L'entorn de treball del científic de dades

Guia de lectures

PID_00270623

Julià Minguillón Alfonso

Temps mínim de dedicació recomanat: 2 hores

Universitat Oberta de Catalunya

Julià Minguillón Alfonso

Doctor enginyer en Informàtica per la Universitat Autònoma de Barcelona (UAB). Professor agregat dels Estudis d'Informàtica, Multimèdia i Telecomunicació de la Universitat Oberta de Catalunya (UOC) en l'àmbit de la ciència de dades. Els seus interessos docents inclouen la programació, la mineria de dades i la visualització, entre d'altres. En recerca, es dedica a analitzar el comportament dels usuaris en entorns virtuals d'aprenentatge i xarxes socials, com ara Wikipedia, amb l'objectiu de millorar els processos de suport a l'aprenentatge i la interacció amb l'entorn.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Julià Minguillón Alfonso (2020)

Primera edició: febrer 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realització editorial: FUOC

Índex

Introducció

Seguint el cicle de vida de les dades, és ben sabut que abans de poder construir i avaluar un model que resolgui un problema en l'àmbit de la ciència de dades, en la majoria de casos cal **recopilar i processar les dades**, les quals poden venir de diferents fonts, en diferents formats, etc. Aquest procés acostuma a ser costós, poc automatitzable i segurament ha de preveure un nombre elevat d'excepcions que cal resoldre després d'una inspecció detallada. Sense que sigui una xifra exacte, es pot considerar que tot el procés de creació d'un model segueix una distribució típica 80-20, en què un 80% del temps es dedica a la preparació de les dades, i el 20% restant a la construcció dels models i a l'avaluació i interpretació dels models construïts. Òbviament, cada projecte o experiment tindrà una configuració diferent, però normalment la preparació de les dades exigeix un procés de vegades molt artesanal que només cal executar un cop i que, una vegada llest, es pot intentar automatitzar per a posteriors ocasions, en què calgui obtenir noves dades seguint el mateix procediment, per exemple, i reproduir totes les passes seguides.

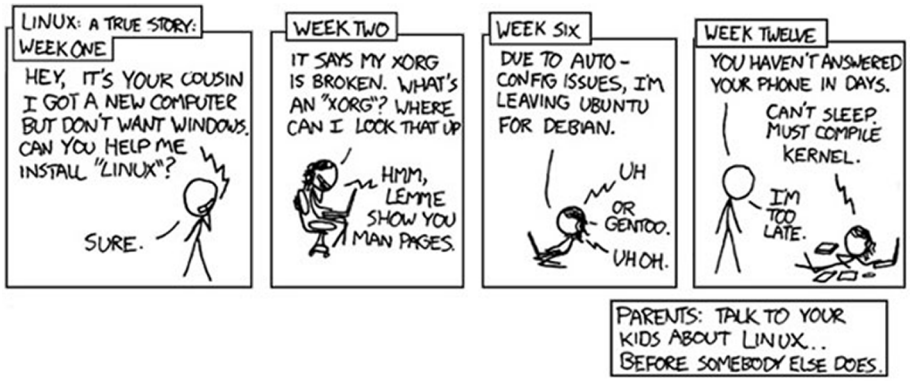
Malauradament (o segurament, per sort) no hi ha una eina general que resolgui tots els problemes en tot moment, de manera que per a cada experiment o projecte que haguem de resoldre ens caldrà utilitzar un conjunt d'eines diferent. Aquest fet, conegut com a *no-free-lunch theorem* en altres àmbits, també es pot aplicar a la ciència de dades. Així, algunes d'aquestes eines necessàries per a resoldre un problema poden venir imposades per la naturalesa del mateix o de les dades, d'altres poden ser part de la infraestructura tecnològica disponible en l'entorn de treball, i d'altres poden ser senzillament les favorites que té cada científic de dades per treballar en el seu dia a dia per a resoldre els problemes típics als quals s'enfronta, com a part del seu entorn de treball. Per tant, un dels primers aspectes a resoldre és disposar d'un **entorn de treball adequat** per a la manipulació de les dades, la creació de models i altres tasques típiques del cicle de vida de les dades. Un cop més, aquest entorn pot venir imposat per la manera com es treballa en una certa empresa, organització, l'Administració, etc., però també es pot veure des de la perspectiva personal del científic de dades que té uns certs recursos (el seu ordinador personal, accés a altres servidors, programari, etc.) que vol utilitzar per a aquesta finalitat.

En aquesta guia ens centrarem en l'entorn de treball necessari per a les fases de captura i, especialment, de preparació de les dades, amb l'objectiu de generar els conjunts de dades que seran utilitzats en la fase d'anàlisi següent. Això determinarà en certa manera els criteris que seguirem a l'hora d'utilitzar unes eines o unes altres i la configuració de l'entorn de treball adequat. Un cop tinguem una màquina llesta per a manipular i analitzar les dades, veurem com usar la línia de comandes, també anomenada *consola*, com a instrument que permet accedir al cor del sistema, i que ens permet realitzar tasques de mante-

niment per a assegurar que l'entorn de treball es manté actualitzat i operatiu, i també del centre d'operacions des del qual durem a terme les tasques que poden ser automatitzades mitjançant les capacitats del llenguatge d'*scripting* que ens ofereix el sistema operatiu.

Així doncs, l'objectiu final d'aquesta guia és proporcionar una visió dels diferents elements que proporciona el sistema operatiu mitjançant l'ús de la línia de comandes, els quals poden ser combinats en petits programes o *scripts* que automatitzen les tasques repetitives que són habituals de qualsevol projecte de ciència de dades. Primer, definirem els components de l'entorn de treball i una proposta general de com es podria configurar. Després, descriurem les comandes típiques que es realitzen des de la consola del sistema operatiu i alguns exemples de problemes que es podrien resoldre d'aquesta manera. Finalment, la guia acaba amb unes lectures històriques recomanades per a entendre els orígens de tots els components que conformen l'univers actual dels entorns esmentats al llarg d'aquest text.

Figura 1. *Linux: a true story*



Font: xkcd.

Objectius

1. Introduir els conceptes bàsics de l'assignatura.

2. Definir els components de l'entorn de treball del científic de dades.

3. Presentar diferents eines per a la manipulació de dades.

4. Introduir la línia de comandes i la programació en *scripting*.

5. Proposar casos d'ús típics de la línia de comandes.

6. Conèixer les diferents lectures clau en aquest àmbit.

1. L'entorn de treball del científic de dades

Actualment, hi ha moltes opcions a l'hora de configurar el que ha de ser un **entorn de treball optimitzat** per a la preparació de les dades usades en la fase d'anàlisi, però de forma simplificada cal començar per decidir els tres nivells típics de qualsevol sistema informàtic: el **maquinari**, el **sistema operatiu** i el **conjunt d'eines o programari**, els quals no són realment independents, sinó que les decisions en un nivell afecten els altres dos, establint forts lligams, com ara que cert programari només s'executa sobre un cert sistema operatiu.

