

Files & the File System

Modern Plain Text Computing

Week 02

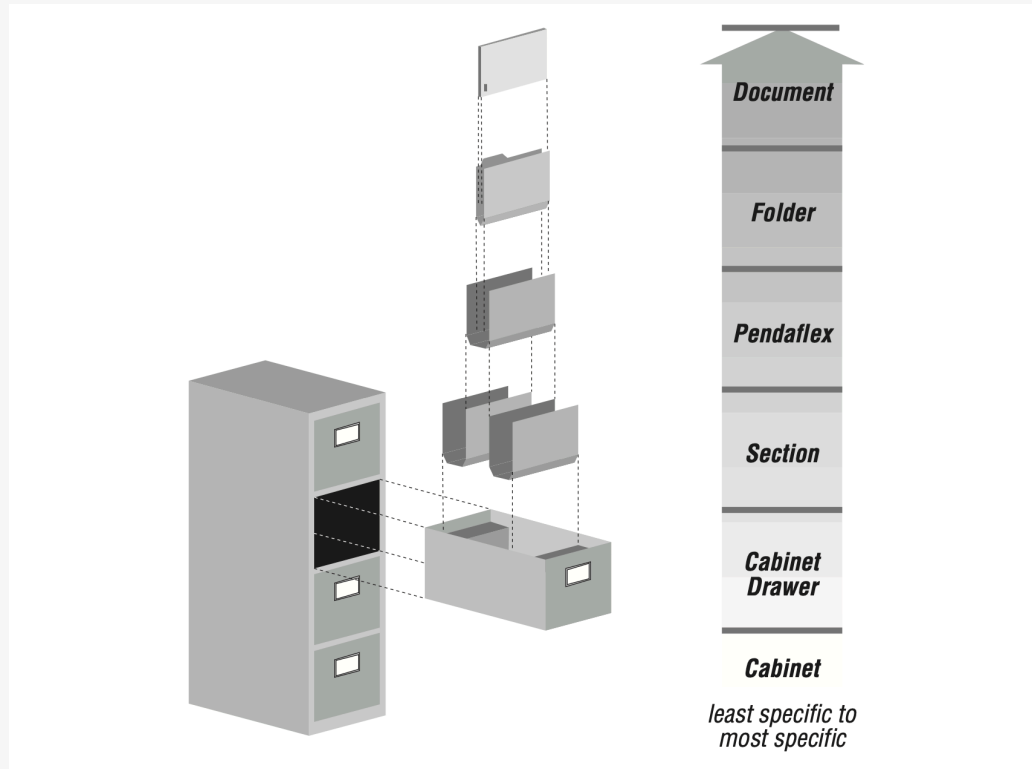
Kieran Healy

kieran.healy@duke.edu

September 2026

Files

What is a file?



You very likely have never used one of these. Maybe you've never even seen one in real life.

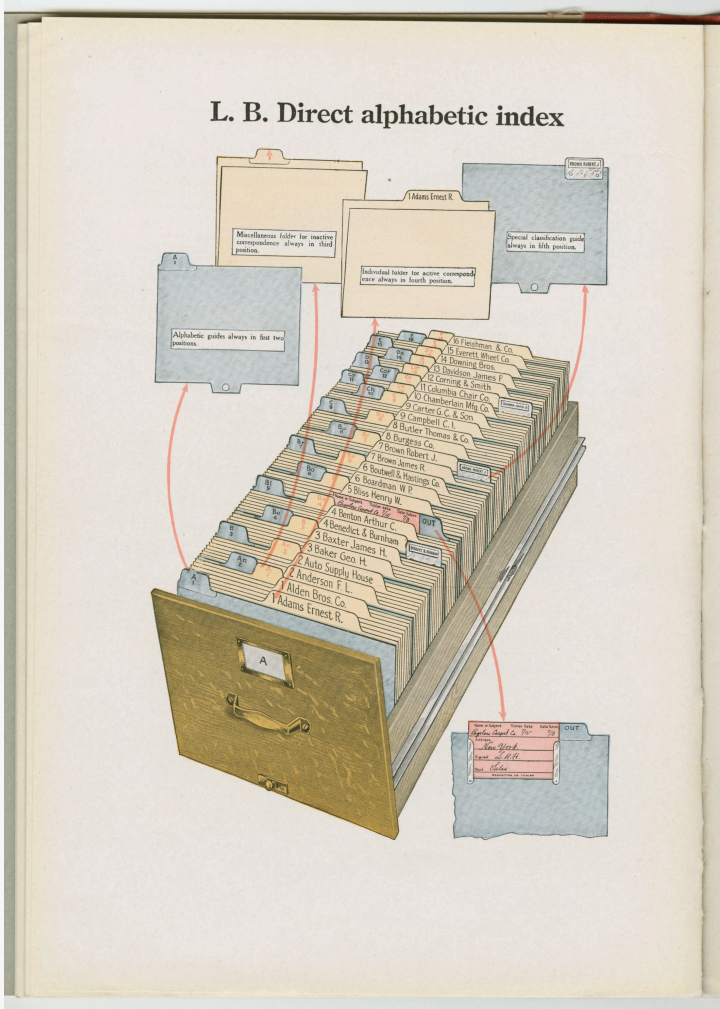
Illustration: Shelley Powers et al. *Unix Power Tools*, 3rd ed. (Sebastopol, CA: O'Reilly Media, 2002), 21.

