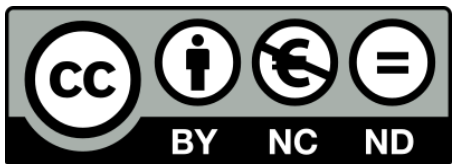
Culture et Histoire de l'informatique

Jean-Pierre Messager (jp@xiasma.fr)

19 décembre 2020 – version 0.99a



Utilisation commerciale interdite sans autorisation

Télécharger ce cours à jour et y contribuer

Il est disponible sur un dépot git

- Accès public à ce cours : https://framagit.org/jpython/culthistinfo
- Si vous créez un compte sur https://framagit.org/
- Transmettez-moi votre identifiant
- Je vous accorde le status *reporter* sur ce projet
  - Ouverture de tickets
- D'autres projets : https://framagit.org/jpython/meta
- Des mises à jours sont régulièrement disponibles
- Chaque changement important de version est accompagné d'un *tag*

Culture et Histoire de l'Informatique

Plan du cours

Calcul et algorithmes

Des ordinateurs humains

Dès l'antiquité on trouve la description d'algorithmes : description détaillée de tâches de traitement que l'on peut réaliser mécaniquement (sans rien y comprendre)

- L'algorithme d'Ératosthène (nombres premiers)
- L'algorithme d'Euclide (plus grand facteur commun)

Les abaques et les bouliers existent pour faciliter des calculs, comptables en particulier.

Systèmes de numération positionnels

Histoire

Découverts trois fois : En Chine – En Amérique par la civilisation Maya – En Inde, repris par le monde arabo-musulman et transmis ensuite en Europe : “chiffres arabes”

Le génie de ces systèmes : le zéro

On a une base, généralement dix, peut être quelconque. On a autant de signes que cette base, un premier signe pour “zéro” et des signes supplémentaires jusqu’à base - 1.

$$douze \rightarrow 2 + 1 * 10 \rightarrow 12$$

Notation avec des chiffres

Représentation et valeur

En général tout nombre s'écrit :

$$n = c_0 * b^0 + c_1 * b^1 + c_2 * b^2 + ... + c_N * b^N$$

on écrit le nombre sous la forme :

$$c_N c_{N-1} \cdots c_1 c_0$$

Pour trouver les chiffres représentant un nombre. On divise par la base on garde le reste de la division, et ainsi de suite... 126 / 10 reste 6 ; 12 / 10 reste 2 ; 1 / 10 reste 1 / (résultat 0 : stop)
En binaire : 126/2 (0) 63/2 (1) 31/2 (1) 15/2 (1) 7/2 (1) 3/2 (1) 1/2 (1) 0 (stop) donc :

$$126_{10} = 1111110_2$$

Quelle base utiliser ?

Compter sur les doigts

- La plupart des cultures utilisent la base dix
- La base vingt se rencontre dans les cultures celtiques, on en trouve la trace en français (influence du gaulois ?), en breton : quatre-vingt-dix, quatre-vingt (au lieu de octante, nonante), tri-ugent
- Systèmes à bases multiples : 10 et 60 (sumériens), qu'on retrouve dans le décompte des heures, des angles
- En informatique on aime bien le binaire (base 2), l'octal (base 8), l'hexadécimal (base 16)
- Le ternaire (base 3) n'est pas inintéressant
- L'URSS a développé un temps l'électronique ternaire, un projet contemporain existe : https://www.ternary-computing.com/

Généralisation des chiffres arabes

Nos chiffres : arabo-indiens

- Depuis la renaissance tout le monde utilise ce système
- Les opérations de calculs deviennent algorithmiques, “mécaniques” : addition, multiplication, la division habituelle est “presque” algorithmique
- Au XVIIe siècle Pascal et Leibniz, philosophes, mathématiciens conçoivent et construisent des calculatrices mécaniques

La mécanisation du calcul par des dispositifs mécanique se poursuit jusqu’à la 2nd guerre mondiale.

Un ordinateur mécanique ?

Calculatrice

- Une calculatrice ne fait que des calculs prédéfinies (addition, soustraction, multiplication, division, puissance, cos, sin, ...)
- Un ordinateur permet de mettre en œuvre un algorithme quel qu'il soit

Premier ordinateur

- En Angleterre Charles Babbage conçoit et commence à construire un véritable ordinateur mécanique
- Ada Lovelace, future baronne et fille de lord Byron, est la première personne dans l'histoire à avoir écrit des programmes au sens propre du terme
- La machine ne sera construire qu'au XXème siècle, on peut la voir à Bletchey Park au musée de l'informatique
- Machine à Différences

Qu'est ce qu'un calcul effectif ?

Quelles procédés mathématiques sont constructifs ?

La crise des fondements

- Le sujet est sur la table dans le contexte de la crise des fondements en mathématique (fin XIXème, début XXème)
- Comment définir la notion de calculabilité ? Comment spécifier mathématiquement ce que une machine potentielle peut faire ou pas ?
- Trois définitions ont été proposées :
  1. Logique combinatoire
  2. Lambda-calcul (Church)
  3. Machines de Turing
- Thèse de Church-Turing : les trois sont équivalents, et toute autre façon de mécaniser un calcul mène à la même chose (quasiment un théorème prouvé, à condition d'en préciser l'énoncé)

Machines de Turing

Alan Turing, définit une machine théorique très simple :

- Ruban de longueur infinie, découpé en cases
- Alphabet : symboles présents ou non sur une case (symbole “blanc”), on peut se ramener à 0/1/blanc (binaire) ou I/blanc (unaire)
- Tête de lecture : placée sur un case, peut la lire et y écrire
- Programme :
  1. Liste d’états : I, II, III, ...
  2. Chaque état décrit une action :
     - Selon ce que contient la case
     - Modifier la case (gomme et crayon)
     - Changer d’état ou non ou s’arrêter (HALT)
     - Déplacer ou non d’une case à gauche ou à droite
- Example en live sur : https://turingmachine.io/