1.1. El maquinari

Com sempre, com més capacitat de còmput i de disc tingui l'ordinador usat per a fer experiments millor, ja que la ciència de dades consisteix precisament a manipular grans volums de dades i construir models complexos amb un cost computacional elevat. Sense entrar a parlar d'infraestructures per a dades massives, és molt fàcil que s'hagin de manipular fitxers amb milions de registres, generant molts fitxers intermedis en el procés, de manera que la capacitat de disc disponible ha de ser avui de l'ordre de centenars de gigabytes (GB) o, fins i tot, de terabytes (TB). Els discs d'estat sòlid (SSD), usats típicament en els ordinadors portàtils, són més eficients, ja que no requereixen parts mòbils, però la seva capacitat és limitada (centenars de GB). Per a grans capacitats d'emmagatzematge, cal usar discs durs convencionals, els quals són més lents i consumeixen més energia però arriben a capacitats de diversos TB. Una altra opció és utilitzar l'emmagatzematge en el núvol, amb una capacitat virtualment il·limitada, però aleshores caldrà tenir en compte la velocitat de transferència dels fitxers, la qual pot ser un coll d'ampolla a l'hora de realitzar els experiments, ja que els fitxers de l'ordre del GB o superior no s'haurien d'anar movent o copiant contínuament.

Una piulada

Una piulada capturada amb totes les seves metadades en format JSON pot ocupar uns 10 KB, de manera que una captura d'un milió de piulades pot ocupar uns 10 GB de disc.

El *dump* de Wikipedia que conté el detall de totes les edicions realitzades[1], sense el contingut de les pàgines ni de la pròpia edició, és un fitxer comprimit d'uns 7 GB per la Wikipedia en castellà, 1,5 GB per la Wikipedia en català i de més de 60 GB per la Wikipedia en anglès.

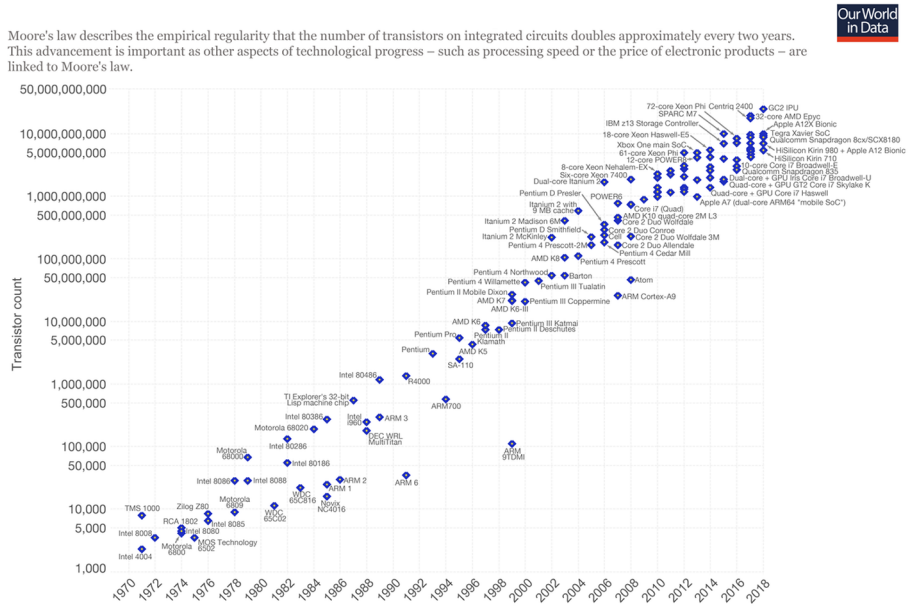
(1)Anomenat `<idioma>wiki-latest-stub-meta-history.xml.gz` on `<idioma>` és «en», «es», «ca», etc.

Pel que fa a la capacitat de còmput, hi ha dos aspectes a tenir en compte. El primer és la **quantitat de memòria RAM disponible**, la qual actualment es mou en l'ordre de GB o desenes de GB. Tot i que parlar de xifres absolutes és complicat, atesa la rapidesa amb què aquestes queden obsoletes, actualment es considera que 8 GB és el mínim recomanable per a poder afrontar experiments de certa mida, tenint en compte que moltes de les eines per a manipular i analitzar les dades es carreguen en memòria abans de començar a realitzar cap

procés. En aquest cas la decisió és senzilla, com més quantitat de RAM disponible millor, i també com més ràpida sigui i més capacitat de transferència tingui millor, sent la coneguda com a DDR SDRAM la més habitual actualment.

Un altre aspecte a tenir en compte és **la potència i velocitat del processador** (la CPU), incloent-hi el nombre de nuclis (també anomenats *cores*) que tingui, els quals es poden usar per a paralel·litzar tasques, quelcom molt habitual en les operacions en què es manipulen dades en format de vectors, taules o tensors, aprofitant al màxim les seves capacitats. També aquí les xifres són ràpidament canviants, d'acord amb la famosa llei de Moore, tal com mostra la figura 2. Actualment, els fabricants principals de CPU com ara AMD o Intel tenen al mercat processadors amb desenes de nuclis, de forma que és possible fer tasques en paral·lel de forma senzilla, sempre per mitjà del programari adequat, òbviament.

Figura 2. Llei de Moore sobre el nombre de transistors en els processadors (1971-2018)



Font: Max Roser. Wikipedia (commons.wikimedia.org/wiki/Transistor_Count).

Finalment, des de fa un cert temps hi ha un nou actor en aquest escenari de còmput, i és el que es coneix com a **unitats de processament gràfic o GPU**. Originalment les GPU estaven dedicades a processar de forma molt eficient les dades necessàries per a la visualització de les imatges en la pantalla del sistema gràfic, però actualment s'utilitzen per a fer càlculs aprofitant la seva arquitectura paral·lela que permet augmentar el rendiment pel tipus de càlculs que són habituals en ciència de dades. De fet, amb l'auge de les xarxes neuronals i el que es coneix com a *deep learning*, l'ús de les GPU s'ha vist reforçat per l'aparició de maquinari dedicat a aquest efecte i també per la disponibilitat de llibreries que aprofiten al màxim les seves capacitats.

1.2. El sistema operatiu

El nivell intermedi que connecta el maquinari amb el programari que el fa servir és el **sistema operatiu**, el qual també es pot considerar programari per si mateix. El sistema operatiu és una capa que aïlla l'usuari de tots els detalls necessaris per a realitzar operacions senzilles, com ara copiar un fitxer, sense importar l'estructura del sistema de fitxers subjacent o altres paràmetres relacionats amb la geometria dels discs. De fet, el sistema operatiu és un programari especial que permet a l'usuari executar altres programes mentre li proporciona serveis bàsics per a accedir a tot el maquinari disponible. En realitat, qualsevol maquinari conté d'una forma o una altra un sistema operatiu que el fa funcionar, com ara els telèfons intel·ligents, les consoles de videojocs, els automòbils, etc.