The file cabinet!



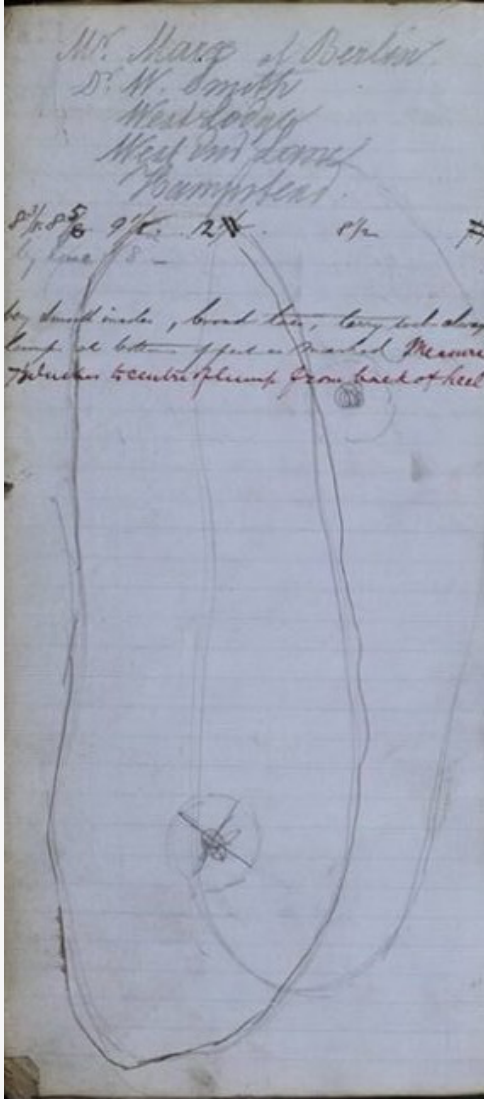
“Could capitalism, surveillance, and governance have developed in the twentieth century without filing cabinets? Of course, but only if there had been another way to store and circulate paper efficiently; if that had been the case, that technology would be the object of this book.” — Craig Robertson *The Filing Cabinet: A Vertical History of Information* (University of Minnesota Press, 2021), 3.

The file cabinet!



“Cabinet logic involves the creation of interior compartments to organize storage space according to classification and indexing systems ... Partitions made from paper, not wood, divided storage space to create rigorous order; these partitions took the form of tabbed manila folders separated by tabbed guide cards. This iteration of the logic dispensed with a separate index to make paper discoverable by utilizing the “very organization of the material and its location” with the “vertical guides serving as locating medium.” Elimination of an index was signaled in filing literature by the terms “direct alphabet index” and “automatic index” ... Without the need to consult a separate index, a clerk grouped papers together on their edge behind tabs labeled with classifications, so any given paper could be found quickly.” — Robertson, *The Filing Cabinet*, 104–5.

Daybooks and Ledgers



Peal and Company (Bootmakers), London. “The collection consists of 660 items, 40 of which are administrative records such as account books and ledgers. The remaining 620 volumes are ‘Feet Books’ recording each order and usually containing an outline drawing of the customer’s feet. Every book includes 150 orders and was given a running number. The earliest surviving book is no. 23 from the 1870s and the last is no. 1002 from shortly before the firm closed in 1965.”

“Mr Marx of Berlin c/o Dr W Smith, West Lodge, West End Lane, Hampstead.” 02/003
(Book 023, c.1870) page 142.

([London Archives Collections Catalogue.](#))

