OTTAWA

PC NEWS

Volume 39, Number 5 May 2022

ARTICLE

Squeezing data out of PDF files

by Alan German

If you want to extract a few items of textual information from a PDF file, it's easy to load the file into a PDF viewer, scroll through the text, identify the items required, and cut and paste them to a text file. But, what if there are 13,717 such files? Clearly, for the sake of efficiency, this is no longer going to be a manual operation and an automated solution is required.

This was the problem posed by a colleague who wished to extract four specific data elements from a series of PDF files. But, another issue was that the specific format of the text to be extracted was not always consistent. Our solution to this problem was to convert each PDF file to text and write a computer program to search through the text for various patterns. The programming language of choice was Python, primarily because a PDF-to-text routine, and an excellent suite of search-and-match algorithms, were already available in the public domain.

Python is an interpreted, object-oriented, high-level programming language that is freely available for multiple platforms from the Python Software Foundation (https://www.python.org). PDFMiner.six is a Python tool that allows information to be extracted from PDF documents. This is community-maintained fork of the original PDFMiner software and can be obtained from GitHub (https://github.com/pdfminer/pdfminer.six).

After downloading the current release of Python (Version 3.10.1), the installation is straightforward with the default installation folder being C:\Users\<Username>\AppData\Local\Programs\Python\Python310. Anyone wishing to try out this programming language would be wise to check the box labelled *Add Python 3.10 to PATH* in the first dialogue box in order to ensure that the program can be easily accessed.

The installation of Python includes a package manager, named pip, that can now be used to install the PDFMiner software from a command window. The instructions for pip on GitHub indicate the installation command as: *pip install pdfminer.six*. Running this command installs the entire suite of programs in the PDFMiner package. One of these is pdf2txt.py which, as the name suggests, will convert a PDF file to text. Anyone considering using PDFMiner should note that only text strings that are comprised of either ASCII or Unicode characters will be processed. Text that is in the form of images, and which would thus require OCR capability, will not be converted.

GitHub also provides a sample command to run the pdf2txt.py program as: *python pdf2txt.py samples/simple1.pdf*. This command loads Python, calls the pdf2txt script, and converts the specified PDF file to text. The program's output is a simple display of the converted text strings being printed to the screen so, clearly, some additional programming is required to make the output more useful.

For those with no prior knowledge of Python, it is useful to know that there is a lot of official documentation, how-to web posts, and video tutorials covering a myriad aspects of the use of this software. In particular, these often include sample programs and code extracts that can be adapted for other uses. It was through such means that a custom Python program was developed to interrogate multiple PDF files and extract the data variables that were of interest to my colleague.

For those who have knowledge of some other, perhaps more conventional, programming languages Python has some unusual characteristics. One unexpected behaviour was identified when reading a PDF filename from a text file and attempting to open the file. The code used was: file_name = f.readline() which reads a line of text from the file system object f and stores the string in the file_name variable. However, a subsequent attempt to open the PDF file using this variable produced an error. Investigation revealed that the variable contained "20191010003_V2_ACM.PDF/n" which caused the operation to fail since a file with a .PDF/n extension didn't

(Continued on page 6)

Inside this issue:

Next Meeting: WEDNESDAY, May 11th, 2022

Next Meeting

Wednesday, May 11, 2022

Topic: Me & My Mac
Speaker: Gillian Villeneuve, MUGOO (Macintosh User Group Of Ottawa)

Apple and Microsoft have always been rivals. Windows may be more common in the workplace and in homes, but the Apple Macintosh (or Mac) has always impressed with its beautiful graphics and user friendliness. Join Gillian Villeneuve from the Macintosh User Group of Ottawa (MUGOO) as she shows us features of her MacBook Pro and shows us how she uses it everyday at home and at work.

Due to COVID-19 restrictions, this meeting will be via Zoom video conference.

Join us at **https://tinyurl.com/opcug-meeting**. The Zoom link will be live at 7:15 pm. The meeting will begin at 7:30 pm.

The above link includes the meeting ID and password. However, if you are prompted for the information, use:

Meeting ID: **924 9556 0898**
Password: **opcug**

Coming Up…

May 2
Tech Talk #2: Basic Photo Editing
Speaker: Lynda Buske

How many times have you looked at a photo and wished you could brighten just the main subject? Or perhaps you had an image printed and don't like how the store cropped it. Are the colours in your photo less vibrant than they were in person? Lynda will show how all these issues can be addressed using free photo editing software available for both Windows and MacOS. [see details to register]

June 8, 6PM - 9PM
Annual Pizza Night

Our Annual Pizza Night is back! This year it will be held at Vincent Massey Park in Picnic Shelter S2.

The pizza will arrive at approximately 6:30 pm. Pop and water, and home-made cupcakes (courtesy of Bob and Debbie Herres) will be provided,

There will be NO presentation afterward.

