L'AMBIENTE UNIX E LE APPLICAZIONI UMANISTICHE

1. Quadro generale

Questo articolo nasce dalla convinzione, che ho acquisito in alcuni anni di esperienze informatiche in differenti settori umanistici, che attualmente l'ambiente operativo Unix[1] rappresenta la vera soluzione della maggior parte dei problemi che incontrano le applicazioni informatiche di carattere umanistico. Poiché tale convinzione non è giustificata tanto dalle possibilità "tecniche" dell'ambiente Unix (che pure sono formidabili, e pure assai poco conosciute), quanto da considerazioni di carattere organizzativo e generale, è necessario chiarire bene questo punto, prima di passare all'argomento specifico dell'articolo, che è quello di illustrare alcune delle possibilità di Unix. Quanto verrà detto non è strettamente connesso alle applicazioni archeologiche; d'altra parte in via metodologica non vi sono differenze specifiche, per quanto riguarda l'informatica, fra l'archeologia e le altre discipline. Perciò questo articolo viene dedicato al primo numero di una rivista come questa, con l'augurio che essa possa essere, come merita, il punto di riferimento di coloro che dibattono seriamente i problemi posti dall'uso delle tecnologie informatiche nell'ambito dell'archeologia.

Comincerò col notare che, a proposito della grandezza o, se si preferisce, della potenza dei computer, da alcuni anni si è stabilizzata una classificazione che, pur essendo per molti versi arbitraria (come sono tutte le classificazioni) tuttavia sembra reggere nel complesso ai cambiamenti repentini che avvengono nel settore[2]. Si distinguono dunque: a) i "main frame", complessi di potenza enorme, misurata in MIPS (million of inputs per second), attorno ai quali è organizzata un'adeguata istituzione (di solito chiamata Centro di Calcolo) che serve intere Università, grandi aziende, o grossi gruppi di ricerca. b) I "mini (computer)", macchine di potenza media, per i quali la misura si riferisce alla capacità di memoria RAM (random access memory) o primaria, in megabyte. In ambito universitario sono le tipiche macchine dipartimentali: potenti, flessibili e affidabili, non richiedono continua sorveglianza né un grosso apparato di gestione. c) I "micro (-computer)", macchine di potenza medio-piccola, basate su una sola scheda (microprocessore), destinate per lo più all'uso di un solo utente,

[1] Unix è un marchio registrato dei Bell Laboratories della AT&T. Indichiamo i manuali fondamentali, da cui si potrà trarre ulteriore bibliografia: B. W. Kernighan, R. Pike, *Unix*, Bologna 1985; S. R. Bourne, *The Unix System*, Reading etc., 1983; H. McGilton, R. Morgan, *Introducing the Unix System*, New York etc., 1983.

[2] Cf. p.es. G. Cioffi, V. Falzone (edd.), *Manuale di Informatica*, Bologna 1987, 713-714.

e variamente conformate in modo da essere da scrivania, oppure portatili, oppure adatte all'editoria, etc. Esse sono strettamente legate all'attività di una persona.

Se invece si prendono in considerazione le concezioni riguardanti gli ambienti di utilizzazione delle tre classi di computer, si ha l'impressione di assistere a continui ripensamenti, dettati non tanto dalle caratteristiche intrinseche delle macchine, e nemmeno tanto da quelle dei software sempre rinnovati, quanto dalle possibilità di organizzazione dei vari tipi di lavoro applicativo svolto per mezzo dei computer.

Quando è iniziata l'espansione dei micro (diciamo dal 1980) essi sono stati considerati utili solo per attività molto collaterali (lavoro spicciolo d'ufficio; al massimo data entry locale in vista di gestione da parte di una macchina grossa), mentre tutti i "veri" lavori di gestione sarebbero stati appannaggio esclusivo dei main frame. I progressi nella costruzione dei micro, e il conseguente incremento della loro potenza, ha poi generato grande confidenza nelle loro capacità[3]. La naturale propensione degli umanisti a lavorare singolarmente ha trovato in questo un notevole incoraggiamento.

Col tempo tuttavia ci si è accorti che l'organizzazione dei Centri di Calcolo, che pure ha alcuni svantaggi di tipo burocratico (in senso lato), ha vantaggi che vanno oltre la pura potenza delle macchine. Infatti i Centri di Calcolo possono offrire apparecchiature speciali e servizi che si giustificano solo se usati in comune da più ricercatori, ed inoltre hanno forme di salvataggio dei dati che scaricano i singoli utenti da alcune preoccupazioni in questo senso.

In questa prospettiva, i mini sono stati considerati dagli umanisti come macchine adatte soprattutto ai dipartimenti scientifici, in quanto richiedono pur sempre competenze che il singolo utente umanista ha raramente, e a prima vista sembrano offrire strumenti che possono trovarsi anche sui main frame. In realtà si comincia a scoprire che all'interno dell'organizzazione dei Centri di Calcolo alcune specifiche esigenze umanistiche trovano raramente soddisfazione, in confronto alle pressioni dell'utenza tecnico-scientifica, e che spesso si sprecano risorse per adattare del software immaginato per impieghi non umanistici ad usi che non gli sono congeniali. Basterà in questo contesto alludere p.es. al problema dei simboli alfabetici "complessi".