Index cards

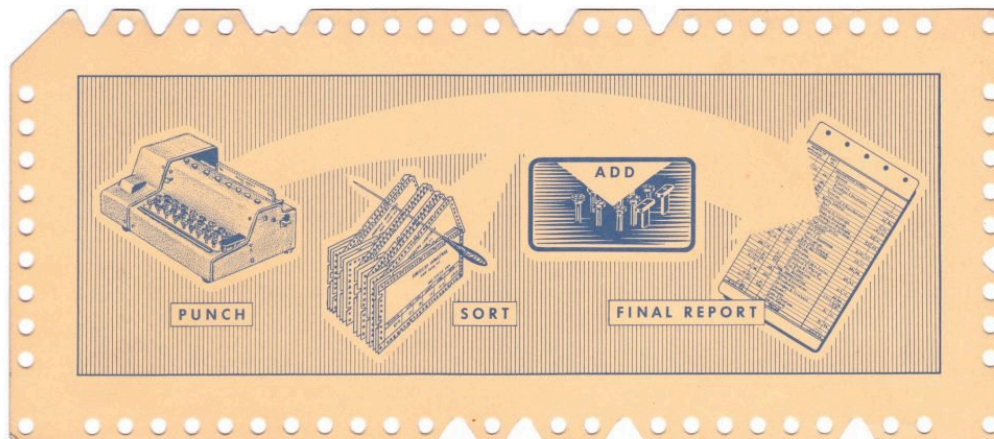


Like a filing cabinet,
but smol

Index cards

OL 462 466 .S27 v.1 <i>Mr. T. Description of and figures of the insects of North America. Illustrated by coloured figures from original drawings executed from nature. By Thomas Say ... [Philadelphia] Philadelphia Museum, S.A. Mitchell, 1824-28.</i>	OL 411 5 33 Say, Thomas, 1787-1834 The complete writings of Thomas Say, on the conchology of the United States. Edited by W.G. Binney. New York, H. Baillière, etc., 1858. Title-page: Description of terrestrial shells....: Phila., Childs & Peterson, 1856; p.1-40. vi, 253p. and 76 plates (pt. col.) 22 cm. Contains unnumbered plate of <i>Helix thyroidea</i> after B. Most of the drawings by Mrs. Lucy Say. --- --- Copy 2 --- t.p. Descrip. of Terrestrial Shells: N.Y., H. Baillière, 1858; p.1-40. Lacks unnumbered plate. 24 cm.
OL 466 .S27 Vault Say, Thomas, 1787-1834. American entomology, or Descriptions of the insects of North America. Illustrated by coloured figures from original drawings executed from nature. By Thomas Say ... [Philadelphia] Philadelphia Museum, S.A. Mitchell, 1824-28. 3 v. 54 col. pl. 24 cm. Vol. 1 has added t.-p., engraved. 1. Insects--Pictorial works. 2. Insects--North America. 1. Title PPAN 28 SEP 83 1834354 ANSWdc 06-12607	OL 473 3 35 Say, Thomas, 1787-1834. The complete writings of Thomas Say on the entomology of North America. Ed. by John L. Le Conte, n. n. With a memoir of the author, by George Ord ... New York, Baillière brothers; London, H. Baillière; etc., etc., 1839. 2 v. 54 (i.e. 53) pl. (54 col.) 24 cm. v. 1 has portrait of Say. 1. Insects--North America. 2. North America--Entomology. 1. Le Conte, John Lawrence, 1825-1893, ed. n. Ord, George, 1781-1838. 2. Say, Thomas, 1787-1834 - Portrait. Agr 5-237 U. S. Dept. of Agr. Lib. for Library of Congress 62115a8 (21)

Edge-Notched Cards

[illegible]

Edge-Notched Cards

GRANT APPLICATION FORM

PROJECT TITLE: Urban Studies Program for in-service training of teachers, sponsored by Humanities and Social Sciences at St. Mary's College and the Graduate Department of Education of the University of Notre Dame.

LOCATION: Notre Dame, Indiana

DATES: 3/1/70 to 3/31/71

AMOUNT: \$10,000

GRANT TOTAL: \$10,000

ED-1000

Ann Sneesby-Koch, **Me and Mr McBee**, National Endowment for the Humanities, 2013.

Edge-Notched Cards

A	b	d	o	II	I	u	m	l	k	III
	Bearb. DE	IX/1	Vorg. Nr.	XV 6942/80	Erf. Nr.	22976				
	Name	Dr. Teske	Vornamen	Werner, Siegfried						
	geb. am	24. 4. 1942	429 F25	in	Berlin					
B	Beruf	Diplomwirtschaftler		soz. St.						
	letzte Tätigk.	Mitarbeiter MFS								
	letzte Arbeitst.									
C	letzte Wohn.									
	Staatsang.	DDR								
	Vorstrafen DDR	keine								
	Vorstr. WD/WB/Ausl.									
D	R/Z									
	Parteizugeh.	s. 1967 SED	Org.	DSF, DTSE						
	früh. Wehrd. verh.									
	Verf. eing. am	11. 9. 1980	durch	MFS	wegen					
	festgen. am	11. 9. 1980	durch		wegen					
E	übern. am		wegen							
F	Tatbestand	§§ 97, 254 StGB								
	zugrundel. Mat.	HVA und Abt. Disziplinar								
G	HB am	12.9.1980	§§ 97, 254 StGB	Grund						
H	Erweit. am		§§	aufgeh. am						
	Abschl. am	31.3.81	mit Übergabe MStA	aufgr.						
I	Abschl. Tatbest.	§§ 97 (1)(3), 99 (1), 254 (1)(2)1, 110 (1), 108 , 63 (2) StGB								
J	Endgült. Abschl.	11. 6. 81	durch OG, Militärstrafsenat							
	mit	lebenslänglich								
K	Tatbestand	§§ 97(1)(2)(3), 110(1), 254(1)(2)1(3) StGB								
	Öffentlichk.	nicht öffentlich								
	Berufung/Protest									