Normalment, quan arrenca l'ordinador i es carrega el sistema operatiu, aquest proporciona una línia de comandes en què l'usuari pot executar accions relacionades amb els elements típics que componen un sistema informàtic, però ja fa temps que pràcticament la totalitat de sistemes operatius també proporcionen una interfície d'usuari gràfica que permet executar la majoria d'aquestes accions de forma més visual, la qual és carregada per defecte. Això ha causat que la majoria d'usuaris desconegui, fins i tot, l'existència de la consola o línia de comandes, i també les operacions que es poden realitzar des de la mateixa, reduint el seu coneixement sobre el maquinari i el propi sistema operatiu a unes quantes operacions bàsiques relacionades amb el sistema de fitxers. Això és percebut com a positiu per molts usuaris que només volen usar el seu ordinador sense haver de preocupar-se'n, però no és el cas de cap usuari que pretengui extraure el màxim del seu entorn de treball.

De fet, en molts casos la tria del maquinari ja implica de certa forma triar el sistema operatiu, ja que aquest va molt lligat als detalls dels dispositius físics que disposa l'ordinador. En el cas dels primers ordinadors personals això era gairebé així, reduint les combinacions disponibles a unes quantes (per esmentar les més habituals):

- Un ordinador personal compatible amb l'IBM PC amb sistema operatiu MS-DOS/Windows (el més habitual).

- Un ordinador personal compatible amb l'IBM PC amb sistema operatiu GNU/Linux (per als més agosarats).

- Un ordinador personal de la marca Apple amb el sistema operatiu Mac OS.

Sense pretendre ser exhaustius, hi havia altres opcions com ara les estacions de treball de la marca Solaris o Silicon Graphics, que executaven diferents versions del sistema operatiu Unix, els ordinadors com, per exemple, el NeXT, o els sistemes operatius com podien ser OS/2 i altres variants d'Unix, però no

eren tan populars com les tres mencionades anteriorment. També hi havia altres sistemes operatius per a servidors de xarxa, però no eren populars entre els usuaris típics.

Cal destacar que en el cas dels ordinadors de la marca Apple, aquests porten un nucli semblant a GNU/Linux però que no ho és realment, que s'anomena *XNU*, i deriva d'un sistema anterior anomenat *Mach*, usat conjuntament amb BSD per a crear el primer sistema operatiu dels ordinadors NeXT, marca creada per Steve Jobs que va ser posteriorment comprada per Apple, donant peu al sistema actual. En la pràctica, els ordinadors Apple es poden considerar equivalents als ordinadors que executen GNU/Linux, tot i que no són exactament idèntics i no totes les eines estan disponibles (en el cas d'Apple).

En l'actualitat, en el moment de comprar un ordinador, aquest ja incorpora el sistema operatiu i no és possible triar-lo, com passa amb els ordinadors Apple. No obstant això, en el cas dels ordinadors PC, alguns fabricants permeten triar entre Windows i GNU/Linux i, així, es pot descomptar el cost de la llicència del primer, un fet conegut com a *Windows tax*.

1.3. El programari

El darrer nivell és el **programari** que aprofitarà totes les capacitats del sistema informàtic resultants de combinar el maquinari i el sistema operatiu. Òbviament, el sistema operatiu determina què es podrà executar i què no, tot i que moltes vegades els desenvolupadors de programari ofereixen versions per diferents sistemes operatius, excepte en el cas del programari propietari, en què és possible que un desenvolupador decideixi optar per una plataforma (maquinari + sistema operatiu) per a explotar al màxim les seves capacitats, oblidant-se de la resta. De fet, en alguns casos, el fet de disposar d'un programari concret era el que establia com havia de ser el sistema informàtic en què s'havia d'executar, com ara 3D Studio MAX, que només es podia executar amb el sistema operatiu Windows (i continua sent així) o el Final Cut Pro que només es pot trobar per Mac OS.

Actualment, excepte en casos molt específics, tot el programari es pot trobar per versions de diferents sistemes operatius, sigui perquè es tracta de programari obert amb el codi font disponible i que es pot compilar per a tenir una versió executable en qualsevol plataforma (maquinari + sistema operatiu), o perquè els fabricants de programari s'aprofiten dels nous entorns de desenvolupament multiplataforma que permeten generar binaris executables per cada plataforma entre les més comunes.

De fet, en un entorn de treball dins de l'àmbit de la ciència de dades, serà molt habitual haver de compilar aplicacions distribuïdes només com a codi font, de manera que caldrà disposar d'un conjunt d'eines (editors, compiladors, depuradors, etc.). No és estrany, doncs, que un dels primers programaris en obert disponible fos un compilador anomenat *GCC* (acrònim de *GNU Com-*

piler Collection) creat pel MIT l'any 1987, capaç de compilar llenguatge C. A partir d'aquell moment va ser possible fer créixer la família d'eines GNU, atès que es disposava d'una eina que permetia convertir-les en codi executable. El propi compilador podia usar-se per a crear noves versions del mateix compilador per altres plataformes i arquitectures, i també per altres llenguatges (entre altres, Fortran), impulsant així la disseminació de les eines GNU/Linux i les aplicacions basades en aquestes.

1.4. Una proposta

Combinant els elements introduïts en els apartats anteriors, és fàcil de veure que hi pot haver tants entorns de treball com científics de dades, atès l'ampli ventall de possibilitats, limitades gairebé només pel pressupost disponible. No obstant això, per a iniciar-se i desenvolupar competències pròpies de l'àmbit de la ciència de dades des d'una perspectiva personal, una bona proposta és optar per un entorn el més flexible possible, basat en l'ús de programari obert, amb una distribució GNU/Linux. L'opció de maquinari més econòmica és un ordinador PC compatible, especialment si ho comparem amb els ordinadors de la marca Apple, ja que podem disposar de més prestacions pel mateix preu (o més econòmic per prestacions similars), tal com mostra l'exercici de la taula següent:

	Dell* XPS	**MacBook Pro**
Pantalla	13 polzades	13 polzades
Processador	Intel Core i5 a 3.9 GHz	Intel Core i5 a 3.9 GHz
RAM	8 GB	8 GB
Disc dur	256 GB SSD	256 GB SSD
Sistema operatiu	Ubuntu Linux 18.04	Mac OS (XNU)
Preu (octubre 2019)	979 $	1.499 $

* Hem triat Dell per la seva popularitat, però hi ha moltes altres marques (Acer, Lenovo, Asus, etc.).

Òbviament hi ha altres aspectes que cal tenir en compte (pes, durada de la bateria, etc.) i que no s'han inclòs en la taula anterior. Per sort, la majoria de fabricants de maquinari permeten simular en línia la configuració d'un ordinador, podent fer comparacions entre diferents marques i models.

Pel que fa al sistema operatiu, en aquest cas ens decantem clarament per GNU/Linux (o XNU en el cas dels ordinadors amb Mac OS), atès que permet treure profit de tota la col·lecció d'eines i participar de forma activa en el moviment de programari lliure. Pel que fa a la distribució triada, les opcions també són gairebé innumerables, però d'entre totes podem destacar Ubuntu (Desktop), per moltes raons. Entre d'altres, té una interfície gràfica molt amigable, és altament personalitzable, d'acord amb les necessitats de cada usuari, hi ha una

gran comunitat d'usuaris i desenvolupadors darrera que asseguren una evolució constant, i funciona en gairebé qualsevol maquinari, incloent-hi els ordinadors antics, i hi ha versions Lite per a màquines amb capacitats limitades. Tot i que hi ha altres distribucions que *a priori* semblen orientades cap a l'àmbit de la ciència de dades, com ara Scientific Linux o Fedora Scientific, generalment és millor optar per una distribució genèrica com Ubuntu i adaptar-la a les necessitats de cada usuari, instal·lant els paquets i mòduls necessaris.