(see article next page)

September 14
Members' Favourites Night

All scheduled regular monthly meetings, weekly Q&A sessions, and a link to OPCUG presentations at the OPL are posted on our website at https://opcug.ca/#upcoming. All events are via video conference until further notice.

2022 CALENDAR

Meetings	Date	Time and Venue
Regular Monthly Meeting	Wednesday, May 11th	7:30 pm via Zoom video conference: **https://tinyurl.com/opcug-meeting** To see all scheduled events, visit https://opcug.ca/#upcoming
Next Q&A Session	Wednesday, April 27th	Until further notice, Q&A sessions are no longer held after regular monthly meetings. Join us on all other Wednesdays for weekly Q&A.
Beer BOF (Wing SIG East)	Wednesday, May 11th	Enjoy a cold brew or other beverage in the comfort of your home during the video conference.

ANNUAL PIZZA NIGHT

Wednesday, June 8, 2022, 6 PM - 9 PM
Vincent Massey Park (Heron at Riverside)
Picnic Shelter S2 (see satellite image below)
Paid parking ($1 per half hour)

Layout of Vincent Massey Park (PDF)

Google Maps

Our Annual Pizza Night is back! This year it will be held at Vincent Massey Park in Picnic Shelter S2. There are 9 picnic tables in the shelter, two of which will be used for pizza, drinks, cutlery, etc., so a few extra chairs brought in by members would be appreciated in case they are needed. All food and drink are free for OPCUG members and their guests.

The pizza will arrive at approximately 6:30 pm. Pop and water will be provided. Homemade cupcakes are for desert, courtesy of Bob and Debbie Herres.

There will be NO presentation afterward.

This will be our first in-person meeting in over 2 years and we have the shelter until 9 pm for much overdue chatting. The club has not organized any activities, but people are welcome to bring their favourite game and break off in small groups near the shelter to play catch, Frisbee, Croquet, lawn bowling, or bean bag (to name a few).

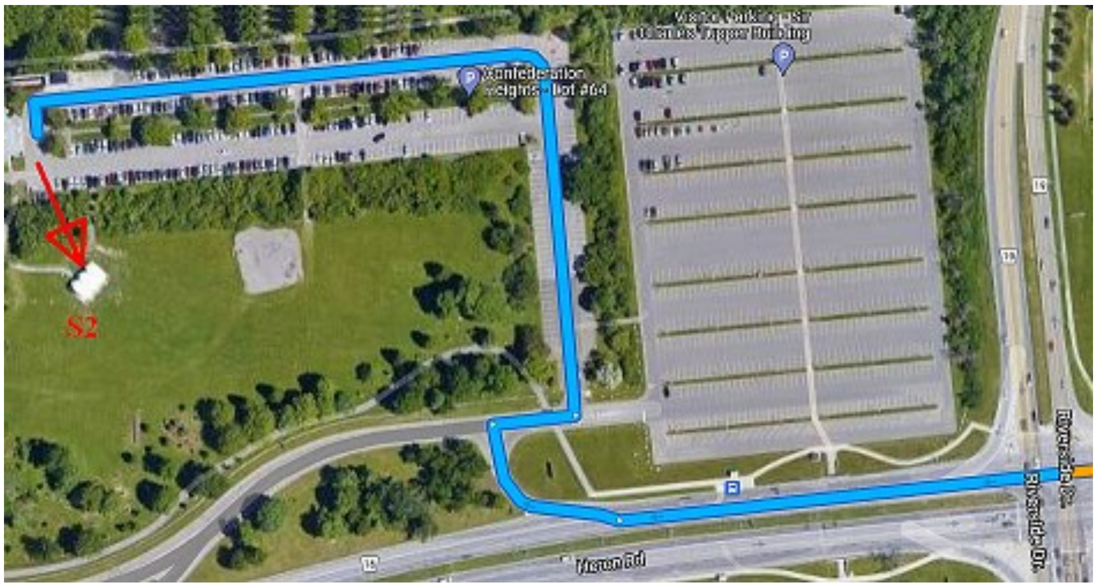
Dogs and alcoholic beverages are not allowed in the park.

Please visit https://ncc-ccn.gc.ca/places/vincent-massey-park for more information.

Garbage detail: There will be recycling bins for pop cans and water bottles (please empty these first), garbage bags for plastic straws and cutlery (these are not recyclable), and yard waste bags for food and paper waste (plates, napkins, etc.). Empty pizza boxes will be collected for composting. Brigitte will bring all this home, so please be mindful when placing items in the recycling bins and yard bags or she will have to sort it! Thank you.

See you there!

