Parts Storage

someonesdad1@gmail.com 14 Mar 2011

In the 70's, I developed a method of keeping track of information that worked well for me. I had file cabinets full of manufacturer's information, lots of lab notebooks, interoffice memos and correspondence, loose-leaf chunks of data, etc. Rather than try to come up with some filing system, I used Pendaflex® folders in the file cabinets and gave each folder a number[1]. Information was filed sequentially as it was received. Each Pendaflex folder would get enough information to make it 25-50 mm or so thick.

Around 1980, I started indexing the information on a computer; when an item was looked up, the user was pointed to a specific numbered folder in a particular file cabinet drawer (or lab notebook, etc.). This was before spreadsheets existed, so we had to write our own apps. But it was easy to store things in a text file. One of the things my coworkers appreciated was seeing my lab notebooks indexed -- they could find things after I left the area and turned my notebooks over to them.

Some people call this method chaotic storage and give Amazon as a prototypical user. Barcodes and databases with remote entry allow complex systems that are efficient as long as people keep the database current whenever a product is moved. I've seen comments on how such systems were used during World War 2, but the principles undoubtedly go back into antiquity, as it doesn't take much cleverness to originate the idea.

The same method works well for keeping a bunch of electronic parts. Unlike resistors which have a natural ordering and thus let you use a storage method described here in `resistor.zip`, batches of different electronic parts can be harder to organize. For example, you can store a bunch of op amps in some compartments, but then you run out of compartments in one box/drawer and have to overflow to another compartment in another box/drawer.

To avoid the stress of trying to plan things out, I've found it's easiest to just stick the parts wherever they fit, then index them in a spreadsheet. I like to use clear plastic boxes, so I give them serial numbers; here's an example of one I'm in the process of filling with parts:

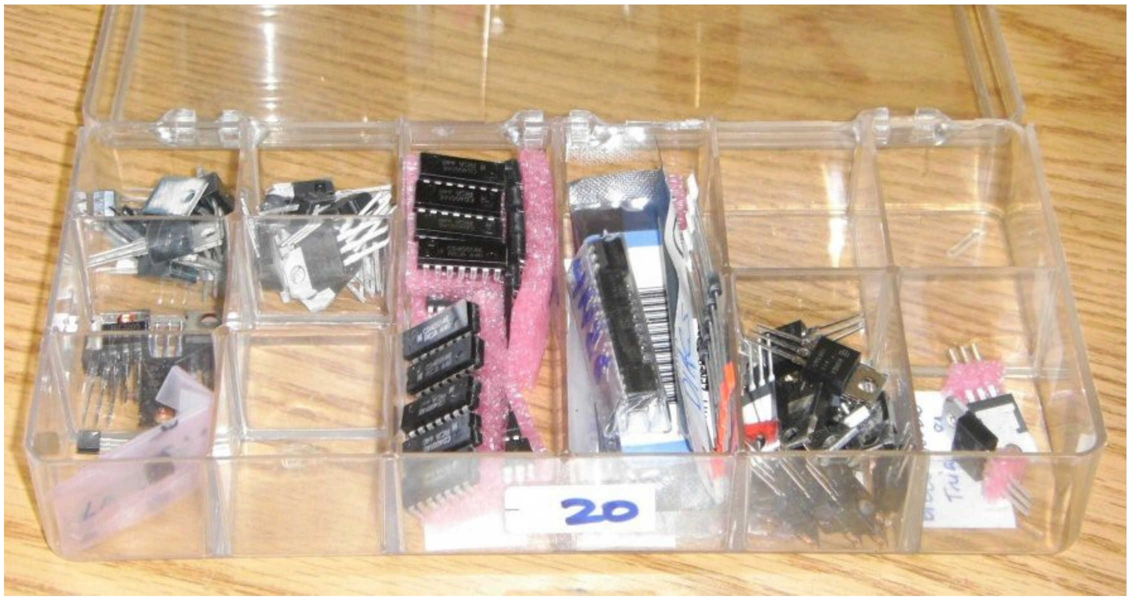


Figure 1

One of the features of this storage method is that I routinely store multiple types of parts in one compartment (as shown in the two center compartments in Figure 1). This isn't a problem, as I