Ma la "crisi" in ambiente umanistico è venuta soprattutto da motivi interni, cioè dai rapporti fra gli stessi grossi centri a carattere umanistico, che pure esistono (CETEDOC, Banche dati bibliografiche, Oxford Computing Centre,

[3] Cf. p.es. i due fascicoli della rivista-bollettino del CNRS francese: « Le Médiéviste et l'ordinateur », 9, printemps 1983; 14, automne 1985. Per l'aspetto storico cf. S. AUGARTEN, *Bit by Bit. An Illustrated History of Computers*, 3 ed., London 1985, cap. 9, 253-281.

Istituto di Linguistica Computazionale di Pisa, Thesaurus Linguae Graecae, etc.[4]) e le singole ricerche. Infatti questi grossi centri sono sorti intorno a imprese particolari, per quanto vaste (p. es. le concordanze latino-medievali del CETEDOC) che spesso prevedevano pubblicazioni in forma non elettronica, e dunque non prevedevano collaborazioni dirette con altre ricerche informatizzate; oppure hanno costituito grossi corpora testuali, che tuttavia difficilmente comprendono i tanti testi minori che spesso interessano i ricercatori[5]. Alcuni centri hanno costituito banche dati biliografiche interrogabili in linea o diffuse su CD-ROM, ma anche qui gli argomenti presi in considerazione sono di solito i più convenzionali.

Si aggiunge che i grossi centri risultano abbastanza restii a diffondere i propri dati su vasta scala e soprattutto in linea (in parte per motivi di "gelosia" o di sfruttamento commerciale, ma spesso per questioni legali assolutamente obiettive); e soprattutto i loro dati sono spesso codificati in modo che poco potrebbero servire per essere sottoposti a procedimenti non previsti al momento della loro codifica.

Per queste, e per altre ragioni che qui non approfondiamo, oggi si fa strada fortemente l'esigenza di ottenere l'accesso pubblico a vaste masse di dati non mediante grosse organizzazioni accentrate, ma mediante la collaborazione della miriade di piccoli centri di ricerca specializzati, esattamente come è accaduto fino ad oggi per mezzo della produzione libraria. Come per mezzo della stampa ogni studioso o piccolo gruppo di studiosi mette a disposizione dei colleghi le proprie edizioni o repertori o monografie o articoli che, raccolti nelle varie sedi di ricerca, forniscono i dati per ricerche ulteriori; così nell'era della comunicazione elettronica si dovrebbero mettere a disposizione i risultati del lavoro di memorizzazione dei dati.

È evidente che questo comporta molti problemi, e solleva parecchie obiezioni. La più seria, per quanto posso vedere, è che in quel modo i dati tendono ad essere intimamente connessi ai giudizi su di essi da parte dello studioso che li ha raccolti e vagliati; e inoltre la loro codifica risentirà delle esigenze dettate dal software a sua disposizione. È opportuno notare che:

a) la "soggettività" dei dati, quando non sia spinta a livelli estremi, del resto comunque non compatibili con un lavoro serio di ricerca, non è affatto incompatibile con i sistemi automatici, anzi ne è un aspetto essenziale. Questo ha

[4] Per informazioni su questi e altri Centri analoghi, cf. I. LANCASCHIRE, W. MCCARTY, *The Humanities Computing Yearbook*, Oxford 1988.

[5] Va detto che fra le realizzazioni migliori in questo campo si possono menzionare il Text Archive dell'Oxford Computing Centre, per le caratteristiche della codifica e la trasparenza nella distribuzione; e in parte minore il Thesaurus Linguae Graecae di Irvin (Illinois), per la completezza dei testi.

insegnato per esempio Gardin[6] proprio in ambito archeologico. Potrei dire (riferendomi alla sintesi da me proposta dell'informatica umanistica[7]) che il momento della codifica dei dati è un momento molto importante, proprio perché non può mai prescindere da un apprezzamento soggettivo di essi, guidato dai risultati della ricerca precedente.

b) Ad ogni modo in ambito informatico occorrerà compiere uno sforzo per rendere i dati utilizzabili in ricerche diverse da quelle che li hanno prodotti, e questo può essere ottenuto in due modi interdipendenti: la massima semplicità nella loro codifica; il riconoscimento di un ambiente operativo comune ai ricercatori, estremamente aperto anche a programmi individuali, che fornisca uno standard utile allo scambio dei dati.

Vorrei proporre un solo esempio: l'organizzazione di una banca dati non è necessariamente legata a questo o quel pacchetto gestionale, ma piuttosto ad un lavoro teorico e metodologico che individui un rapporto fra la formalizzazione della sua struttura e la realtà che essa sarà chiamata a riprodurre e su cui si faranno le analisi in modo automatico.