Résultats mathématiques

Parmi les théorèmes prouvés

- Tout ce qui est intuitivement calculable est calculable avec une machine de Turing
- Il existe une machine de Turing universelle
  - Un programme pour une machine de Turing peut être représenté par un nombre. Il suffit d'utiliser des convention pour décrire états, actions, etc.
  - Il existe (quel que soit le codage) une Machine de Turing dite universelle qui prend en entrée sur le ruban :
    - Le codage d'une machine de Turing
    - L'entrée supposée et qui va l'exécuter (la "simuler") sur l'entrée
- Il n'existe pas de machine de Turing prenant en entrée une machine de Turing et son entrée et qui prédit si cette dernière s'arrêtera ou pas (indécidabilité du problème de l'arrêt)

La bonne et la mauvaise nouvelle

La mauvaise

Si on peut fabriquer un ordinateur, on ne pourra jamais en faire un autre qui fait quelque chose de plus que les autres...

- On pourra le faire plus vite
- On pourra le faire moins cher
- On pourra le faire avec plus de couleurs
- mais rien de plus (et rien de moins)

La bonne

Universalité

En programmation ça s'exprime par la propriété d'être *Turing complete* pour un langage : tous les langages de programmation permettent de faire strictement la même chose (il suffit de pouvoir y programmer une machine de Turing universelle)

- Pascal, C, Python, JS, ... sont Turing complete
- SQL (de base, standard ISO) ne l'est pas
- CSS (HTML 5) est Turing complete !!

Sortira-t-on un jour de cette limitation/universalisme ?

Un autre modèle de computabilité ?

- On entend parler d'ordinateurs quantiques depuis longtemps
- C'est un sujet controversé et si c'est possible on perdra sans doute l'universalité (comme les calculateurs analogiques d'autrefois)
- Des entreprises communiquent beaucoup (IBM, Google, Microsoft, ...) mais présentent peu de résultats...
- On parle aussi de machines non exactes (inspirées du cerveau biologique)

Ordinateur et électronique

Architecture physique

- L'architecture typique d'un ordinateur, dite de Von Neumann est constituée de composants distincts :
  - L'unité de calcul arithmétique et logique (CPU)
  - La mémoire de travail qui contient programmes et données
  - Un bus de communication entre les deux
  - Tout le reste : des périphériques
- Circuits TTL et des mémoires à tores de ferrite (jusqu'aux années 60/70)
- Une révolution : la création des transistors
- Circuits intégrés
- L'évolution générale : plus petits, moins consommateurs d'énergie, moins chers, plus rapides
- https://www.youtube.com/watch?v=HdcLRMv3D3g

Systèmes d'exploitation

Définition

Un logiciel qui présente à l'utilisateur et au dévoloppeur une machine plus simple plus générique que la machine physique.

- Au lieu de lire bit par bit, octet par octet une bande, une carte, un disque on adrese des fichiers par leur nom, leur chemin d'accès
- Des périphériques différents sont présentés de façon homogène : disque dur magnétique, disque SSD, disquette ; port série, carte son, port parallèle, ...

Cf. Andrew Tanenbaum : Principe et Implémentation des Systèmes d'Exploitation (version plus récente : Modern Operating Systems)

Avantage/Inconvénients

- Simplifient la programmation énormement et permet d'envisager l'interaction entre programmes
- Consommation de ressources (temps, mémoire, disque)

Histoire des systèmes d'exploitation

Systèmes historiques (années soixantes)

- CTSS (Compatible Time Sharing System), Tenex
- Multics : projet ambitieux impliquant les grands industriels de l'époque (IBM, Honeywell, AT&T, General Electric, ...)
  - Très sécurisé
  - Multi-utilisateurs
  - Multitâche
- Le développement de Multics s'enlise
  - Le language PL/I supposé être utilisé prend du retard
  - Il est trop ambiteux, trop complexe
  - Son développement est organisé de façon bureaucratique, par des sociétés concurrentes

D'UNICS à UNIX

AT&T (American Telegraph & Telecom), Bell Laboratories

- Ken Thompson récupère un PDP-7 de DEC qui n'est pas utilisé
- En quelque jours il développe en langage machine un début de système d'exploitation
  - Multitâche
  - Système de fichier
  - Shell (ligne de commande)
  - Il l'appelle UNICS par dérision vs. MULTICS
  - Principe de conception KISS (Keep it Simple Stupid)
- Il éveille l'intérêt des autres membres de l'équipe (Kernighan, Ritchie, ...)

Un système portable, puissant et simple

UNIX

- L'équipe réécrit UNICS le renomme UNIX
- Sur un mini-ordinateur, le DEC PDP-11
- Création du langage C (dérivé du langage B, qui dérive de BCPL)
- Réécriture en C du noyau, du shell, des utilitaires
- Multitâche, multi-utilisateur, sécurité sur les ressources (propriétaire, droits d'accès)
- Système de fichier homogène (données, programmes, périphériques)

Les seventies

AT&T utilise le système en interne

- Pour la rédaction de brevets
- Pour le développement de logiciel
- AT&T n'a pas le droit de le vendre (loi anti-trust)
- AT&T fournit le code d'UNIX pour une somme symbolique à qui le demande
- Un article est publié dans la presse scientifique : UNIX Time-Sharing System. https://www.bell-labs.com/usr/dmr/www/retro.pdf

UNIX est porté sur d'autres architectures

- Interdata pour commencer, plus tard VAX
- Le compilateur C est portable

UNIX sort du laboratoire

Des Universités se procurent rapidement UNIX

- University of California, Berkeley en particulier
- Cambridge au Royaume Uni, Sydney en Australie
- Elles ajoutent des fonctionnalités, des utilitaires
- Elle portent UNIX sur d'autres architectures

Les années quatre-vingts

Le monopole d'AT&T saute...