interne Ablage

640

4.7

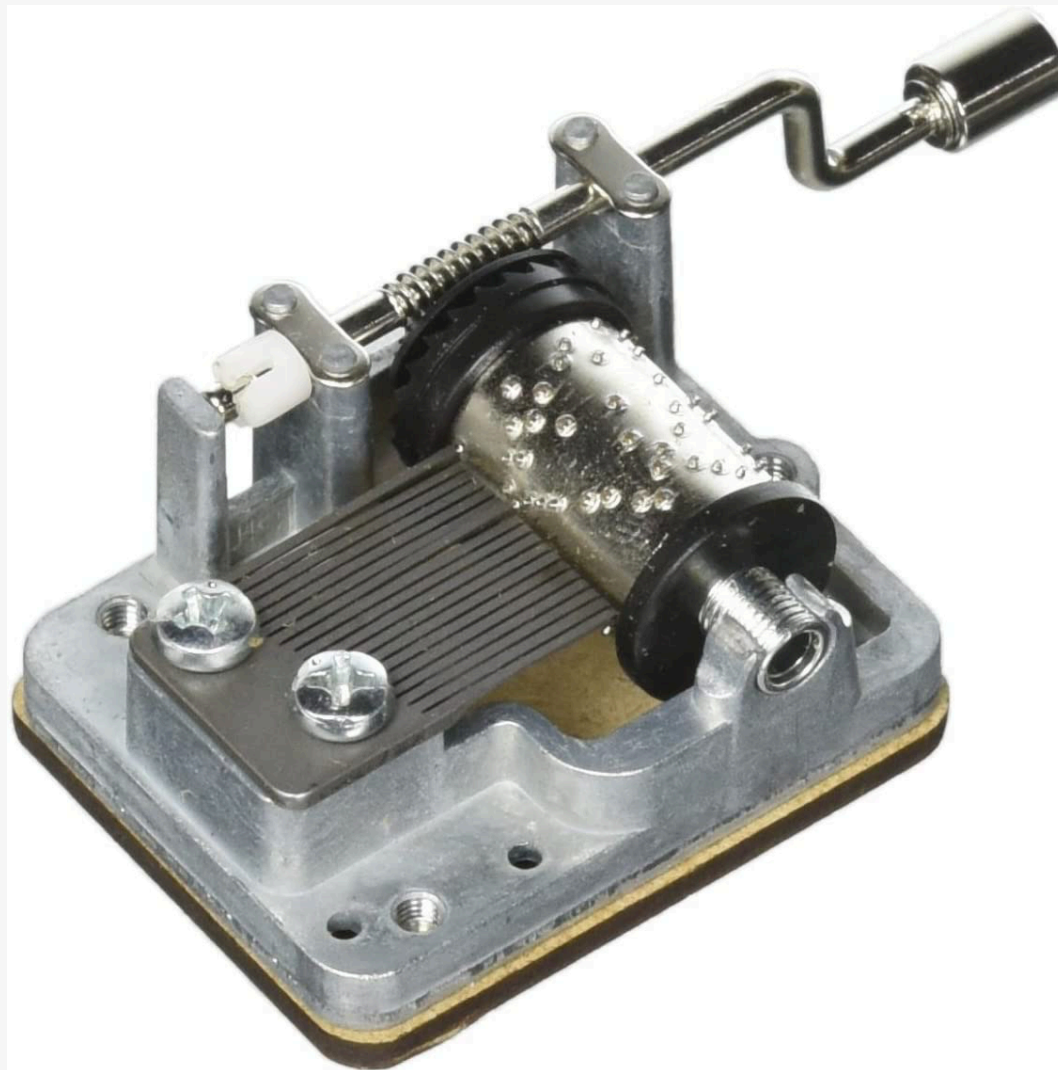
ME 81/6

Ges. Ankl./Vert./Mitw. v. Koll.

Stasi record card for
Werner Teske, a
former Stasi employee
sentenced for
espionage, from 1981.
(Source.)

Automating Information and Control

A music box



A Jacquard Loom



Jacquard Loom Cards



Tabulation Machines

Hollerith Machines



Hollerith Machines

1	2	3	4	CM	UM	Jp	Ch	Oc	In	20	50	80	Dv	Un	3	4	3	4	A	E	L	a	g
5	6	7	8	CL	UL	O	Mu	Qd	Mo	25	55	85	Wd	CY	1	2	1	2	B	F	M	b	h
1	2	3	4	CS	US	Mb	B	M	O	30	60	O	2	Mr	O	15	O	15	C	G	N	c	i
5	6	7	8	No	Hd	Wf	W	F	5	35	65	1	3	Sg	5	10	5	10	D	H	O	d	k
1	2	3	4	Fh	Ff	Fm	7	1	10	40	70	90	4	O	1	3	O	2	St	I	P	e	l
5	6	7	8	Hh	Hf	Hm	8	2	15	45	75	95	100	Un	2	4	1	3	4	K	Un	f	m
1	2	3	4	X	Un	Ft	9	3	i	c	X	R	L	E	A	6	0	US	Ir	So	US	Ir	So
5	6	7	8	Ot	En	Mt	10	4	k	d	Y	S	M	F	B	10	1	Gr	En	Wa	Gr	En	Wa
1	2	3	4	W	R	OK	11	5	l	e	Z	T	N	G	C	15	2	Sw	FC	EC	Sw	FC	EC
5	6	7	8	7	4	1	12	6	m	f	NG	U	O	H	D	Un	3	Nw	Bo	Hu	Nw	Bo	Hu
1	2	3	4	8	5	2	Oc	O	n	g	a	V	P	I	Al	Na	4	Dk	Fr	It	Dk	Fr	It
5	6	7	8	9	6	3	O	p	e	h	b	W	Q	K	Un	Pa	5	Ru	Ot	Un	Ru	Ot	Un