L'umanista che progetta una banca dati prende di solito come punti di partenza (perché l'ambiente e le precedenti esperienze cartacee lo invitano a fare questo) due fattori: 1. il programma di gestione che userà; 2. la "scheda" dei dati che dovrà immettere. Questo atteggiamento (non possiamo chiamarlo metodo, perché è un modo di operare basato su elementi psicologici e vecchie abitudini piuttosto che sulla riflessione metodologica) è oggi da considerare errato alla radice, dopo la produzione degli studi (ancor prima che dei programmi) riguardanti il sistema RELAZIONALE di gestione delle banche dati.

Ricordiamo intanto brevemente che questo sistema si basa piuttosto su una visione teorica[8], e dunque non è legato in linea di principio alle realizzazioni tecniche, che vengono "a valle". Esso prevede la costituzione di una pluralità di archivi (contrariamente al singolo archivio in cui vengono inseriti i dati nel modello gerarchico), che vengono espressi sotto forma di "tabelle", nelle quali ciascun campo di notizie (all'interno di un record) può essere riempito con una sola notizia riferita al soggetto del record. I problemi derivati dalla necessità di riferire più notizie dello stesso tipo riguardanti un certo soggetto vengono risolti con la moltiplicazione dei record o con l'aggiunta di archivi. Il soggetto dei record, che resta la base della concezione degli archivi, viene identificato nei passaggi da un archivio all'altro in quanto mantiene inalterato almeno uno degli at-

[6] J.-C. Gardin, *Archaeological Constructs*, Oxford 1983 (cf. soprattutto cap. 3, 31-61).

[7] Cf. T. Orlandi, *Informatica umanistica*, Roma 1990, cap. 2, 29-41.

[8] P. Atzeni, C. Batini, V. De Antonellis, *La teoria relazionale dei dati*. Torino 1983 (con bibliografia).

tributi ad esso assegnati. Si aggiunge che un soggetto di record in uno degli archivi può essere un attributo di un altro soggetto in un altro archivio.

Si noti che questo sistema consente di utilizzare le singole tabelle o archivi come parte di un sistema complesso diverso da quello per il quale sono stati concepiti, purché siano stati concepiti correttamente in ambito relazionale. Siamo cioè in linea perfettamente coerente con la necessità di scambio di dati fra studiosi che li utilizzano per scopi diversi, ed in contesti diversi.

Ma soprattutto il sistema relazionale può essere svincolato dai programmi che vengono usati in concreto per gestirlo in modo automatico. Per questo motivo si rivela errato l'atteggiamento di chi pensa di dover basare l'architettura del suo sistema su un programma concreto (punto 1 sopra). Dato poi che gli archivi sono più di uno, si rivela altrettanto errato l'atteggiamento di chi vuole immaginare prima di tutto la "scheda", cioè l'ipotetica area in cui sono riunite tutte le notizie riguardanti i soggetti (considerati tutti di un solo tipo, dunque riuniti in un solo archivio) che interessano la sua ricerca[9].

Questo della scheda è un problema delicato che va approfondito nei suoi aspetti nascosti, di solito non considerati dagli operatori. Infatti la "scheda" in senso tradizionale non ha senso nel sistema relazionale, se la si vuole prendere come base per costruire un archivio. D'altra parte ha senso, una volta progettato correttamente l'archivio, e deciso il programma che lo gestirà, costruire la "maschera" di una scheda che consenta di introdurre i dati in maniera rapida e facile, non in relazione all'architettura dell'archivio, ma in relazione a come i dati si presentano materialmente allo studioso. Dunque è giusto porsi il problema della scheda, ma bisogna ben comprenderne la portata, e soprattutto comprendere che essa comporta un programma aggiuntivo (che del resto è previsto dai linguaggi di programmazione dei DBMS relazionali) che faccia da tramite fra la scheda stessa e la collocazione reale dei dati nei diversi archivi del sistema.

2. L'AMBIENTE UNIX

È utile a questo punto ricapitolare i dati della questione come abbiamo voluto porla per introdurre poi il discorso Unix. Nella gestione di data base si rivela essenziale l'architettura "teorica" di un sistema, non i programmi di gestione automatica; e possiamo aggiungere che, per quanto riguarda il campo della gestione di testi (che qui non affrontiamo[10]) si rivela essenziale la codifica dei testi, cioè un passaggio preliminare a forte carattere teorico, e non i programmi

[9] Una raccolta di esempi di "schede" in P. MOSCATI, *Archeologia e Calcolatori*, Firenze 1987, 15-53.

[10] Per una trattazione sintetica, con bibliografia, rimando a T. ORLANDI (cit. alla nota 7), 117-131.

di gestione di tali testi, vuoi per l'analisi, vuoi per la stampa.