(click image to view larger)



(overlaid graphics courtesy of Jocelyn Doire)

ARTICLE

Password strength

by Chris Taylor

I have written before about various aspects of passwords: *Protecting your passwords* (https://opcug.ca/Reviews/ProtectPasswords.html); *Passwords* (https://opcug.ca/Articles/1912NEWS.pdf); and *Home wireless security* (https://opcug.ca/Articles/1909NEWS.pdf).

I still see people either using weak passwords or following outdated advice on what makes for a good password.

Weak passwords

Weak passwords are those that are pretty easy for an attacker to crack. They include;

- Short passwords. Passwords should ideally be 15 characters or longer.
- Dictionary words. Attackers use many dictionaries when attempting to crack passwords including different languages and specialty dictionaries such as medical, quotations, common phrases, rhymes, etc.
- Common passwords known to attackers. One list of the top 10,000 passwords includes the 12-character password "businessbabe" as the 3,611th most common password. Clearly, trying to be a little clever is not enough.
- Something associated personally with you that an attacker might be able to research: your name; the name of your child/spouse/pet/work/hobby/street/nickname/etc.; favourite food…well, you get the idea.

Complex passwords

There is a lot of advice encouraging complex passwords such as, "Think of a phrase and then take the first letter of each word or parts of words, change some to capitals, insert some punctuation, …". So you start with something like *Whatever you do, don't forget the password* and end up with *WuD,d4getTpword*. This is undoubtably a strong password. But I don't think it is very easy to remember and therefore, perhaps it's not a good password.

Better approaches

There are two approaches I recommend: use extremely complex passwords and a password manager; or use four or more seemingly unrelated words strung together.

Password managers

In my article *Protecting your passwords* (https://opcug.ca/Reviews/ProtectPasswords.html), I talked about the process I followed to find a password manager. One benefit of using a password manager is that you don't even have to try to remember your passwords, so you can use very long and complex passwords. Most password managers can generate extremely strong passwords for you, if you want. So fill your boots with passwords that look like *ZW6UBd6Eal&Ob~H,Km*!w0Bu*. Just make sure the password used to open your password manager is easy for you to remember, yet hard to crack. Which brings us to…

Unrelated words strung together

Even though I use a password manager, I prefer to use passwords that I can easily type and *might* be able to remember. The trick is to make them very hard to crack yet easy *for me* to remember. Length wins over complexity. A 15-character password comprised of all lowercase letters wins over a 10-character complex password with upper & lowercase letters, numbers, and symbols.

An easy trick is to string together a series of four or (preferably) more words. Make sure the words are unrelated or make for a nonsense phrase. Some examples might be *ParisIsMadeOfWine*, *WindyRoadsAreHardToClimb*, *TheMoonIsFurtherThanAfrica*, or *PickUpThePhoneAndWrite*. Don't forget about length.

Password strength meters

Be wary of sites that promise to measure how strong your password is. The 2015 paper *Zipf's Law in Passwords* (https://eprint.iacr.org/2014/631.pdf) found that the password *password$1* was rated as "very weak" by Dropbox, "fair" by Google, and "very strong" by Yahoo.

Many password strength meters will rate a string of twenty "P"s followed by the letter "A" as being weak. The password checker at Kaspersky (https://password.kaspersky.com/) complained "A password change is long overdue! Bad news. Repeating character sequences. Your password could be cracked faster than you can say 'Oops!'" But it isn't a weak password. If an attacker has no idea you used the same capital letter twenty times followed by a single different letter, they won't start by guessing the 650 possibilities that match that pattern.

Steve Gibson's *How Big is Your Haystack* site (https://www.grc.com/haystack.htm) more accurately reports that a massive cracking array (making one hundred trillion guesses per second) would take 1.71 million centuries to exhaustively search a 21-character password that contains only uppercase letters. As Steve Gibson notes, once you force an attacker to begin an exhaustive search, the most important factor is password length. Be sure to read all Steve has to say about passwords!

Final words

The National Institute of Standards and Technology (NIST) no longer recommends requiring complex passwords. Many websites have not caught up to up-to-date thinking about passwords. Security guru Bruce Schneier was frustrated when the strong password *:s^Twd.J;3hzg=Q~* generated by his Password Safe program was rejected by a website because it did not contain 2 numbers. You will likely encounter situations where you still have to live with at least some complexity (mix of upper & lower-case letters, numbers, and special characters) in your passwords.

Fortunately, there seem to be few situations where you are required to periodically change your password—another outdated idea no longer recommended by NIST and other authoritative bodies. A strong password does not become weaker with age. You only need to change a strong password if you have reason to believe it has been divulged somehow. ♦♦♦

THROUGH THE LENS

A guide to digital photography for computer enthusiasts. After the click of your camera, you're only half done!