Eines

Algunes d'aquestes eines estan descrites a datascience.recursos.uoc.edu.

Entre d'altres, caldrà disposar com a mínim de les eines següents pel que fa a poder descarregar i manipular fitxers:

- **Compiladors** de C, C++ i Fortran, per a compilar llibreries científiques i altres eines que usen còmput numèric o simbòlic. El compilador GCC (de fet, un *front-end*) proporciona compiladors per a la majoria de llenguatges típics.

- Un **editor de text** adequat per a l'edició de codi font. Les opcions també són innombrables: des de les eines del sistema operatiu com ara Vi o l'editor Emacs, passant per Nano o Joe, fins a editors més visuals que usen el sistema de finestres per a oferir entorns de visualització WYSIWYG (*what you see is what you get*). És interessant que l'editor de text inclogui un mode per a l'edició de codi font, tenint en compte de quin llenguatge es tracta, ajudant així amb la sintaxi de les comandes, variables, estructures de control, etc.

- Eines per a la **manipulació de fitxers** CSV, JSON, XML, RDF, etc., és a dir, fitxers de text amb un format intern orientat a la descripció de dades en format tabular o jeràrquic, amb o sense metadades. Entre d'altres, destaquen CSVKit o Jq, per exemple.

- Eines per a accedir a **recursos web**, punts d'entrada SPARQL o API en general, com ara Wget o cURL, que permeten descarregar pàgines web i realitzar altres operacions mitjançant el protocol HTTP. Altres eines semblants però amb característiques addicionals que simplifiquen el seu ús són Aria2 i HTTPie.

R o Python

Per exemple, sobre «falsa» dicotomia entre l'ús d'R o Python vegeu www.kdnuggets.com/tag/python-vs-r.

I pel que fa a les eines més enllà de les primeres fases del cicle de vida de les dades, darrerament hi ha hagut una espècie de «falsa» dicotomia entre l'ús d'R o Python com a motor del sistema emprat per a analitzar i visualitzar les dades. Diem «falsa» perquè realment no cal triar entre els dos mons com si fossin incompatibles, atès que poden conviure raonablement bé en un mateix sistema. Sí que és cert, però, que R i Python mostren diferències molt importants que poden fer optar per un o l'altre, ja que parteixen de premisses molt diferents. Es pot dir que **R** és un entorn semblant a una «supercalculadora», en què es poden fer càlculs complexos i obtenir els resultats immediatament, usant un llenguatge amb una sintaxi i semàntica particulars però típica de programaris cientificomatemàtics. D'altra banda, **Python** és un llenguatge de programació

general que permet establir els passos que calen per a resoldre un problema i obtenir la solució desitjada, usant les llibreries adequades. Sigui com sigui, qualsevol científic de dades hauria de conèixer tots dos mons, usant cadascun en funció dels objectius i de les circumstàncies, sense haver de triar entre un o l'altre. Un entorn habitual en la caixa d'eines del científic de dades és **Anaconda**. Es tracta d'una distribució que inclou tot el que cal per a manipular dades, crear models, avaluar-los i implementar-los, i també generar informes i gràfics, tant per a R com per a Python. De fet, Anaconda és tan complerta que es pot dir que ho inclou gairebé tot (la instal·lació ocupa més de mig gigabyte), per la qual cosa pot no ser necessari en molts casos en què només es vol treballar amb un entorn més concret, com ara la dicotomia abans esmentada, R o Python:

- R + R Studio + R markdown + Shiny
- Python + Numpy + SciPy

1.5. Entorns virtuals

Un cop triat l'entorn de treball, l'opció natural és instal·lar el sistema operatiu escollit, fer les configuracions oportunes i, després, instal·lar tot el programari desitjat. Això inicia un cicle continu d'instal·lacions i actualitzacions, especialment per temes de seguretat que sempre cal tenir presents, a banda de les millores en les prestacions del maquinari, del sistema operatiu i del programari. Atesa la diversitat de maquinari, de sistemes operatius (i de distribucions en el cas de GNU/Linux) i de la combinació de programari i llibreries, a la pràctica molts cops el que pot passar és que el que funciona en una màquina pot no funcionar en una altra «idèntica», especialment per un problema de versions del programari necessari i les seves dependències, és a dir, del programari requerit per altres programaris, com ara les llibreries del sistema. De fet, la figura 3 mostra en clau d'humor la diversitat de comandes usades amb GNU/Linux per a la instal·lació de nous paquets o mòduls de programari, en funció de la distribució usada.

Figura 3. Instal·lador «universal»

Font: xkcd.com/1654.

Normalment, en un entorn personal això no representa cap problema, atès que es poden instal·lar totes les dependències i resoldre aquest tipus d'errors. Però en un entorn de producció d'una empresa, un canvi en una llibreria o versió pot tenir resultats catastròfics, provocant que moltes dependències es trenquin i s'hagin de resoldre, deixant tot el sistema en un estat inconsistent.

Una possible solució a aquest problema és l'ús d'**entorns virtuals**, els quals permeten reproduir un escenari concret de forma que es pot executar sobre qualsevol combinació de maquinari i sistema operatiu (potser amb algunes restriccions). Un cop es té una instal·lació que es considera adequada, aquesta s'encapsula en forma de màquina virtual que després pot ser executada per un programari, de forma que podem assegurar el seu funcionament independentment de la plataforma subjacent. Òbviament això té un cost pel que fa al rendiment i, en alguns casos en què es requereixi un maquinari molt específic no serà possible, però és una solució molt popular per a distribuir, per exemple, en una distribució de GNU/Linux entre els estudiants d'una assignatura concreta en què calen unes eines i configuracions específiques.

Hi ha diferents nivells:

1) Màquines virtuals com a caixes negres, usant eines com ara Parallels Desktop, VirtualBox o anteriorment VMware, amb les quals és possible crear una màquina virtual que s'executa de forma aïllada, sense interactuar amb la resta de processos que s'estan executant en la màquina que fa d'hoste executant el programari adequat.

2) Eines com Docker, que permeten executar en un mateix servidor diferents màquines virtuals que es poden comunicar entre elles mitjançant canals. Com que es comparteix el mateix sistema operatiu de base, es tracta de «contenidors» més lleugers que les màquines virtuals anteriors, que han de contenir tots els elements necessaris, incloent-hi el sistema operatiu.

3) Eines com Kubernetes, que permeten gestionar una «orquestra» de contenidors que es poden executar en recursos distribuïts i en el núvol, afegint un nivell d'abstracció respecte als anteriors, ja que no depenen d'una màquina concreta en què s'hostatgen les màquines virtuals o contenidors. De fet, Kubernetes permet usar Docker com a contenidors de les aplicacions que es volen distribuir.