Il fatto è però che l'attenzione ai programmi fin qui data dagli studiosi ha una sua causa molto reale. Se infatti i programmi effettivi di gestione non hanno alcun potere di migliorare la costruzione teorica dei sistemi (che è ciò che davvero importa), essi hanno però il potere di impedire — in determinati casi — molte delle libertà di azione necessarie alla corretta costruzione teorica. Inoltre essi possono impedire lo scambio di dati con studiosi che usano programmi diversi. Da questo punto di vista si comprendono le prese di posizione e le polemiche degli studiosi nei riguardi della bontà o meno dei programmi e dei pacchetti applicativi; ma ritengo che esse si basino su una situazione in via di superamento e pertanto debbano almeno cambiare obiettivo.

D'altra parte si nota che gli strumenti che davvero consentono libertà e coerenza nello standard dei dati sono quelli che fanno parte di un ambiente operativo, e quindi bisognerebbe concentrarsi assai più sulle caratteristiche dei sistemi operativi (o meglio ancora ambienti di sviluppo) che non su quelle dei pacchetti applicativi. Questa attenzione all'ambiente operativo, prima e più che ai singoli programmi, può fornire la soluzione soddisfacente ai problemi illustrati sopra.

È vero peraltro che oggi gli strumenti presenti nei sistemi operativi sono per lo più assai rudimentali, rispetto alle esigenze degli utenti. Occorre però chiedersi se e quali sistemi operativi presentino invece strumenti che si possono considerare ottimi, e soprattutto adatti alle ricerche umanistiche. Mi sembra in effetti che Unix sia uno di questi, e poiché non sembra molto conosciuto e sfruttato in questo campo, penso sia utile proporne le caratteristiche generali e dare alcuni esempi specifici relativi alla sua utilizzazione [11].

Unix è un sistema operativo "portabile": esso può essere installato su qualsiasi computer, indipendentemente dal modello, e indipendentemente dalla potenza della macchina. Le varie sue versioni sono installabili oggi sui personal come sui supercalcolatori [12]. Unix consente inoltre la piena portabilità dei programmi che siano stati implementati al suo interno, cosa che non si verifica facilmente per altri ambienti operativi.

Unix consente il lavoro a molti utenti per volta, ma soprattutto consente a ciascun utente di effettuare contemporaneamente più operazioni. Esso è inoltre particolarmente versato nel campo delle comunicazioni (messaggi, posta elettronica, scambio di file) sia a livello locale, sia attraverso le reti di trasmissione dati, sia semplicemente attraverso la rete telefonica.

[11] Una buona sintesi a livello non specialistico in L. CEROFOLINI, *Ambienti di sviluppo e sistemi di documentazione*, in G. ADAMO (ed.), *Trattamento, edizione e stampa di testi con il calcolatore*, Roma 1989, 3-28.

[12] Recentemente sono stati diffusi anche dei package (come mks-toolkit della Mortice Kern Systems) che simulano perfettamente Unix in ambiente DOS su personal della classe 8086.

Vi sono caratteristiche più specifiche che possiamo solo accennare per motivi di economia in un articolo, ma che meriterebbero un buon approfondimento perché si rivelano con l'uso essenziali alle applicazioni umanistiche. Esse riguardano: 1) il file-system. Esso permette di suddividere dati e testi in un grande numero di file, che tuttavia al momento della gestione possono riprendere senza altri passaggi l'unità iniziale. Questo è importante per questioni di indicizzazione, come per la gestione di basi di dati relazionali, come per molti altri lavori. 2) Gli strumenti di sviluppo. Essi sono molte decine, ciascuno con compiti ben specificati, e nello stesso tempo con la possibilità di variare i parametri applicativi secondo esigenze diverse. Alcuni li vedremo all'opera negli esempi più oltre, ma sono una minima parte rispetto a quelli esistenti. 3) Redirezione e pipe. Ognuno degli strumenti prevede che come input e output siano indicati dei file a piacere, ovvero che si formi una catena in cui l'output di uno strumento diventi l'input di quello impiegato successivamente. I risultati che si possono ottenere con queste procedure sono davvero incredibili.

Queste caratteristiche di Unix fanno sì che esso sia particolarmente adatto a lavori nei quali le esigenze vengono molto "parcellizzate", cioè vengono scisse in un certo numero di componenti fondamentali; ma nello stesso tempo l'ambiente vada mantenuto quanto più possibile unitario. Concretamente possiamo dire che se, p. es., col sistema operativo MS-DOS è possibile utilizzare package per la gestione di banche dati, package per indicizzazioni e concordanze, package per la gestione di testi, package per le comunicazioni; tuttavia ciascuno di essi avrà caratteristiche particolari che impongono input particolari e forniscono output particolari, e il passaggio di file dall'uno all'altro sarà per lo meno complicato, quando non impossibile. Con Unix invece tutto si svolge in un medesimo ambiente, sotto il controllo costante dell'utente, e con la possibilità di intervenire in moltissimi modi sofisticati in ognuno dei passaggi del procedimento a cui sono sottoposti i dati.

Aggiungiamo che su Unix esiste una vastissima bibliografia che ne esamina approfonditamente tutti gli aspetti, in vista di svariate applicazioni, cosa che non avviene per i sistemi operativi per macchine della classe mini o main-frame, per i quali si può solo ricorrere ai manuali delle case, che risultano di solito troppo ampi per certi versi, troppo sintetici per altri.