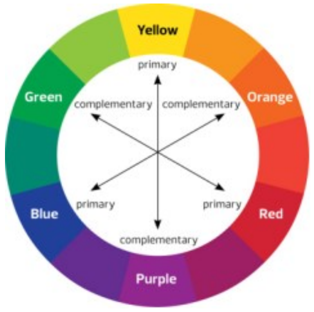
Complementary Colours

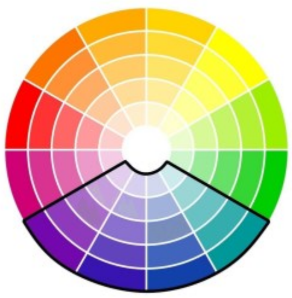
by Lynda Buske

Complementary colours are the opposite hues on the colour wheel. They comprise one primary color plus a secondary color created by mixing the other two primaries. For instance, blue and orange are complementary colours because blue is a primary and orange is made up of the remaining two primary colours, red and yellow. By the same logic yellow and purple (red+blue) are complementary as are red and green (blue+yellow).

The wheel illustrates the range of complementary colours as you rotate around and find the corresponding colour on the opposite side.

The advantage of using complementary colours is that they can make each other look brighter and stand out in a more distinct manner than remaining colour pairings.





Autumn in Canada is a great time for photographing with complementary colours since the red autumn leaves are often paired easily with green leaves/trees and orange leaves look great against a blue sky. Having an equal proportion of two complementary colours can be overwhelming so some photographers suggest having one as a smaller part of the photo or as an accent colour only. This would work well for clothing on a model where you might not want a purple and yellow striped dress but a pale-yellow dress with a mauve scarf might look nice. You don't necessarily have to pair colours with the same intensity of shade. For instance, a dark green could go well with a light red or pink.

In the photo below, you can see how much more striking the orange and yellow leaves are against the blue sky than the evergreen.

Don't get too hung up on exact opposites across the colour wheel. Blue goes nicely with yellow and dark turquoise goes well with light pink even though the colours are not directly across from one another on the wheel.

Blue and yellow *(right)* are close enough to complementary as are turquoise and pink *(below)*.

If you are not satisfied with how your camera captured the colour, you can make adjustments with photo editing software. You can deepen colours using a saturation function or you can change the tint or temperature (warmth or coolness of the light). More advanced programs will allow you to emphasize a single colour in your photo if, for instance, you wish to intensify just the orange leaves but leave the rest of the image as it was originally captured. ♦♦♦

Lynda regularly gives presentations for the OPCUG at the ***Ottawa Public Library*** *(https://opcug.ca/opl-presentations/).*
This article is also in PDF format on the OPCUG website (https://opcug.ca/digital-photography/).

PDF files

(Continued from page 1)

exist! This error was avoided by stripping the escaped newline character from the file_name variable using: file_name = file_name.rstrip("\n").

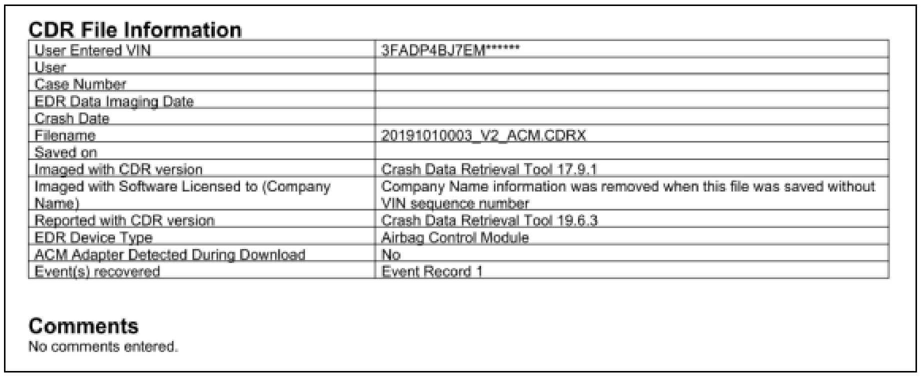
Another feature of Python that initially resulted in errors was the fact that code indentation is critical since no clause terminators (e.g. end, end if) are used. For example, in Python a conditional if clause starts with *if {expression is true}:* which is followed by four spaces to indent the code with the action(s) to be taken. The end of the conditional statement is identified by a subsequent line of code not being indented. Note also that the familiar if -then command is replaced by an if-colon code sequence

Because we were new to Python, both the need for the colon in conditional statements, and the strict requirements for code indentation, initially created a number of runtime errors. A piece of software that proved exceedingly useful to further develop and modify our code was PyCharm. This is an integrated development environment (IDE) for Python that provides a Python editor, together with built-in testing and debugging features. The lines of code are numbered in the editor which really simplifies the process of reviewing and correcting any errors that are detected. The Community Edition of PyCharm is free software and can be obtained from Jet Brains (https://www.jetbrains.com/pycharm).

