité du personnel informatique,

– il pourrait servir de référence aux constructeurs de périphériques ou d'autre matériel,

– il fournirait un cadre de base pour le développement de systèmes particuliers ou de systèmes expérimentaux, directement utilisable,

– il favoriserait l'intégration future de nouveaux éléments matériels.

Nous sommes bien loin d'un tel objectif.

Pour prendre le problème de la normalisation par un côté plus terre à terre, regardons si des systèmes actuels pourraient constituer une norme de facto acceptable malgré leurs imperfections. Pour ce faire, examinons les raisons du succès de quelques systèmes en jetant un coup d'œil sur le passé récent.

4. La situation des années 70

A cette époque, quelques grands constructeurs dominent le marché. La plupart du temps, chaque type d'ordinateur d'un même constructeur dispose de son propre S.E. incompatible en grande partie avec le modèle voisin. Des programmes de conversion permettent les transferts entre systèmes; le logiciel, souvent en assembleur, n'est pas transportable. De plus, l'accès aux ordinateurs est difficile car ils sont isolés dans les «centres de calcul». Deux faits font évoluer la situation: l'arrivée des *microprocesseurs* et la prise de conscience de la valeur des *langages structurés*.

Les grands laboratoires industriels et universitaires vont rapidement se trouver confrontés aux problèmes suivants: plusieurs sites de développement hétérogènes avec difficultés de communication; dispersion d'efforts de «cross development»; utilisation de petits systèmes promus par les constructeurs de microprocesseurs, d'un confort d'emploi douteux et nécessitant souvent l'écriture de logiciel de développement complémentaire.

Au «Computing Science Research Center» des Bell Laboratories, des recherches sur la portabilité des programmes avaient déjà abouti à l'écriture de PFORT, programme qui vérifie automatiquement qu'une source FORTRAN est conforme aux normes ANSI de l'époque. Cependant, confrontés à leurs problèmes de développement, les chercheurs réalisèrent que les S.E. des machines étaient un obstacle aussi important à la portabilité que leur architecture hardware. Ils prirent donc l'option de développer leur propre S.E.: UNIX. La première implémentation de UNIX fut réalisée sur PDP–11 (1972) et fut menée de front avec l'écriture du langage structuré C. Bientôt C tourna sur un Honeywell 6000 et sur IBM 370. Les programmes écrits sur PDP–11 tournèrent sans difficultés sur les autres systèmes. Ces succès encouragèrent le projet d'écrire une version portable de UNIX en langage C. Elle fut réalisée sur un Interdata 8/32 (32 bits) en 6 mois (1977).

A l'EPFZ, au début des années 70, une intense activité de promotion de PASCAL conduit *N. Wirth* aux USA. Il y rencontre un écho beaucoup plus favorable à PASCAL qu'en Europe, au point que plusieurs universités développent et distribuent des compilateurs PASCAL. *K. Bowles*, professeur à l'Université de San Diego, entreprend un projet de diffusion de PASCAL basé sur un S.E. écrit lui-même en PASCAL et dont la portabilité se fonde sur l'existence d'une machine virtuelle et d'un noyau minimal d'entrée-sortie (BIOS) proche de celui du déjà célèbre CP/M. Son approche suscite un vif intérêt surtout de la part des utilisateurs de microprocesseurs moins répandus que les 8080 ou Z80. Vers 1979, le système est repris sur une base commerciale par Softech Microsystems qui continue son développement en essayant de fonder une base normée (P-code de la machine virtuelle à partir de la version IV.0).

On ne peut pas parler de norme sans mentionner CP/M. Ce petit système, dérivé amélioré d'ISIS d'Intel est apparu à l'époque de la mise sur le marché du Z80. Il est écrit en assembleur 8080, compatible avec celui du Z80. Son sort est intimement lié au succès de ces deux processeurs. Une importante quantité de logiciel en assembleur, des applications commerciales, des traitements de textes, les compilateurs des principaux langages de programmation sont disponibles sous CP/M.

Ce rappel de l'évolution illustre les fondements d'une normalisation de S.E.: large diffusion, portabilité.

5. Nouvelles exigences

Avec l'évolution actuelle vers les 16 et 32 bits, de nouveaux problèmes logiciels apparaissent avec beaucoup plus d'acuité: l'utilisation rationnelle de la puissance de calcul et de l'espace d'adressage, la synchronisation de processus et le partage de ressources. On constate deux faits:

1. Aucun des langages évolués classiques n'apporte une solution fiable aux problèmes dits «temps réel»;

2. Les S.E. offerts par certains grands constructeurs ont les primitives nécessaires mais ne sont pas publics ni adaptables à un environnement autre que la machine pour laquelle ils sont faits.