3. Esempi applicativi

Vogliamo ora tentare di mostrare alcuni esempi concreti di quelle funzioni che fanno di Unix un magnifico strumento umanistico. Dato che ci rivolgiamo soprattutto ad un ambiente archeologico, tratteremo specificamente di procedimenti relativi alle banche dati. In questo caso quello che ci proponiamo di mo-

strare è la duttilità e la potenza del linguaggio disponibile, e nello stesso tempo la perfetta trasparenza dei dati e dunque la loro portabilità in qualsiasi condizione ipotizzabile.

Per essere concreti, abbiamo scelto un esempio che consiste in una banca dati realmente esistente, progettata attraverso un package specifico installato su un sistema operativo non Unix, e cioè l'Rdb, per il sistema operativo VMS della Digital, in particolare per la macchina VAX. Il motivo della scelta è dovuto al fatto che sulla banca dati di cui parleremo è stato pubblicato un interessantissimo resoconto, che non solo ne mostra il fondamento metodologico e la struttura completa anche nei particolari, ma anche spiega in maniera eccellente e adatta ad un pubblico di studiosi umanistici i principi del sistema detto "relazionale", che attualmente appare di gran lunga il più adatto per le esigenze umanistiche [13]. Ci è sembrato opportuno che l'esemplificazione venisse fatta proprio su un problema pensato indipendentemente da Unix.

Non intendiamo affatto mettere in concorrenza i due sistemi, ciascuno dei quali ha i suoi pregi, e può essere consigliabile in diverse circostanze. Quello che interessa è solo di mostrare con un esempio concreto gli strumenti di Unix all'opera.

La banca dati è costruita sul contenuto di un gruppo di documenti riunito nei volumi manoscritti delle cosiddette Notti Coritane, che riportano l'attività di un gruppo di eruditi di Cortona, riuniti nell'Accademia Etrusca, che era volta a descrivere e studiare vari tipi di oggetti (reperti archeologici e naturalistici, dipinti, disegni, incisioni, monete e medaglie, antichi manoscritti, etc.) ed inoltre ad illustrare fatti di storia locale e riunire notizie artistiche e scientifiche. Si tratta dunque di avere indici, elenchi e raffronti di tutto il materiale di cui si parla nei manoscritti delle Notti Coritane.

Secondo i principi del sistema relazionale, i file (o tabelle) vengono costruiti a partire dai soggetti fondamentali, o "entità" su cui si intende indagare [14]. In questo caso si sono scelti: PERSONAGGI — OGGETTI — LUOGHI — EVENTI, e per maggiore flessibilità si sono anche distinti: ATTRIBUTI (dei personaggi) — TIPOLOGIA (degli oggetti) — EVENTI (di cui si è trattato nel corso delle riunioni) — AZIONI (nel senso di rapporti generali fra persone e cose etc.: autore di, scopritore di, etc.) — COLLOCAZIONE (delle notizie all'interno dei manoscritti).

Sempre secondo i principi del sistema relazionale, sono stati costruiti altri

[13] F. Fidecaro, A. Tosi, *Le Notti Coritane. Applicazione del modello relazionale*, « Bollettino d'Informazione (Centro di Elaborazione Automatica di Dati e Documenti Storico Artistici, Pisa) », 10 (1989) fasc. 2, 195 pp.

[14] *Ibid.*, 11.

file (o tabelle) che riguardano le relazioni poste (oggettivamente o soggettivamente) fra le "entità" sopra citate, in particolare fra Personaggi, Oggetti, Luoghi, Eventi, dando origine a file il cui nome indica chiaramente lo scopo: P-O (relazioni personaggi-oggetti); P-L (relazioni personaggi-luoghi); e così via per tutte le possibilità.

Così nel file PERSONAGGI i record sono divisi nei campi: nome — identificatore del personaggio. Nel file ATTRIBUTI: titolo1 — titolo2 — titolo3 (si possono avere dunque fino a tre titoli. Qui i principi del sistema relazionale sono stati un po' coartati, ma ciò è giustificato dalla poca importanza) — qualifiche accademiche — professione — identificatore del personaggio — identificatore della collocazione. Nel file TIPOLOGIA: tipologia — codice del tipo. Nel file OGGETTI: oggetto (sua descrizione) — quantità — materiale — tecnica — data — codice del tipo — identificatore dell'oggetto. Nel file LUOGHI: luogo — edificio — appartamento — identificatore del luogo. Nel file EVENTI: evento (sua descrizione) — data — identificatore dell'evento. Nel file AZIONI: azione (sua descrizione) — codice dell'azione. Nel file SEGNATURA: identificatore della collocazione — volume — pagina.