A nivell personal no és habitual usar el tercer nivell i rarament el segon. En canvi, l'ús de màquines virtuals pot ser útil per a fer proves i assegurar que el codi desenvolupat es pot executar correctament independentment de la plataforma triada. En entorns de producció reals, usar eines del tercer nivell és cada cop més habitual, atès que els desenvolupadors es poden desentendre dels detalls de la plataforma en què s'acabarà executant el seu codi, creant eines que s'executen com un servei, el que s'anomena *SAAS*[2].

(2)Acrònim de *Software As A Service*.

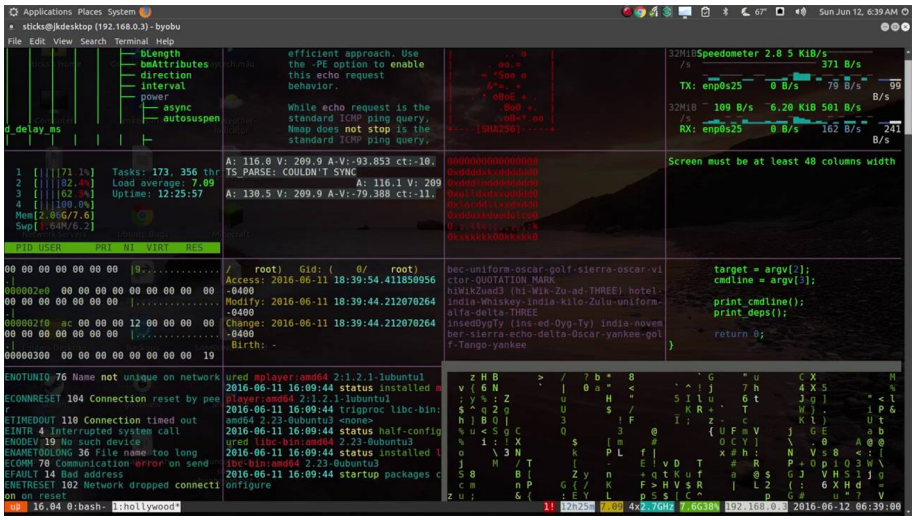
2. La línia de comandes

És una mica paradoxal que un cop configurat un sistema informàtic amb unes capacitats de còmput, gràfiques i d'emmagatzematge punteres i que aprofiten l'estat de l'art en els tres nivells descrits anteriorment, ara ens centrem en un dels elements més primigenis, gairebé un vestigi, que encara està disponible (però en molts casos amagat) en els sistemes operatius actuals, i ens referim a la **consola o línia de comandes**.

Cal tenir en compte que les interfícies gràfiques d'usuari actuals utilitzades pels sistemes operatius moderns són relativament recents, atès el cost del maquinari necessari per a visualitzar amb una resolució adequada els elements que les componen. Per exemple, Windows 3.0, que es pot considerar la primera versió popular del sistema operatiu de Microsoft per a ordinadors personals, va ser llançat l'any 1990, gairebé deu anys després de l'aparició de l'ordinador personal d'IBM el 1981, el qual incorporava MS-DOS com a sistema operatiu. Anteriorment ja hi havia altres propostes com la d'Apple Macintosh, de l'any 1984, o la de Commodore Amiga, de 1985. En l'actualitat, tots els sistemes operatius moderns ofereixen una interfície gràfica per defecte, relegant la consola a un segon terme per tasques molt concretes o considerades només per a administradors experts.

De fet, l'ús de la consola sembla que s'ha relacionat, erròniament, amb un ús limitat a experts, segurament per la seva utilització en pel·lícules, en què un *hacker* tecleja ràpidament comandes incomprensibles i resol un problema clau per al desenvolupament de la narració. Alguns exemples molt coneguts són *Matrix*, amb la pantalla de fòsfor verd plena de símbols, *Jurassic Park* o *Tron Legacy*, entre d'altres. De fet, la popularitat de la consola és tal que fins i tot hi ha una aplicació que simula una pantalla digna del millor *hacker*, anomenada com no podria ser d'una altra manera *Hollywood*, tal com mostra la figura 4.

Figura 4. Exemple de pantalla típica de *hackers*



Font: elaboració pròpia.

Sigui com sigui, la consola és una eina que permet realitzar operacions bàsiques de manteniment del sistema operatiu, especialment quan aquestes involucren moltes passes diferents o afecten molts elements diferents (fitxers, directoris, processos, etc.). No es tracta de triar entre la interfície gràfica i la consola, però aquesta darrera no pot ser desconeguda per a qualsevol persona que utilitzi el seu ordinador de forma intensiva i avançada.

No pretenem amb aquesta guia ser exhaustius pel que fa al que es pot fer des de la línia de comandes, per això ja hi ha manuals i altres materials docents que descriuen totes les seves capacitats. Ens limitarem a descriure les operacions més típiques que donen peu a treure tot el profit del sistema operatiu i el maquinari disponible més directament relacionades amb la ciència de dades.

Vegeu també

Els manuals i materials docents es proporcionen a l'apartat «Bibliografia».

2.1. Manipulació de fitxers

A banda del manteniment del propi sistema operatiu, una de les tasques habituals que es realitzen des de la línia de comandes és la **manipulació de fitxers**. Aquesta tasca té diferents nivells en funció de les operacions realitzades:

- **A nivell del sistema de fitxers**: això inclou llistar, crear, renombrar, moure i esborrar directoris de l'arbre de directoris, i també les mateixes operacions amb els fitxers que estan en aquests directoris. El sistema operatiu proporciona comandes bàsiques per a fer totes aquestes operacions.

- **A nivell del contingut dels fitxers**: això inclou operacions per a detectar patrons mitjançant expressions regulars, substituir caràcters o cadenes, o convertir formats, tant a baix nivell (la codificació usada, per exemple UTF-8) com a alt nivell (per exemple, entre CSV i JSON), tot i que en alguns casos això requerirà el suport d'alguna eina addicional.

Com a mostra de les possibilitats de la línia de comandes del sistema operatiu, la comanda següent busca tots els fitxers amb extensió «.zip» des del directori arrel de l'usuari i els mou a un directori anomenat «zips» dins del directori de fitxers temporals.

```
find /home/usuari -iname '*.zip' -exec mv '{}' /tmp/zips/ \;
```

Això també es pot fer des de la interfície gràfica, fent una cerca i després movent tots els fitxers trobats, però segurament amb un cost major, especialment si el nombre de fitxers trobats és elevat, a banda que aquesta comanda es pot automatitzar per a executar-la automàticament cada dia a una hora concreta, per exemple.