Our program code had to loop through a list of PDF filenames, convert each PDF file to text, search through the resulting text in order to identify the locations of a number of specific elements, extract the data elements of interest, and then write the values of these elements to a file.

For our purposes, a version of the pdf2txt script that was structured as a function was used to convert each PDF file to a single text string. While the PDF files did contain a number of tables and charts, when converted, the data elements of interest all appeared as simple text strings. So, with the entire file reduced to a single text string, the required variables could be extracted as sub-strings. Interestingly, the pdf2txt utility includes the option to use layout analysis to structure the resulting text in various ways. For our purposes, we disabled this feature (laparams = None) in order to force the output to be a simple text string.

The other feature of Python that we found most useful was the ability to search text strings in multiple ways using the re (regular expressions) module. The reasons for using this technique can be seen by looking at the following extract from a sample PDF file:



CDR File Information

User Entered VIN	3FADP4BJ7EM******
User	
Case Number	
EDR Data Imaging Date	
Crash Date	
Filename	20191010003_V2_ACM.CDRX
Saved on	
Imaged with CDR version	Crash Data Retrieval Tool 17.9.1
Imaged with Software Licensed to (Company Name)	Company Name information was removed when this file was saved without VIN sequence number
Reported with CDR version	Crash Data Retrieval Tool 19.6.3
EDR Device Type	Airbag Control Module
ACM Adapter Detected During Download	No
Event(s) recovered	Event Record 1

Comments

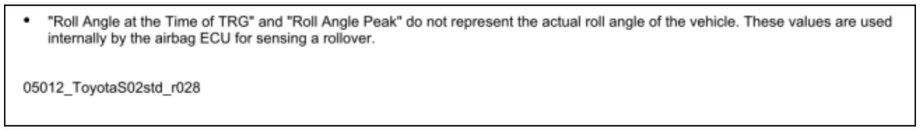
No comments entered.

Three of the variables of interest are depicted here, namely: User Entered VIN, Filename, and Event(s) recovered. For each of these variables, we used the re.search construct to locate and extract a text string that appears between two other text strings. For example, with the PDF file converted to the text string *str*, the User Entered VIN was extracted as the variable *VINstr* using:

```
VINstr = 'String not found'
result = re.search('User Entered VIN(.*?)User', str)
if result:
    VINstr = result.group(1)
```

In this code, re.search extracts all the characters (.*?) that occur between the strings "User Entered VIN" and "User". These two strings correspond to the titles used in lines 1 and 2 of the table so that VINstr becomes "3FADP4BJ7EM******". Note that, should the search fail, VINstr retains the value initially set as "String not found".

Identifying the Module ID variable proved to be a little more complex mainly because the text that immediately preceded the Module ID was not constant. This problem is illustrated by the following extract from one of the PDF files:



- "Roll Angle at the Time of TRG" and "Roll Angle Peak" do not represent the actual roll angle of the vehicle. These values are used internally by the airbag ECU for sensing a rollover.

05012_ToyotaS02std_r028

The problem is that the text that precedes the Module ID is the end of the last bullet point in a series of "data limitations" that spans several pages. These data limitations differ widely such that there is no well-defined ending point for the text. Consequently, our previous strategy of extracting a sub-string between two known text strings cannot be employed to define the Module ID.

Instead, we access the power of Python's regular expressions and search through the converted text in order to match a defined pattern. For example, in the current instance we can use the fact the module commences with the string 05012_ and ends with an underscore and a revision number as _r028 in order to search for a string matching the pattern '0\d\d\d\d_.*_r\d\d\d'.

Special characters are defined for regular expressions and use the \ character as an escape sequence. For example, \d represents any numerical digit from 0 to 9. So, our sample pattern begins with a 0 (zero), followed by any four numerical digits (\d). and an underscore character. Similarly, the pattern is to be terminated with an

(Continued on next page)

PDF files *(Continued from previous page)*

underscore character, followed by r, and any three numerical digits. The .* sequence indicates that any number of alpha-numeric characters may appear between the two matching patterns.

In practice, several such patterns were found to be required to match all the Module ID's in the set of PDF files under consideration. These various patterns were identified by setting the ModuleIDstr variable for any unmatched Module ID's to "String not found". The variables identified by the Python program were read into an Excel spreadsheet and filters applied to all the data columns so that rows containing "String not found" could readily be identified. A review of the associated PDF files then identified the Module ID's that had not been matched and additional matching patterns were developed to accommodate these instances. The result was a series of conditional statements that searched the converted text string (str) for the defined patterns as indicated in the following code extract.

```
ModuleIDstr = 'String not found'
# Search for ID commencing with five digits as 0xxxx
result = re.search(r'0\d\d\d\d_.*_r\d\d\d', str)
    if result:
        ModuleIDstr = result.group()
# Search for ID commencing with five digits and _Volvo as 1xxxx_Volvo
result = re.search(r'1\d\d\d\d_Volvo.*_r\d\d\d', str)
    if result:
        ModuleIDstr = result.group()
```

The programming exercise described here should demonstrate that Python offers a set of ready-made, simple-to-use, yet extremely powerful tools. While the current project involved processing PDF files as simple text, and searching through the resulting string for specific sub-strings, Python is frequently used for many other applications, including programming micro-controllers. The software is available at no cost and there is a wealth of documentation and sample code available which together provide a great opportunity for anyone who wishes to try their hand at developing custom programs.