Come si vede, vi sono dei campi in ciascun file (quelli chiamati in genere: identificatore di...) che si ripetono e consentono di rimandare (in gergo, che riprendiamo volentieri dalla pubblicazione, "navigare") da un soggetto all'altro su cui si fanno le ricerche. Questo, e la stessa struttura dei file, permette di immaginare gli algoritmi abbastanza semplici che guidano la "navigazione", e di fare sia singole ricerche in linea, sia di produrre indici anche complessi.

Cominciamo dalle interrogazioni in linea[15]. Immaginiamo che il punto di partenza sia un oggetto che ha attirato la nostra attenzione (p. es. una "Caricatura del musico Carnacci celebre buffo di Roma"; v. p. 59-60) e vogliamo avere tutte le notizie che lo riguardano. In questo caso lo strumento essenziale sarà il "grep" (eventualmente con le varianti "egrep" e "fgrep", su cui non ci soffermiamo). Questo comando permette di effettuare su un file o su un gruppo di file in qualche modo omogeneo (possono appartenere ad una directory; oppure possono avere qualche cosa in comune nel nome) la ricerca di una sequenza qualsivoglia di simboli alfanumerici e speciali, e mostra le linee che li contengono, eventualmente precedute dal nome del file in cui si trovano.

Dunque scrivendo il comando:

⟨grep "Caricatura del musico Carnacci" oggetto⟩

[15] Negli esempi, i comandi in linea sono indicati dai segni ⟨. . .⟩, e i risultati dai segni : . . . :. Gli esempi sono molto realistici, ma abbiamo dovuto omettere qualche particolare che avrebbe nuociuto alla chiarezza. Si tratta soprattutto di qualche apice che talvolta occorre aggiungere. Ma chi abbia una conoscenza elementare di Unix può facilmente utilizzare gli esempi seguenti in modo concreto.

dove oggetto è il nome del file, si ottiene di visualizzare la linea che immaginiamo sia la seguente:

: Caricatura del musico Carnacci celebre buffo di Roma@1@carta@disegno@ 0009@0209 :

dove:
@ è il simbolo di separazione dei campi della scheda (una scheda si identifica con una linea del file)
0009 è l'identificatore della tipologia
0209 è l'identificatore dell'oggetto.

Scrivendo il comando:

⟨grep "0009" tipologia⟩

verrà visualizzata la seguente linea:

: stampe@0009 :

e sapremo che si tratta di una stampa.

Scrivendo il comando:

⟨grep "0209" P-O⟩

dove P-O è il file che contiene le relazioni fra gli oggetti, i personaggi, e le "azioni" (da intendere in modo molto lato) che riguardano gli oggetti, verranno visualizzate le seguenti linee:

: 0122@0209@0015@0073:
: 0131@0209@0106@0073:
: 0075@0209@0084@0073:
: 0254@0209@0067@0073:

Nel terzo campo si trova il numero che identifica l'azione, mentre nel primo campo si trova il numero che identifica il personaggio implicato dall'azione. Si tratta di usare sempre il "grep" sui file opportuni, e cioè "personaggi" e "azioni" per mettere in chiaro di che si tratta. Faremo un solo esempio di richiesta:

⟨grep "0015" azione⟩

: incisa/e da@0015 :

⟨grep "0122" personaggi⟩

: Tuscher Marco@0122 :

Volendo conoscere anche gli eventuali attributi del personaggio si potrà procedere di nuovo con grep sul file "attributi", che ci dà l'occasione di proporre un passo in avanti nel procedimento. Infatti il record (o linea) di questo file è abbastanza lungo, e possiede 7 campi (cf. sopra). Il risultato "bruto" di "grep" sarebbe dunque (qui invento totalmente il contenuto dei campi):

: Marchese di Villaverde@Cavaliere@Gran Ciambellano@Incisore@Accademico Etrusco@0122@3241 :

e a noi interessava solo il campo 5.

È opportuno a questo punto introdurre lo strumento chiamato "awk", che ha moltissimi usi, di tipo sia editoriale sia di gestione di dati. In sostanza esso considera singolarmente le linee di un file, e all'interno di esse distingue dei settori (i nostri campi) considerando come separatore un simbolo che viene specificato di volta in volta. Ogni settore viene rappresentato dal simbolo $, seguito dal numero progressivo del settore: $1, $2, $3, etc. Dando p. es. questo comando:

⟨ awk -F@ {print $5}⟩

verrà visualizzato solo il quinto campo della linea in questione. Nel nostro caso, si dovrà mettere in "pipe" tale comando, dopo quello di "grep":

⟨ grep "0122" attributi | awk-F@ {print $5}⟩

: Accademico Etrusco :

In realtà "awk" è un vero e proprio linguaggio di programmazione, che si piega ad usi infiniti. Noi qui e in seguito indicheremo soltanto alcuni dei più semplici. È possibile per esempio ottenere degli output più eleganti:

⟨ grep "0122" attributi | awk -F@ {print "Il personaggio è un "$5}⟩

: Il personaggio è un Accademico Etrusco :

Ci sembra insomma di aver chiarito fin qui come è possibile navigare attraverso le varie tabelle relazionali rimanendo sempre esclusivamente nell'ambito del sistema operativo. Compiamo ora un passo ulteriore, utilizzando la possibilità che Unix offre, di raccogliere un certo numero di comandi in un testo, e poi farli eseguire dal sistema senza bisogno di intervenire direttamente. Nel nostro caso sarà essenziale introdurre uno strumento che gestisca le "variabili", perché altrimenti non sarebbe possibile passare da un file all'altro, mediante gli identificatori, senza intervento del ricercatore interessato.