2.2. Gestió de processos

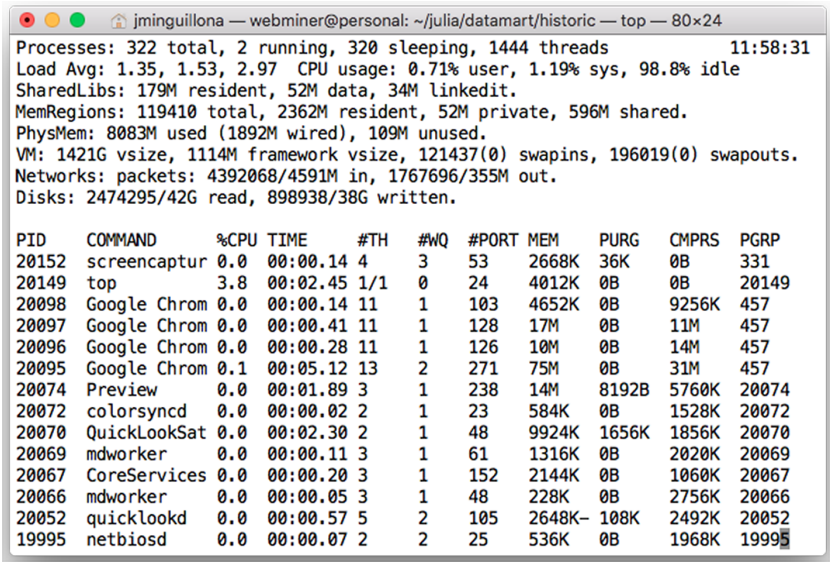
Una altra tasca típica que es realitza des de la línia de comandes és la **supervisió dels processos** que estan en execució. En aquest cas, algunes de les operacions més habituals són les següents:

- **Saber quins processos s'estan executant**: la comanda *ps* (*process status*) mostra els processos en execució en aquell moment exacte; la comanda *top* ho fa de forma dinàmica, actualitzant les dades a cada moment i també mostrant el consum de recursos com ara la càrrega de cada CPU i la memòria RAM usada, tal com mostra la figura 5. Una opció més amigable és *htop*, que afegeix més funcionalitats un cop instal·lada. Un cop identificat un procés pel seu identificador, és possible canviar el seu estat amb la comanda *kill* que, com el seu nom indica, permet interrompre l'execució de processos. Quan un procés executa diversos subprocessos, és possible eliminar-los tots amb la comanda *killall*.

- **Executar una comanda en *background***: si en el moment d'executar una comanda afegim & al final, aquesta comanda passa a executar-se en segon pla, alliberant la línia de comandes per a seguir executant altres operacions. Amb la comanda *nohup* es pot indicar que el procés que es vol executar en segon pla no ha de ser interromput, assegurant la seva execució completa.

L'execució de la comanda *top* genera un resultat semblant al que mostra la figura 5.

Figura 5. Exemple d'execució de la comanda *top*



Font: elaboració pròpia.

Aleshores, si volem cancel·lar o suspendre un dels processos en execució, només cal conèixer el seu PID i mitjançant la comanda *kill* canviar el seu estat, per exemple per a matar el procés anomenat *preview* faríem:

```
kill -9 20074
```

2.3. Automatització de tasques

Finalment, una de les característiques més interessants de la línia de comandes és que, de fet, es tracta d'un entorn integrat que permet desenvolupar i executar conjunts de comandes agrupats en el que es coneix com a ***scripts***. Els *scripts* (fitxers de text amb codi font) permeten crear seqüències de comandes que aprofiten les possibilitats que ofereix el sistema operatiu per a concatenar operacions, com ara els *pipes* o l'ús de fitxers temporals. A més, el llenguatge que interpreta i executa els *scripts* també proporciona els elements típics de qualsevol llenguatge de programació, com ara l'ús de paràmetres, variables, estructures de control (bifurcació i bucles), funcions, etc. Això permet crear petits programes que automatitzen la resolució dels problemes típics amb què es troba un científic de dades quan ha de resoldre un nou repte.

El sistema operatiu proporciona una línia de comandes o ***shell*** que és un programa que permet executar operacions i altres programes de forma interactiva. De fet, el primer *shell* en entorns Unix es va dir *sh* i des d'aleshores molts altres entorns han fet servir aquestes sigles, com ara Ksh, el popular Bash o Zsh. Tot i que hi ha una *shell* per defecte, normalment és possible usar diferents *shells*, indicant-ho en la primera línia del fitxer de text que conté l'*script*, tal com mostrava l'exemple de la figura 3 que usava Bash com a intèrpret de les comandes.

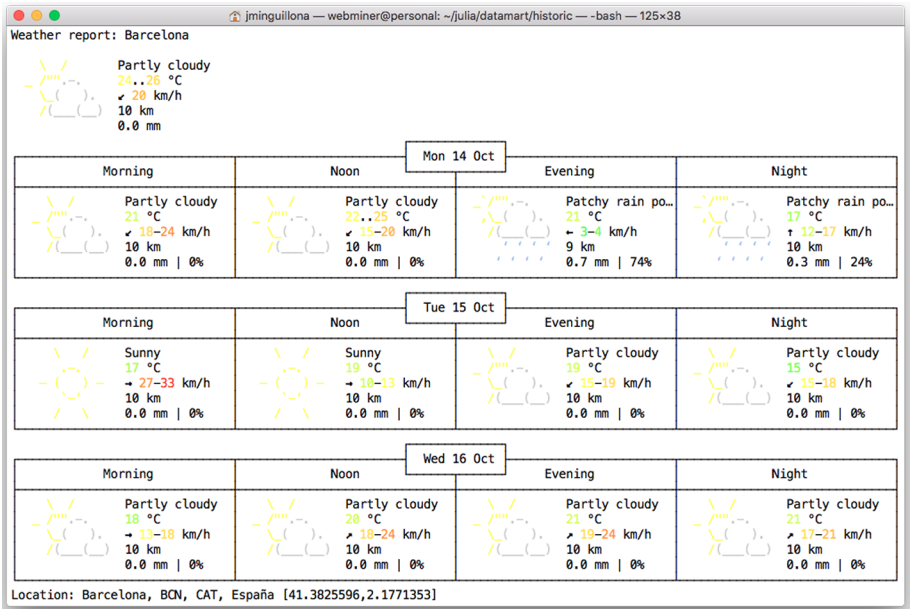
Com a exemples avançats del que es pot arribar a fer amb els *scripts*, hi ha una col·lecció anomenada *Bash Snippets* que inclou eines per a fer canvis de moneda amb el canvi actual, accedir a la predicció meteorològica o encriptar

i desencriptar fitxers, entre d'altres, combinant les capacitats de Bash amb altres eines com les descrites amb anterioritat. Així, un cop instal·lats els Bash Snippets, disposem d'una comanda per a obtenir la predicció en una ciutat concreta, de la forma següent:

```
weather Barcelona
```

El resultat obtingut per pantalla, usant només caràcters ASCII, és el següent:

Figura 6. Resultat de l'execució d'un *script Bash (weather)*



Font: elaboració pròpia.

Tot i que queda fora de l'abast i objectius d'aquest material docent, recomanem fer una ullada als Bash Snippets per a veure com aprofiten al màxim els recursos que el sistema operatiu ens proporciona per a treballar des de la línia de comandes.

3. Escenaris d'ús

Com hem comentat anteriorment, l'ús de la consola permet executar *scripts*, els quals resolen algun problema concret combinant comandes i eines diferents. Sense entrar en detalls tècnics sobre com fer-ho, alguns possibles problemes que es poden resoldre usant les possibilitats que ofereix la línia de comandes són com els que es descriuen a continuació.