11015

Hollerith Machines

1	2	3	4	CM	UM	Jp	Ch	Oc	In	20	50	80	Dv	Un	3	4	3	4	A	E	L	a	g		
5	6	7	8	C	F1	L	O	M	F3	d	Mo	25	55	85	F7	Y	1	F8	1	F9	B	F	M	b	h
1	2	3	4	CS	US	Mb	B	M	0	30	60	0	2	Mr	0	15	0	15	C	F10	N	c	F11		
5	6	7	8	No	Hd	Wf	W	F	5	F5	65	1	3	Sg	5	10	5	10	D	H	O	d	k		
1	2	3	4	Fh	F21	Fm	7	1	10	40	70	90	4	0	1	3	0	2	St	I	P	e	l		
5	6	7	8	Hh	Hm	8	2	15	45	75	95	100	Un	2	4	1	3	4	K	Un	f	m			
1	2	3	4	X	Un	Ft	9	3	i	c	X	R	L	E	A	6	0	US	Ir	Sc	US	Ir	Sc		
5	6	7	8	Ot	En	F20	10	4	k	d	Y	S	M	F	B	10	1	Gr	En	Wa	Gr	En	Wa		
1	2	3	4	W	R	OK	11	5	F17	Z	T	F16	G	C	F14	Sw	F13	EC	Sw	F12	EC				
5	6	7	8	7	F19	1	12	6	m	f	NG	U	O	H	D	Un	3	Nw	Bo	Hu	Nw	Bo	Hu		
1	2	3	4	8	5	2	Oc	0	n	g	a	V	P	I	Al	Na	4	Dk	Fr	It	Dk	Fr	It		
5	6	7	8	9	6	3	0	p	o	h	b	W	Q	K	U	F15	a	5	Ru	Ot	Un	Ru	Ot	Un	

11015

11015

Hollerith Operators



Demonstrating a older card-puncher, probably to show how things had improved with census tabulation methods. This is likely the “Before” picture with a roll from the 1890 Census. The card-puncher is a **Pantograph**.

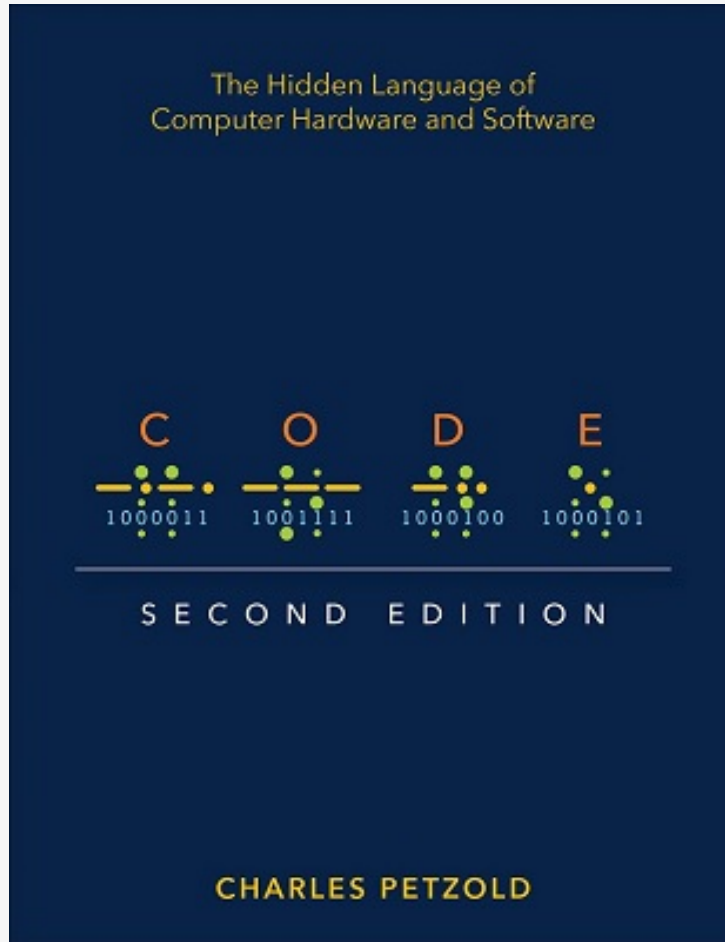
Hollerith Operators



Same woman as the previous photo; her colleague on the right is demonstrating the newer, faster IBM Type 001 Key Puncher. (Again, probably a re-enactment / demo of earlier techniques.)