Lo strumento che usiamo è "read". Scrivendo questo comando:

⟨ read VARIABILE1 VARIABILE2 VARIABILE3⟩

si darà il nome a tre elementi di una riga (ciascuno deve essere separato da uno spazio) che viene scritta lì per lì ovvero viene fatta provenire dal risultato di un comando precedente attraverso un "pipe". Per preparare opportunamente tale riga, si farà uso di "awk" in modo analogo a quanto abbiamo mostrato sopra.

Ecco dunque come appare un testo che riassume (se così possiamo esprimerci) la navigazione mostrata in precedenza (gli a-capo sono importanti; abbiamo numerato le righe solo per le successive annotazioni - Unix non lo prevede):

```
1 grep "Caricatura del musico Carnacci" oggetto
2 awk -F@ {print $5 " " $6} | read TIPO OGGETTO
3 grep $TIPO tipologia | awk -F@ {print "L'oggetto è del tipo" $1}
4 grep $OGGETTO P-O | awk -F@ {print $3 " "$1}
5 while read AZIONE PERSONAGGIO
6 do grep $AZIONE azione | awk -F@ {printf "  %s:",$1}
7 grep $PERSONAGGIO personaggi | awk -F@ {print $1}
8 done
```

Commenti: alle righe 2, 3, 4, 6 si vede come viene usata la possibilità di indicare ad "awk" dei testi da stampare, messi fra virgolette, inclusi eventualmente degli spazi, per rendere più elegante l'output. — Alla riga 6 viene usato il comando "printf" invece di "print", per evitare che l'output seguente vada a capo. — Alle righe 5-7 abbiamo introdotto un ciclo di "while", che fa parte degli strumenti di Unix, per rendere possibile l'elenco di tutte le azioni connesse con l'oggetto in questione. Questo fa parte di ciò che viene chiamato il linguaggio "shell" di Unix, ma non intendiamo qui approfondire questo settore, che pure è di grande interesse anche per l'utente umanista.

Per rendere di uso generale questo che, come si vede, diventa un vero e proprio programma, sostituiamo alla riga 1 questi altri comandi abbastanza evidenti, dopo quanto abbiamo detto:

```
1 echo "Inserire l'oggetto da cercare:"
2 read RICERCA
3 grep $RICERCA oggetto ... (etc. ...)
```

Diamo poi il comando che trasforma qualsiasi file (che naturalmente contenga quanto si deve) in uno strumento alla pari di quelli usati finora. Se il nostro file si chiama, p. es., "ricerca", si scriverà:

⟨chmod + x ricerca⟩

e in tal modo, da questo momento in poi, scrivendo:

⟨ricerca⟩

si avrà il seguente risultato:

```
: Inserire l'oggetto da cercare: :
⟨Caricatura del musico Carnacci⟩
: L'oggetto è del tipo: stampe :
:       incisa/e da Tuscher Marco :
:       disegnato/a da Ghezzi Pier Leone :
:       raffigura Carnacci :
```

Aggiungiamo poi, per inoltrarci un po'di più nelle caratteristiche di "awk", che per la verità alcune delle operazioni doppie sopra utilizzate (grep... awk) possono essere semplificate e anche rese più mirate dalla capacità di "awk" di agire solo su quelle linee che in uno dei campi contengano una sequenza voluta. Pertanto la riga 3 può essere sostituita da:

⟨awk -F@ /$TIPO/ ~$2 {print "L'oggetto è del tipo " $1} tipologia⟩

Abbiamo dunque visto come sia agevole navigare in una banca dati di tipo relazionale con gli strumenti di Unix. Spostiamo ora l'attenzione dalle ricerche del tipo in linea, alle possibilità di ottenere indici di vario genere di tutto il materiale della banca dati. In parte, come si comprenderà bene, i procedimenti potranno essere gli stessi. Quello che funziona per un caso, può essere ripetuto (si ricordi il ciclo "while") per tutti i casi del medesimo tipo.

Ma per ottenere indici generali vi sono strumenti più adatti, che è opportuno presentare anche perché possono essere utilizzati per molte altre esigenze, che il lettore individuerà da solo partendo dalle proprie esigenze. Gli strumenti a cui alludiamo sono soprattutto: "join" e "sort". Quest'ultimo riordina secondo l'ordine alfanumerico le linee di un file, in maniera molto flessibile. Infatti esso può distinguere un certo numero di campi nella linea, mediante un simbolo di divisione che viene indicato dall'utente, e ordinare le linee tenendo conto solo dei campi voluti, e inoltre dando una certa priorità all'interno di quei campi.