3.1. Dades en obert

Cada cop és més habitual que l'Administració publiqui dades en obert, mitjançant portals en què es possible trobar conjunts de dades que descriuen molts dels aspectes relacionats amb la nostra societat, com ara dades sociodemogràfiques, del trànsit, contaminació, etc., tal com mostra l'exemple de la figura 7. En molts casos, el periodisme de dades comença amb l'anàlisi d'un o més conjunts de dades que poden aportar llum per a respondre alguna pregunta relacionada amb fets rellevants de la nostra societat. Descarregar aquests fitxers des del seu origen, netejar-los i filtrar la informació desitjada i fusionar-los en un de sol per a la seva anàlisi posterior són tasques que es poden automatitzar usant *scripts* des de la línia de comandes.

Figura 7. Categories del catàleg de datos.gob.es

Font: datos.gob.es.

Per exemple, cada dia es publica la llista de preus dels carburants a totes les estacions terrestres (també hi ha l'equivalent de les marítimes), la qual es pot descarregar en diferents formats, entre els quals com a fitxer JSON, mitjançant la comanda següent (és una sola línia):

```
curl "https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/
EstacionesTerrestres/" | jq
```

Si ens quedem amb els primers elements de la llista retornada per la comanda anterior processada amb *Jq*, una eina orientada a manipular documents JSON, obtenim:

```
{
 "Fecha": "14/10/2019 16:18:57",
 "ListaEESSPrecio": [
 {
 "C.P.": "01240",
 "Dirección": "CALLE GASTEIZBIDEA, 59",
 "Horario": "L-D: 24H",
 "Latitud": "42,842917",
 "Localidad": "ALEGRIA-DULANTZI",
 "Longitud (WGS84)": "-2,519194",
 "Margen": "D",
 "Municipio": "Alegría-Dulantzi",
 "Precio Biodiesel": null,
 "Precio Bioetanol": null,
 "Precio Gas Natural Comprimido": null,
 "Precio Gas Natural Licuado": null,
 "Precio Gases licuados del petróleo": null,
 "Precio Gasoleo A": "1,279",
 "Precio Gasoleo B": null,
 "Precio Gasolina 95 Protección": "1,359",
 "Precio Gasolina 98": null,
 "Precio Nuevo Gasoleo A": null,
 "Provincia": "ÁLAVA",
 "Remisión": "dm",
 "Rótulo": "ES DULANTZI REPSOL",
 "Tipo Venta": "P",
 "% BioEtanol": "0,0",
 "% Éster metílico": "0,0",
 "IDEESS": "14209",
 "IDMunicipio": "1",
 "IDProvincia": "01",
 "IDCCAA": "16"
 },
 …
 ]
}
```

Com es pot veure, per a cada estació tenim la seva localització (i geolocalització) i també el preu de diferents carburants, per la qual cosa es pot usar per a fer un seguiment i anàlisi dels preus d'una zona al llarg del temps, per exemple.

3.2. Generació de corpus de dades

Wikipedia és el recurs digital col·laboratiu més gran construït per la humanitat, contenint milions d'articles sobre temes diversos en més de 300 idiomes diferents. Per tant, és una base de coneixement incommensurable, que creix cada dia. A més, projectes com **Wikidata** afegeixen una capa semàntica que permet establir relacions entre els elements i fer cerques més complexes, habilitant el que es coneix com a **web semàntica**. Molts projectes de recerca usen

Wikipedia com a base per a la construcció de corpus de paraules, usats per entrenar sistemes de traducció automàtica, sistemes intel·ligents de pregunta/resposta, etc., sent necessari descarregar grans fitxers de dades que caldrà processar posteriorment.

De forma periòdica (cada pocs dies) es publiquen abocaments de memòria (*dumps*) de tot el que hi ha a Wikipedia, incloent-hi els continguts però també les edicions realitzades, dades dels usuaris, etc. Els *dumps* són fitxers XML o SQL, els quals es poden processar per a extreure la informació desitjada. En paral·lel, la Wikipedia disposa d'una API que permet accedir a dades de forma dinàmica, mitjançant consultes parametritzables. A més, Wikidata inclou un punt SPARQL per a fer consultes més avançades, no solament del contingut sinó de les relacions entre les entitats representades a les pàgines de Wikipedia.

Per exemple, des de la línia de comandes podem executar la comanda següent (és una sola línia):

```
curl "https://www.wikidata.org/w/api.php?format=json&action=
wbgetentities&ids=Q1492" | jq '.entities[].claims.P625'
```

Amb la comanda *curl* fem una petició a l'API de Wikidata, amb l'objectiu d'obtenir totes les entitats (*wbgetentities*) relacionades amb l'element Q1492 (que correspon a la pàgina de Wikipedia sobre Barcelona). El resultat és un fitxer JSON que és formatat adequadament mitjançant la comanda *jq*, amb la qual ens quedem amb la propietat *P625* que es correspon a la geolocalització de l'element, en aquest cas la ciutat de Barcelona. El resultat obtingut és:

```
[
 {
 "mainsnak": {
 "snaktype": "value",
 "property": "P625",
 "hash": "608b0ebef62ccb5a0b98f9f5c8953fa949ede0e1",
 "datavalue": {
 "value": {
 "latitude": 41.414802777778,
 "longitude": 2.1335555555556,
 "altitude": null,
 "precision": null,
 "globe": "http://www.wikidata.org/entity/Q2"
 },
 "type": "globecoordinate"
 },
 "datatype": "globe-coordinate"
 },
 "type": "statement",
 "id": "q3042433$A2E9939B-838B-4D6F-B383-14C5F49B0F03",
 "rank": "normal"
 }
]
```

És cert que aquest tipus de tasques també es podrien fer usant Python, per exemple, però la línia de comandes ens ofereix un entorn de treball en què es poden executar consultes com aquesta de forma senzilla, combinant comandes i eines típiques d'un entorn de treball d'un científic de dades.

4. Lectures relacionades

Com a tancament d'aquesta introducció general a la programació en *scripting*, hi ha uns quants articles que, tot i ser molt antics, resulten molt interessants per a entendre el raonament lògic que ha seguit el desenvolupament de totes les eines relacionades amb la consola del sistema operatiu i que permeten entendre millor per què algunes coses encara són com són en l'actualitat.

4.1. El llenguatge de programació de comandes d'Unix

Publicat l'any 1978 per Stephen Bourne, creador de la primera consola de comandes completa per Unix (*shell*), aquest article descriu el llenguatge de programació (segons paraules de l'autor) que permetia gestionar els recursos d'un ordinador executant el sistema operatiu Unix.

L'article es limita a descriure la sintaxi i les comandes bàsiques que es poden executar des de la línia de comandes, i també agrupar-les en *scripts* incloent tota la parafernàlia típica dels llenguatges de programació com ara variables, estructures de control, operacions i la gestió d'errors. No obstant això, és molt interessant, atès que introdueix conceptes com la concatenació d'operacions (*pipe*) o la redirecció de l'entrada/sortida, sent realment un exemple de com amagar detalls del sistema operatiu per a fer operacions més complexes de forma senzilla, però sempre des d'un plantejament de llenguatge de programació, no simplement d'*scripting*.