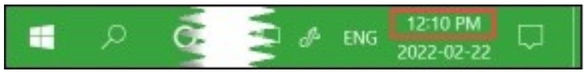
And, as a final note, our 13,717 PDF files yielded a total of 365 unique Module ID's identified using 11 different matching expressions, including a number of similar modules with different revision numbers (_rXXX).

♦♦♦

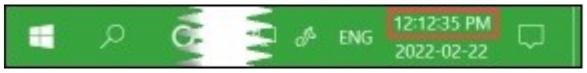
Quick Tip 44: Add seconds to the Taskbar clock

by Chris Taylor

The Windows 10 clock on the taskbar shows hours and minutes.



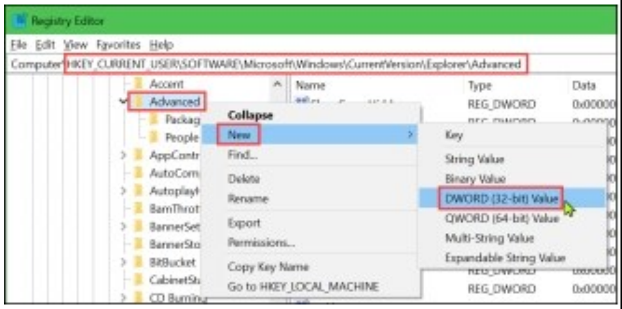
A simple registry hack can add seconds to the display.



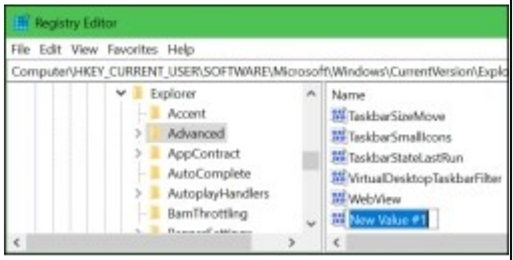
Caution: while it is fairly straightforward to edit the Windows registry, don't make random changes. There is no *Undo* and changes take effect with no *Save* command. If this makes you nervous, see the penultimate paragraph. Having a good image backup of your computer is—as always—advised.

Open the ***Registry Editor*** (***Start|Windows Administrative Tools***). In the left pane drill down to:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced

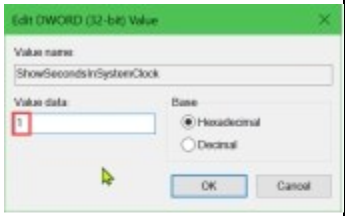


Right-click on ***Advanced*** and choose ***New*** | ***DWORD (32-bit) value*** from the context menu.



In the open dialogue box, overwrite the default ***New Value #1*** and name the value ***ShowSecondsInSystemClock***.

If you accidentally click off ***New Value #1***, right-click it, and choose ***Rename***.



Value name: ShowSecondsInSystemClock

Value data: 1

Double-click the ***ShowSecondsInSystemClock*** value in the right pane, set the ***Value data*** to ***1***, and click OK.

Close the Registry Editor, sign out of Windows and sign back in. Your clock will now display seconds.