Pour combler ces lacunes, de nouveaux langages apparaissent: CHILL, PASCAL concurrent, mCP, MODULA, PORTAL, ADA, pour n'en citer que quelques-uns (dont 2 suisses!) et de nouveaux S.E., dont par exemple MP/M issu de CP/M, RMX 80/86 d'INTEL, SOLO de Brinch Hansen, etc.

Parmi les systèmes adaptés au nouveau contexte, UNIX a l'avantage d'être issu d'un environnement 16 bits, multi-utilisateur. Il est particulièrement bien placé pour devenir un standard de développement. Deux firmes importantes l'ont choisi comme outil de développement de base: Zilog va reconstruire le système pour le Z8000 en le nommant Zeus. Le système d'accès aux fichiers sera étendu de manière à éviter que deux utilisateurs modifient simultanément le même fichier, et la souplesse d'interfaçage des CRT sera augmentée. Whitesmiths Ltd. a développé une implémentation de UNIX pour

mini- et microordinateurs. Appelé IDRIS, ce système est spécialement conçu pour les machines bas de gamme. Il faut cependant 60 kbit de code du type PDP-11 pour supporter la partie résidente du système et la plus grande passe du compilateur C. Zeus-UNIX-IDRIS pourraient former le noyau d'une famille compatible de systèmes.

6. Systèmes à vocation spécifique

Un nombre important d'applications utilisent des systèmes hôtes, mais substituent en fait leur propre organisation des disques à celle de l'hôte. Il s'agit de certaines bases de données. Parmi celles-ci, il faut mentionner la première d'entre elles à être normalisée (ANSI Standard X11.1–1977). Il s'agit de MUMPS, créée en 1966 dans le Laboratory of Computer Science du Massachusetts General Hospital à Boston. Le laboratoire s'était donné pour tâche de créer un système répondant aux exigences suivantes:

- travailler sur un mini-ordinateur,
- disposer d'une base de données multi-utilisateurs, structurée hiérarchiquement,
- travailler en time-sharing,
- offrir un puissant outil de manipulation de chaînes de caractères
- disposer d'un langage de commande simple utilisable par des non-professionnels de l'informatique.

Le résultat fut le MHG Utility Multi-Programming System, un système autonome, et un langage (MUMPS). Divers dialectes de ce langage furent implémentés sur différents matériel et un Users Group (MUG) fut créé. Un MUMPS Development Committee (MDC) fut créé également, dont le travail aboutit à la normalisation du langage.

Aujourd'hui, des versions mono-utilisateur existent pour 8080 ou Z80, distribuées par divers constructeurs (IMSAI, Cromemco, Sol, Northstar). Les techniques utilisées par les implémenteurs de MUMPS sur différents microprocesseurs les ont amenées à une portabilité toujours plus raffinée qui a pour conséquence de rendre le système plus facile à maintenir, en particulier en vue du remplacement des disquettes par des disques Winchester.

7. Résumé de quelques S. E. répandus

CP/M: Mono-tâche, mono-utilisateur; processeurs Intel 8080, 8088, 8086 et Zilog Z80. Système séquentiel, organisation disque à accès aléatoire. Interface aux périfériques par un BIOS écrit par l'implémenteur et accédé par une «Jump table».

Langage système par mots clés. Presque tous les compilateurs courants sont disponibles, ils sont souvent écrits en assembleur et génèrent du code machine. Beaucoup de logiciel d'application ainsi que des éditeurs et du traitement de texte sont disponibles. La faculté d'adaptation à un autre hardware est faible, vu le peu de portabilité du logiciel.

MP/M: comme CP/M, sauf multi-tâches, multi-utilisateurs.

p–UCSD: mono-tâche, mono-utilisateur (version II). Toutefois, IBS a développé un hardware sur lequel le système UCSD tourne en version multi-utilisateur. A partir de la version IV, ce système reste mono-utilisateur, mais supporte l'écriture de processus concurrents, sans que le langage système, ni l'interface aux périfériques ne soivent revus.

Le système est séquentiel, bien que dans le langage PASCAL, le multi-tasking soit possible avec une synchronisation par sémaphores. Organisation de l'espace-disque: blocs contigus gérés par un algorithme d'allocation dynamique. Nature du «langage système»: menu à structure arborescente. Logiciel disponible: langage PASCAL étendu, FORTRAN 77, BASIC évolué, annoncé: COBOL. Le logiciel d'application disponible croît très vite et il existe deux Users Groups (USUS USA et USUS UK), par l'intermédiaire desquels du logiciel gratuit est obtenable.

Faculté d'adaptation très élevée par l'existence d'une machine virtuelle en plus d'un BIOS du type CP/M. La machine virtuelle est simulée avec les processeurs suivants: Z80, 8080, DEC PDP–11 et LSI–11, 6502, 6800, 9900, bientôt 8086, Z8000, 68000. De plus, il existe des processeurs exécutant directement le code de la machine virtuelle (Western Digital).