Posteriorment (l'any 1983) el mateix autor va publicar una versió més actualitzada en una de les revistes més importants de l'època pel que fa a la disseminació de la informàtica entre el públic general, anomenada *Byte*, en un monogràfic sobre el sistema operatiu Unix. La revista també és un molt bon exemple del boom que es va viure durant els anys 80 quant a la informàtica de consum, els anuncis que hi apareixen són un aparador perfecte del que estava passant.

Bourne, S. R. (1978). «UNIX time-sharing system: The UNIX shell». *The Bell System Technical Journal* (vol. 57, núm. 6, pàg. 1971-1990). Disponible a: ieeexplore.ieee.org/abstract/document/6770407.

Bourne, S. R. (1983). «The Unix shell». *Byte magazine* (vol. 10, núm. 8). Disponible a: archive.org/stream/byte-magazine-1983-10/1983_10_BYTE_08-10_UNIX#page/n187/mode/2up

4.2. El projecte GNU

Aquest article de Richard Stallman (conegut per *rms*) de l'any 1998, gran defensor del programari lliure i desenvolupador de molt programari que forma part de l'univers GNU/Linux, descriu com va ser dissenyat i creat a partir de la necessitat creada per la impossibilitat de compartir programari amb els ordinadors disponibles al laboratori del MIT on treballava, ple de màquines executant UNIX però amb llicències propietàries. Així va néixer el projecte GNU, acrònim recursiu de *GNU is not Unix*, nom típic en aquest context.

L'article presenta la línia argumental d'Stallman i altres desenvolupadors a l'hora de decidir els aspectes lligats a les llicències, els primers desenvolupaments (era especialment important disposar d'un compilador per a poder crear programes lliures a partir de codi obert), i també la creació de la Free Software Foundation (FSF) i l'impuls al programari lliure. També explica perquè parlem de GNU/Linux, que no és més que l'ús de Linux com a nucli en què executar tota la família d'eines que componen GNU.

L'article acaba llistant els perills del moviment, tenint en compte que es basa en la feina voluntària de milers de desenvolupadors, tot i que són pocs els que participen en projectes crítics. Entre d'altres, Stallman menciona les patents, els detalls ocults en el maquinari (cosa que dificulta el desenvolupament de programari per als nous dispositius), el programari propietari, amb Microsoft al capdavant com a enemic número u, i la dificultat de documentar tot de forma adequada.

Stallman, R. (1998). «The GNU project». *Open Sources*. Disponible a: noemalab.eu/org/sections/ideas/ideas_articles/pdf/stallman_eng.pdf.

4.3. Els llenguatges d'*scripting*

Aquest article, escrit per John Ousterhout l'any 1998, és un recull de les particularitats dels llenguatges d'*scripting* que els fan diferents (i interessants!) respecte als llenguatges de programació tradicionals. L'autor destaca que els llenguatges d'*scripting* es basen en l'existència d'altres components ja existents (potser programats en altres llenguatges de programació) i que, per tant, són més una espècie de cola que permet enganxar operacions per a aconseguir el resultat obtingut, afavorint la resolució de petits problemes i l'experimentació.

L'autor fa un recull de diferents llenguatges d'*scripting* (òbviament, una mica obsolet avui dia) i compara diferents eines desenvolupades amb llenguatges de programació tradicionals (C, C++, Java, etc.) amb versions més lleugeres programades amb un llenguatge d'*scripting* com és Tcl. L'argument principal per usar llenguatges d'*scripting* és que quan s'han de combinar tecnologies

diferents, com ara la creació d'interfícies gràfiques, l'accés a recursos d'internet (en aquella època quelcom encara molt poc desenvolupat) i l'ús de diferents *frameworks* per a la creació d'aplicacions.

Ousterhout, J. K. (1998). «Scripting: Higher level programming for the 21st century». *Computer* (vol. 31, núm. 3, pàg. 23-30). Disponible a: ieeexplore.ieee.org/abstract/document/660187.

4.4. Els orígens del llenguatge AWK

En un article de l'any 1979, Alfred V. Aho, Brian W. Kernighan i Peter J. Weinberger van presentar el llenguatge anomenat *AWK* (no de forma fortuïta, com es pot deduir), orientat a la detecció i processament de patrons. Segons els autors, AWK és una extensió de les funcionalitats de la comanda *grep*, tant pel que fa a l'ús d'expressions regulars per a detectar patrons com a les accions que es poden realitzar per a processar-los.

L'article descriu els elements que caracteritzen el llenguatge: la seva estructura, orientada a analitzar línies de fitxers de text, amb un bloc que s'executa un cop abans de processar cap línia, un bloc que s'executa per cada línia d'entrada i un bloc final que s'executa un cop després de processar totes les línies; el concepte de registre (de fet, la línia) i de camp; la sortida amb les operacions *print* i *printf* (com en el llenguatge C); les expressions regulars per a analitzar patrons; les funcions incorporades (també molt semblants a les del llenguatge C) i les estructures de control.

Els autors finalitzen l'article amb les consideracions de disseny, tenint en compte els objectius del llenguatge, orientat a resoldre petits problemes relacionats amb els fitxers de text, i una comparació del cost de diferents operacions realitzades amb AWK i altres comandes del sistema operatiu o llenguatges de programació. Tot i que avui en dia aquestes xifres estan del tot obsoletes, serveixen per a mostrar un dels objectius dels autors: la facilitat d'ús d'AWK (en un sentit de coherència entre totes les seves possibilitats), per sobre de la seva eficiència per a operacions concretes com explicar línies, quan existeix un comando específic (wc) per a realitzar aquesta tasca de forma molt més eficient.

Aho, A. V.; Kernighan, B. W.; Weinberger, P. J. (1979). «Awk—a pattern scanning and processing language». *Software: Practice and Experience* (vol. 9, núm. 4, pàg. 267-279). Disponible a: onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380090403.

4.5. Espai de recursos de ciència de dades

Finalment, recomanem als estudiants explorar l'espai de recursos de ciència de dades de la UOC, on es poden trobar descripcions i petits exemples d'ús d'eines relacionades amb aquests materials (i altres pròpies de l'àmbit). L'enllaç a aquest recurs és: «Espai de recursos de ciència de dades».

Bibliografia

Albing, C.; Vossen, J. P.; Newham, C. (2007). *Bash Cookbook: Solutions and Examples for bash Users*. California: O'Reilly Media.

Dougherty, D.; Robbins, A. (1997). *Sed & Awk, UNIX Power tools* (2a. ed.). California: O'Reilly Media.

Friedl, J. E. F. (2006). *Mastering regular expressions*. California: O'Reilly Media.

Janssens, J. (2014). *Data Science at the Command Line: Facing the Future with Time-tested Tools*. California: O'Reilly Media. Disponible a: www.datascienceatthecommandline.com.

Shotts, W. (2012). *The Linux command line: a complete introduction*. San Francisco: No Starch Press. Disponible a: linuxcommand.org/tlcl.php.