Programmable Computers

Logic from Sand



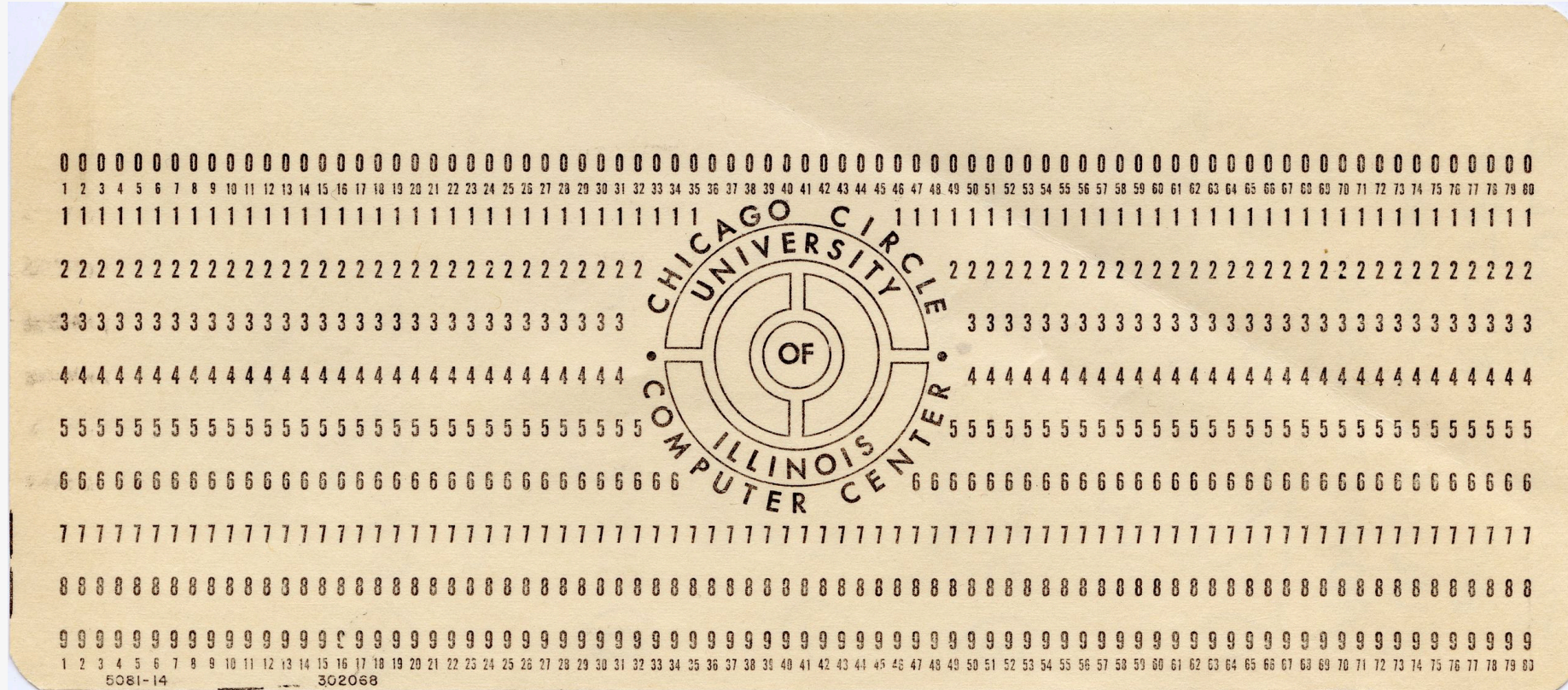
The best book to read about how the guts of a programmable computer works is Charles Petzold's *Code: The Hidden Language of Computer Hardware and Software*, 2nd ed. (Microsoft Press, 2022).

IBM punch cards



In the longer term, punch card writers got much more efficient. And now they could be fed into machines that could use them to run programs instead of just tabulate the punches.

IBM punch cards



An IBM punch card is 80 columns wide. The first CRT terminals displayed 80 columns of text for this reason. You'll see 80 columns of text pop up as a standard in all kinds of places.

Big Iron



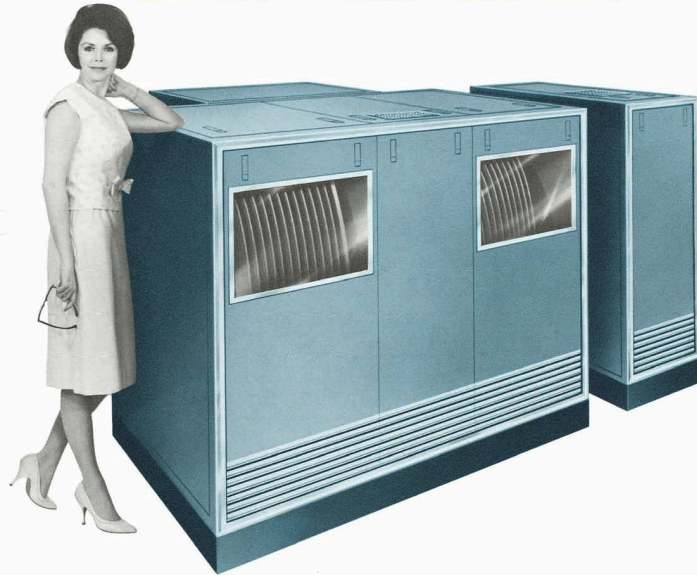
No screens! Paper in, paper out for the operator; magnetic tapes for storage in the background.

This is an IBM/360, the most important class of mainframe in the 1960s and early 1970s.

One thing that's hard to convey in pictures is the way that—because of all the daisy-wheel or tractor-fed printing, mechanical card processing, and huge reels of tape spinning up and down—rooms like this were *loud*.