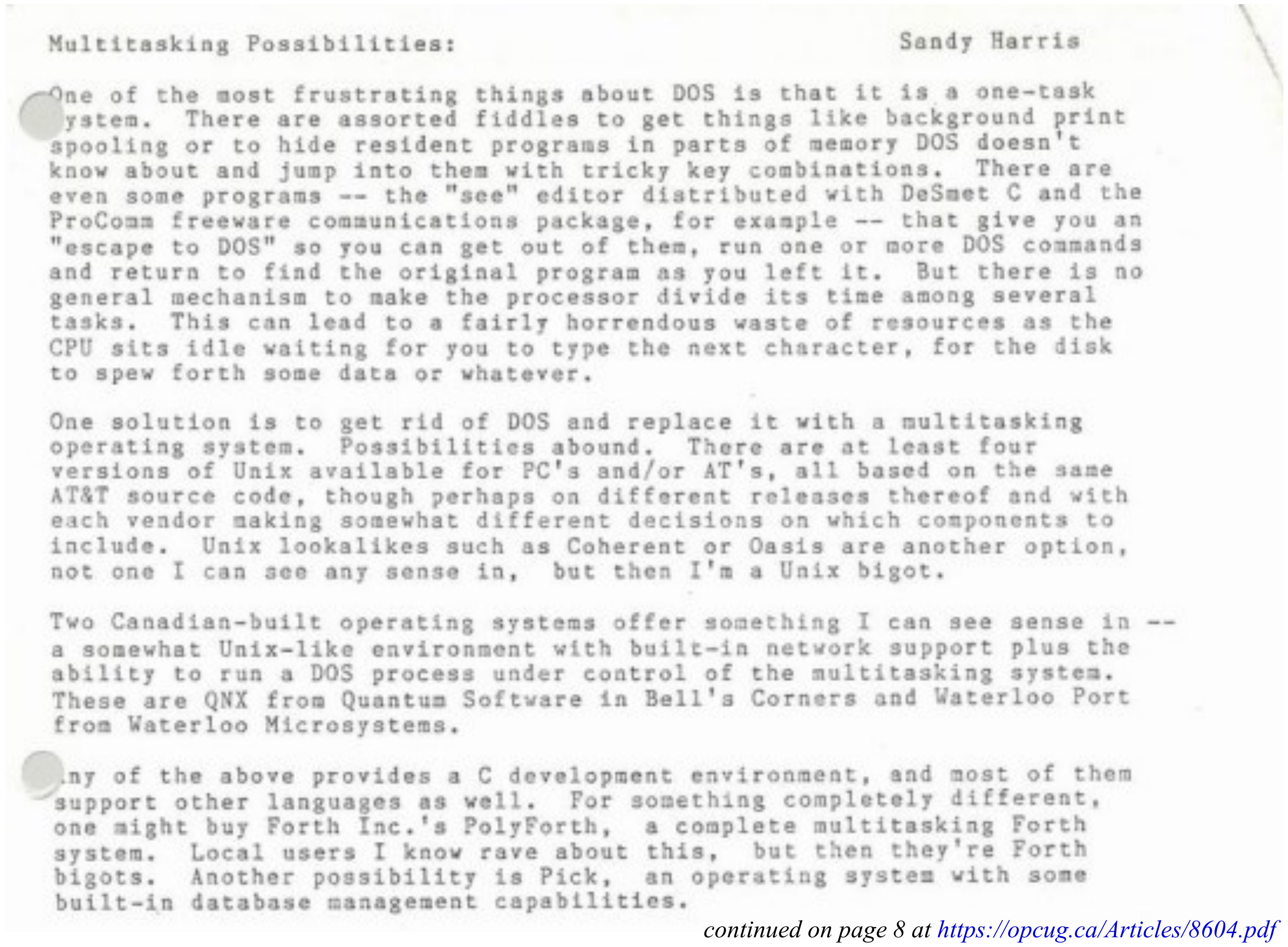
If you want to change the clock back to not show seconds, edit the ***ShowSecondsInSystemClock*** value and change the ***Value data*** to ***0***.

If you are uncomfortable with using the Registry Editor, you can download ***https://opcug.ca/downloads/TaskbarClockSeconds.zip***. See the ReadMe.txt file in the ZIP file for instructions.

As of the writing of this article, this does not work in Windows 11.

BLAST FROM THE PAST - 1986

Below is an article from the **April 1986** newsletter about some early multitasking operating systems, with talk of DoubleDOS, MultiLink, DOSamatic (early freeware), PASCAL, and much mention of UNIX and UNIX-based/inspired systems, including Ottawa-based QNX. GNU is mentioned before its release as an open source O/S and still operates today, having morphed into LINUX.



Multitasking Possibilities: Sandy Harris

One of the most frustrating things about DOS is that it is a one-task system. There are assorted fiddles to get things like background print spooling or to hide resident programs in parts of memory DOS doesn't know about and jump into them with tricky key combinations. There are even some programs -- the "see" editor distributed with DeSmet C and the ProComm freeware communications package, for example -- that give you an "escape to DOS" so you can get out of them, run one or more DOS commands and return to find the original program as you left it. But there is no general mechanism to make the processor divide its time among several tasks. This can lead to a fairly horrendous waste of resources as the CPU sits idle waiting for you to type the next character, for the disk to spew forth some data or whatever.

One solution is to get rid of DOS and replace it with a multitasking operating system. Possibilities abound. There are at least four versions of Unix available for PC's and/or AT's, all based on the same AT&T source code, though perhaps on different releases thereof and with each vendor making somewhat different decisions on which components to include. Unix lookalikes such as Coherent or Oasis are another option, not one I can see any sense in, but then I'm a Unix bigot.

Two Canadian-built operating systems offer something I can see sense in -- a somewhat Unix-like environment with built-in network support plus the ability to run a DOS process under control of the multitasking system. These are QNX from Quantum Software in Bell's Corners and Waterloo Port from Waterloo Microsystems.