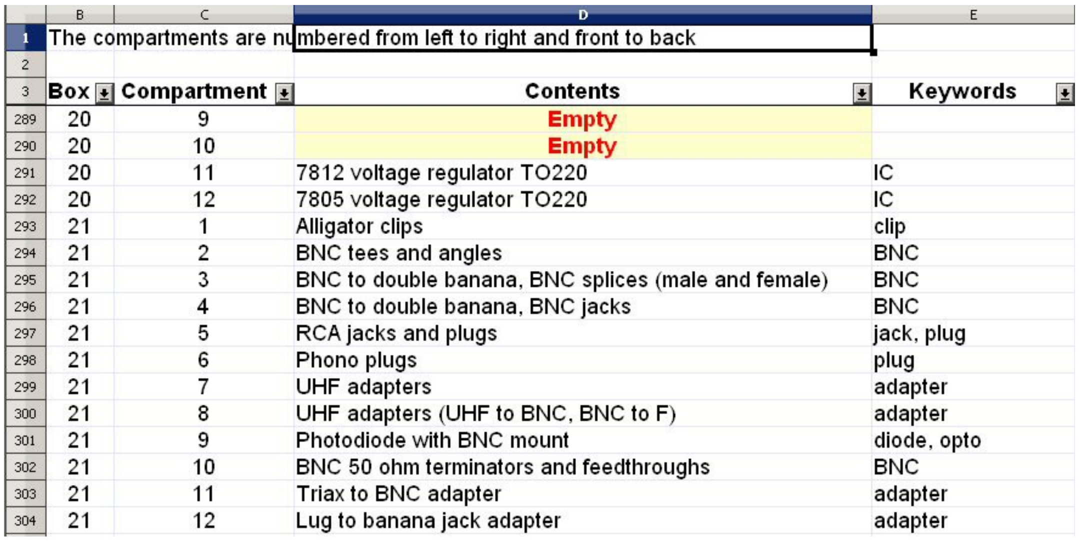
1 Pendaflex is a brand name; it was commonly used to denote the type of hanging folders seen in file cabinets.

make sure that I can distinguish the parts somehow, either by small bags, sticking the IC into some foam, etc. For example, for the ICs stuck into foam, I write the part number in big letters on the back of the foam so that I don't have to go find my microscope to read those tiny letters. ☺ Or, if there are not too many parts in the compartment, I just toss them all in together and can sort them out later when they're needed.

Here are the fields I use in the spreadsheet:

Box	Serial number of the box containing the part.
Compartment	Which compartment in the box contains the part. I number from left to right and front to back.
Contents	Gives the part number and perhaps a few key specs.
Keywords	Keywords that let me find similar parts quickly.

I use an Open Office spreadsheet and the title row gets the autofilter set up on it (Excel has similar functionality). This lets me find things quickly by just picking the keyword. Here's a picture of what the spreadsheet looks like:



	B	C	D	E
1	The compartments are numbered from left to right and front to back			
2				
3	Box	Compartment	Contents	Keywords
289	20	9	Empty	
290	20	10	Empty	
291	20	11	7812 voltage regulator TO220	IC
292	20	12	7805 voltage regulator TO220	IC
293	21	1	Alligator clips	clip
294	21	2	BNC tees and angles	BNC
295	21	3	BNC to double banana, BNC splices (male and female)	BNC
296	21	4	BNC to double banana, BNC jacks	BNC
297	21	5	RCA jacks and plugs	jack, plug
298	21	6	Phono plugs	plug
299	21	7	UHF adapters	adapter
300	21	8	UHF adapters (UHF to BNC, BNC to F)	adapter
301	21	9	Photodiode with BNC mount	diode, opto
302	21	10	BNC 50 ohm terminators and feedthroughs	BNC
303	21	11	Triax to BNC adapter	adapter
304	21	12	Lug to banana jack adapter	adapter

I clearly mark empty compartments so that it's easy to find a storage location when needed.

