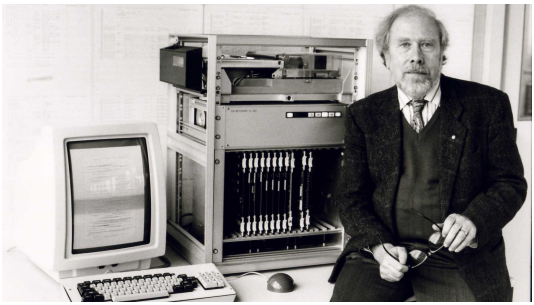
Hommage à Niklaus Wirth

Tristan Nitot [1]



Niklaus Wirth en 1984 à côté de Lilith à ETH Zurich. ©N. Wirth

Dans toutes les industries, il y a des figures légendaires. Dans le numérique (qui pour moi rassemble le matériel informatique et le logiciel), il y a indéniablement Niklaus Wirth [2]. Même si aujourd'hui, dans un monde qui va vite, peu de gens se souviennent de ce scientifique suisse qui vient de s'éteindre le premier janvier juste avant ses 90 ans. Et pourtant, quel parcours, quelles contributions, quelle sagesse et, chose plus rare encore dans ces métiers, une étonnante humilité.

On ne saurait résumer la vie de Niklaus Wirth en quelques mots : il est né en Suisse en 1934, a étudié à l'ETH Zurich, puis obtenu un doctorat en informatique à Berkeley. C'est là qu'il a découvert les langages informatiques et les compilateurs. Il a obtenu l'ACM *Turing Award* (le prix Nobel de l'informatique) en 1984. Il a inventé de nombreux langages, dont le célèbre langage Pascal [3] mais aussi Modula-2 [4].

1. Entrepreneur du numérique, https://www.linkedin.com/in/nitot.
2. https://fr.wikipedia.org/wiki/Niklaus_Wirth.
3. https://fr.wikipedia.org/wiki/Pascal_(langage).
4. https://fr.wikipedia.org/wiki/Modula-2.

DOI:10.48556/SIF.1024.23.133 **1024** – Bulletin de la SIF, numéro 23, avril 2024, p. 133–134

Mais réduire la carrière de Niklaus Wirth aux langages informatique serait une erreur. Il inventait des systèmes informatiques, comprenant un système d'exploitation, un environnement de développement avec un langage et un compilateur, avec les interfaces homme-machine.

Il a fait deux passages d'une année au Xerox PARC (*Palo Alto Research Center*), s'inspirant de la citation d'Alan Kay qui y officiait : « *Les gens qui font du logiciel sérieusement devraient construire leur propre matériel* ». C'est ainsi qu'en 1980, quatre ans avant l'arrivée du Mac, Niklaus Wirth a commencé à développer Lilith (cf. image page précédente), une des premières stations de travail avec une souris et un affichage graphique haute résolution, sans arriver au succès commercial des solutions américaines.

En 1992, dans le manuel du système Oberon [5], il explique que malgré la loi de Moore qui stipule que « *la puissance des semi-conducteurs double tous les deux ans* », les logiciels deviennent plus gros et moins optimisés au même rythme. On a appelé cela la loi de Wirth : « *En dépit de multiples bonds en avant, le matériel accélère moins vite que le logiciel ne se ralentit* ». Le système Oberon, qui était composé d'un système d'exploitation, d'un langage et d'un ordinateur, visait à contredire la loi de Wirth. En 2013 (il a alors 79 ans !), sortait une nouvelle version d'Oberon où Wirth est allé jusqu'à fabriquer son propre microprocesseur sur la base de circuits FPGA [6].

Wirth a aussi publié dès 1995 un plaidoyer pour le logiciel frugal [7] où il explique les origines de la loi de Wirth dans le fait que les auteurs de logiciels rajoutent des fonctionnalités inutiles pour inciter leurs clients à acheter la nouvelle version, ce qui rend le logiciel plus « gras », plus lent, et fait les affaires des fabricants de matériel, dont la précédente génération est devenue *de facto* obsolète. Les clients rachètent donc du matériel pour remplacer l'ancien qui fonctionne pourtant très bien. De nos jours, presque 30 ans plus tard, la notion d'obsolescence programmée est dorénavant connue de tous, et on réalise que cela fait 50 ans que les industries du matériel et du logiciel l'ont institutionnalisée.

Pourtant, alors que l'on doit réduire l'empreinte écologique de l'activité humaine pour faire face à l'effondrement de la biodiversité et au réchauffement climatique, et que le numérique pollue plus encore que le transport aérien, l'appel de Niklaus Wirth à plus de simplicité, d'optimisation, de sobriété et de frugalité, donc d'élégance, est plus que jamais d'actualité.

Merci pour vos contributions, professeur Wirth, puissent les communautés du numérique vous rendre hommage en suivant vos principes !

5. http://www.projectoberon.net/wirth/ProjectOberon/PO.System.pdf.
6. https://fr.wikipedia.org/wiki/Circuit_logique_programmable.
7. https://liam-on-linux.dreamwidth.org/88032.html.