Any of the above provides a C development environment, and most of them support other languages as well. For something completely different, one might buy Forth Inc.'s PolyForth, a complete multitasking Forth system. Local users I know rave about this, but then they're Forth bigots. Another possibility is Pick, an operating system with some built-in database management capabilities.

continued on page 8 at https://opcug.ca/Articles/8604.pdf

CLUB HISTORY IN 1986

(taken from https://opcug.ca/history/)

Oct 1986: University of Ottawa Computer show on Oct 18th and 19th.

Jun 1986: To get publicity, the club gives a copy of Ashton Tate's Framework as a door prize.
Chris Taylor organizes a software contest for the best original program or enhancement to an existing program.
The courses made a profit of $80. The instructors are to get $20 per night.
A contest is organized to design a logo. The club's name inclusion of IBM is misleading; many now have clones.

Mar 1986: All new members get disk 00, which is an index to the entire software library.
The first SIG meeting on Packages (now known as suites) is held with 16 members.

Feb 1986: Access to the Heath board is now working. The committee is looking at getting our own system.
The software library now has 101 disks.

Jan 1986: There are many options to establish a BBS including one at Carleton, QNX and FIDO (which would be free and available 24 hours per day)
Possible meeting topics include representatives from Borland or Microsoft, ham radio, Michael Cowpland and his new laser printer, VNIX, Software Kinetics.
Mike Luckham gives a course on programming in assembly language.

OTTAWA PC NEWS

Ottawa PC News is the newsletter of the Ottawa PC Users' Group (OPCUG), and is published monthly except in July and August. The opinions expressed in this newsletter may not necessarily represent the views of the club or its members.

Member participation is encouraged. If you would like to contribute an article to Ottawa PC News, please submit it to the newsletter editor (contact info below). Deadline for submissions is three Sundays before the next General Meeting.

To receive the monthly newsletter by email, send an email to:
opcug-newsletter+subscribe@googlegroups.com (leave subject and body fields blank)
You do **not** need to create a Gmail or Google Groups account.

To subscribe to other OPCUG Google Groups member services, go to:
https://opcug.ca/google-groups-how-to/

Group Meetings

OPCUG meets on the second Wednesday in the month, except July and August, at the Riverside United Church, 3191 Riverside Drive, Ottawa. Parking is free at the church. OCTranspo bus #90 stops nearby. Details at https://opcug.ca/venue/.
(NOTE: Due to COVID-19 safety guidelines, all our events are via video conference until further notice. Details at https://opcug.ca/venue/)

Meetings are 7:30–9:00 p.m. followed by a Q&A Session until 10 p.m.

OPCUG Membership Fees:	$20 per year
Mailing Address:	3 Thatcher St., Nepean, Ontario, K2G 1S6
Web address:	**https://opcug.ca**
Follow us on Facebook:	**https://www.facebook.com/opcug**
Follow us on Twitter:	**https://www.twitter.com/opcug**

President and System Administrator		
Chris Taylor	**chris.taylor@opcug.ca**	613-727-5453
Meeting Coordinator		
Lawrence Patterson	**meetings@opcug.ca**	
Treasurer		
Alan German	**alan.german@opcug.ca**	
Secretary		
Gail Eagen	**gail.eagen@opcug.ca**	
Membership Chairman		
Mark Cayer	**mark.cayer@opcug.ca**	613-823-0354
Newsletter		
Brigitte Lord (editor/layout/e-distribution)	**brigittelord@opcug.ca**	
Public Relations		
Lawrence Patterson	**PR@opcug.ca**	
Facilities		
Bob Walker		613-489-2084
Webmaster		
Brigitte Lord	**webmaster3@opcug.ca**	
Privacy Director		
Wayne Houston	**privacy2@opcug.ca**	
Special Events Coordinator		
(Mr.) Jocelyn Doire	**jocelyn.doire@opcug.ca**	
Director w/o Portfolio		
Karen Wallace-Graner	**karenwg@opcug.ca**	

Q&A

Q&A HAS GONE ON-LINE! WEEKLY!

Because of the pandemic, the OPCUG is holding weekly Q&A sessions in Zoom video-conferences.

Join us every Wednesday (except on regular monthly meeting nights) at 7:30 pm to discuss computer issues. Questions (and answers) on any computer-related issue are welcome. Or, do you have a favourite computer program or topic that you would like to share with the group? Send your questions, answers, or the details of what you would like to share to: SuggestionBox@opcug.ca

Everyone is welcome to attend Q&A sessions and to ask questions about their specific computer-related problems. Join us at: https://tinyurl.com/opcug-meeting (if you use the Zoom client, the meeting ID is **924 9556 0898** and the password is **opcug**).

OPCUG

Users helping users for over 40 years