- AT&T peut vendre UNIX dans l'industrie
- Surtout à partir d'UNIX System V Release III (ex. Microsoft Xenix)
- Berkeley continue à proposer UNIX (BSD) courtoisement
- Procès intenté par AT&T...

UNIX se répand dans l'industrie

- Un nouveau de machine : les stations de travail
- Sun Microsystem et autres...
- Ils adoptent naturellement UNIX
- Certains reprennent BSD, d'autres achètent une licence AT&T

Fondamentaux d'UNIX

Unix is user-friendly — it's just choosy about who its friends are.
(Anonyme)

Pour les utilisateurs et les développeurs

- Nombreux utilitaires qui font une chose et le font bien
- Shell qui permet de les faire communiquer entre eux (possible parce que on a le multitâche et des i/o simples)
- Système de fichier hiérarchique et simple qui contient données mais aussi des fichier "spéciaux" qui correspondent à des périphériques

Illustration

Documentaire de 1982, AT&T Archives : The UNIX Operating System
https://www.youtube.com/watch?v=tc4ROCJYbm0

GNU ?

GNU : GNU's Not UNIX

- Au début des 80s, Richard Stallman, ingénieur au MIT, tombe sur un bug dans un pilote d'imprimante
- Il cherche les sources pour corriger... introuvables
- Un collègue les lui fournit mais il prévient : interdit de distribuer les sources ou même le correctif
- Stallman est furieux. Il démissionne, il lance le projet GNU
- Écrire un OS complet + applications librement distribuable
  0. Droit d'usage sans licence pour toute domaine
  1. Droit d'étudier et de modifier le programme, ceci implique l'accès au code source
  2. Droit de distribuer le programme
  3. Droit de distribuer des versions modifiées du programme
- 1991 : Il est fonctionnel
  - Compilateur (GCC), Bash, commandes UNIX usuelles, ...
  - Manque le noyau, Hurd, il prend du retard
- En 91 : la justice tranche en faveur de Berkeley contre AT&T

Où en est-on en 1990 ?

État des lieux matériel...

- Stations de travail assez chères (mais répandues dans l'industrie et les Universités) : Sun Sparc, HP, IBM RS, Silicon Graphics, ... toutes sous UNIX
- IBM PC et compatibles, processeur 386 (32 bits, MMU, superviseur)
- Mainframes (S/390) et Minis (AS/400) IBM (banques, assurances)
- Apple Macintosh Classic (Motorola 68000)
- Machines "hobbyists" : 8 bits (Commodore 64, Amstrad, MSX, Sinclair, ...) (Z80, 6502), en fin de vie, 32 bits (Amiga, Atari ST) (Motorola 68000)

Où en est-on en 1990 ?

État des lieux logiciels

- Systèmes propriétaires IBM sur mainframes et minis
- DOS/Windows 3.x sur IBM PC, IBM tente de pousser OS/2
- Quelques UNIX propriétaires sur IBM PC, chers et peu commodes
- MacOS Classic (Mac OS v1 à v9)
- GNU fonctionne mais comme environnement utilisateur complémentaire sur les autres UNIX, pas encore de noyau (Hurd…)

Pendant ce temps (1991) à Helsinky en Finlande

Un étudiant en L3 informatique, Linus Torvalds, aquiert un PC 386, et il veut un vrai OS dessus...

Mais...

- Il découvre Minix fournit avec l'ouvrage de Andrew Tanenbaum, c'est un quasi-UNIX mais... incomplet, pas beaucoup de pilotes et il est propriétaire (il faut avoir le livre pour l'utiliser, il est interdit de distribuer des versions modifiés)
- Il décide d'écrire son propre noyau en utilisant Minix + GNU comme environnement de dev, au bout d'un an plus besoin de Minix et Linus le publie sur Internet et choisi la licence GPL (GNU)

Collaboration et Internet

Software is like sex : it's better when it's free. — Linus Torvalds

De Freax à Linux

- Il se répand très vite (et se développe avec beaucoup de contributeurs) dans le monde universitaire
- GNU/Linux : un environnement complet, libre, autonome
- The Story of Linux (2011) : https://www.youtube.com/watch?v=5ocq6_3-nEw
- How Linux is Built : https://www.youtube.com/watch?v=yVpbFMhOAwE
- Nom de code Linux (2002, 1h) : https://www.youtube.com/watch?v=79_IMeks4wY

Étapes de l'industrialisation de Linux (199x - 1999 - 2020)

Distributions de GNU/Linux

- Pionners : Yggdrasil, Slackware, ...
- Puis Debian GNU/Linux, Red Hat
- Mandrake/Mandriva (FR), SuSE (DE)
- Ubuntu (variante de Debian), CentOS (identique à Red Hat), Fedora
- Aujourd'hui, 4 familles (les deux 1ères représentent 90% du parc)
  - Famille Debian (Debian, Ubuntu, Mint, etc.)
  - Famille RedHat (RHEL, CentOS, Fedora)
  - Plus marginal SuSE/OpenSuSE
  - Gentoo, ArchLinux, Alpine, ...
- Beaucoup de choses dans l'informatique embarquée et temps réel (Yocto, Buildroot, Android, etc.)

Pendant ce temps chez Apple...