UNIX: Multi-utilisateur, multi-programmable, géré en time-sharing. Organisation disque basée sur une structure d'arbre.

Langage système utilisant des symboles spéciaux, dont une caractéristique essentielle est la possibilité de définir des «tuyaux de programmes». Etendue du logiciel disponible: plusieurs versions de certains langages, les langages courants, beaucoup de logiciel d'application, beaucoup de logiciel de communication, le langage C pour l'écriture de système. Faculté d'adaptation à un autre hardware: se fait en écrivant l'ensemble des routines I/O et un générateur de code pour le compilateur C (pour autant qu'il n'existe pas sur le matériel cible). Ce travail est plus complexe que l'adaptation du système p–UCSD par exemple, vu que la notion de machine virtuelle n'existe pas pour UNIX. La portabilité résulte d'une bonne conception du générateur de code du compilateur de C.

8. Conclusions

Les objectifs d'une normalisation des S. E. sont indépendants d'autres soucis normatifs fondamentaux: communication, bus et langages de programmation. Ces derniers, en particulier, peuvent être distribués sur différents S. E. Etant donné que les S. E. existants ne comprennent pas encore systématiquement un interface universel de communication directe, il est nécessaire de transmettre l'information par support magnétique. Les formulaires de commande de logiciel prennent aujourd'hui l'allure d'une table à plusieurs entrées dont l'une est le format des supports magnétiques, comme par exemple pour la commande du système p–UCSD ou du mCP.

Le prix de développement et de diffusion du logiciel pourrait être diminué efficacement en réduisant le nombre d'entrées dans la table. Ce problème est en fait très complexe vu la diversité des configurations hardware et l'étendue du spectre des tailles en kbit des S. E. existants. Il est vraisemblable cependant que deux familles de S. E. coexisteront: ceux destinés aux petites machines genre home computer et ceux destinés aux professionnels de l'informatique.

Dans la première catégorie, il existe une norme de fait avec CP/M. Il est cependant à craindre que CP/M ne parvienne pas à rompre ses liens avec les processeurs 8080–8088–Z80. Le système p–UCSD est mieux adapté à une plus large diffusion et devrait concentrer les nouveaux développements ou même prendre en charge des logiciels existants sur d'autres systèmes du même genre pour favoriser le transfert futur vers des systèmes plus performants basés sur des matériels à 16 bits.

Dans la deuxième catégorie, il existe aussi une norme de fait avec UNIX. Des efforts sont actuellement faits pour ame-

ner un véritable UNIX sur matériel 8 bits. (Il existe des pseudo-UNIX mono-utilisateur, qui n'ont de UNIX que le langage de commande, mais ne permettent pas d'échange de logiciel avec la communauté des utilisateurs de UNIX.)

Quoiqu'il en soit, dans le contexte des 16 et 32 bits, UNIX se répand rapidement. Cependant, tant que ADA n'est pas sérieusement sur le marché, il est trop tôt pour faire des pronostics. ADA est déjà distribué en version micro-ADA par des maisons de logiciel ou des constructeurs aux USA, sur des systèmes issus de la philosophie du p–UCSD. D'autre part, il n'est pas exclu que la notion de APSE ne conduise pas en fait à la définition peut-être plus formelle d'un S.E. normalisé.

Bibliographie

[1] *B. Weiner* and *D. Swartz:* Adapting UNIX to a 16-bit microcomputer. Electronics 54(1981)6, p. 120...124.

[2] *P. J. Plauger* and *M. S. Krieger:* UNIX-like software runs on mini- and microcomputers. Electronics 54(1981)6, p. 125...129.

[3] *S. C. Johnson* and *D. M. Ritchie:* UNIX time-sharing system: Portability of C programs and the UNIX-system. Bell Syst. Techn. J. 57(1978)6, p. 2021...2048.

[4] *P. J. Plauger* and *M. S. Krieger:* C-language's grip on hardware makes sense for small computers. Electronics 54(1981)6, p. 129...133.

[5] *R. J. Rothstein:* MUMPS: une épidémie qui s'étend. Paninformatic –(1980)1, p. 19...21 + Nr. 2, p. 6+8.

[6] *R. T. Walters* and *S. L. Johnson:* Strategy for an extensible microcomputer-based MUMPS system for private practice. IEEE Symposium on Computer Application in Medical Care (1979), p. 457...463.

[7] *P. Brinch Hansen:* The architecture of concurrent programs. Englewood Cliffs, N.J., Prentice-Hall. 1977.

Adresse de l'auteur

Th. Gagnebin, Hermes Precisa International, 8, rue des pêcheurs, 1400 Yverdon.