Storage

*more of everything
you need and want
in a random-access
mass memory...*

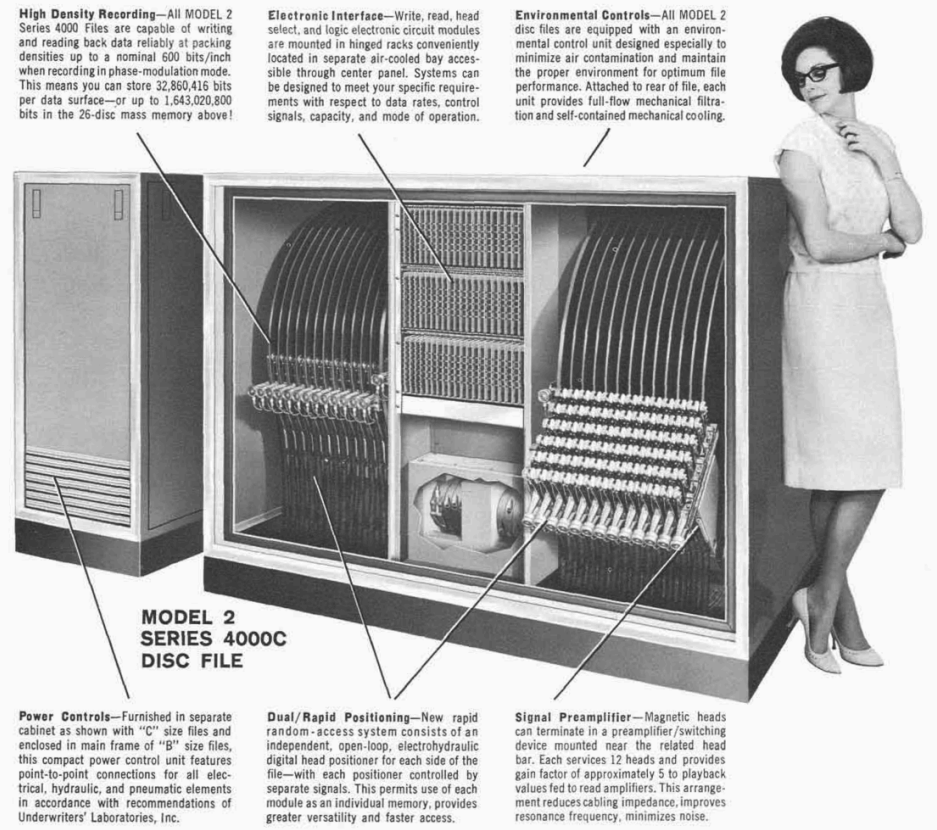


BRYANT Model-2 DISC FILES
SERIES 4000

Copyright 1965—Ex-Cell-O Corporation • All rights reserved

Notice that the “File”
here is the machine
itself, or at most a
single disk platter.

Storage



The older way of speaking is still with us, as when we speak of someone's "Application File" or "Tenure File"; that is, a file is a collection of related documents.

But the newer way, where "file" means "a single document", is now dominant, especially in computing.

What Files Are

A file is a metaphor

Your computer does not have “files” in the way that a filing cabinet has files.

A file is an [abstraction](#), a way of naming and organizing data on your computer that at a lower-level is “just zeros and ones” (and at a lower level than *that* is just electromagnetic patterns in some physical substrate that can be *interpreted* as zeros and ones).

The file metaphor in computing dates most prominently to the development of the Unix operating system in the early 1970s.

Files are organized in filesystems.

There are many kinds of files

As many as there are kinds of application.

Files have the name someone gives them. `My Thesis`, `term_paper`, and so on.

There's a longstanding (though weak) convention about using file extensions, tagged on to the end of a name, to signal *to users* what kind of file it is:

`term_paper.docx`, `.xlsx`, `.ppt`, `.pdf`, `.sqlite`, `.png`, `.jpg`, `.ps`, `.mp3`,
`.mp4`, `.gif`, `.csv`, `.Rmd`, `.qmd`, `.md`, `.txt`.

Files don't know what their extension is, a bit like how electrons don't know what color the outside of their copper wire is.

Binary and Plain Text files

Understanding the general notion of “encoding information” is a very rich and deep topic that, sadly, we are going to skip.

If a file is in some binary format then in general you won't be able to read its contents just by looking inside it. You will need an application that understands the file's particular format; i.e. the way that information in it is encoded.

A `.jpg` file uses a set of rules to store numbers that can be interpreted as corresponding to things like the hue and location of a pixel. But *you* won't see a picture if you look inside a `.jpg` file using a text editor. You'll need an application that knows how to read `.jpg` files.

What is Plain Text?

Text files, though, are sort of special. What's visibly in them appears to correspond much more closely to what they represent. A plain text file seems to represent the letter "A" with a symbol that looks like an "A". So much so that we can say it *is* an "A".

That means that when you look at a text file you can see what is in it immediately. And editing the contents of the file is the same as editing its text.

There's still an "encoding" of course! It's still necessary to have an application that can read the text file and display it on a screen, etc. But what's inside seems much closer to being immediately interpretable "just by looking", because most of it is letters and numbers.

But wait!

But wait!

I thought you said computers just store ones and zeros?

But wait!

I thought you said computers just store ones and zeros?

Yes this is true. In **ASCII** encoding, for instance, an “A” is really just conventionally the symbol represented by the seven-bit binary number **1000001**, which exists on some sort of storage medium (an SSD, a Hard Disk, a floppy disk, a punch card, a reel of paper, whatever) in such a way that some device can read its contents.

But wait!

I thought you said computers just store ones and zeros?

Yes this is true. In **ASCII** encoding, for instance, an “A” is really just conventionally the symbol represented by the seven-bit binary number **1000001**, which exists on some sort of storage medium (an SSD, a Hard Disk, a floppy disk, a punch card, a reel of paper, whatever) in such a way that some device can read its contents.

ASCII is the **American Standard Code for Information Interchange**. It was first specified in 1963.