Fin des années 80

- Apple a des gros problèmes financiers, Steve Jobs fait appel à l'ancien directeur de Pepsi Cola pour relancer l'entreprise
- Celui ci expulse rapidement Steve Jobs de l'entreprise
- Steve Jobs et une partie de l'équipe d'Apple créent NeXT
  - Conçoit une station de travail (NeXTCube, NeXTStation)
  - Adaptent un UNIX BSD comme environnement logiciel de base
  - Développent un environnement graphique spécifique (les autres UNIX ont généralement X11/Motif/CDE)
  - Développent un environnement de développement logiciel très en avance sur son temps, basé sur GNU GCC
- "Mac de luxe", destiné à un public étudiant

Apple a un problème...

Il faut un nouvel OS

- Mac OS Classic est plus tenable (faux multitâche, vieillissant)
- Plusieurs tentatives de développer un nouvel OS en interne ou avec l'aide d'IBM
- Envisage d'acquérir un OS existant, deux candidats :
  - BeOS (créé l'ancien directeur d'Apple France, Jean-Louis Gassée)
  - NeXTStep (de Steve Jobs)
- NeXTStep est choisi, devient Mac OS X (plus tard iOS)
- Apple et NeXT fusionnent, Steve Jobs en reprend la direction

Les autres UNIX...

Très synthétiquement...

- L'histoire d'UNIX est foisonnante : https://www.levenez.com/
- GNU/Linux a finit par dominer le marché, il reste en gros :
- Les BSD libres (FreeBSD, OpenBSD, NetBSD)
- macOS et iOS
- IBM AIX, HP/UX, Oracle (ex-Sun, sigh) Solaris, ...
- Linux est fournit par Microsoft dans Azure, mais aussi dans Windows 10 : Windows Subsytem for Linux

Un autre système : VMS

DEC a conçu un système issu du PDP-11 dans les 70s

- VAX (sur lequel tourne aussi UNIX BSD)
- Un OS : VMS
- Architecture pas très éloignée d'UNIX
- Pas du tout les mêmes commandes
- Très robuste, très sécurisé
- Assez répandu dans l'industrie jusqu'aux années 90
- Encore maintenu et utilisé, porté sur Intel64
- DEC a été racheté par Compaq, Compaq a fusionné avec HP

Et Microsoft ?

Dans les années 70

- Créé par William Gates et Paul Allen
- Fournissent un interpréteur du langage BASIC pour les premiers micro-ordinateurs à destination des hobbyists
- Cf. la lettre ouverte écrite par Bill Gates contre le piratage

IBM contacte Microsoft

IBM se lance dans la micro-informatique

- En 1980, soucieux du succès de l'Apple II (Jobs, Wozniak), IBM décide de lancer un micro-ordinateur
- IBM décide de sous-traiter le développement du logiel (OS)
- IBM contacte l'éditeur Digital Research pour obtenir CP/M (pour Z80, proche du Intel 8086) qui ne donne pas suite
- IBM contacte Microsoft qui n'a rien en magasin mais accepte quand même
- Microsoft rachète Quick and Dirty OS, semblable à CP/M écrit par un développeur (Tim Paterson) en lui cachant que c'est pour IBM
- En le modifiant un peu, Microsoft le propose à IBM : MS-DOS

Les années PC

Dans les petites entreprises le PC décolle

MS-DOS évolue

- 2.0 : apparition des répertoires et gestion des disques durs
- 3.x, 4.x (complètement bugué), 5.x (corrige les bugs), 6.x
- 7.x (partie de Windows 95)
- C'est un OS très primitif : gestion des fichiers et de la mémoire, guère plus
- Limitation à 640Kio max (Bill Gates : "Personne n'aura jamais besoin de plus")

Le matériel aussi évolue, de 16 à 32 bits

Le PC évolue : processeur Intel 286 et surtout 386

- IBM lance le projet OS/2 : moderne, multitâche, multi-utilisateurs et graphique
- Microsoft participe au développement d'OS/2
- En parallèle Microsoft développe une surcouche graphique pour MS-DOS : Windows
- Présenté à IBM comme transition vers OS/2
- Windows 1.0 échec, Windows 2.0 échec
- Windows 3.0/3.1/3.11 : succès énorme, ergonomie proche du Mac
- Multitâche "coopératif" : une application peut bloquer/planter le système

Et la concurrence ?

Concurrent de DOS/Windows

- DR-DOS (Digital Research)
- GEM (Digital Research), interface graphique
- Geoworks Ensemble, interface graphique

Alternatives

- Microsoft Xenix, SCO UNIX, ...
- Novell Netware (serveurs réseau)

De Windows 3.1 à Windows NT

Le torchon brûle

- Microsoft a planté un couteau dans le dos d'IBM
- Windows est présenté comme un OS à part entière (ce qu'il n'est pas)
- IBM pousse OS/2 (compatiblité Windows), semi-échec commercial

Microsoft souhaite faire de Windows un "vrai" OS

- Transition (toujours DOS) : Windows 95, 98, Me
- Projet Windows NT *New Technology* (plus de DOS)
  - Microsoft recrute le chef chez DEC du développement de VMS
  - Reste compatible avec les Windows basés sur DOS
- NT 3.51 se positionne comme concurrent d'UNIX
- Windows 2000 connaît un certain succès côté serveur
- Windows XP est la première version grand public basée sur NT

Microsoft Windows au XXIème siècle

Those who don't understand Unix are condemned to reinvent it, poorly.
– Henry Spencer

*"Linux is a **cancer** that attaches itself in an intellectual property sense to everything it touches."*
*"There's no company called Linux, there's barely a Linux road map. Yet Linux sort of springs organically from the earth. And it had, you know, the characteristics of **communism** that people love so very, very much about it. That is, it's free."*
– Steve Ballmer (CEO Microsoft)

"Microsoft loves Linux."
– Satya Nadella (CEO Microsoft)

Microsoft au XXIème siècle

Dominant sur le poste de travail, présent sur les serveurs

- Après Windows XP...
- Windows Vista est une catastrophe
- Windows Seven le remplace avantageusement
- Depuis Windows 8, Windows 10
- Diverses versions de Windows Server 20xx
- Pourquoi pas de Windows 9 ? Pour éviter qu'une appli ancienne pense qu'elle tourne sous 95 ou 98 et se plante. Malin.