I also save the spreadsheet to a CSV file. This lets me use a python script (see appendix) to quickly find things from the command line (much of my computer time is at a command line). For example, typing `comp IC` yields:

```
12:4 4001 CMOS quad 2-in NOR
12:4 74LS04 hex inverter
20:1 LM285Z-1.2 voltage reference
20:1 LM285Z-2.5 voltage reference
<etc.>
```

The numbers are the box and compartment number. If I just type `comp`, I get a list of the keywords.

While I dreaded the data entry to the spreadsheet, it really wasn't too bad for around 20 boxes (it has taken a day or two of my time, but I spread it out over a couple of weeks). Here's what my shelf of parts currently looks like (when I get more boxes, I'll probably need one more shelf of parts):

One nice thing about the method is that I have a few parts stored elsewhere in the house. I can leave these where they are, yet still know what's in those storage locations. I don't bother keeping track of quantities, but that's easy to add to the spreadsheet if desired.

As of this writing, there are 318 distinct parts stored in these boxes.

Oh, one other tip. For that day when you need a part and the computer isn't working, print out the spreadsheet sorted by keywords.

Appendix: python script

```
'''
Search for keywords in the components database.
'''

from __future__ import division
import sys, getopt, functools, csv, operator
from collections import defaultdict

beginning_lines_to_ignore = 3
data_file = "d:/p/electronics/spreadsheets/ElectronicParts.csv"
nl = "\n"

manual = '''{name} [options] [keyword [keyword2...]]
  Searches the components database for the indicated keywords ANDed
  together and prints out matches.  If no keyword is given, a list of all
  keywords is printed.

Options
  -a        Dump all records
  -h        Show help
  -k        Show list of keywords
  -o        Combine with OR rather than AND
'''
```

```
def StreamOut(stream, *s, **kw):
    k = kw.setdefault
    # Process keyword arguments
    sep     = k("sep", "")
    auto_nl = k("auto_nl", True)
    prefix  = k("prefix", "")
    convert = k("convert", str)
    # Convert position arguments to strings
    strings = map(convert, s)
    # Dump them to the stream
    stream.write(prefix + sep.join(strings))
    # Add a newline if desired
    if auto_nl:
        stream.write(nl)

out  = functools.partial(StreamOut, sys.stdout)
outs = functools.partial(StreamOut, sys.stdout, sep=" ")
dbg  = functools.partial(StreamOut, sys.stdout, sep=" ", prefix="+ ")
err  = functools.partial(StreamOut, sys.stderr)

def ParseCommandLine(d):
    d["-a"] = False
    d["-k"] = False
    d["-o"] = False
    try:
        optlist, args = getopt.getopt(sys.argv[1:], "ahko")
    except getopt.GetoptError, str:
        msg, option = str
        out(msg + nl)
        sys.exit(1)
    for opt in optlist:
        if opt[0] == "-a":
            d["-a"] = True
        if opt[0] == "-h":
            out(manual)
            exit(0)
        if opt[0] == "-k":
            d["-k"] = True
        if opt[0] == "-o":
            d["-o"] = True
    return args

def GetData():
    C = csv.reader(open(data_file, "rb"))
    d = defaultdict(list)
    for i, row in enumerate(C):
        if i < beginning_lines_to_ignore:
            continue
        loc = ':'.join(row[1:3])
        descr = row[3]
        keywords = row[4].replace(",", " ")
        for kw in keywords.split():
            s = "%-8s %s" % (loc, descr)
            d[kw].append(s)
    return d

def main():
    opts = {} # Options dictionary
    args = ParseCommandLine(opts)
    d = GetData()
    if opts["-a"]:
        all = []
        for i in d:
            for j in d[i]:
                all.append(j)
        all.sort()
        for i in all:
            out(i)
        exit(0)
    if not args:
        out("Keywords:")
        kw = []
        for i in d:
            kw.append(i)
        kw.sort()
        for i in kw:
            out("   ", i)
        exit(0)
    sets = []
    for kw in args:
        if kw in d:
            sets.append(set(d[kw]))
    if len(sets) > 1:
        if opts["-o"]:
            for i in sets[1:]:
                sets[0] = sets[0].union(i)
        else:
            for i in sets[1:]:
                sets[0] = sets[0].intersection(i)
    if sets and sets[0]:
        sets = list(sets[0])
        sets.sort()
        if sets:
            for i in sets:
                out(i)

main()
```