ASCII

The venerable and now outdated **ASCII** character set: 26 uppercase letters; 26 lowercase letters; 10 digits; 32 printable symbols; and 33 control characters ultimately derived from telegraph code, stock-ticker, and teletype machines.

Binary	ASCII	Decimal	Hexadecimal	Octal
0000000	null	0	0	0
0000001	start of header	1	1	1
0000010	start of text	2	2	2
0000011	end of text	3	3	3
0000100	end of transmission	4	4	4
0000101	enquire	5	5	5
0000110	acknowledge	6	6	6
0000111	bell	7	7	7
0001000	backspace	8	8	10
0001001	horizontal tab	9	9	11
0001010	linefeed	10	A	12
0001011	vertical tab	11	B	13



Modern Text: Unicode and UTF-8

ASCII is a seven bit system that only has 2^7 or 128 “code points” — i.e. individual slots that could represent anything. It left out all kinds of things. (Other alphabets, for instance. Also any diacritics or accents. And any number of symbols.)

Eight-bit computers allowed for 256 code points. The second 128 never had a single standard for what they should represent. The most common extension was **ISO-8859-1** or “Latin1” encoding, but there were others too. This created conflicts and confusion when a program or application expecting text encoded according to one standard was fed text encoded with a different standard.

Modern Text: Unicode and UTF-8

Encoding conflicts are why you still sometimes see this sort of thing on web pages: “CafÃ©” or “Caf ☐ ” instead of “Café”.

It is surprisingly difficult to establish the encoding of a large text file that doesn't explicitly declare how it's encoded in some sort of metadata. (You can guess, but it can be super-annoying.)

Nowadays this has *mostly* been resolved by the adoption of **Unicode** and its simplest and most widespread encoding, **UTF-8**, which extends ASCII to 1,112,064 code points. It uses between one and four eight-bit elements to represent particular character glyphs.

Many older datasets may still be encoded in something other than UTF-8, however.

Organizing Files

Input/Output

Beginning in the 1970s, computing rapidly moves away from print I/O and towards screens.

Storage capacity and processing power increase radically (and get much smaller) with the development of hard drives and integrated circuits.

We get to a point where our “Teletype” interface with the machine is purely metaphorical: this is the *command line* or *console*.

And after that, in the late 1970s and early 1980s, an entirely new set of metaphors gets introduced: files represented by “icons” inside “windows”, first on on a metaphorical “desktop” and then later on a more abstract touch-based surface.

A late-model teletype (TTY) machine



The DEC VT-100 Terminal (1978)



The IBM PC (1981)



The Apple Macintosh (1984)



The macOS Terminal app icon



This is where we came in

The “Office” and “Engineering” models really start to diverge in the 1980s

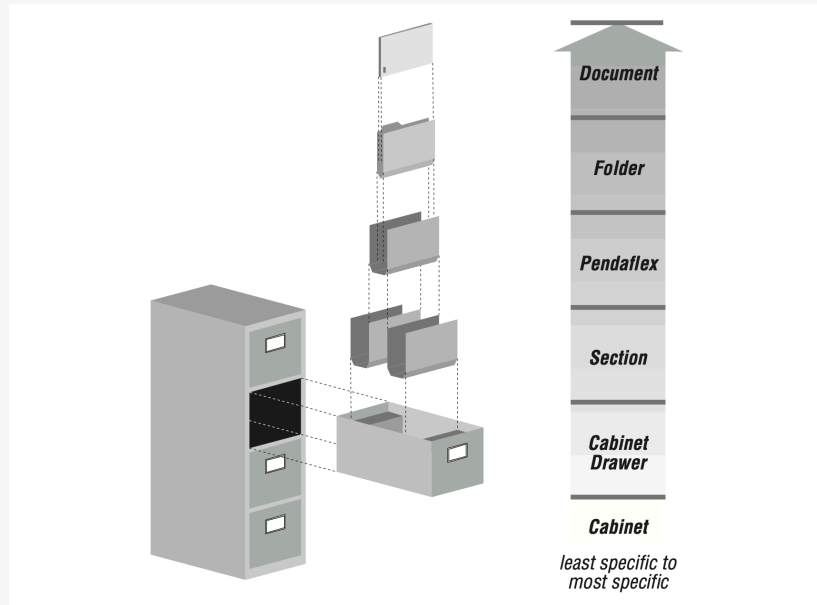
A lot of computing gets done using the Engineering model and its metaphors, even as the Office model comes to dominate.

But many of these **newer systems remain built on top of the world made out of the older metaphors**. And in particular, the idea of *named files* living in a *hierarchical file system* that are acted on in sequence through *written instructions* remains extremely important for many computing tasks.

Especially the stuff we need to do.

Back to the file system

Files



Our data is stored — or represented as being stored — in a *file system*.

This is, again, a way of organizing items for our benefit.

The UNIX operating system developed at Bell Labs codifies the modern “file” metaphor.

Files are named items that live in a [hierarchical file system](#). “Ordinary” documents like `notes.txt` are thought of as files, which seems natural to us now.

The hierarchy is made of [folders](#) or “directories” that, like a filing cabinet, can nest inside one another and inside larger storage units.

By navigating the hierarchy from its **root**, we can trace a [path](#) to any particular file.