Réseaux et Internet

Dès les années 60, et au delà

- Des protocoles réseaux permettent à des ordinateurs de communiquer entre eux
- Mainframe IBM (SNA), Novell IPX, DecNET, ...
- LAN : Appletalk, NETBEUI (IBM/Microsoft)
- Au niveau hardware : Token Ring, Ethernet (LAN)
- WAN : réseaux téléphonique (Modems), RNIS et liaisons spécialisées

Internet Protocol

1974 : Internet Protocol (TCP/IP)

- Développé sur les bases d'Arpanet (projet du département de la défense USA), réseau à grande échelle, résistant aux pannes
- Les universitaires qui y ont partipé se lancent dans la création des protocoles d'Internet
- Interconnexion de réseaux entre eux, d'ordinateurs de toutes architectures et exécutant divers OS
- Les normes sont construites de façon collaborative
- Normes officielles : Requests for Comments (RFC)
- Premières implémentations : BSD UNIX, VMS, autres UNIX
- Principe : transport de paquets sans chemin prédéfini
- Principe : le réseau est bête, l'intelligence est aux extrémité

La guerre des réseaux

Dans les années 80 les opérateurs télécoms s'y mettent

- Ils veulent imposer le modèle OSI
- Plus complexe que TCP/IP, basé sur des circuits déterminés en début de dialogue, facturable à la seconde
- Heureusement TCP/IP va se généraliser (France, RENATER 1984) à l'Université d'abord puis au delà

Internet pour le grand public et les entreprises

- Milieu des années 90
- Microsoft a boycotté Internet et proposé MSN
- Six mois plus tard, ils ne juraient plus que par Internet et la première version de IE est sorti
- Les premières piles logicielles TCP/IP de Windows sont reprises de BSD UNIX

Au niveau applicatif pour les utilisateurs

Premières applications

- Mail (SMTP, POP, IMAP)
- Usenet (forums de discussion, NNTP)
- telnet pour le login à distance
- IRC pour les conversations
- X11 (X-Window System) pour les applis graphiques à distance
- Gopher (l'ancêtre du Web)
- FTP pour l'échange de fichiers

Évolution des protocoles

- IPv4, adresses sur 32 bits : `66.228.47.22` (4 294 967 296 addresses possibles, épuisé en 2019)
- IPv6 (1998), adresses sur 128 bits : `2600:3c03::f03c:91ff:fe82:68b2` (340 282 366 920 938 463 463 374 607 431 768 211 456 adresses possibles)

World Wide Web

Le Web

- En 1991, à Genève, au CERN, Tim Berners-Lee et Robert Cailliau inventent le Web
- Protocole de requètage et de fourniture de documents : HTTP
- Format de document avec liens hypertexte : HTML
- HTTP s'inspire de SMTP, HTML s'inspire de SGML (deux technos des 70s)
- But : faciliter la communication de publications scientifiques
- Développé sur UNIX NeXTStep (serveur et client)
- Protocole et code source publiquement disponible

L'histoire du Web

Serveurs et navigateurs

- CERN HTTPD (serveur) devient NCSA HTTPD qui s'accompagne d'un jeu de rustines (patch) qui l'accompagne : "A patchy server"
- D'où le nom du serveur Apache (A-pa-chee)
- Navigateurs : Mosaic, Netscape, MS Internet Explorer, Mozilla Firefox, Google Chrome, Microsoft Edge

Modèle applicatif

- Le modèle d'origine est documentaire (on demande et on obtient un document)
- Le modèle qui s'est imposé plus tard est applicatif (site de commerce par exemple)

Applications Web

Dynamisme côté serveur

- Au lieu de lire un fichier pour envoyer du contenu au navigateur, quand une requête HTTP (basée une url http ://...) a lieu :
  - Un programme est exécuté sur le serveur
  - La sortie du programme est envoyée au client
- Ce programme va récupérer des informations qui peuvent être extraite de l'url (`http ://serveur.example.com/... show.php ?productID=123&customer=12`) ou bien dans le flux d'octet transmis (GET/POST)
- Typiquement ce programme interroge et met à jour une base de données

Améliorer l'expérience utilisateur

Dynamiste côté client

- Netscape (l'éditeur du navigateur *Navigator* qui longtemps a dominé le marché) a créé le langage JavaScript (rien à voir avec Java !)
- S'exécute dans le navigateur
  - Accès aux actions de l'utilisateur
  - Accès à la page HTML affichée (peut la modifier)
  - Peut déclencher des requêtes HTTP
- HTML a beaucoup évolué pour arriver à HTML5/CSS

Modèle MVC : Modèle/Vue/Contrôleur

Séparation des préoccupations

- Ce modèle était déjà bien connu des développeurs d'applications avec une interface graphique (ou même textuelle)
- M : Modèle (métier) : étant donné un productID quels sont les caractéristiques de ce produit (extrait de la base de données)
- V : Vue, déclenchement d'une fonctionnalité du modèle (récupérer les infos d'un produit, modifier une commande), lié à un modèle d'url, par exemple
  `http ://.../prodInfo.php?productID=123`
- C : Contrôleur, lien entre les requêtes HTTP et les diverses vues
- On y trouve souvent un langage de macros pour écrire des gabarits HTML

Langages et frameworks côté serveur

Rationnaliser le développement côté serveur

- Au début le langage de choix : Perl (UNIX/Linux), ASP (MS Windows)
- Ensuite est arrivé PHP (Personal Home Page), il est plus facile d'accès que Perl pour les non-programmeurs
- PHP est passé du statut de gadget à langage complet pour le développement Web, des frameworks sont apparus (Simphony, Zend, Laravel)
- En pratique on rencontre bien d'autres langages : Java, C#, LISP, Objective-C, ..., Python (frameworks Web Django, Flask, ...) et même JavaScript (node.js)
- https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/

Côté client...