Se "sort" è uno strumento che (sia pure in modo meno sofisticato) si trova generalmente nei sistemi operativi, "join" è invece uno strumento che, per quanto ci consta, è peculiare di Unix. Esso agisce su due file, e mette a confronto in sequenza le loro linee. Le linee sono distinte in campi (questa volta separati necessariamente dallo spazio). Nel tipo di impiego più semplice, quando il primo campo delle due linee corrisponde, esso presenta in output tale campo (una volta sola), e di seguito gli altri campi sia del primo che del secondo file.

Ritorniamo all'esempio fatto prima, e supponiamo di voler elencare le Notti Coritane di cui è data notizia negli omonimi volumi, in ordine di data, aggiungendo il luogo in cui furono tenute, e l'indicazione di volume e pagina della notizia. I file che entrano in gioco saranno: eventi, luoghi, L-E (relazione fra luoghi ed eventi). Ricordiamo come sono organizzate le loro linee (cioè le schede e i relativi campi:
eventi: evento @ data @ Ide (identificatore dell'evento)
luoghi: luogo @ edificio @ appartamento @ Idl (identificatore del luogo)
L-E: Ide @ Idl @ Code-r (tipologia degli eventi) @ Scheda (vol. e pag.)
Si tratterà prima di tutto di porre all'inizio delle linee i campi "di riferimento". Mediante "awk" è agevole ricomporre le linee dei file, in modo che il primo campo in eventi sia Ide, il primo campo in luoghi sia Idl. Quindi sceglieremo in eventi le linee che contengono le Notti Coritane, e le metteremo in ordine

secondo Ide, ottenendo un file che chiameremo eventi.new:

⟨grep "Notte Coritana" eventi | sort -t@ -1 ⟩ eventi.new⟩

Quindi ordiniamo L-E secondo il campo Ide, ottenendo un file che chiamiamo L-E.new:

⟨sort -t@ -1 L-E⟩ L-E.new⟩

Se ora diamo il comando:

⟨ join eventi.new L-E.new ⟩ temp01 ⟩

otteniamo un file temp01 in cui le linee sono così composte
temp01: Ide @ Evento @ Data @ Idl @ Code-r @ Scheda
Mediante un ulteriore passaggio in "awk" elimineremo i campi Ide e Code-r che non servono, e porremo Idl al primo posto. A questo punto agiremo sul file luoghi:

⟨sort -t@ -1 luoghi⟩ luoghi.new⟩

e di nuovo agiremo attraverso "join":

⟨join temp01 luoghi.new⟩ temp02⟩

Le linee del file temp02 saranno così composte:

temp02: Idl @ Evento @ Data @ Scheda @ Luogo @ Edificio @ Appartamento

Come si vede, abbiamo ottenuto riunite in ciascuna linea tutte e solo le notizie che volevamo dare nell'elenco. Agendo con "sort" e "awk" metteremo le linee in ordine di data, ed elimineremo il campo Idl che non serve. Di nuovo "awk" potrà servire, come abbiamo visto sopra, a stampare l'indice in una forma elegante.

4. Conclusione

Io spero che risulti da quanto abbiamo esposto, che un umanista che prenda un minimo di tempo e compia un minimo sforzo mentale per informarsi sugli strumenti che Unix mette a disposizione, può gestire i propri dati in maniera semplice e diretta con due vantaggi fondamentali: (1) affrancarsi in modo sostanziale dai rapporti di dipendenza dagli specialisti di informatica e dagli analisti, che per quanto amichevoli spesso tarpano la "fantasia" (scientifica, beninteso) dello studioso nello sperimentare sempre nuovi modi di analizzare i dati di cui dispone. (2) Mantenere i propri dati codificati e memorizzati in maniera tale da essere facilmente scambiati con quelli di colleghi con cui intrattenga rapporti, soprattutto (ma non esclusivamente) se anch'essi possono disporre dell'ambiente Unix sulle macchine a cui hanno accesso.

A mio modo di vedere, solo questa è la via mediante la quale si può realizza-

re l'accumulo spontaneo di conoscenze e dati su supporto magnetico (o comunque gestibile con sistemi elettronici), attraverso la collaborazione di singoli studiosi, perfettamente indipendenti, piuttosto che demandare questo compito a grossi centri nazionali o internazionali, che per quanto bene orientati e organizzati avranno per loro natura una tendenza in qualche modo egemonizzante.

TITO ORLANDI
Dipartimento di Studi Storico-religiosi
Università di Roma "La Sapienza"

ABSTRACT

The trend in the computing for research in the humanities, unlike instruction or social sciences applications, seems to be the coexistence and exchange of many little or medium-size data-bases (both textual and 'factual') rather than large ones, developed in big institutions. This requires two main conditions: a common operating environment, and standards in the organization and encoding of data. In archaeology, as in other disciplines, Unix offers a convenient solution for problem 1, and relational data-base theory for problem 2. An example is given on how a data-base may be organized and managed exclusively with the native tools of Unix and plain ASCII files.