JavaScript pose quelques soucis...

- JavaScript a été conçu en 15mn dans un restaurant
- Ça se sent... https://www.destroyallsoftware.com/talks/wat
- Il y a de plus en plus de code JS dans les pages (AJAX)
- Des frameworks ont été développés pour contourner les défauts du langage : jQuery, Angular, react, ...
- Un "sur-langage" plus strict a été créé par Microsoft : TypeScript
- L'avenir : WebAssembly (un langage de bas niveau qui est la cible de compilateurs d'autres langages : JS, Python, ASP.NET, TypeScript)

Les APIs REST

- API : Application Program Interface
- REST : REpresentational State Transfer

Un concept pas nouveau : Appels de fonction à distance (RPC)

- Nouveau : utiliser le Web
- Une requête HTTP demande une action (lire/écrire/modifier/supprimer)
- Le serveur traite la requête et renvoie des infos dans un format structuré (pas de HTML) XML ou (surtout) JSON
- Un example : Un call sur `https ://.../productInfo ?id=42` va renvoyer `{ 'id':42, 'name':'Cup', 'price':12, 'disp':True }` (format JSON)
- Le client est typiquement une application, un script, ... pas nécessairement un navigateur Web

Le Cloud

En général le terme qualifie des choses assez diverses

- SaaS : Software As a Service. Ex : Gmail, Saleforce, Google Doc, Office 365, ...
- PaaS : Platform As a Service. Sur un serveur mutualisé vous avez accès à une zone permettant de placer des applications Web dans un langage donné (PHP, Perl, Python, ...), et à une base de données (Hébergement Web)
- IaaS : Infrastructure as A Service. Se substitue à l'hébergement de serveurs physiques en utilisant des machines virtuelles
- C'est IaaS qui est important ici

Virtualisation de systèmes

Pas une idée nouvelle

- IBM dans années 60 (S/390, puis zSerie)
- L'OS ne tourne plus directement sur le matériel mais au dessus d'un hyperviseur, très léger, qui fournit à l'OS un environnement identique à la machine physique, et ceci en plusieurs instances parallèles
- Le matériel (CPU) est conçu pour limiter l'impact sur les performances
- On fait tourner ainsi plusieurs instances de l'OS, ou même plusieurs OS différents

Sur PC

Arrivée de la virtualisation dans le monde PC

- VMWare est le pionner de l'introduction de cette techno sur processeur Intel
- Intel et AMD ont ajouté des fonctionnalités dans leur processeurs pour aider à la virtualisation (comme IBM en 196x)
- D'autres hyperviseurs sont apparus : Xen (Linux), KVM (Linux), Hyper-V (MS Windows), VirtualBox

Le Cloud

Cloud IaaS et virtualisation

- Propose la création, exécution de machine virtuelles à la demande à des clients divers
- Amazon a proposé ce type de service en premier dans leurs data centers
- Services fournis :
  - Un VPC (un réseau Ethernet virtuel)
  - Des images disques
  - Des machines virtuelles dans ces VPCs
  - Contrôle par une console Web et une API REST
- D'autres services peuvent être fournis en plus (du style PaaS), bases de données, frontaux Web, stockage, ordonnanceurs de conteneurs, ...
- Fournisseurs actuels : Google Compute Engine, Microsoft Azure, Digital Ocean, Outscale (API compatible AWS), ...
- Plate forme OpenStack (libre)

Cloud et DevOps

C'est quoi le DevOps ?

- C'est l'idée d'avoir en développement des environnements aussi semblables que possible à la production
- Ces environnements sont déployés automatiquement
- Les tests sont exécutés automatiquement
- Les déploiements en production aussi

Le Cloud dans tout ça ?

- Le Cloud, permettant d'automatiser les déploiements, rend cette démarche beaucoup plus effective
- Les outils qui le permettent
  - Suivi de version (GIT) : Infrastructure as Code
  - Les langages de script (Bash, Python, ...)
  - Les outils de déploiement d'infra (Terraform, ...)
  - Les outils de déploiement de configuration (ansible, puppet, SALT, ...)

Conteneurs

Quasi-VM légères

- Un conteneur est une façon de virtualiser un système de façon plus légère
- Des environnements autonomes pour un ensemble de processus (tâches) au dessus d'un système exploitation
  - Une partie du système de fichier (chroot)
  - Un sous-ensemble des processus
  - Un espace de nommage des utilisateurs, groupes, etc
  - Des canaux de communication réseau
- Le même effet qu'une VM, mais tous les conteneurs étant constitués de processus sous le contrôle d'un noyau unique, OS de l'hôte et OS des systèmes invités ne peuvent être différents
- Linux (cgroups), BSD, Solaris (slices)
- Aucune accélération matérielle nécessaire
- Le niveau de cloisonnement est inférieur

Conteneurs, devops et Cloud

VMs et conteneurs

- On fait tourner volontier des conteneurs dans des VMs
- Ou des machines physiques (si l'OS de production est l'OS du poste du développeur)

Technologies

- LXC (Linux Containers)
- Docker
- Concurrents de Docker : rkt *("rocket")*, LXD, runC, CRI
- Orchestrateurs : Kubernetes, Rancher, ...

Langages de programmation

La CPU exécute quoi ?

- Des instructions de bas niveau (opérations) représentées par des nombres binaire en RAM/ROM, accompagnés d'opérandes
- La mémoire contient aussi des données : nombres entiers, décimaux, texte, symboles, etc...
- On peut programmer avec des cables, des interrupteurs, un clavier numérique
- On peut lire un programme à partir d'une carte perforée, d'une bande perforée, magnétique, disquette, disque, ...
- On peut représenter les opérations par des mnémoniques

```
01100011 00000001           # INC A
00000110 00000101           # ADD A, B
00011000 00000100 00001011 # MOV B, 11
```

- Un programme d'assemblage transforme mnémoniques et données en représentations et opcodes/opérandes binaires

Programmer en assembleur...

Premiers langages compilés

- L'assembleur est fastidieux et non-portable entre CPU
- En 1951, Grace Hopper écrit un compilateur d'un langage de plus haut niveau (A-0 System) puis en 1959 pour le langage COBOL
- Il convertit quelque chose de plus lisible en assembleur puis en binaire

Langages interprétés

- Ils sont exécutés sous le contrôle d'un interpréteur
- En pratique souvent compilés sous une forme intermédiaire (byte code) qui est exécuté par une machine virtuelle (pas au sens de la virtualisation)

Premiers langages

Impératifs

- COBOL
- FORTRAN, ALGOL

Fonctionnels

- LISP

Leur descendance

- ALGOL : C, Pascal, Modula, Ada, BASIC, ...
- LISP : Common LISP, Scheme, CaML, Haskell

Leur influence

- Python, JavaScript, ... tout ou presque

Programmation orienté objet

Premiers

- Smalltalk, Common LISP

Par la suite

- C++
- Java, C#

Programmation logique

- PROLOG

Catalogue de langages de programmation

- http://www.rosettacode.org/

L'état de l'art

Programmation système, jeux, calcul

- C, C++, Lua
- Golang, Rust

Traitement de données, Web, ...

- Perl, Python, Ruby, PHP, JavaScript
- Java, C#

Calcul et statistiques

- Scala, Julia
- R, Python

Statistiques sur les langages

- https://www.tiobe.com/tiobe-index/
- https://stackoverflow.blog/2017/09/06/incredible-growth-python/

Bases de données

Un système de fichier est déjà une base de données...

Bases de données relationnelles

- Basé sur une théorie mathématique : l'algèbre relationnelle créée par Edgar Frank Codd chez IBM
- Langage de requête SQL, normalisé, lisible
- SGBDR propriétaires : IBM DB2, Oracle, Microsoft SQL Server ; SGBDR libres : MySQL/MariaDB, PostgreSQL (mon préféré), SQLite
- On définit un schéma : tables, champs, types, relations

```
SELECT produits.nom, fabriquants.nom, prix, stock
  FROM produits, fabricants
 WHERE fabricant.id = 42
   AND produits.fabricant = fabricants.id
   AND prix <= 50
   AND (nom LIKE '%jeu%' OR nom LIKE '%game%' )
```

NoSQL

Au début “No SQL” puis, plus modestement, “Not only SQL”

- Structurer les données au moment de la requête, pas à l'avance
- Au lieu d'avoir des tables avec des champs prédéfinis on a des dictionnaires clés/valeurs et des textes indexés
- Pur NoSQL : MongoDB, Elastic Search/Lucène, Cassandra
- Mixtes : PostgreSQL surtout

Bonne référence : https://use-the-index-luke.com/

D'autres modèles existent

- Graph database (réseaux)
- Bases géospatiales (PostgreSQL + PostGIS par example)
- Textuel, ...

Cryptographie

C'est de là que tout est parti...

- C'est pour casser les codes de l'armée nazie que l'informatique basée sur l'électronique est née (Alan Turing)
- Chiffrer et authentifier est très utile en informatique... Vous ne voudriez pas que votre numéro de carte bancaire soit déchiffré par quelqu'un d'autre que le commerçant.

De l'antiquité jusqu'à récemment...

- Le chiffrement est symétrique : on chiffre et on déchiffre avec la même clé
- Ex : Jules César (permutation circulaire de l'alphabet), Vigenère, Enigma, etc... puis DES, 3DES, Blowfish, ...
- Très rapide, plutôt sûr, inconvénient : il faut convenir à l'avance d'un secret partagé (la clef)

Le chiffrement asymétrique

En 1977

- Trois mathématiciens inventent le premier algorithme de chiffrement asymétrique
- Ronald Rivest, Adi Shamir et Leonard Adleman : RSA
- Vous créez une paire de clefs : une dite publique, l'autre dite privée
- La clef publique peut être publiée...
- La clef privée doit rester ABSOLUMENT privée
- Si on utilise une des deux clefs pour chiffrer, l'autre clef (et elle seule) permet de déchiffrer

Le chiffrement asymétrique en pratique

Que peut-on faire ?

- Si vous chiffrez avec votre clef privée, qui peut déchiffrer ?
  - Il faut la clef publique, donc tout le monde en principe
  - Rien de caché... Mais ça garantit que le message vient de vous
- Tout le monde peut chiffrer avec votre clef publique, qui peut déchiffrer ? Seulement vous !
- On peut chiffrer avec la clef publique du destinataire et aussi sa clef privée à soi : authentifie l'expéditeur et garantit que seul le destinataire peut lire le message

En pratique

- Pour authentifier on chiffre avec une clef privée un condensé du message (MD5, puis plutôt SHA)
- On utilise la clef publique pour échanger des clefs symétriques, dites clefs de session car le chiffrement symétrique est plus rapide

Les certificats

Un petit souci

- Comment s'assurer de l'authenticité d'une clef publique sans l'avoir reçue en main propre ?
- On a une liste pré-installée de clefs publiques de tiers de confiances (certificats racines), États, organisations diverses (voir la configuration de votre navigateur Web)
- L'autre participant à un échange va vous envoyer sa clef publique signée i.e. chiffrée par une des clefs privées correspondant aux certificats racines
- C'est comme ça que fonctionne SSL (par exemple HTTPS)
- Si vous voulez profiter de ceci sur le Web public, sans payer de tiers de confiance, regardez le service "Let's Encrypt"

Pour finir...

Une drôle d'histoire...

- Des machines qui, sans information, (logiciel) ne servent à rien
- Des machines universelles, qui ne sauront jamais faire plus de choses, qui ne progresseront donc jamais, sinon...
  - En étant plus rapides, plus petites, moins chères
  - En s'intégrant à des contextes nouveaux : entreprises, domiciles, avions, trains, voitures, télévisions, téléphones, ...

Un univers où, au fond, rien de fondamental ne change...

- Mais où les usages évoluent sans cesse
- Un exemple : Xerox a construit un prototype de tablette dès le début des années 80 : graphique, tactile, multi-média, communication sans fil, et sous UNIX !
- Il a fallu attendre trente ans pour que ce soit pertinent : réseau GSM, baisse du prix, du poids, ... et sous UNIX (Android/Linux, iOS) !

Recherches personnelles

À vous !

Choisissez deux thèmes parmi les trois présentés là : https://framagit.org/jpython/culthistinfo/-/tree/master/Docs/Atelier

- Un autre OS aux laboratoires Bell : Plan 9
- UNIX (vraiment ?) contre Linux au tribunal
- GIT

Rédigez une synthèse à partir de sources trouvées sur l'Internet

- Wikipedia (surtout en anglais) est un bon point de départ
- Ne vous contentez pas pour autant de Wikipedia
- Une page maximum par sujet
- Livrable au format PDF, MarkDown, texte simple, ODT

Un dernier mot

Vous aurez l'occasion de participer à des projets divers

- En entreprise, pour d'autres organisations, pour des États
- Pour vous même, et vous pourrez publier des logiciels libres ou y contribuer

Une question que vous pourrez vous poser...

- L'informatique intervient dans les aspects les plus intimes et les plus importants de la vie
- L'informatique joue un rôle sans doute central dans l'histoire future de l'humanité
- Échanger avec nos amis, avec des inconnus
- Accéder et contribuer à la culture, à la politique
- Interagir avec l'administration, acheter des biens et des services

Quand vous interviendrez dans ces projets pour les mener à bien, aurez-vous rendu le monde meilleur ? moins bon ? inchangé ?

Références

Computing : The Human Experience : https://computingthehumanexperience.com/

Modern Operating Systems, Andrew Tanenbaum, 4e ed. 2014, Prentice Hall

Unix : A History and a Memoir, Brian W. Kernighan, 2018, Princeton University Press

Éric Lévénez's site : https://www.levenez.com/

The Art of Unix Programming, Eric Steven Raymond, 2003 : https://www.arp242.net/the-art-of-unix-programming/

The UNIX-HATERS Handbook : https://web.mit.edu/~simsong/www/ugh.pdf

The Design of the UNIX Operating System, Maurice J. Bach, 1986 Prentice/Hall International

Introduction à l'informatique, Isabelle Tellier : https://www.univ-orleans.fr/lifo/Members/Isabelle.Tellier/poly_intro_info/index.html

A Commentary on the Sixth Edition UNIX Operating System, John Lions, 1977 : https://warsus.github.io/lions-/

Just for Fun : The Story of an Accidental Revolutionary, Linus Torvalds et David Diamond, 2002 Harper Business

La cathédrale et le bazar, Eric S. Raymond, 1998 : https://archive.framalibre.org/IMG/cathedrale-bazar.pdf

Le manifeste GNU, Richard Stallman, 1985 : https://www.gnu.org/gnu/manifesto.fr.html

Cyberstructure – L'internet, Un Espace Politique, Stéphane Bortzmeyer, 2018 C&f Editions (https://www.bortzmeyer.org/)

Understanding the Digital World, Brian W. Kernighan, 2017 Princeton University Press

OLD-COMPUTERS.COM : https://www.old-computers.com/

The National Museum of Computing, Bletchley Park, Buckinghamshire, UK : https ://bletchleypark.org.uk/

Cryptonomicon, Neal Stephenson, 2001 Le livre de poche

La cité des permutants, Greg Egan, 1999 Le livre de poche

The Art of Computer Programming, Donald Knuth, 1969-2020 Addison-Wesley

Microserfs, Douglas Coupland, 1992 Éditions 10-18

Le hold-up planétaire, Roberto Di Cosmo et Dominique Nora :https ://www.dicosmo.org/HoldUp/HoldUpPlanetaire.pdf

Confession d'un voleur – Internet : La liberté confisquée, Laurent Chemla : http ://www.confessions-voleur.net/confessions/

TODO

- TODO : compléter la partie langages de programmation, ajouter l'histoire des micro-contrôleurs et processeurs (du Z80 jusqu'à RISC V), loi de Moore, ajouter la notion de preuve de programme, calcul numérique et calcul formel
- TODO : les jeux vidéos, la téléphonie mobile, le "big data", l'histoire des interfaces graphique (Xerox Star, ...), de la bureautique, détails sur les licences FLOSS, la distinction oss/free software
- TODO : complexité de programmes, P vs NP, Donald Knuth et algorithmique, LaTeX :-), histoire des échecs, des prévisions fausses
- TODO : IA, OpenData, méthodologies de dev.